Open innovation meets Changeability: Strategic design analyses for Cyber-physical Industry platforms

Doctoral Dissertation of:

Supervisor:
Prof. Gaetano Cascini

Tutor:
Prof. Stefano Bruni

Opponent:
Prof. Olivier L. de Weck,
Massachusetts Institute of Technology

February, 16th 2016
Abstract

Nowadays, engineering design has to face challenges related to customers’ differentiation and high demand for novelty and innovation. At the same time, sustainability has become a priority, as resources consumption and waste generation are seriously threatening the quality of life of future generations.

A viable concept to address these challenges is provided by cyber-physical industry platforms. Industry platforms can take advantage of an independent community of developers as external sources of innovation; furthermore, their modularity allows customers to tailor the product’s features and update them easily during the utilization phase, thus increasing the lifecycle value and reducing the need for product substitution. In particular, cyber-physical systems have some technological characteristics that make them ideal as industry platforms. However, engineering literature lacks a comprehensive methodology to support the design of cyber-physical industry platforms.

This thesis provides an initial strategic design methodology blueprint, based on three subsequent design analyses. In the first part, the effect of system architecture on changeability is investigated, so that the system can be easily modified during its lifecycle. Then, a value analysis of the platform configurations is carried out and the most valuable platform configurations and modules are highlighted. Finally, as every industry platform is coupled with a multi-sided market, the third part provides a socio-technical simulation of the platform ecosystem, whose results can determine both the technical features a platform should have and the most appropriate strategic management of the ecosystem.

Two practical applications are provided. The former scrutinize the effect of modularity and architectural features over a large sample of numerically-generated architectures; the latter employs the three analysis for the strategic design of a modular, customizable smartphone. Both applications illustrate the usefulness of the proposed analyses and provide suggestions for future cyber-physical systems design.

The change propagation sensitivity analysis demonstrates that system architecture does play a role in determining the change propagation behaviour, therefore it is possible to architect systems to reduce change propagation. The subsystems’ degree, the system diameter, the number of component loops and the structural complexity are the four most correlated features with the change propagation indices proposed.

As far as the customizable smartphone case study is concerned, the distributions of change indices indicate that the evaluation of changeability depends on several factors, like the agent performing the change and the lifecycle phase during which the change is made; furthermore, a change in the platform core or the CPU is likely to generate many propagated changes. The Logit value analysis forecasts that basic screen modules, mono loudspeakers, the fingerprint reader, the high-speed interface and the 750 mAh batteries will be the most appreciated modules in a customizable smartphone. Finally, the sociotechnical simulations highlights how sustaining the platform ecosystem can be as important as initiating it correctly. The sensitivity analysis statistically proves that the initial number of modules is fundamental for the entire growth of the communities and that each month of product development saved can increase the developers’ community size by five participants. From the same analysis, it appears that a 12.5% rate of malfunctioning platform architectures is sufficient to make the entire market collapse.

A long-term scenario for the development of the research field about cyber-physical systems and ecosystem innovation concludes the work. Three coupled research directions are envisioned: cyber-physical social systems, massive customization and fluent engineering design. The first one focuses on the interactions between human behaviour and cyber-physical systems, the second one explores the freedom of choice offered by customization-intensive products and its implications on both the customers’ attitudes and the logistics chain management; the last one studies how agile product development methods and lean manufacturing can benefit from a constant feedback of data from the actual system operations.
Acknowledgements

After studying systems for some years, it becomes natural to notice the importance of the interrelations between the individual and the community that surrounds him. Now that I look back at the past three years of great struggles, satisfaction, frustration, and excitement, I can only be extremely grateful to all the people I shared my professional and personal life with.

First and foremost, I want to thank Professor Gaetano Cascini for introducing me to the engaging topic of Engineering design, for guiding me when I was entering the quicksand of research and for always giving me his honest and impartial opinion. I am really thankful for your time and dedication.

During my PhD, I had the chance to be part of the wonderful Kaemart research group, which was invaluable, not just for the precious suggestions and the continuous support, but also for the enjoyable moments we spent together, inside and outside the university campus.

Just beyond them, separated by Via La Masa, I can see Professor Stefano Bruni, who had the patience of listening to my presentations during the PhD evaluation and suggested me how to refine and improve my contents.

I cannot forget about the colleagues in the System Dynamics research group and in the Industrial engineering department, who assisted me in what could be mistakenly judged as minor projects, because they were big personal milestones.

I am deeply indebted to Professor Olivier de Weck, because he trusted an unknown PhD student met at a conference and gave him the opportunity to contribute to an extraordinary project in an exciting environment. From you I have learned the importance of looking at the bigger picture and the value of being passionate about research.

I also want to thank Narek Shouqarian, Kaushik Sinha, Ioana Josan-Drinceanu and Olga Willner, with whom I shared so much time discovering the peculiarities and eccentricities of industry platforms design (and the joy of Chinese cuisine!).

My ten months in Cambridge would have not been the same without the colleagues and friends in the Strategic Engineering Research Group, the TPP program and the MIT-Portugal program. You made my stay in the US a truly unforgettable experience.

I acknowledge the support and help of the Google ATAP team, especially Paul, Roshni, Anil, Andrew and Beatriz, without whom this thesis would have been an empty shell of theoretical speculations. I really hope that all your efforts will be repaid by the success of Project ARA.

If the last three years have been so enjoyable, a big role is due to you, my friends. From the cold Swiss mountains to the hot beaches in Crete, passing through nasty pubs in Milan, tough training camps, trips to automotive expositions, crazy Friday nights and cosy Saturday dinners, I have always been overwhelmed by your love and affection.

A PhD is a path, not a goal, and I would have never finished this effort without my family. From the very beginning you have always been there, when I did need you and when I did not need you, never too close, never too far. My gratefulness goes beyond all the words of appreciation I could write.
PART 1.0
Chapter 1. Introduction .............................................................................................................. 13
  1. The bigger picture .................................................................................................................. 13
  2. Cyber-physical systems as industry platforms ................................................................. 14
  3. Research questions and thesis outline .............................................................................. 16
Chapter 2. State of the art ......................................................................................................... 21
  1. Cyber-physical systems design ........................................................................................... 21
      1.1 Technical systems, complexity and systems architecture ........................................ 21
      1.2 Cyber-physical systems ............................................................................................... 23
  2. Platform design .................................................................................................................... 28
      2.1 Definitions and classification ...................................................................................... 28
      2.2 Internal and supply-chain platforms ......................................................................... 29
      2.3 Industry platforms ....................................................................................................... 30
  3. Innovation ............................................................................................................................. 34
      3.1 Definitions and classification ...................................................................................... 34
      3.2 Open innovation ............................................................................................................ 38
      3.3 Innovation diffusion .................................................................................................... 41
  4. Engineering changes and change Ilities ............................................................................. 44
      4.1 Engineering changes and technology infusion ............................................................. 44
      4.2 Change-related Ilities’ frameworks .............................................................................. 48
      4.3 Mapping the Ilities on the Engineering change classification ....................................... 50
  5. Value ...................................................................................................................................... 53

PART 2.0
Chapter 3. Methodological framework .................................................................................. 57
  1. A model for industry platforms ............................................................................................ 58
  2. Implications for product design and development ............................................................ 60
Chapter 4. Architecting changeability ..................................................................................... 63
  1. A classification for change-related Ilities ........................................................................... 63
  2. The Changeability Investigation Technique ....................................................................... 66
  3. 8AM800, a numerical tool for change propagation .......................................................... 71
      2.1 Architectures generation ............................................................................................... 72
      2.2 Change Propagation Matrices generator .................................................................... 74
      2.3 Change indices evaluator ............................................................................................. 75
      2.4 Main file ....................................................................................................................... 75
  3. An initial validation .............................................................................................................. 75
PART 3.0

Chapter 7. Influence of architectural features on change indices ................................................. 107
1. Generating architectures and changes ...................................................................................... 107
2. Comparison of change propagation indices distribution for integral, modular and star architectures .......................................................... 108
3. Choice of architectural indices .................................................................................................. 114
4. Correlation analysis .................................................................................................................. 116
5. Conclusions and design recommendations ................................................................................ 119

Chapter 8. Strategic design of a customizable smartphone ............................................................... 121
1. Project ARA, the modular customizable smartphone ................................................................. 121
2. Case study characterization ......................................................................................................... 123
3. Change propagation evaluation for ARA architectures ............................................................. 130
PART 4.0
Chapter 9. Conclusions and future research directions ................................................ 162
  1. Summary of the activities ......................................................................................... 162
  2. Research results and contributions ......................................................................... 164
  3. Limitations and future developments ........................................................................ 165
  4. Cyber-physical systems innovation ........................................................................... 166

References ...................................................................................................................... 168

APPENDICES
Appendix A. A definition of complexity........................................................................ 189
  1. A small literature review .......................................................................................... 189
  2. Systems and holons .................................................................................................. 190
  3. Complexity ............................................................................................................... 191
  4. Complicatedness ...................................................................................................... 191
Appendix B. Correlation plots from change propagation analysis ............................... 193
  1. ICL ............................................................................................................................ 193
  2. ICI ............................................................................................................................ 204
  3. OCR .......................................................................................................................... 215
Appendix C. Conjoint analysis questionnaire .................................................................. 227
Appendix D. Sensitivity analysis for ecosystem simulations .......................................... 237
  1. Inputs of sensitivity analysis .................................................................................... 238
  2. Users side equilibrium point .................................................................................... 239
  3. Developers side equilibrium point .......................................................................... 242
  4. Users side maximum ............................................................................................... 245
  5. Developers side maximum ...................................................................................... 248
Part 1.0

Introduction
Chapter 1

Introduction

1. The bigger picture

The twenty-first century lives in a small, interconnected world. The spreading of communication devices, the ubiquitous presence of the Internet, quick and cheap means of transport and low international trade barriers are factors that interwove the Earth (Bang and Markeset 2012a) and gave rise to a worldwide community of human beings, which is currently at the peak of the human empathic bonds (Rifkin 2009). As a result, people and materials move easily and information is shared fast and globally.

This framework affects the engineering design of products. As international trade increases due to trade agreements (Subramanian et al. 2006), products have to be designed to serve a large community of potential customers with diverse needs and mind-sets. For example, many Western companies failed to understand the needs of developing countries’ customers, and therefore developed unsuccessful strategies and products (Dawar and Chattopadhyay 2002; London and Hart 2004; Meyer and Tran 2006; Ray and Ray 2011). Furthermore, the availability of product variants increases the demand of product variants: it has been observed that the availability of choices leads to an increasing importance of niche markets (Anderson 2006), which in turn increases the availability of choices, leading to a self-enforcing phenomenon (Lindemann et al. 2009). The increase in pressure due to global competition and the need of product differentiation (Bang and Markeset 2012b) also increase the push for innovative products (Huff et al. 2013; Schilling 2013). While in some industries innovation is a competitive asset, in other markets it is necessary for survival.

At the same time, the connectedness is paid in terms of natural resources consumptions (Rifkin 2009), to the point that some authors question the very foundation of the current economic paradigm (Klein 2014). Sustainable development refers to a complex topic hard to characterize. (Ciegis et al. 2009) states that “none of hundreds of sustainable development definitions found in the literature include all the aspects of the concept and provide perfect understanding of it”. In the same article, the authors conclude that the most suitable definition is the one provided by the Brundtland commission: sustainable development is “the development that satisfies the needs of the current time period without jeopardizing the ability of future generations to satisfy their needs”. However, the World Wildlife Fund (2012) estimated that in 2008 the human global footprint exceeded the available Earth resources by more than 50%. According to another report, humans resource consumption increased by the 50% in the last 30 years, and it now accounts for 60 billion tonnes of raw material per year (Sustainable Europe Research Institute 2009), even if the efficiency in the use of resources has increased by the 30%. If these trends remains constant in spite of the world population growth, not only the natural environment, but also the human race will have to face serious threats.

Not only human beings are consuming more than the Earth can sustain, but they are also generating waste that it is hard to recoup. An example is the generation of electronic waste (Widmer et al. 2005; Robinson 2009; Zhang et al. 2012), also known as e-waste. E-waste is generated by electronic devices like personal computers, smartphones, music players or control units. The lifecycle of these products is shortened when the pace of innovation increases, and thus the e-waste generation rate escalades; it is estimated that more than 20 million tonnes per year were generated in the middle of the 2000s, but further diffusion of electronics and increased wealth in Asia and Eastern Europe can increase this figure (Robinson 2009). E-waste is at the same time dangerous and difficult to recoup. E-waste contains toxic materials like lead, mercury and cadmium; furthermore, burning insulated wires release one hundred times more dioxins than burning domestic waste.
Recycling procedures are currently employed in rich countries, but most e-waste is currently landfilled in developing countries, often after semi-illegal shipping that violates the 1989 Basel Convention on the Control of Transboundary Movements of Hazardous Wastes and their Disposal. Developed countries export nearly the 80% of their total e-waste (Hicks et al. 2005). The lack of rigorous procedures and controls over the e-waste disposal process can lead to dangerous consequences for the environment and the human health, like the Guiyu case study in (Robinson 2009) highlights.

The sustainable use of natural resources and the correct management of waste are two key enablers to “realize the future we want for all”, according to the 2012 UN Development Agenda (UN System Task Team 2012). At the time of writing, several political, social and technical alternatives are under evaluation, and the research community is investigating several directions; however, there is still no consensus on how to make the current way of living more sustainable in the next future.

2. Cyber-physical systems as industry platforms

Interconnectedness involves also technical systems. In the last few years, a new category of networked products, generically called “cyber-physical systems” (Gunes et al. 2014), “smart interconnected products” (Porter and Heppelmann 2014; Porter and Heppelmann 2015) or the “Internet of things” (Evans 2011; Greengard 2015) drew the attention of both the public and the research community. The term is still ambiguous and comprises several research themes (Gunes et al. 2014), but the common features of cyber-physical systems is the ability of their heterogeneous subsystems to connect to a network (e.g. the Internet), through which they can also interact with human beings. Thanks to this feature, they are able to provide advanced emergent properties called “smart” features, which are born from the subsystems sensing ability, high computation capacity and data sharing. Cyber-physical systems can be considered an evolution of mechatronic systems; they not only integrate software and hardware, but also network interfaces and protocols. Several research areas are involved in the engineering design of these complex systems, like for example control system theory, electronics, mechanics, computer science and network science; yet, all these branches have to be integrated in a holistic framework to design these technical systems. To this end, cyber-physical model will be conceptually described with an industry platform model.

A platform is “a set of sub-systems and interfaces forming a common structure from which a stream of products can be developed” (Meyer and Lehnerd 1997). All platforms may be subdivided into two parts: the core and the periphery (Gawer 2009a). The core is composed by those subsystems that remain stable across the platform variants, also called platform configurations; the periphery changes from variant to variant and is usually composed of modules. Cyber-physical systems are platforms whose core is the network structure and interface, while modules are sensors, computer-human interfaces, computation units, databases and mechatronic systems (Figure 1.1).

In cyber-physical systems, it is common that a firm is specialized in designing or producing one subsystem, (e.g. a certain class of sensors); therefore, to be more precise, cyber-physical systems are considered an industry platform. Industry platforms are “products, services or technologies that are developed by one or several firms, and that serve as foundations upon which other firms can build complementary products, services or technologies” (Gawer 2009b). The overall cyber-physical system emerges from the interaction of different and partially independent entities that must be coordinated like a community (West and Lakhani 2008).
Classifying cyber-physical systems as particular industry platforms is not just a conceptual clarification; it permits to leverage the knowledge on product platform design in order to highlight opportunities and challenges in the engineering design of these technical systems. A peculiar feature of platform is their ability to be changed easily, which can be called “changeability”. Platforms are the technical foundation of mass customization (Fogliatto et al. 2012; Ferguson et al. 2014), thanks to their ability to cope well with change (Saleh et al. 2009) and their economies of scope (Gawer 2014). While most physical platforms are customized at the point of sale with assemble-to-order strategies (Wortmann et al. 1997), the intrinsic flexibility of network interfaces in cyber-physical systems allows a continuous customization and evolution of the system over time. In this respect, cyber-physical systems are partially designed and partially evolved, like many engineering systems (de Weck et al. 2011) and can be “updated” to increase the overall lifecycle value of the system (Browning and Honour 2008). However, the ability to change involves serious risks, if the system design is not well thought out. Customizability requires customers to make choices over different product variants, which could lead to frustration or even dissatisfaction (Schwartz 2004; Franke et al. 2009). Moreover, Cyber-physical systems can be considered “complex” (Sinha et al. 2013) because of their structure and behaviour, therefore have an inclination to propagate change (Clarkson et al. 2004b; Giffin et al. 2009). Change propagation occur when undesired changes are required in different parts of the technical system after an initial, desired change. If changes propagate in a cyber-physical system during the design, they can increase the development costs and delay the development process, but if they propagate during the utilization, the system can incur in serious disservices or even damages.

A second key point in industry platforms is the contribution of independent developers to the platform as a whole, which can be framed inside modular innovation and open innovation literature (Gawer 2009b; Schilling 2013). Open innovation is “the use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation, respectively” (Chesbrough et al. 2006). In case of industry platforms, not only knowledge, but also virtual commodities, physical products or services are provided, as an inflow towards the platform or an outflow from the firms. If appropriate toolkits are provided, also users can be involved in this innovation process, so that their expertise is exploited (von Hippel and Katz 2002; von Hippel 2005). The ability to integrate different devices easily during the entire lifecycle was previously a prerogative of software system and is a research topic worth undertaking. At the same time, platform innovation poses some challenges. Coordinating several firms can be a complex task, not only because of the additional coordination workload, but also because knowledge is only partially available across the stakeholders. Then, the technical feasibility of the platform is endangered by the continuous creation of new components: the system interfaces must be flexible enough to accommodate new (and potentially innovative) subsystems over a long lifetime. Finally, industry platforms thrive if there is a community that
supports them (Gawer 2009a); in economic terms, every industry platform is associated to a multi-sided market (Rochet and Tirole 2003; Evans and Schmalensee 2010) composed by groups entities with different goals connected by cross-side network externalities. Cyber-physical systems are currently being developed in several engineering branches, for example manufacturing (Hermann et al. 2015), infrastructures (Fang et al. 2012; Ghosh et al. 2012; Celino and Kotoulas 2013) or home automation (Ko et al. 2012), and their true potential has yet to be fully exploited. Cyber-physical systems, studied as industry platforms, fit perfectly in the general framework provided in the previous Section, as they can meet individual customers’ needs and foster innovation while reducing resources consumption. As mentioned before, these technical systems are able to change easily thanks to their modular structure and their network interfaces; therefore, they can be very easily customized at any time by the system stakeholder. Moreover, innovation can be leveraged from a wide community, which can bring specific knowledge and expertise to the system while exploiting the resources of the platform. External firms do not need to design the entire system, but they can focus on their “module”, thus, even SMEs or users are empowered when they enter the platform community. At the same time, ecosystem innovation (Gawer and Cusumano 2014) capitalizes on customization, since a wide community providing many subsystems can generate a variety of platform configurations that is unmatched by traditional, closed platforms. Finally, cyber-physical industry platforms can limit the consumption of resources by increasing the efficiency of material use and decoupling well-being from material consumption (UNEP 2011). If industry platforms can be personalized intensively with a well-designed choice support, customers can have just the product features they want without undesired side elements; furthermore, customization per se can generate satisfaction and pleasure (Franke and Schreier 2010). Furthermore, cyber-physical systems can be updated subsystem after subsystem; the malfunctioning or outdated parts are substituted, while the rest of the system is preserved. This feature has the potential to reduce material consumption significantly compared to integral technical systems, which are disposed as a whole. If more and more electronic devices becomes cyber-physical systems, the amount of e-waste can be reduced significantly (Hakkens 2015). Finally, “green” innovations, like alternative energy production, can benefit greatly from innovation communities like the one that are born with industry platforms (Dougherty and Dunne 2011).

3. Research questions and thesis outline

The previous section introduced cyber-physical systems, modelled them as industry platforms and showed opportunities and risks for these systems. This thesis means to provide a set of descriptive studies about cyber-physical industry platform design in order to increase designers’ knowledge on the technical system and its community so that those risks are mitigated. The thesis followed a set of well-known guidelines called Design Research Methodology (Blessing and Chakrabarti 2009). The DRM is composed of four phases (Table 1.1): Research clarification, Descriptive study I, Prescriptive Study and Descriptive Study II. The Research Clarification phase focuses on formulating a realistic and worthwhile research goal, Descriptive Study I elaborates the initial description of the existing situation, while Prescriptive Study develops a support to correct and improve the initial situation. Descriptive Study II evaluates the impact of the support and its ability to realise the desired situation. After the Research Clarification and some inevitable iterations in the research activity, it was clear that some aspects of industry platforms design were still obscure. In particular, three desired outcomes are worth investigating: providing changeability through engineering design, defining key platform requirements and supporting successful modules and their developers (Figure 1.2).

As mentioned previously, changeability is threatened by the propagation of changes through the technical systems interfaces. While several research works tried to understand and anticipate change propagation phenomena (see Chapter 2, Section 4), it is still not clear what is the influence of the system architecture on change propagation and how to evaluate the changeability of novel technical systems. Cyber-physical systems can be represented as industry platforms composed by a core and a periphery. In the conceptual phase of the system design, it is not possible to predict all the components that will be part of the periphery, as the community will evolve over time following the market’s dynamics, but the requirements for the core, the interfaces and the system as a whole have to be specified. For example, the minimum set of subsystem, the core structure and the interfaces’ degree of openness have to be established (Gawer 2009b). On the other hand, a good design of the platform as a whole is important, but not sufficient for the commercial success of the platform, as the features of the cyber-physical system are inseparable from the community that provides the peripheral components. The platform supporter therefore is interested in seeding the initial
community correctly, in order to initiate the multi-sided dynamics, which is particularly sensitive to the initial launch (Evans 2009). In fact, multi-sided markets can survive and thrive only if they reach a minimum number of community members, called “critical mass.” For these reasons, understanding what are the most relevant modules in the platform configurations and supporting the developers of those modules is paramount.

In order to increase the designers’ knowledge about the best practices in cyber-physical systems design, I decided to implement a Type 2 research project, focusing on numerical Descriptive study. The design activity has been modelled with three numerical simulations (Figure 1.2) that can anticipate the effects of strategic design decisions on the cyber-physical system and its community; hence, designers can anticipate unwanted consequences and increase the likelihood of desired outcomes. Strategic design concentrates on conceptual, systemic design decisions that affect, and are affected, by the technical system lifecycle; since the platform and its ecosystem are in constant evolution, strategic design is the most suitable design approach for cyber-physical systems conceptual development and the most adequate framework for their design activities.

Table 1.1: Types of design research projects; adapted from (Blessing and Chakrabarti 2009)

<table>
<thead>
<tr>
<th>Research Type</th>
<th>Research Clarification</th>
<th>Descriptive Study I</th>
<th>Prescriptive Study</th>
<th>Descriptive Study II</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td>Initial</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Review-based</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td>Initial</td>
</tr>
<tr>
<td>4</td>
<td>Review-based</td>
<td>Review-based</td>
<td>Review-based Initial/Comprehensive</td>
<td>Comprehensive</td>
</tr>
<tr>
<td>5</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td>Comprehensive</td>
<td>Initial</td>
</tr>
<tr>
<td>6</td>
<td>Review-based</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td>Comprehensive</td>
</tr>
<tr>
<td>7</td>
<td>Review-based</td>
<td>Comprehensive</td>
<td>Comprehensive</td>
<td>Comprehensive</td>
</tr>
</tbody>
</table>

The first design analysis means to anticipate change propagation in architectures by employing statistical model of changes and Monte-Carlo methods. First, various facets of changeability are classified according to the engineering change properties; then, it is illustrated how changeability can be increased by limiting change propagation. The prediction of change propagation takes advantage of the Changeability Investigation Technique, an evolution of the Changeability Assessment Technique described in (Koh et al. 2013). The CIT is employed to research the correlation between architectural features and change propagation, but it can also highlight the platform architectures less sensitive to change propagation or the subsystems that should be monitored more carefully by Change Management. This part of the thesis answers two research questions: “How can Changeability be precisely defined and measured?” and “How does system architecture influence change propagation?”

The second pillar of the thesis is the Value analysis for product platforms, a method that adapts the Value analysis from Value Engineering (Dell'Isola 1997) in order to customizable platforms. The methodology consists of five phases, from Goal definitions to Results presentation and implementation, and takes into consideration both the customers preferences and the platform technical features. The analysis is based on Logit value, a definition of value that is consistent with stated choice models (Louviere et al. 2000) and utility theories (Bowman and Ambrosini 1998; Rao 2014), so that value is related to the likelihood that a customer chooses a certain platform configuration among a finite set of platform configurations. This allows a robust evaluation of value, which is based on conjoint analyses questionnaires data and not on subjective criteria. The value analysis can help designers addressing the question “How can platform architectures be ranked according to their potential success?”
The third and last design analysis studies the ecosystem dynamics and the effect of socio-technical variables on the multi-sided market critical mass and equilibrium. The research question “What socio-technical factors can foster an industry platform and help reaching the critical mass?” is explored thanks to an agent-based model representing a two-sided community. Users are agents that want to buy the platform core and the modules in order to maximize the use value; their rational choices are modelled according to stated choice models. Developers are agents that try to maximize their profits by designing and selling complementary products; they compute the net present value of investments and choose to enter the community, develop new modules, retire old modules or leave the community accordingly. The two communities are connected by the exchange of money and goods thanks to the cyber-physical system. The more Users join the community, the higher are the potential revenues for Developers; the more Developers, the higher the number of platform configurations and the likelihood that a User finds the perfect match for her needs. These positive feedback loops, called cross-side network externalities, are the core of the community dynamics and can be influenced by the platform supporter thanks to technical and managerial decisions. For example, the number and type of the modules at the product launch, the system-level performances, or the market resistance to innovation are variables that can determine the success or the failure of a cyber-physical platform.

The three analysis have been applied to a cyber-physical system designed by Google ATAP team called Project ARA, which can be described as a modular, customization-intense smartphone. The close interaction with the ARA project management provided not only a unique perspective on the design challenges, but it also feedbacks to quantify the effectiveness and the usefulness of the analyses results. For this reason, even though the research can be classified as Type 2, it also provided some insights about Prescriptive studies and Descriptive study II.

Figure 1.3 illustrates the structure of this thesis. Chapter 2 reviews the literature about cyber-physical systems, industry platforms and other relevant branches of knowledge; Chapter 3 provides a general modelling framework and clarifies the challenges for the design of a cyber-physical system as an industry platform. Chapter 4, Chapter 5 and Chapter 6 describe the methodological outline of the three analyses, respectively; for each analysis, both the mathematical characterization and the process are explained. The Changeability Investigation Technique is employed in Chapter 7 for a general sensitivity analysis about which architectural features influence change propagation, while Chapter 8 describes extensively the use of the analyses to support the design of Project ARA. Finally, Chapter 9 summarizes the research results and provides future research directions.
Figure 1.3: Thesis structure

Some parts of the research activities presented in this thesis have already been presented at conferences (Colombo and Cascini 2014; Colombo et al. 2015). All the research activities have been carried out independently by the author, who exploited the collaboration with the co-authors in order to refine and improve the contents of the research.
Chapter 1

Introduction
Chapter 2

State of the art

This chapter provides a brief introduction to the main topics of the thesis. Since the research covers a wide range of subjects, a compromise between breadth and depth had to be found. This chapter is meant as a general preface to the actual research work; smaller, more focused reviews are distributed along Part 2 in order to provide precise information about specific aspects of the design analyses.

The remainder of the chapter is divided into 5 Sections. Section 1 illustrates the basic terminology and concepts of technical systems design and details the specificities of cyber-physical systems. Section 2 concerns platform design, and Section 3 details various aspects of innovation, from the development of an invention to the diffusion of innovation in the market. Section 4 describes the features of engineering changes and reviews nine major frameworks about change Ilities; finally, Section 5 provides a brief, multi-disciplinary survey about the concept of value.

1. Cyber-physical systems design

1.1 Technical systems, complexity and systems architecture

According to the Oxford English Dictionary, a system is a set of things working together as parts of a mechanism or an interconnecting network; in the case of technical systems, the set of things is an artefact composed by elements which share an intended function, i.e. the artefact teleology (Gero and Kannengiesser 2004). System thinking is the process of understanding how things influence one another within a whole; it is not a unique or well-defined technique or method, but it is an awareness that there is a mutual influence between the parts and the whole. Whether a system is an ontological entity or a gnosiological concept in the mind of the designer is not essential; however, it is important to recognize that every object can be decomposed into smaller elements that are combined together. The use of systems thinking in engineering design is a fruitful exercise when the relationship between different designed artefacts and the relationship between the artefacts and the environment are not negligible. In this sense, “Systems thinking incorporated into the theory of technical systems presents the opportunity to treat the problem as a whole. This is necessary pre-condition for consistently successful design and other engineering efforts” (Hubka and Eder 1988).

When artefacts are considered as systems, they can be decomposed into elements and interrelationships between elements, but it often observed than some properties of the system as a whole cannot easily traced back to the elements properties: as Aristotle affirms in the second book of Metaphysica, “the whole is more than the sum of the parts”. These properties are called “Emergence” or “emergent behaviours”, and can be observed only in the system as a whole. Typical examples of emergence are evolution, intelligence, organized social behaviours or the fluctuation of prices in markets. The origin of emergence is still debated (Kim 2006; Bedau 2008; Pernu and Annila 2012) and even subject to criticism. An interesting interpretation (Corning 2002) considers emergent properties as the results of positive or negative synergies inside system, synergies being “the combined (cooperative) effects that are produced by two or more particles, elements, parts or organisms - effects that are not otherwise attainable”. (Crawley et al. 2015) gives three practical examples of emergence in technical systems: function, performances and lifecycle properties. Functions can have different representations and definitions (Vermaas 2009); in systems design they usually indicate what the systems does (NASA 2007; Eisner 2011) as opposed to its structure (i.e. the list of its elements and interfaces). Performances indicate how well a system performs its functions, while system lifecycle properties (also called “Ilities”) are
defined as “desired properties of systems, such as flexibility or maintainability […]”, that often manifest themselves after a system has been put to its initial use. These properties are not the primary functional requirements of a system’s performance, but typically concern wider system impacts with respect to time and stakeholders than are embodied in those primary functional requirements. The itilies do not include factors that are always present, including size and weight” (de Weck et al. 2011). All these three aspects will be taken into consideration for the strategic design analyses.

One of the most relevant features of systems is complexity, which can approximatively defined as a measure of the “size” of the system. Literature about complexity is wide and diverse: just (Lloyd, 2001) lists more than 40 measures of complexity, but many others have been proposed in literature. However, five key aspects can be traced in several definition: the size of the system (in terms of parts or interfaces), the observer’s effort, the relation with modelling language, the information content, and a sub-classification of different aspects related to complexity. In this thesis, complexity will be defined as a measure of the information content in the models of a system; a more precise definition and the explanation for this choice are given in Appendix A. The evaluation of complexity is not unique, since it depends on the model of the system; in the field of technical systems design one of the preferred representations is the system architecture.

A system architecture can be described as “an abstract description of the entities of a system and the relationships between those entities” (Crawley et al. 2004). It generally links the function to the structure (Ulrich 1995; Crawley et al. 2015) and can be represented with different levels of granularity, which depend on the number of hierarchical level the system has been decomposed into (Chiriac et al. 2011). Architecture can be represented formally or informally; popular formal modelling languages are SysML (Friedenthal et al. 2008), Object-Process Methodology (International Standards Organization 2015) and Design Structure Matrices (DSM) (Eppinger and Browning 2012). SysML is a graphical modelling language developed by joint initiative between the Object Management Group and INCOSE. It utilizes nine diagrams to represent different aspects of the system: its requirements, its behaviour (i.e. how the system change its state through time) and its structure. The Object-process methodology is a modelling language that combines functions, structures and different types of interactions on the same diagram. Design structure matrices are adjacency matrices that represent the interfaces between systems components. When they represent elements from multiple domains, for example requirements or organization, they are also called Multi-Domain Matrices (MDM) (Lindemann et al. 2009). DSM have three advantages: they are based on a mathematical item, they are a synthetic representation of the architecture and they are consistent with graph theory. However, DSMs may require a significant amount of time to be generated (Suk Suh et al. 2011) and they are not as intuitive as graphical representations.
System architectures can fall into two main categories, depending on the interfaces topology and the mapping between functions and structure (Ulrich 1995; Selva 2012; Kahn et al. 2013). Modular architectures have a one-to-one mapping between functions and physical elements; they are generally composed by clusters of physical components called modules. Modules can have many different definitions (Gershenson et al. 2003), but from a structural perspective, they are sub-systems that have a high density of interfaces between elements of the clusters and few interfaces outside the cluster (Hölttä-Otto et al. 2012). A star architecture is a variant of modular architectures where one of the modules (called bus) connects several other components facilitating the sharing of resources across the system. Integral architectures share functions across physical elements or, alternatively, have physical components that provide more than one function. Their interface layout does not show any particular pattern, and resources are shared uniformly across the system. Figure 2.1 provides a network representation of the three types of architecture discussed. Modularity brings several benefits (Gershenson et al. 2003), in particular with reference to engineering changes (see Section 4.2), but usually reduces the system performances or increases resource consumption (Ulrich 1995). Few real system architectures are neither completely modular nor completely integral, as they stand in between these two extremes. Modularity metrics are abundant in literature (Gershenson et al. 2004), but they are scarcely correlated one to the other (Hölttä-Otto et al. 2012) and can depend on the granularity of the system model (Chiriac et al. 2011); thus, they have to be chosen carefully.

1.2 Cyber-physical systems
“Cyber-physical systems” (abbreviated CPS) is a term introduced by Helen Gill in 2006 at the National Science Foundation (Lee and Seshia 2011) in order to indicate “a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities” (Baheti and Gill 2011). Since then, the terms achieved great popularity and has been used (and maybe even abused) to designate systems that connect sensors, embedded computers, sensors actuators in order to achieve “smart features”. (Lee 2008) defines cyber-physical systems as “integrations of computation with physical processes”, while the CPS Steering Group considers CPS as “systems [that] use computations and communication deeply embedded in and interacting with physical processes to add new capabilities to physical systems” (CPS Steering Community 2008). Acatech (2011), the German national academy for science and engineering, gives the following definition: “Cyber-physical systems are systems with embedded software […] which: (1) directly record physical data using sensors and affect physical processes using actuators; (2) evaluate and save recorded data, and actively or reactively interact both with the physical and digital world; (3) are connected with one another and in global networks via digital communication facilities (wireless and/or wired, local and/or global); (4) use globally available data and services; (5) have a series of dedicated, multimodal human-machine interfaces”. In 2013, a report for the National Institute of Science and Technology described CPS as “as smart
systems that encompass computational (i.e., hardware and software) and physical components, seamlessly integrated and closely interacting to sense the changing state of the real world” (Energetics Inc. 2013). After reviewing a large set of works on cyber-physical systems, (Gunes et al. 2014) concludes that CPS are “complex, multi-disciplinary, physically-aware next generation engineered systems that integrate embedded computing technology (cyber part) into the physical phenomena by using transformative research approaches”. (Porter and Heppelmann 2014) dissert about “smart, connected products” that “combine hardware, sensors, data storage, microprocessors, software and connectivity in myriad ways”.

![Cyber-physical system model](image)

Figure 2.2: Cyber-physical system model; adapted from (Gunes et al. 2014)

These definitions highlight several aspects of cyber-physical systems, but they all recognize the importance of the integration between computation (software and embedded computing), sensing and actuation (hardware) and communication (network). (Gunes et al. 2014) represents these elements inside a model (Figure 2.2) that bridges the cyber world with the physical world. The physical world is sensed through one or more sensors connected into a network, which transmit the data to the communication network. The cyber world, which is constituted by algorithms, databases and computational elements, receives the data and elaborates the decisions; these in turn are transformed into actuation thanks to the actuators network.

While cyber-physical systems always connect physical and cyber elements, the hierarchical level of the cyber-physical interfaces defines at least four types of cyber-physical systems (Stucheli and Meboldt 2013), as Figure 2.3 shows. If the communication network bounds together technical systems that can operate independently, it determines an External network CPS; typical examples are network of independent devices or plants, like smart power grids. If the technical system is constituted by well-defined sub-systems that are connected together by a communication bus or network, like cars or airplanes, it can be considered an Internal network CPS. If connectivity is embedded into the lowest system hierarchical level, i.e. the single components, and the data are shared inside the technical system, the technical system is an Internal element network CPS. Finally, if the most basic components are able to communicate outside the technical system, the cyber-physical system has a Pervasive network; an example of this are car sensors that signal malfunctioning to the car owner through text messages.
Cyber-physical systems are not an entirely new class of technical systems; they are born from the combination of several previous technical areas that now can form a synergy thanks to the computational power and miniaturization of embedded systems and the pervasiveness of communication networks. (Gunes et al. 2014) identifies eight domains that are interlaced with cyber-physical systems; however, two more can be considered (Figure 2.4).

Figure 2.3: classification of CPS according to (Stucheli and Meboldt 2013)

Figure 2.4: knowledge domains related to cyber-physical systems; partially adapted from (Gunes et al. 2014)

Big data refers to datasets that store such a high quantity of data that human analysis, traditional statistical techniques or data visualization are not able to evaluate them. Cloud computing allows users to access to a shared pool of configurable computing machines through an on-demand service. Systems-of-systems are
Chapter 2

State of the art

combination of technical systems that have a common function, but are designed, managed and operated by independent stakeholders. Mechatronic studies how to control and operate physical devices through electronic components and software. Cybernetics is the cross-disciplinary study of feedback loops in technical, social and neurological systems. The Internet of things is a commercial term that identifies technical systems that can connect to the Internet; in the beginning, it focused on RFID tags, but now it comprises any kind of device and it is almost synonymous with cyber-physical systems. The Web of things is an extension of the Internet of things and means to connect physical devices to the world-wide-web using standard web technologies. Finally, machine to machine communication is a subset of the Internet of things or the Web of things where data are shared across machines without interactions with human beings. Another relevant aspect of cyber-physical systems is their relationship with human beings. Therefore, the framework from (Gunes et al. 2014) can be supplemented by human-machine interfaces and socio-technical systems, both of which study how human beings as individuals and as community affect and are affected by technical systems.

All the reports and articles cited in this sub-Section agree on the impact that cyber-physical systems can have on several sectors. A 2008 National Intelligence Council report (The National Intelligence Council 2008) includes the Internet of things among the six disruptive technologies that “offer the potential to enhance or degrade US power over the next fifteen years”. Porter and Happlemann (2014; 2015) state that smart connected devices are able to radically transform competition and companies organization, while the company Cisco in a white paper forecasts that 50 billion devices will be connected to the Internet by 2020 (Evans 2011). More specifically, five main areas will be affected by cyber-physical systems development, according to (CPS Steering Community 2008; Baheti and Gill 2011; Sztipanovits et al. 2012; Energetics Inc. 2013; Gunes et al. 2014; Wolf and Feron 2015), whose contribution has been summarized in Table 2.1. This list is by no means comprehensive, but it highlights the importance of cyber-physical system development and the variety of their potential applications.

The cited sources recognize however that some design and management challenges have to be tackled before the benefits of cyber-systems can be exploited. (Gunes et al. 2014) identifies 21 system lifecycle properties that have to be embedded into cyber-physical systems. (Lee 2008) highlights how traditional models of electronics are unsuitable to represent the concurrent nature of physical phenomena. (Energetics Inc. 2013) details five common areas and five specific areas whose challenges need to be addressed. Safety and reliability need to be improved with better models and cost-effective verification and validation procedures. Machine-to-machine networks have to endure uncertainty conditions well, and the development community has to coordinate efficiently during the product development; moreover, machine-to-human interactions should be seamless. As noted before, the abstraction infrastructure must be able to bridge digital and physical components. As far as architecture is concerned, several system lifecycle properties should be addressed, design methodology have to be developed and trustworthy and holistic evaluation procedures have to be defined. Finally, education and training play a fundamental role in the diffusion and correct functioning of cyber-physical systems.

Please note that this thesis addresses many of the challenges found in literature:

- The use of a platform model for cyber-physical systems allows to leverage previous literature on product and industry platform design in order to tackle challenges related to system architecting;
- The limitation of change propagation increases the reliability of the system during the lifecycle and it allows cyber-physical systems to face unknown or uncertain external conditions better;
- The evaluation of the systems’ value provides useful information for the holistic evaluation of the technical system and helps the customization process, thus reducing the complicatedness of human-machine interfaces;
- The ecosystem simulation studies how developers behaves not only during the product development, but also during the overall platform lifecycle.
Chapter 2

State of the art

Table 2.1: cyber-physical systems main areas of application

<table>
<thead>
<tr>
<th>AREA OF APPLICATION</th>
<th>POTENTIAL CONTRIBUTIONS</th>
<th>RESEARCH INITIATIVES</th>
<th>EXAMPLES IN LITERATURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturing</td>
<td>Optimize productivity in the manufacture of goods or delivery of services</td>
<td>Industry 4.0 Industrial Internet</td>
<td>(Dworschak and Zaiser 2014; Lee et al. 2014; Hermann et al. 2015; Lee et al. 2015)</td>
</tr>
<tr>
<td></td>
<td>Resource-efficient production</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Highly customized products</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tailored adjustments to the human workforce</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emergency response</td>
<td>Emergencies prevention</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Faster response to calamities</td>
<td>SmartAmerica Challenge Strategic Foresight Initiative</td>
<td>(Gelenbe and Wu 2013)</td>
</tr>
<tr>
<td></td>
<td>Increased reach of operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transportation</td>
<td>Integrated air traffic control</td>
<td>NextGen air traffic control Volvo Intellisafe autopilot Ertico-ITS Europe consortium</td>
<td>(Luettel et al. 2012; Xiong et al. 2015)</td>
</tr>
<tr>
<td></td>
<td>Autonomous air/ground vehicles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Networked mobility</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Water distribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Traffic control</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Building automation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Health care and medicine</td>
<td>Continuous health monitoring</td>
<td>Medical Device Plug-and-Play (MD PnP) Interoperability program HealthSuite Digital Platform</td>
<td>(Istepanian et al. 2004; Haque et al. 2014)</td>
</tr>
<tr>
<td></td>
<td>Health devices customization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Telemedicine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Medical devices interoperability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Platform design

2.1 Definitions and classification

A product platform is a technical system that can provide different functions or assume different structures from the combination of standardized subsystems. Literature defines them as “a set of sub-systems and interfaces forming a common structure from which a stream of products can be developed” (Meyer and Lehnerd 1997) or, more generally, as “the assets (components, processes, knowledge, people, or relationships) shared by a set of products” (Robertson and Ulrich 1998). The set of all platform variants form a product family (Simpson et al. 2014). All platforms may be subdivided into two parts: the core and the periphery (Baldwin and Woodard 2009). The core is composed by those subsystems that remain stable across the platform variants, also called platform configurations; the periphery changes from variant to variant and is usually composed of modules (Figure 2.5). Platforms generate economies of scope (Gawer 2014), i.e. a reduction of cost that happen when joint development or production happens: the core can be used across several variants with minor modifications; usually, the cost of adapting the core is lower than the cost of designing new product parts. For this reason, product platforms have enjoyed great popularity in sectors where mass customization of complex technical systems is appreciated, like for example in automotive, aeronautics or consumer electronics.

Mass customization (Pine 1993; Fogliatto et al. 2012) is “a product development approach that allows for the creation of goods that minimize the trade-off between the ideal product and the available product […], while maintaining system costs comparable to mass produced products” (Ferguson et al. 2014). At a first glance, mass customization should be a catch-all strategy to maximize customer’s value while reducing the costs. Furthermore, the inherent uniqueness of certain platform configurations can provide value on its own (Franke and Schreier 2008). For example, a case study about innovation toolkits showed the price customers are willing to pay for self-design watches doubles the price of standardized watches (Franke and Piller 2004). Nevertheless, some studies have also underlined that this value is actually mediated by several factors, like process effort and enjoyment (Franke and Schreier 2010) or the level of insight into customers’ own preferences (Franke et al. 2009); choice can lead to frustration due to complicatedness (Valenzuela et al. 2009), post-decisional regret (Zeelenberg et al. 1998), expectations disillusion (Diehl and Poynor 2010) and conflicting desires (Chatterjee and Heath 1996; Gourville and Soman 2005). These issues led to the development of configurators (Trentin et al. 2013) and recommender systems (Park et al. 2012) that guide customers into the choice process.

In (Gawer 2009a), three types of product platforms are highlighted: internal, supply-chain and industry platforms. Internal platforms are product platforms whose core and periphery are entirely designed inside a single firm; supply-chain platforms are managed by a central firm, but a fixed and well-defined (“closed”) supply-chain provides the core and/or modules that are integrated by the central firm. Industry (or external) platforms are “products, services or technologies that are developed by one or several firms, and that serve as
foundations upon which other firms can build complementary products, services or technologies.” Every industry platform therefore is associated to a community of complementors and generates a multi-sided market (see Section 2.3). Traditionally, the body of literature that treats internal and supply-chain platforms can be associated to the engineering design community, while industry platforms were the object of management and economics research. (Gawer 2014) tries to unite these two stream of literature by providing the following unified conceptualization: “Technological platforms can be usefully seen as evolving organizations or meta-organizations that: (1) federate and coordinate constitutive agents who can innovate and compete; (2) create value by generating and harnessing economies of scope in supply or/and in demand; and (3) entail a technological architecture that is modular and composed of a core and a periphery.” This thesis continues the research about a unified approach for internal and external platforms by using engineering design methodologies, typically used in internal platforms, in order to analyse industry platforms; however, the two streams of literature will be presented separately in order to preserve the distinction between the previous literature streams.

2.2 Internal and supply-chain platforms

![Diagram](Figure 2.6: Issues in internal product platform design (adapted from (Jiao et al. 2007)))

As mentioned earlier, engineering design research focuses almost exclusively on internal or supply-chain platforms. (Jiao et al. 2007) reviews platform design literature and clusters it into three groups (Figure 2.6) thanks to Axiomatic design domains (Suh 2001). At the front-end, designers need to define the product portfolio that satisfy customers’ needs with the correct functionalities; then product family design maps the functional requirements into the design parameters and defines the platform architecture. Back-end issues include process design and supply chain design. While strategic design is relevant in all these issues, this thesis focuses on product definition and product design; therefore, this Section addresses only the first two topics. (Cameron and Crawley 2014) define benefits and costs in product platform strategies and warn about divergence phenomena that occur when platforms achieve less commonality than it was first intended. According to the authors, most companies do not realize the need for high upfront development and inventory investments at the beginning of the platform development; these initial costs are balanced only later through economies of scope and economies of scale; therefore, only a long-term perspective can justify platform design. (Pirmoradi et al. 2014) states that front-end issues involve customer involvement, product portfolio design, product family positioning, and transition or mapping from customer needs to functional requirements; relevant research areas in this phase are (1) Product portfolio and product family positioning, (2) Market-driven product family design, (3) Product family modelling and (4) Platform and product family configuration issues. The first topic segments the market into different clusters, generates variants for each cluster and identifies possible product adjustments to meet customers’ specific needs; market-driven product family design exploit customers’ preferences to inform design decisions. Product family modelling means to represent completely and coherently the set of product variants enabled by the platform, while configuration issues are related to what should be shared among the platform variants and what can be customized. Market-driven platform design is a top-down approach to platform design (Simpson et al. 2001). As market-driven platform design is interested in measuring customers’ needs and translating them into technical
requirements, stated choice questionnaires and conjoint analyses (Chandukala et al. 2007; Rao 2014) can be employed. (Ferguson et al. 2011) highlights the potential of conjoint analysis in mass customization and underlines strengths and weaknesses of stated choice models for engineering design. In particular, the authors conclude that “Determining a sufficient level of granularity for assessing consumer preferences is a critical issue.” (Kazemzadeh et al. 2009) takes advantage of conjoint analysis in order to improve the requirements specification for product families. Stated preferences highlight the most appreciated features of a product, which are translated into requirements thanks to the House of Quality matrix. Conjoint analysis has also been employed to determine the best product family portfolio. In (Kumar et al. 2009), for example, an advanced market segmentation grid is derived from a nested logit model; the grid is then combined with product’s features and estimated cost, which are given as an input to a commonality optimization algorithm. A general inquiry on commonality optimization from conjoint analyses can be found in (Turner et al. 2011), where bottom-up and top-down methods are compared. The article mentions that top-down methods lead to well-informed decisions, at the price of a more complex analysis.

Between front-end and back-end issues, designers have to address challenges related to the definition of platforms functions and structures (Pirmoradi et al. 2014). In this field, four research areas can be highlighted: (1) trade-off between commonality and variety, (2) family and platform configuration and optimization, (3) metrics for design and assessment of platforms and (4) design support systems. An exemplary article of this area is (Suh et al. 2007), which describes a method to generate platform variants that can face market uncertainties. Once critical platform elements are identified, platform alternatives are generated and evaluated in terms of economic performances thanks to a scenario analysis. Alternatively, platform design may be influenced by the commonality vs. variety trade-off. Commonality measures the differentiation between different platform combinations; on one hand, the use of similar modules can increase the economies of scope and standardization, on the other, platform differentiation covers more market niches and reduces variants overlap. In order to find the optimal products variety, literature has proposed commonality metrics and computation algorithms; for example (Thevenot and Simpson 2007) proposes CMC as the measure of commonalities and redesigns a family of staplers thanks to a genetic algorithm optimizer, while (Hölttä-Otto et al. 2008) represents product families as dendrograms and increases commonality thanks to a partition algorithm. A detailed comparison of commonality metrics can be found in (Thevenot and Simpson 2007). Fixson (2007) summarizes 160 references about modularity and commonality and classifies them according to subjects (product, process, organization and innovation), effects (performance/quality, variety, costs and time) and research methods (theory-building, frameworks, process modelling mathematical modelling, simulation, experiments, case studies and reviews).

2.3 Industry platforms

Industry platforms involve four actors in a network of mutual benefits (Eisenmann et al. 2009), as shown in Figure 2.7. Users exchange products or services through the platform, which is maintained by two entities: the platform provider and the platform sponsor. The platform provider is the point of contact between users, while the platform sponsor design the platform and holds the Intellectual property rights over it. Depending on the uniqueness or multiplicity of firms in the two roles, different industry platform models can be employed: proprietary (one provider and one sponsor; e.g. game consoles), licensing (many providers and one sponsor; e.g. Android OS), join venture (one provider and many sponsors, e.g. Japan Nuclear Fuel Co.) and shared (many providers and many sponsors; e.g. DVD).

Industry platform literature can be subdivided into two main streams: the one considering managerial and coordination aspects, and the one dealing with multi-sided markets. (Gawer 2009a) considers industry platforms as the final step into the evolution process of product platforms. The author proposes business actions to create a new platform when none existed before (“coring”) and to beat the competing platforms by fostering the platform community (“tipping”). From a technology management perspective, industry platforms involve several actors contributing in different ways and with different competences to the whole project. (Le Masson et al. 2009) proposes the study of collaboration platforms that are naturally formed by the institutions involved into the industry platform development. According to the case study results, collaborative platforms for platform design will emerge if (1) actors lack some capabilities and cannot internalize them, (2) none of the actors has a clear view over all the potential sources of value and (3) the design process itself creates the capabilities and improves the knowledge about the potential sources of value. Moreover, collaborative platforms not only aim at finalizing the industry platform design, but they also provide capabilities and share perspectives over the market landscape.
Another relevant topic is the degree of “openness” in industry platforms, i.e. the degree of the restrictions on participation in platform development, commercialization or use (Eisenmann et al. 2009). (Schilling 2009) identifies a continuum from wholly proprietary to wholly open (Figure 2.8). Wholly open systems are not protected by patents or secrecy, and are the opposite of wholly proprietary (i.e. protected) architectures. In between the two extremes, several degrees of openness can be chosen: limited licensing allows platform sponsors to stop any module from being commercialized, while liberal licensing allows wide modifications of the platforms modules, while modifications to the core must be approved by the sponsor. Finally, moderate licensing protects the copyright of the platform core, but allows external entities to provide the modules freely. According to the same article, the platform sponsor strategy should be informed by the firm resources and the industry environment; however, a wholly open strategy should be always avoided for technologies that require standardization and compatibility, because openness tends to fragment the platform. The previously cited (Eisenmann et al. 2009) identifies horizontal and vertical competitive strategies, focusing especially on mature platforms.

As noted in (Gawer 2014), most of the literature on industry platforms is a derivation of economics literature on network effects and two- or multi-sided markets. Two-sided (multi-sided) markets are “markets involving two (multiple) groups of agents interacting via ‘platforms’ where one group’s benefit from joining a platform depends on the size of the other group(s) that joins the platform” (Armstrong 2006). The reinforcing relationship between the multiple sides of the market is called indirect network effect or cross-side network externalities, not to be confused with same-side networks externalities, which arise within agents in the same group. As (Hagiu and Wright 2015) explains, “A cross-group network effect arises if the benefit to users in at least one group (side A) depends on the number of users in the other group (side B) that join. An indirect network effect arises if there are cross-group network effects in both directions (from A to B and from B to
A). In this case, the benefit to a user on side A depends on the number of participants on side B, which in turn depends on the number of participants on side A.

Given this mutual influence in the two sides, the most critical phase to establish successful two-sided markets is the launch, when the two communities are small and the platform provider has to initiate the two-sided dynamics. This “chicken-and-egg” problem has been addressed by several research works. A relevant stream of literature addressed this challenge through mathematics and pricing policy, with the intent of maximizing platform profits or the community social surplus (Rochet and Tirole 2003; Rochet and Tirole 2004; Parker and van Alstyne 2005; Armstrong 2006; Rochet and Tirole 2006). Other articles tackled the challenge differently. (Boudreau and Hagiu 2009) focused on non-monetary policies modelling the platform provider/sponsor as a regulator; as a result, the authors found that in all case studies the “regulatory role played by multi-sided platforms was pervasive and at the core of their business model”. (Evans 2009) proposes the metaphor of catalyst and chemical reactions to describe the role of industry platforms. The two (or more) communities are chemical reagents, whose interactions are eased by the presence of the catalyst, i.e. the platform. The author considers some strategies for “getting both sides on board”, comprising both monetary and non-monetary incentives; among those, the article suggests pre-commitment investments by the platform provider and zig-zagging, i.e. subsidizing one side of the market at a time.

![Figure 2.9: One-side equilibrium curves, two-sided equilibrium point and critical mass represented on a state-space; adapted from (Evans and Schmalensee 2010)](image)

In (Evans and Schmalensee 2010), the authors focus on the problem of market equilibrium and critical mass, taking advantage of a “state-space” representation of the market: a Cartesian plane where the axes are the number of adopters in the two communities (Figure 2.9). All possible market trajectories can be represented in this space; moreover, three special elements can be drawn: one-side market equilibriums, two-side equilibrium points and the critical mass. One-side equilibrium is the set of combinations of Side A adopters and Side B adopters that maintains the number of adopters in one side constant. In a market state-space, one-side equilibriums take the shape of curves; there are as many equilibrium curves as there are sides in the multi-sided market. When the equilibrium curves intersect, they generate a two-side (multi-side) equilibrium point, i.e. a condition where all the sides are fixed in time and, even if the single adopters may change, the total number of adopters remains constant. In a market state-space, one-side equilibriums remain constant. In Figure 2.9, there are three equilibrium points: one in the origin, one in the middle of the plane and the other in the top-right corner of the plane. The two-sided market is attracted by these equilibrium points, which can be of two types: stable and unstable. Once a two-sided market reaches a stable equilibrium point, it cannot change its size anymore. The origin of the Cartesian plane is always a stable equilibrium point, but it is highly undesirable, since it determines the failure of the industry platform: as mentioned earlier, literature has focused on monetary and non-monetary strategies to avoid this market collapse. (Evans and Schmalensee 2010) defines the boundary of the null equilibrium attraction space as the
“critical mass”, i.e. the minimum number of adopters that allows the market to reach a non-null equilibrium two-side equilibrium; then, it advances a mathematical characterization of the equilibrium curves and two-side equilibrium.

In (Kumar et al. 2010), the authors propose a discrete-time mathematical model that can describe the evolution of two-sided markets and the market equilibria (Equation 2.1).

\[
\begin{align*}
    n_A(t+1) &= (1-\varepsilon_1)n_A(t) + \varepsilon_2 g_A(n_B(t)) \\
    n_B(t+1) &= (1-\varepsilon_3)n_B(t) + \varepsilon_4 g_B(n_A(t))
\end{align*}
\]  

(2.1)

\(n_A\) and \(n_B\) are the number of adopters in Side A and Side B, respectively; \(t\) is the discrete variable representing time. \(\varepsilon_1\) and \(\varepsilon_3\) are called death-rates and represent the number of adopters that leave the market from time \(t\) to time \(t+1\). \(\varepsilon_2\) and \(\varepsilon_4\) are named birth-rates, and represent the effect of same-side network externalities, like word-of-mouth or prestige among peers. Finally, \(g_A\) and \(g_B\) are the affinity curves, the functions that model cross-side network externalities. According to this model, the equilibrium curves can be determined as:

\[
\begin{align*}
    n_A^*(t+1) &= n_A^*(t) \\
    n_B^*(t+1) &= n_B^*(t)
\end{align*}
\]  

(2.2)

And then:

\[
\begin{align*}
    n_A^* &= \frac{\varepsilon_2}{\varepsilon_1} g_A(n_B) \\
    n_A^* &= \frac{\varepsilon_4}{\varepsilon_3} g_B(n_A)
\end{align*}
\]  

(2.3)

Equation 2.3 determines the shape of the equilibrium curves; by finding the intersection between the curves, it is possible to derive the two-side equilibrium. (Sinha et al. 2015) determines the conditions according to which the equilibrium is stable or unstable and proposes a method to infer the affinity curves from Monte-Carlo simulations (Figure 2.10).
3. Innovation

3.1 Definitions and classification

Innovation is a common word in business and academic language, and many definitions can be found. Innovation comprises two main themes: the generation of creative ideas and the conversion of ideas into business or other useful applications (Roberts 2007). For example, (Amabile 2012) considers innovation “the successful implementation of creative ideas”, while the OECD (OECD 1991) defines it as “an iterative process initiated by the perception of a new market and/or new service opportunity for a technology-based invention which leads to development, production, and marketing tasks striving for the commercial success of the invention”. The latter definition also underlines that innovation is iterative in nature and that innovation concerns product development as well as other business domains, like production and marketing. A discovery or an invention becomes an innovation when it reaches the market and diffuses among potential adopters (Smith and Barfield 1996). (Myers and Marquis 1969) summarizes these concepts: “Innovation is not a single action, but a total process of interrelated sub-processes. It is not just the conception of a new idea, nor the invention of a new device, nor the development of a new market. The process is all these things acting in an integrated fashion.” From a practical perspective, this continuous process can be subdivided into at least two phases: during the product development, when ideas are transformed into products, and during the product launch, when products diffuse into the market. The former topic is covered in Sections 3.1, the latter in Section 3.2.

New product development (NPD) processes are the sequence of technical and managerial operations that transform innovative ideas into products ready to be launched on the market. Two main classes of NPD processes can be considered. The most popular innovation process model is the waterfall-based funnel model, illustrated in Figure 2.11. The initial phase is a wide screening of the raw ideas in order to find the most successful one; then, the approved ideas are turned into projects and developed; of these, just few are actually launched on the market. On the contrary, spiral-based NPD processes highlight the iterative nature of innovation and the interconnections between different organizations. For example, (van der Duin et al. 2007) proposes the Cyclic Innovation model (CIM), a model that considers four mutually influencing dynamic processes (Figure 2.12). The CIM identifies exchanges of information as positive and negative feedback processes between four processes (scientific exploration, technological research, product development and...
market transition), mediated by four agents (hard knowledge infrastructure, manufacturing and process industry, services and soft knowledge structures).

Figure 2.11: Funnel model of innovation; adapted from (Schilling 2013). Blue dots are the ideas, white dots are discarded ideas that are not brought to the following phase

Figure 2.12: Cyclic innovation model; adapted from (van der Duin et al. 2007)

Innovation can be fostered or dumped by factors internal as well as external to the firm. (Calantone et al. 2010) analyses 134 articles on innovation in order to test six hypotheses about moderators of innovation. Thanks to bivariate analyses, multivariate structural equation models and moderator analyses, the author showed that
technological turbulence (but not market turbulence) promotes innovation directly, and that both customer and competitor strategic orientation are positively related to innovation. Moreover, mechanistic organizational structures (Miller 1987) exert a positive impact on innovation. The general notion that newly developed products show higher performances than older products is confirmed by the statistical analyses.

Literature about innovation is large and rather difficult to synthesize, but several classification methods have been proposed in order to separate different types of innovation. (Schilling 2013) provides four axes of classification:

- **Radical vs. incremental innovation**: the radicalness of innovation (also called newness or innovativeness) measures how “different” is an innovative solution with respect to what exist either in the firm or in the market. (Garcia and Calantone 2002) highlights that even though several definitions of innovativeness exist, all of them are related to the degree of discontinuity in marketing and/or technological factors. Innovativeness can be measured at firm level (micro-perspective) or at global level (macro-perspective).

  According to Schilling, radicalness is a combination of newness and differentness; however, the definition of radical innovation is quite debated. For example, (Cross 2008) considers an innovation radical when it leads to the development of both the market and the technology; (Garcia and Calantone 2002) identify radical innovation by the initiation of a new technology and new marketing S-curves. The same article proposes a more nuanced scale that goes from radical innovation to imitative innovation, which is described in Table 2.2. “Really new” innovations are radical with respect of either the market dimension or the technology dimension, but not both; (Cross 2008) differentiate between market innovation and technology substitution, depending on what part of the innovation is radical. Incremental innovations provide novel features or improvements in existing markets and existing technologies, while imitative innovations are innovations that try to mimic other innovations, but are new to the firm.

  Table 2.2: Innovation classification based on innovativeness according to (Garcia and Calantone 2002) and (Cross 2008)

<table>
<thead>
<tr>
<th>INNOVATION</th>
<th>INNOVATIVENESS IN MARKETS</th>
<th>INNOVATIVENESS IN TECHNOLOGY</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radical</td>
<td>High</td>
<td>High</td>
<td>Steam engine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>World Wide Web</td>
<td></td>
</tr>
<tr>
<td>Really new (Technology substitution)</td>
<td>Low</td>
<td>High</td>
<td>Canon laserjet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sony walkman</td>
<td></td>
</tr>
<tr>
<td>Really new (market innovation)</td>
<td>High</td>
<td>Low</td>
<td>Early commercial jet-lines</td>
</tr>
<tr>
<td>Incremental</td>
<td>Medium</td>
<td>Medium</td>
<td>Digital automotive control systems</td>
</tr>
<tr>
<td>Imitative</td>
<td>Low</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

- **Product innovation vs. process innovation**: this classification determines what is innovated. Product innovation means to improve the outputs of an organization – goods, services or both. Process innovation increase the efficiency and the effectiveness of the methods and techniques that the organization uses to generate the output, i.e. how the organization conducts its business. As Schilling notes, product innovation and process innovation can complement each other, in that a new product may require an innovative process to be created, or an innovative process can allow the design of novel products. Utterback and Abernathy (1975) consider the time evolution of these two types of innovation for a single development cycle (Figure 2.13). In an initial phase, the focus is on product innovation, while the process is immature; then, as the product features stabilizes and a dominant design emerges from the market, the focus shifts on process improvement and costs reduction. As observed in (Adner and Levinthal 2001), this dynamics does
not depend only on the firm capabilities and strategies, but it is affected by the market. More specifically, in the first phases of the development cycle, innovation intends to meet market requirements, while later innovation has to face competition among suppliers and a market constituted by "technologically satisfied" consumers.

Figure 2.13: Rate of innovation over time for product and process innovations; adapted from (Utterback and Abernathy 1975)

- **Competence-enhancing vs. competence-destroying innovation**: this classification relates to the knowledge accumulated by the organization and the effects of the innovation. If an innovation leverage previous knowledge and skills, and enriches them, it is a competence-enhancing innovation; on the other hand, if the newness of the innovation replace knowledge and skills in the organization, the innovation is classified as competence-destroying. These two aspects can vary from one organization to another, as organizations competences are unique. This classification is extremely relevant in open and ecosystem innovations, as it will be highlighted in the next sub-section.

- **Components/Modular vs. architectural innovation**: as defined in Section 1.1, the architecture is a synthetic representation of technical systems that highlights functional and structural interrelation between the system components. Components innovation, also called modular innovation, focuses on improving the systems components, but it maintains the interface layout inside the technical system; on the contrary, architectural innovation changes the arrangements of the components with minor modifications in the components. (Henderson and Clark 1990) defines an innovation that changes both architecture and components as “radical”; however, since most of the literature provides a different definition of radical innovation, a more correct term could be “system-wide innovation”. Since a change in architecture has usually strong implications on the systems performances and the process, architectural innovation have usually higher innovativeness than components innovation, and their effects tend to reverberate also on the market and the organization’s competences.

While (Schilling 2013) considers just these four classifications, a recent streams of literature called open innovation allows a further differentiation:
Open vs. closed innovation: an innovation process can be classified as open or closed depending on the knowledge flows across the organization’s boundaries. If the innovation process occur inside the firm without exchanges of information across the organization’s boundary, it is classified as “closed” innovation; if the opposite is true, the innovation is “open”. Chesbrough, the father of open innovation, introduced open innovation in opposition to “previous” closed innovation practices (Chesbrough 2003); however, (Trott and Hartmann 2009) proves that these two concepts are not in antithesis and that antecedents to open innovation can also be found in traditional innovation processes. This classification is linked with the competence enhancing vs. competence destroying innovation classification.

The collaboration of developers community in industry platforms can be seen as a particular case of open innovation (Gawer 2009a); therefore, a particular attention is given to open innovation processes, benefits and challenges.

3.2 Open innovation
Open innovation is “the use of purposive inflows and outflows of knowledge to accelerate internal innovation, and expand the markets for external use of innovation, respectively” (Chesbrough et al. 2006). The author takes of the previously described funnel model of innovation in order to explain the differences between closed and open innovation (Figure 2.14). While in purely closed innovation discarded ideas or projects do not generate value, in the open innovation paradigm, they can be licensed to external companies as intellectual property, or generate spin-offs and enter new markets. The use of internal knowledge by external entities is called outbound open innovation. On the contrary, external sources of knowledge can be leveraged in to order to foster novel ideas at the beginning of the NPD process or during the process as technology insource. In this case, the inflow of external knowledge is called inbound open innovation. Modular innovation insourcing is related to the topic of technology infusion, which is detailed in Section 4.1.

Figure 2.14: funnel model of open innovation; adapted from (Chesbrough et al. 2006). Blue dots are the ideas, white dots are discarded ideas that are not brought to the following phase, green dots are inbound innovation ideas and orange dots are outbound innovation ideas.
In its seminal work, Chesbrough (2003) proposes six open innovation principles, in contrast with closed innovation principles:

1. Not all of the smart people work for us, so we must find and tap into the knowledge and expertise of bright individuals outside our company;
2. External R&D can create significant value; internal R&D is needed to claim some portion of that value;
3. We don’t have to originate the research in order to profit from it;
4. Building a better business model is better than getting to market first;
5. If we make the best use of internal and external ideas, we will win;
6. We should profit from others’ use of our IP, and we should buy others’ IP whenever it advances our own business model

Since then, literature about open innovation grew rapidly, and it is now a structured branch of innovation management. Dahlander and Gann (2010) list four types of open innovation, which are presented in Table 2.3.

<table>
<thead>
<tr>
<th></th>
<th>INBOUND INNOVATION</th>
<th>OUTBOUND INNOVATION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pecuniary</strong></td>
<td>Acquiring</td>
<td>Selling</td>
</tr>
<tr>
<td><strong>Non-pecuniary</strong></td>
<td>Sourcing</td>
<td>Revealing</td>
</tr>
</tbody>
</table>

Acquiring is the pecuniary form of inbound innovation, so openness regards how organizations license-in and leverage expertise from outside. A challenge in acquiring knowledge from the outside is the ability of the internal team to understand and take advantage of this knowledge: if the external inputs are too similar to the internal competences, it brings little value, but if it is too different, its integration within existing practices requires significant resources. Selling is the outbound counterpart of acquiring and usually takes the shape of an intellectual property licensing strategy. Some of the most common issues in selling innovation are the reluctance of organizations in selling their own intellectual property, the required transaction costs and the difficulty in evaluating the real value of the intellectual property. The non-pecuniary form of outbound innovation is called revealing and seeks indirect, non-financial benefits from the knowledge dissemination. Typical benefits are feedbacks from the market in terms of evaluation, improvements and prestige. The literature on Users innovation (von Hippel 2005) explores the advantages of this form of open innovation. Clearly, the major challenge in revealing information is choosing what to reveal freely and how to capture the value that this choice generates. The last open innovation strategy is sourcing, the use of external knowledge through non-pecuniary exchanges, like crowdsourcing. Papers on the topic warn that while there is an initial positive effect of sourcing, relying too heavily on external sources of innovation leads to wasted resources or even has a negative impact on the overall innovation process.

Studies on inbound innovation are more common than studies on outbound innovation; (West and Bogers 2014) clusters literature on inbound open innovation into four areas: (1) obtaining innovation from external sources, (2) integrating innovation, (3) commercializing innovation and (4) interaction mechanisms between the firm and its collaborators. Obtaining innovation can be further differentiated into searching for ideas and innovations, enabling and filtering ideas and innovations, and acquiring innovation. In this area, which is the most popular so far, literature has focused on obtaining the maximum number of innovation possible; only recently works tried to understand what is the number of innovation that balance effectiveness and efficiency. Furthermore, organizations like firms and universities are the most studied source of external knowledge. The second area, integrating innovation, deals with antecedents and barriers to innovation, as well as to the absorptive capacity of an organization. One of the most serious barriers to the integration is the “not-invented-here” syndrome (Chesbrough 2003), which reduce the trust in innovations not developed in the internal R&D department. Moreover, some of the most successful examples of open innovation involve external acquisition of knowledge followed by internal research and development of the concept, which is not integrated as a “black-box” (Christensen et al. 2005). The absorptive capacity measures the firm ability to utilize external knowledge; literature is not unanimous on the antecedents and effects of the absorbing capacity, but it has also studied the interrelation between internal and external innovation, which can be seen either as complementary
or competitive. The commercialization of innovation is a central part of open innovation even in inbound innovation, at least in Chesbrough’s intentions, but few articles in literature consider this aspect. Financial performances may not be the main reason for managers to adopt inbound open innovation practices, but evidence from literature show mixed results about the monetary returns of open innovation. Finally, interaction mechanisms entail how two or more organization relate in order to exchange knowledge. This interaction can be dyadic, or take the shape of networks and communities.

Networks and communities of innovation are central to open innovation: as (West and Bogers 2014) noticed, it is not possible to study inbound/outbound innovation without considering the outbound/inbound counterpart and the necessary relationship structure between firms. Additionally, these “innovation ecosystems” are becoming increasingly important as the complexity of the innovation and the span of knowledge required for innovations increases (Dougherty and Dunne 2011). Finally, this topic is strongly related to the design and management of platforms; for example, it well complements the study of collaboration platforms in (Le Masson et al. 2009). Innovation networks are more relevant for internal and supply-chain industry platforms, while innovation communities are associated with the supply-side of industry platforms.

According to (Vanhaverbeke 2006), [open] innovation networks are “collaborative efforts of specialist companies each providing complementary intermediate goods and services”. (Chesbrough et al. 2006) dedicates an entire section to innovation networks, from which important conclusions can be drawn. First, innovation networks can be a reference business model for several industries, from electronics, to pharmaceutics and biotechnology, but also alternative energy and health-care (Dougherty and Dunne 2011). Secondly, the value of open innovation must be assessed across the larger system of collaborations among firms. Inside these networks, the innovation network manager plays a fundamental role in coordinating (industry shaping) and establishing goals (industry foresight) for all the firms. On the other hand, firms have to choose a governance model to tie themselves with other firms. Ties can be classified according to two axes: formal or informal and deep or wide (Simard and West 2006).

![Diagram: nature of ties in open innovation networks](image)

**Figure 2.15: nature of ties in open innovation networks; adapted from (Simard and West 2006)**

Formal ties are “contractually agreed upon, planned channels for knowledge exchange between organizations, such as a strategic alliance” (Simard and West 2006); on the opposite side of the spectrum, informal ties are relationship between firms that do not depend on explicitly stated relationships. Deep ties are the ones that link firms inside a sector or an area of expertise, while wide ties connect firms from different sectors, markets or knowledge. Depending on the combination between tie classes, the four scenarios in Figure 2.15 can be imagined. In general, deep ties are easier to manage, but have less potential for innovation than wide ties.
because the knowledge exchanged is similar. Wide informal ties have high innovation potential, but are difficult to coordinate, while in wide informal ties understanding and assigning roles is challenging.

Open innovation communities are defined as an ongoing voluntary association of individuals (or even organizations) that are organized or leveraged by for-profit actors (West and Lakhani 2008). As remarked in (von Hippel 2007), communities differ from networks in having membership, identity, and group loyalty. Studies in this area usually consider software communities built around an open-software industry platform, like for example the Linux operating systems or the Apache software foundation, but other “open” design examples can be found also in manufactured goods (Shah 2005; Raasch et al. 2009); moreover, customers with a strong affiliation with a brand can also form this kind of community (Füller et al. 2008). (von Hippel 2007) describes three conditions for users’ communities to strive: (1) some users innovate, (2) some users freely reveal their innovations and (3) users can manufacture their innovation “cheaply”. (Ebner et al. 2009) proposes virtual competitions in order to create engineering communities, and recommends that innovativeness of ideas are encouraged, that the topic of the competition is wide enough to attract a large number of participants, that the incentive structure is adequate to the challenge and that all stakeholders are involved since the early stages of the competition.

All these forms of “ecosystem innovation” can be very relevant for product platforms. Two study are described here, one descriptive, the other prescriptive. Adner and Kapoor (2010) test six hypotheses about competition, technological interdependence and innovator’s performances in innovation ecosystems led by a “technology leader”. The ecosystem is divided into technology leader, component suppliers and complement suppliers; components are parts of the technical system, while the complements are products that can be bundled with the components in order to increase the overall customer’s satisfaction. It is found that while technology challenges in components increase the technology leader competitive advantage, technology challenges in complements slow down the technology leader learning curve and increase the likelihood that rivals become more competitive. (Nambisan and Sawhney 2011) differentiates between hub firms as integrators and hub firms as platform leaders and identifies three orchestrating processes. To quote the article, “an innovation integrator primarily focuses on envisioning the core innovation and integrating partners’ contributions to create the final product or offering, whereas a platform leader focuses on defining and developing the core innovation (platform) and facilitating partners’ complementary innovations that expand its reach and range.” In both cases, the management of the network is influenced by three elements: modularity, network openness and network embeddedness (i.e. the number and strength of the interfaces between organizations); these in turn determine three hub firms’ orchestration process. Managing innovation leverage entails identifying opportunities for innovation leverage across the network, defining and coordinating the roles of the firms in the network, and assigning resources to each firm according to the asset needs. Managing innovation coherence means to provide a common ground for the innovation activities, through seminars, information sharing platforms and vocabulary building. Finally, managing innovation appropriability seeks to build trust among members through transparent use and share of IP over the network.

3.3 Innovation diffusion

As mentioned at the beginning of this Section, an invention becomes an innovation only when it is successfully introduced into the market. This sub-section will focus on models that describe and try to predict the diffusion of an innovative product among potential buyers. As stated in (Peres et al. 2010), “diffusion research seeks to understand the spread of innovations by modelling their entire life cycle from the perspective of communications and consumer interactions”. The models proposed can also be interpreted as technology future analysis (Porter et al. 2004; Nikulin 2013) or marketing models; the difference rely in the ultimate goal of the analyses more than in the mathematical models themselves.

The most popular and widely recognized model of technology diffusion is the S-curve model. S-curves are a family of curves whose shape reminds of the letter “S”, the most conventional S-curve is the logistic function, a sigmoid curve having equation:

\[ x(t) = \frac{L}{1 + \exp[-\alpha(t - t_0)]} \]  

(2.4)
Where $x$ is the cumulative number of adopters in time, $L$ is the final market saturation, $\alpha$ measures the steepness of the curve and $t_0$ is the half-life time of the curve. If the sigmoid curve describes the diffusion of an innovation in a market, four categories of adopters can be highlighted (Figure 2.16). In the early launch, early adopters are the first class of customers that buy the product; they usually account for the 13-14% of the market. Early adopters play a very relevant role as initiators of the word-of-mouth and the first class to provide feedbacks to producers. Early adopters are followed by the early and late majority, which increase the number of adopters almost linearly; finally the laggards are the adopters with the highest resistance to innovation.

![Adopters growth curve and classification of adopters](image)

**Figure 2.16: Adopters growth curve and classification of adopters**

S-curves have been observed frequently in product diffusion. Figure 2.17 shows the cumulative number of adopters over time for many common goods. It can be noticed how the diffusion speed has increased in the last 30 years.

![Diffusion of innovation of household commodities from 1900 to 2005](image)

**Figure 2.17: diffusion of innovation of household commodities from 1900 to 2005; retrieved from (Slicker City 2013)**
Chapter 2

State of the art

The first and most influencing model that tried to represent diffusion phenomena in the market is due to Bass (1969). According to this model, innovation diffusion of durable goods with potential \( m \) is a contagious process governed by company-driven influences like advertising (represented by the parameter \( p \)) and customers-driven influences like word-of-mouth (represented by the parameter \( q \)). Every customer is connected to all the others in a homogeneous social network, and the probability that a customer chooses an innovative product is linear with the previous number of adopters. The flow of incoming new adopters \( n \) at time \( t \) is given by:

\[
n(t) = \left( p + q \frac{N(t)}{m} \right)(m - N(t))
\]

(2.5)

Where \( N(t) \) is the cumulative number of adopters until time \( t \). \( N(t) \) is distributed as a sigmoid curve, while \( n(t) \) has bell-shaped profile. The parameters \( p, q \) and \( m \) can be estimated from historical data; however, the model predictive power has been subject to criticism. Furthermore, the model three parameters cannot be easily correlated to real-world practices and people social network are not homogeneous in nature. In order to overcome these three limitations, novel models have been developed.

According to (Peres et al. 2010), literature on innovation divided into diffusion within markets and technologies diffusion across markets and brands. Inside the first group, models focus on social network influence, the importance of network externalities, take-off and saddle prediction and the sequence of technology generations; the second group measures the influence of geographic differences in innovation diffusion and competition between brands. Given the scope of this thesis, two aspects of the literature will be scrutinized: the effect of social network and externalities, modelled thanks to agent-based models.

Agent-based models are numerical representation of complex adaptive systems (Holland 2006) that model heterogeneous properties and behaviours at micro-level in order to understand the overall behaviour of the macro-level (Garcia 2005). Each model is constituted by a set of agents, defined by their characteristics and behaviours, an environment and the rules of interactions. Agent-based models are particularly suitable to simulate human systems (Bonabeau 2002) and economic systems (Heckbert et al. 2010; Chakraborti et al. 2011; Chen et al. 2012), even though the verification and validation of their results is often critical (Garcia et al. 2007; Gürçan et al. 2013; Smajgl and Barreteau 2014).

Agent-based innovation diffusion models can be clustered into cellular automata-based and social network-based models. The former class refers to an evolution of the Bass model, while the latter class are complex computational models based on simulated interactions. Agents’ behaviour usually consists in a non-linear, stochastic model that defines whether an agent purchases a product or not; as mentioned, the interactions among agents and with the environment are defined by the social network. A comprehensive and detailed review of the literature on agent-based innovation diffusion models is provided in (Kiesling et al. 2012), where models are classified according to internal decision rules and social influence. In particular, decision rules determine the behaviour of the agents; they can be inspired by algebraic models, utilitarian rules, state-transition models, opinion dynamics models, social psychology or econometrics. Among the various results, literature studied in depth two issues: the importance of social network and the role of early adopters.

The importance of the network topology (Newman 2003) have been consistently assessed across the literature in the field (Kiesling et al. 2012). Models assume that customers purchasing behaviour is mainly affected by word-of-mouth and that satisfied customers promote goods with other potential customers. Under these assumptions, the diffusion dynamics is affected by the network topology and the features of the initial adopters, who spread the word across the social network. In order to study the effect of topology on innovation diffusion, (Abrahamson and Rosenkopf 1997) combines a social network composed by a high-density core and a low-density periphery with a threshold adoption model. After simulating several initial seeding conditions, the article concludes that the global density of the network as well as local idiosyncrasies in the connectivity of boundary nodes have a great influence on innovation diffusion. A more recent and detailed research paper (Delre et al. 2010) performs a sensitivity analysis over the parameters of an innovation diffusion threshold model. Innovations diffuse more quickly in scale free networks than in regular lattices, sensitivity to word-of-mouth in scale free networks damps the diffusion of innovation and increase the variability of the results.

One of the main goals of the research field is the study of the so-called “hubs” or “opinion leaders”, i.e. customers with high centrality in the network, who can influence several other potential adopters. The simulations in (Delre et al. 2010) confirms that the influence of opinion leaders (i.e. the number and strength
of their social links) is positively correlated with innovation diffusion in a network. A similar differentiation of social influence is given in (Eck et al. 2011), which assess the importance of informational and normative influence on followers. The former refers to the propensity to trust other people opinion as a true picture of reality, while the latter describes how prone an agent is to others’ expectations. If opinion leaders can be identified early, their purchase can be subsidized, and the product spreading is enhanced. According to (Cho et al. 2012), distance centrality is the best indicator of opinion leaders if the target of the marketing process is the maximum cumulative number of adopters, while sociality centrality is the best indicator if the fastest speed of diffusion is the goal.

Interestingly, the role of early adopters varies if a product is a platform or not (Frattini et al. 2014). Non-platform innovation is more sensitive to imitation mechanisms, while platform innovation is influenced by dissemination mechanisms. Imitation mechanism is subject to normative influence, while dissemination mechanisms are subject to informational influence. In this respect, it has been observed that innovativeness influences the amount of word-of-mouth, while quality and performances affect the valence (positive or negative) of word-of-mouth (Moldovan et al. 2011). Moreover, network externalities slow down the innovation diffusion process, since the majority waits for the early adopters to reach a critical mass before they can exploit the platform (Goldenberg et al. 2010). This behaviour has strong implications for industry platforms, and adds a second critical mass problem to the one described in Section 2.3.

Agent-based model have been utilized also to study the dynamics of two-sided markets, in particular payment card competition. (Alexandrova-Kabadjova et al. 2011) simulates a payment card market constituted by merchants and customers distributed in a regular lattice. Customers and merchants are able to subscribe or unsubscribe to new cards according to a stochastic model that depends on the cards accepted by the merchant or the cards owned by customers, respectively. The simulations reproduce the tendency to a scenario dominated by a small set of cards that can be observed in the current payment card market; this behaviour is mediated by the market inertia and the information shared across merchants and customers.

4. Engineering changes and change llities

4.1 Engineering changes and technology infusion

Engineering changes are a critical aspect of engineering design. According to some reports, engineering changes consume more than 30% of the R&D resources in a firm (Soderberg et al. 1989; Langer et al. 2012), and can account for a major of the total project costs (Burati et al. 1992; Chang et al. 2011).
### Table 2.4: List of definitions for change in technical systems

<table>
<thead>
<tr>
<th><strong>SOURCE</strong></th>
<th><strong>DEFINITION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Wright 1997)</td>
<td>An engineering change (EC) is a modification to a component of a product, after that product has entered production</td>
</tr>
<tr>
<td>(Huang and Mak 1999)</td>
<td>[Engineering changes are] the changes and modifications in forms, fits, materials, dimensions, functions, etc. of a product or a component</td>
</tr>
<tr>
<td>(Terwiesch and Loch 1999)</td>
<td>Engineering change orders (ECOs) [are] changes to parts, drawings or software that have already been released</td>
</tr>
<tr>
<td>(Huang and Mak 2003)</td>
<td>Engineering changes are changes and/or modification in fits, functions, materials, dimensions, etc. of a product and constituent components after the design is released</td>
</tr>
<tr>
<td>(Fricke and Schulz 2005)</td>
<td>The expression “changes” […] encompasses all kinds of changes, whether changes of needs, requirements, specifications, already built components, processes, cost, schedule, and so on.</td>
</tr>
<tr>
<td>(Tavčar and Duhovnik 2005)</td>
<td>Engineering change is a modification of a product after the product has entered production</td>
</tr>
<tr>
<td>(Jarratt et al. 2011)</td>
<td>‘An engineering change is an alteration made to parts, drawings or software that have already been released during the product design process. The change can be of any size or type; the change can involve any number of people and take any length of time.’</td>
</tr>
<tr>
<td>(Siddiqi et al. 2011)</td>
<td>A design change is considered to be a process in which a design that had been previously considered as finished or completed is revised</td>
</tr>
<tr>
<td>(Vianello and Ahmed-Kristensen 2012)</td>
<td>Engineering change is a modification to a component of a product, after the original design has been completed</td>
</tr>
<tr>
<td>(Hamraz et al. 2013)</td>
<td>Engineering changes are changes and/or modifications to released structure (fits, forms and dimensions, surfaces, materials etc.), behavior (stability, strength, corrosion etc.), function (speed, performance, efficiency, etc.), or the relations between functions and behavior (design principles), or behavior and structure (physical laws) of a technical artefact</td>
</tr>
</tbody>
</table>

The first challenge when dealing with engineering change is choosing a definition (Table 2.4). All definitions agree upon the fact that an engineering change is a modification of something that was considered to be “fixed” or have already been released. Differences arise about when the change happens and what is changed. According to some authors, engineering changes should be limited only to the modifications during the design phase; according to others, all lifecycle phases can be considered. Another point under discussion is what is changed. Among the others, authors cite components, forms, materials, drawings, structure, behaviour and functions, needs, requirements, specifications, processes, schedule. In the context of this paper, engineering changes can occur during the entire system’s lifecycle and involve only changes in the technical system (Hubka and Eder 1988). It must not be forgotten anyway that several other supporting or contiguous systems, like manufacturing, organization or markets, and the design process do play a fundamental role in generating and constraining engineering changes.

The remainder of the section will examine in depth seven features of engineering changes, as derived from literature (Figure 2.18).
Every system lives a double life: as a physical realization and as a representation. The first is the concrete set of elements perceived to form a whole, while the second one is the abstract concept of a whole. This dualism was highlighted in Soft Systems Thinking (Checkland 2000), but it can be found, under various names, in other fields. The holon is clearly separated from the system in (Checkland 1988); Chen and Crilly (2015) use the words instance and characterization, while Gu and co-authors (2009) separates the design aspect from the product. Engineering design makes use of several representations, like blueprints, mathematical models or bills of material; more generally, every individual reasons in terms of representations of systems, as claimed in (Checkland 1988). In this framework, engineering changes are separated according to the object of change, either the physical system (instance of a system) or its representation (holon of a system).

The model of change described in (Ross et al. 2008) defines the mechanism of change as the path from one system’s state to another, without specifying how the transition is performed. Generally, three change mechanisms are identified: modification (a change in components or interfaces parameters), substitution and addition/removal. These three change mechanisms can be applied on system’s components, on system’s interfaces or on both. As an example, let us consider a change in a bridge design (holon) under the circumstance of deformation higher than expected. If the size of a beam is increased, the designer is performing a modification, whereas if some new beams are added, the mechanism is an addition. On the other hand, substitution occur if the beam is replaced by one having a different section. Modification requires that components’ parameter can be changed, therefore it can be applied either to holons, which consists of information, or to instances that allow such a change, for example tunable mechatronic systems. On the other hand, substitution can be applied to both holons and systems with changeable interfaces.

Every engineering change has an initiator, i.e. something that generates that change (Table 2.5). In the field of engineering change management, two main change initiator categories have been identified, native changes and propagated changes. Native changes are original modifications that can be desired or undesired, while propagated changes are undesired modifications due to other changes inside the system. Native changes can be further differentiated into initiated and emergent changes (Clarkson et al. 2004a; Eckert et al. 2004), depending on the system’s boundary. Initiated changes are generated by an outside source, while emergent changes are “caused by the state of the design, where problems occurring across the whole design and throughout the product LC can lead to changes” (Eckert et al. 2004). The most typical initiated change is due to change in requirements, while emergent changes can arise because of problem in communication, errors in the design phase, defects or wear. Several papers (Chang et al. 2011; Vianello and Ahmed-Kristensen 2012) distinguish the reasons for change into generic and specific to the project; the latter can be separated according to the stakeholders’ degree of control. In (Langer et al. 2012) four major sources of change are highlighted: insufficient clarification of requirements, human error, insufficient external communication and inadequate processes, methods or tools for support. Siddiqi et al. (2011) specifies changes that arise from original errors, change in requirements and market shifts.
Table 2.5: Classification of changes according to initiator

<table>
<thead>
<tr>
<th>Change</th>
<th>Initiator</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiated change</td>
<td>Reason external to the technical system</td>
<td>Change in requirement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Market shifts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Innovations</td>
</tr>
<tr>
<td>Emergent change</td>
<td>Reason internal to the technical system</td>
<td>Ambiguous communication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Errors in design</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wear</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inadequate design processes, methods or tools</td>
</tr>
<tr>
<td>Propagated change</td>
<td>Another change inside the technical system</td>
<td></td>
</tr>
</tbody>
</table>

Literature also provides some figures about the relative frequency of changes. In the automotive industry, 77% of the changes are emergent or propagated, while only 23% of them are initiated; the most common origin of change is documentation (28% of the total), cost correction accounts for the 16% circa, while both design corrections and inventory issues generate a 9% of the total changes. In emergent changes, requirements change account for just the 0.7% of the total. On the other hand, Griffin et al. (2009) showed how, in a complex cyber-physical system, changes in requirements leads to propagated changes in other requirements 48% of the times. A survey involving 50 firms in automotive and electrical equipment industries showed that 56% of the changes are emergent, while the rest is initiated (Deubzer et al. 2006); among the emergent changes, the 61% is unavoidable.

(Giffin et al. 2009) calls the components that have the tendency to reduce change propagation “absorbers”; in this paper, the more general term buffer is introduced to describe component, interface or a set of components can be modified without generating further changes. Buffers play a fundamental role in change-related utilities, since they limit change propagation and therefore make easier to change a system.

As noted in (Ross et al. 2008), every change must be performed by an agent, which could be external or internal to the system. Typical external agents are designers, technicians, suppliers or users of the system. In formal change management during the design phase, three distinct roles can be identified (Siddiqi et al. 2011): the person who initiates the change, the verifier and the executor of the change; in other life-cycle phases or in less formal environments the roles can be accomplished by the same agent.

The life-cycle phase of the system plays a fundamental role in characterizing changes. Literature often assumes that every time a system goes through a phase in the life-cycle, the cost for change increases by a factor of ten (Boehm 1981). Furthermore, the number of changes depend on the life-cycle phase. (Langer et al. 2012), for example, analyses responses from a pool of 90 Danish firms and states that 60% of changes occur during the production phase or after the product has been released to the market. Similarly, (Vianello and Ahmed-Kristensen 2012) compare changes in two projects: an aero-engine (as typical variant design) and a drilling equipment for the oil industry (representative of customizable products). In the first case, 8% of the changes occurred during the product development, 76% during the manufacturing phase and 16% during service. In the second case, 15 changes over 100 were due to issues during the development, 55 changes out of 100 happened during installation, and the remaining during service.

Most of the change-related utilities frameworks proposed in the previous section made clear that changes require time and resources to be performed. Another measure of the resources need can be the number of subsequent (propagated) changes (Giffin et al. 2009; Siddiqi et al. 2011; Pasqual and de Weck 2011) or the risk that the change poses to the system: (Langer et al. 2012) differentiates average changes and critical changes according to the impact that change has on the system’s product development.

Technology infusion refers to the practice of proving modular innovation through the addition of innovative components in legacy systems blueprints. In complex technical systems, novel technologies are first developed in laboratories and R&D departments; once they are mature enough, they are introduced inside the final product. However, the introduction of these new technologies can have repercussions on the technical system the novel part is infused into; therefore, technology infusion can be studied from a change propagations.
perspective. (Smaling and de Weck 2007) evaluates the benefits and costs of a set of possible technology infusion opportunities thanks to a utility-based assessment. First, a large set of potential innovative solution is generated; then, a fuzzy Pareto-front filtering selects the most promising solutions, taking into consideration also the uncertainty surrounding the system. The third step is the computation of the resources required for change in the technical system, and the potential increase in stakeholders’ utility. Finally, the candidate solutions are represented in an opportunity vs. risk tradespace, which informs decisions about technology development. (Suk Suh et al. 2011) evolves the previous methodology by focusing on the estimation of value, both in terms of utility and costs, and in terms of net present value of the R&D investment. The updated methodology is then applied to the evaluation of a novel technology infusion in a printing system.

4.2 Change-related Ilities’ frameworks
This sub-section means to analyse a set of relevant change-related ilities’ frameworks from literature. Navigating the huge literature about change ilities can be challenging, because of the breadth of the topic. (de Weck et al. 2011) has computed that in 2010, around 10000 referenced journal articles had “adaptability” as a central theme, while around 5000 articles concerned “flexibility”. Such a huge body of literature is hard to summarize with a bottom-up approach; the task is made even more daunting by the fact that some keywords, like “change” or “flexibility”, are used commonly in conversations and in other fields of knowledge. Furthermore, in recent years a series of frameworks on Ilities have been published. Therefore, instead of directly analysing the entire body of literature, this work analyses a selected number of updated frameworks, whose combined knowledge can cover a big part of the literature in the field.

To give the research a solid foundations, a rigorous set of rules were employed during the literature research. First, works needed to concern definitions, characterizations or organization of change-related Ilities. In fact, a surprising number of papers propose methodologies to increase the “flexibility” or “adaptability” of products or processes, without defining the meaning of the Ility mentioned (Ryan et al. 2013). Secondly, only sources published from 2005 onwards have been selected. This allows having an up-to-date overview of the discussion, while guaranteeing that a sufficient number of sources are compared. Thirdly, articles or books had to be classified as reviews of previous literature or had to mention specifically Ilities frameworks. Given the three rules above, nine major frameworks have been found, which summarize the knowledge of approximatively 330 independent sources in the fields of life-cycle properties and engineering design. A short description of the nine frameworks’ content follows.

(Saleh et al. 2009) proposes a literature review on flexibility in different domains. The paper recognizes that flexibility is a word often abused; therefore, it explores its meaning in decision theory, management, manufacturing systems, in design process and in products (flexible systems). As far as the engineering design field is concerned, flexible design processes deal with the uncertainty in requirements during the early stages of product development, while flexible systems are able to “satisfy changing requirements after the system has been fielded”. Therefore, flexibility is the ability to respond to a change in requirements during the utilization/operation phase.

In the book “Engineering systems: Meeting human needs in a complex technological world”, de Weck et al. (2011) dedicate a chapter to the definition of several Ilities. Flexibility can manifest itself in the regime of operation or redesign, and it is an umbrella term for several other Ilities, like Evolvability (the property of being redesigned easily), Adaptability (ability to be changed easily in response to a change in the external environment) or Agility (the ability to be changed quickly). Similar properties are Reconfigurability, thanks to which a system can change structure to perform multiple functions, and Scalability, which involves changes in size to support something. Finally, Extensibility allows systems to fulfil new functions or new sets of functions. The chapter also stresses the importance of these Ilities for the success of systems, taking as an example TCP/IP protocol, whose flexibility was crucial to insure the growth and diffusion of Internet and the World Wide Web.

Similarly to Flexibility in the previous citation, (Fricke and Schulz 2005) proposes the use of Changeability as an umbrella term for other Ilities. In particular, Robustness is the ability to be insensitive towards changing environments, Agility the ability to change quickly, Flexibility the ability to be changed easily and Adaptability the ability to adapt to changing environment. While Adaptability and Robustness do not require the implementation of changes from external, Agility and Flexibility do. The article then proceeds with the enumeration of techniques to foster each Ility during systems design.
Three papers track the gradual evolution of a framework on change ilities. (Ross et al. 2008) considers change-related ilities as means to increase the life-cycle value of a system (Browning and Honour 2008). The article starts with a model of change in systems and then associates different ilities to various aspects of the model. Changes can be characterized by a change agent, a change effect and a change mechanism. Changeability “is determined by the number of acceptable change paths that can be taken by a system.” Flexibility is determined by changes performed by agents external to the system, while Adaptability is needed when the agents are internal to the system. Modifiability measures the resources to add or subtract an attribute, while Scalability increases or decreases the value of an attribute. The approach of relating change ilities to change properties is very interesting and it has inspired the classification in Section 4.3. The framework is expanded in (de Weck et al. 2012), which considers Changeability the most generic change ility, as a changeable systems is able to alter its operations or form, and consequently possibly its function, at an acceptable level of resources. Furthermore, Evolvability is defined as the ability of a system design to be inherited and changed across generations. Agility is again defined in terms of quickness of change, while system Reconfigurability allows changing components arrangement and links reversibly. Finally, in (Mekdeci et al. 2014), the main ilities under evaluation are Viability and Pliability. Viability is “the current likelihood that an engineered system will provide acceptable value to its stakeholders, over its life era”. Pliability is related to different viabilities, in that a pliable system is able to remain viable by switching between other viable instances as specified by the architecture’s pliable set (Mekdeci et al. 2014). A system instance is the physical realization of a system blueprint that is allowed by the system architecture.

(Ryan et al. 2013) starts with a research question: is it possible to better define change ilities starting from literature consensus? As the authors claim, while the majority is not always right, “it does tend to shift the burden of cogency onto the minority.” The literature review classifies ilities according to five questions related to the inherent presence of changes, the measure of change efficiency, the source of change (internal or external to the system), the possibility to foresee a change and the life-cycle phase. Five life-cycle properties are identified: Flexibility is “the measure of how easily a system's capabilities can be modified in response to external change”; Adaptability is “the measure of how effectively a system can modify its own capabilities in response to change after it has been fielded”. Robustness, which is not here considered as a change-related ility, is “the measure of how effectively a system can maintain a given set of capabilities in response to external changes after it has been fielded”, while Agility defines “a measure of how quickly a system's capabilities can be modified in response to external change”. Finally, Versatility allows systems to have from the beginning capabilities beyond stated requirements.

Several of the previous frameworks associate the need of change ilities because of changes in requirements. On the other hand, (Chalupnik et al. 2013) considers change ilities as a protection from the uncertainty that usually envelope the design of a system, especially in early design phases. In this respect, change ilities are associated to the reliability of a system, as they can mitigate the risk that a system incur in case of unexpected events. Adaptability is defined as the “ability of a system to be modified in order to do its basic job in uncertain or changing environments”, Versatility allows systems to “to do jobs not originally included in its requirements.” Flexibility, which is again recognized to be very vaguely defined in literature, is associated with “the ability of a system to change its states, i.e. sets of capabilities together with operating conditions.”

(Gu et al. 2009) considers how to design a system so that it can be easily changed during its lifecycle. The article differentiate between two different adaptabilities: Design Adaptability refers to the ease with which an existing design can be modified to accommodate changes in requirements; mass customization can be considered as a special application of Design Adaptability. Product Adaptability is the “the capability of a physical product to be adapted to satisfy the changed requirements”. A personal computer that can be updated with more performing components is an example of Product Adaptability.

The nine frameworks proposed share many similarities. All of them agree on the difficulty of defining change ilities and tried to give definitions according to differentiating principles that authors considered relevant. Some of them proposed a model for systems’ change, like (Ross et al. 2008; de Weck et al. 2012; Mekdeci et al. 2014), others considered different lifecycle phases, like (Saleh et al. 2009; Gu et al. 2009); others evaluate if a change inside the system is necessary (Fricke and Schulz 2005). Requirements are very often cited as the origin of changes in system, and there is a strong consensus about the bond between Agility and quickness of change.

At the same time, there are many differences and inconsistencies between these frameworks. The main issue is the use of the same words to indicate different aspects of change, which in turns generates ambiguity. For
example, does Flexibility relate to internal or external sources of change? Is Adaptability about the Utilization phase, or any system’s phase? (Ryan et al. 2013) tries to solve the issue by examining the literature and the semantics behind the definitions, but in many cases, the questions that are supposed to clarify the meaning of the Ilities are left unanswered. A model for change in systems can support a coherent classification. Indeed, the framework in (Ross et al. 2008; de Weck et al. 2012; Mekdeci et al. 2014) is backed by a conceptual model of changes in systems, but it provides a high-level perspective on changes in the system, without detailing the impact of change on systems’ structure or functions. Therefore, it does not explicitly consider some aspects of change, like change propagation, which is a critical aspect in complex systems design.

4.3 Mapping the Ilities on the Engineering change classification
The Engineering change classification will now serve as the basis for a comparison of the previous definitions of change-related Ilities, in order to clarify the literature. The Ility definitions exposed in Section 4.2 are reframed according to the engineering change features in Section 4.1. This task was not trivial, as several times the definitions either do not precisely indicate which engineering change features are involved or indirectly assume some of them. Often, interpretation of the definition’s connotations was needed, as well as deduction from the context. I apologize to any source authors if this process led to misinterpretations of the original meaning.

The results of this process can be appreciated in Table 2.6. It can be noted that the Tables are quite sparse, as, in many cases, definitions cover just one or two aspects of the Engineering change. As far as the object of change is concerned, (Gu et al. 2009) is the only framework that clearly separates changes in holons and changes in systems’ instances. The other frameworks either do not specify the object of change, or define some Ilities on holons and other on instances. Change buffers differentiates clearly two Ilities: Reconfigurability, which absorbs changes by rearranging the interfaces between components, and Pliability, which allows systems to change into different architectures. The most distinctive differentiation factor is the life-cycle phase. Some frameworks focus on just the operations phase, while others consider the entire system lifecycle. Remarkably, Evolvability is always applied to the redesign phase. Change mechanism was perhaps the most ambiguous aspect of change-related Ilities, and the one that was most subject to interpretation. Curiously, many frameworks mention the ability to increase in size or functionalities (Scalability), but few consider also the Subtraction mechanism (van Voorthuysen et al. 2015). Agents are often not well specified, but there is a clear demarcation between agents internal to the system (or the system itself) and agents external to the system. In a similar fashion, initiator can be subdivided into two groups: internal and external to the system. All the Ilities collected focus on external (initiated) changes, mainly because of changes in requirements (during the design phase) or the instance’s environment (during the operations phase). (Fricke and Schulz 2005) makes a distinction between changes that need to be implement and changes that does not. Finally, all Ility frameworks focus on two types of resources: money and time. As already mentioned, there is consensus on Agility, as the Ility that reduce the time for changes. The other Ilities seems to be more concerned about monetary expenditures.
<table>
<thead>
<tr>
<th>ARTICLE</th>
<th>ILITY</th>
<th>OBJECT</th>
<th>BUFFER</th>
<th>LC PHASE</th>
<th>MECHANISM</th>
<th>AGENT</th>
<th>INITIATOR</th>
<th>RESOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saleh et al. (2009)</td>
<td>Flexibility</td>
<td>Instance</td>
<td>After launch</td>
<td></td>
<td></td>
<td>External (changing requirements)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>de Weck et al. (2011)</td>
<td>Reconfigurability</td>
<td>Instance</td>
<td>Configurations (arrangements and links)</td>
<td>Operations</td>
<td>(Interface) Substitution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flexibility</td>
<td></td>
<td>Design, operations, redesign</td>
<td>Modification, Substitution, Addition/Subtraction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptability</td>
<td></td>
<td>Design, operations, redesign</td>
<td>External (environment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Agility</td>
<td></td>
<td>Design, operations, redesign</td>
<td>External (environment)</td>
<td></td>
<td></td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extensibility</td>
<td>Functions</td>
<td>Operations and redesign</td>
<td>Addition</td>
<td>External (Environment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scalability</td>
<td></td>
<td>Operations and redesign</td>
<td>Addition/Subtraction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chalupnik et al. (2013)</td>
<td>Adaptability</td>
<td></td>
<td>Redesign</td>
<td></td>
<td>System itself External (Environment)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ryan et al. (2013)</td>
<td>Flexibility</td>
<td>Instance</td>
<td>System’s states</td>
<td>Modification</td>
<td>External Focus on efficiency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptability</td>
<td></td>
<td>Operations</td>
<td></td>
<td>External (Requirements)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Agility</td>
<td></td>
<td>External (Requirements)</td>
<td></td>
<td>Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gu &amp; Nee (2009)</td>
<td>Adaptability</td>
<td>Holon</td>
<td>Design</td>
<td></td>
<td>External (Requirements)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptability</td>
<td>Instance</td>
<td></td>
<td></td>
<td>External (Requirements)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6: Mapping the changeilities to the Engineering change features (part 1 of 2)
<table>
<thead>
<tr>
<th>ARTICLE</th>
<th>ILITY</th>
<th>OBJECT</th>
<th>BUFFER</th>
<th>LC PHASE</th>
<th>MECHANISM</th>
<th>AGENT</th>
<th>INITIATOR</th>
<th>RESOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fricke and Schultz (2008)</td>
<td>Agility</td>
<td>No distinction</td>
<td></td>
<td>External (implementation necessary)</td>
<td>Time</td>
<td>“Easily”</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flexibility</td>
<td>No distinction</td>
<td></td>
<td>External (implementation necessary)</td>
<td>System itself</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptability</td>
<td>No distinction</td>
<td></td>
<td>External to the system</td>
<td>Internal to the system</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ross et al. (2008)</td>
<td>Flexibility</td>
<td>External to the system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adaptability</td>
<td>Internal to the system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scalability</td>
<td>Modification</td>
<td></td>
<td>Substitution, Addition/Subtraction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>de Weck et al. (2012)</td>
<td>Agility</td>
<td>Holon or Instance</td>
<td>Accessible level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changeability</td>
<td>Holon Instance</td>
<td></td>
<td>Redesign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evolvability</td>
<td>Holon Instance</td>
<td></td>
<td>Additions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extensibility</td>
<td>Holon Instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reconfigurability</td>
<td>Holon Instance</td>
<td></td>
<td>Change of arrangements</td>
<td>Addition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mekdeci et al. (2014)</td>
<td>Pliability</td>
<td>Instance Architecture</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6: Mapping the change Ilities to the Engineering change features (part 2 of 2)
5. Value

Value is one of the main concepts in economic history, dating back even to Adam Smith’s “The Wealth of the nations”. It is not hard, therefore, to find several definitions and perspective on the concept of value; it is much harder to find a synthesis of the various viewpoints. Given the scope of the thesis, we will focus on the subset of literature that explains why customers choose certain products and how this notion can be used for customizable platforms. In this framework, value is approached from four main areas of investigation: economics, marketing, engineering design and mass customization (Figure 2.19).

As mentioned above, value has been at the centre of deep discussions in Economics. Classical economists distinguish between “use value” and “exchange value.” Use value refers to the benefits offered by a product as perceived by potential customers, while exchange value is the monetary conversion of value when a product is exchanged. Consumers are willing to trade exchange value to gain use value, as long as the price (exchange value) does not exceed the price the customer is willing to pay (use value). Consumer surplus is the difference between the maximum amount a consumer is willing to pay for a good and the amount he must actually pay to purchase the good in the marketplace (Besanko and Braeutigam 2011). It is important to underline that use value is always perceived, i.e. is a subjective evaluation of the customers. When a product is sold, buyers want to maximize the consumer surplus, while sellers want to maximize the profit, which is the difference between price and costs. Profit is the value captured by the firm (Bowman and Ambrosini 1998). The theory of value as illustrated above leads to several microeconomics conclusions.

Marketing has a perspective on value similar to Economics, but it is also driven by a more practical goal: understanding why customers choose a certain product among others. A comprehensive literature review about value in marketing is given in (Lindgreen et al. 2012); here, the notion of value as a discriminant in customers’ choice is addressed. The basic assumption in many marketing choice models (Chandukala et al. 2007) is the existence of a scalar measure of consumer preferences called consumer utility, which can be used to rank several choice alternatives. Consumers try to maximize their utility given some form of monetary constraints;
the geometrical locus of the maximum utilities given certain budgets is called “indirect utility function”. Several choice models can be found in literature; depending on the goals of the model and the assumptions in the modelling, some can be more appropriate than other (Ben-Akiva 1997; Chandukala et al. 2007). How to populate the parameters inside these models is the subject of Conjoint analysis (Green and Rao 1971; Ben-Akiva and Lerman 1985; Rao 2014). Conjoint analysis is “a set of techniques ideally suited to studying customers’ choice processes” (Rao 2014); a conjoint analysis presents itself as a questionnaire where respondents have to make trade-offs between product features, including price. Conjoint methods can be subdivided into rating-based methods and choice-based methods. In the first category, respondents are invited to rate a combination of products features; in the latter category, respondents have to choose between two or more products, each one with specific features. Both categories have points of strength and weaknesses; some hybrid (Green et al. 1981) and adaptive (Johnson 1987) methods were developed in order to overcome the limits of traditional methods.

While Economics and Marketing focus on the effects of value on markets and customers, how to generate that value in a product is a typical topic of Engineering design; in particular, a set of tools and methods called “Value analysis and engineering” has been developed (Miles 1961). Value analysis allows designers to understand what part of a technical system has low value, while value engineering increase the value of a system by identifying solutions to technical problems. Value, in this area, is defined as benefits over costs: “value is the most-cost-effective way to reliably accomplish a function that will meet the user’s needs, desires and expectations” (Dell’Isola 1997). As far as the Value Engineering methodology is concerned, it consists of several phases, which form the job plan (Kasi 1994; Dell’Isola 1997); those phases can be summarized as: (1) Information gathering, (2) Alternatives generation, (3) Evaluation of solutions, (4) Proposal development and (5) Presentation and implementation.

As the object of the Value analysis is a product platform, a systemic perspective can be very useful. Systems engineering brings a different perspective on the topic, since it highlights three key points (Cook and Wissmann 2007; Saavedra et al. 2014; Crawley et al. 2015): the presence of multiple stakeholders, the emergent properties and the life-cycle value. In a systemic perspective, value cannot be attributed to a single stakeholder, but to several ones; thus, needs have to be elicited, prioritized and managed in the Stakeholder requirements definition process and in the Requirements management process (Haskins 2006; Bjtan et al. 2013) at the beginning of the technical development. Furthermore, in every system is possible to highlight two different set of properties and behaviours: local and emergent (Crawley et al. 2015). Emergent properties and behaviours cannot be associated to a single or a subset of the system, and usually are associated to the major part of the value in a system (Cao and Duan 2013). Finally, value cannot be assessed in a single moment in time, but throughout the entire life-cycle (Browning and Honour 2008), since complex technical systems can have a long life. This topic is related to the value of adaptable and customizable systems, which would need to be further explored.
Part 2.0

Strategic design analyses outline
Chapter 3

Methodological framework

The Introduction laid out the benefits of cyber-physical industry platforms. Industry platforms can deliver value to customers thanks to a high level of customization, they can be innovated by a large community of Developers providing inbound innovation and they can potentially have a longer lifecycle than integral products, thus generating less material waste. However, the current design methods do not addresses the design for lifecycle of industry platforms, even though it is critical, and it is not clear how strategic design decisions influence the outcome of the product development activity, i.e. the cyber-physical system.

This chapter describes the framework that unifies the three design analyses developed, which will be further detailed in the remaining three chapters of Part 2. Section 1 provides a general model that synthesized the interplay between industry platforms and two-sided markets. The implications for the design process are detailed in Section 2, which also offer an overview over the three design analyses that constitutes the methodology.

Figure 3.1: generic systemic representation of an industry platform and its different supervisors
Chapter 3

Methodological framework

1. A model for industry platforms

In the framework of this thesis, (modular) cyber-physical industry platforms are cyber-physical systems developed by one or several firms that exploit the benefits of changeability in order to offer other firms the opportunity to build complementary products, services or technologies (Gawer and Cusumano 2014). The focus will be on complementary products and technologies; complementary services will be addressed in future works.

It is possible to separate two parts in product platforms: the core and the modules (Figure 3.1). The core (often called simply “platform”) is the physical or digital part of the system that supports and coordinates the other modules; in cyber-physical systems, the core can be constituted by physical components, software algorithms, data structures, protocols and interfaces. Platform modules, or modules, are the complementary products and technologies offered by the external firms, here generically called “(module)s developers.” The architecture of the core can be integral or modular, but for sake of simplicity, only the modules external to the core will be considered. In this thesis, the term industry platform designates the whole, generic technical system. The platform architecture is the abstract description of the platform entities and their interrelations (Crawley et al. 2004); in this representation, the number and interfaces of generic modules are considered. A platform configuration, on the other hand, instantiate the platform configuration with the specific modules designed by specific developers. The difference in the two models’ granularity depends on the particular application of the model and the goals of the modeller.

Every industry platform is associated with a multi-sided market (Gawer 2009b), but in this thesis, only two sides will be considered: the developers and the users (Figure 3.2). As mentioned above, developers are individuals or firms that provide modules complementary to the industry platforms. Users are individuals that can exchange monetary value for the use value of the platform and its modules (Bowman and Ambrosini 1998). The platform ecosystem consists of the community of users who have adopted the platforms (the “adopters”), as well as the developers who offer at least one module in the market (the “supporters”) and the modules in the market.

It is often said that multi-sided markets are characterized by strong network externalities (Rochet and Tirole 2003; Armstrong 2006; Rochet and Tirole 2006): the more agents join one side, the more the other side desires to join the market. Using cybernetic terminology, there is a double, circular positive feedback loop linking the two sides, which, in the case of industry platforms, is mediated by the presence of the modules. In fact, users
are attracted by the number of modules offered in the market, because they determine the function provided by the platform configurations; for users, the number of developers is not as relevant as the number of modules offered. The two-sided market is therefore defined as “asymmetrical”: developers receive benefits from the increased number of adopters, since the market potential increases, but users gain advantages from the number of modules in the ecosystem, which is indirectly related to the number of developers.

![Diagram of Socio-Technical Model of Asymmetrical Industry Platforms](image)

Figure 3.3: socio-technical model of asymmetrical industry platforms highlighting the positive feedback loops (full arrows) and ownership relations (dotted arrows)

The platform model and the asymmetrical two-sided market model are coupled, as it can be seen in Figure 3.3. The architecture and the technical specifications of the core determine the basic features of the platform, as well as the potential features added by modules; the core and the modules together determine the emergent properties of the technical system. Users may desire to acquire specific platform configurations because of the functions and performance offered; this determines a profit for the platform developer as well as the module developers. If the market proves to be profitable, more developers join the ecosystem, or supporters decide to increase the variety of modules offered; this in turn increases the function and performances of the platform, thus increasing its attractiveness in the eyes of potential adopters.

On the other hand, if the ecosystem does not offer what the users need, the platform as a whole is not attractive, the module developers cannot cover the development investments and the whole market collapses (Evans and Schmalensee 2010). This scenario can arise either because of poor technical design or because of wrong strategic decisions. In the first case, for example, the core may not be adaptable enough to accommodate for novel modules, thus limiting the creativity of developers; the overall platform may be penalized excessively by the modular architecture, which usually is less performing than an integral one, or technical malfunctioning can arise from particular platform configurations. In the second case, the developer’s community may have offered the wrong set of modules, or the module central to the several successful configurations may have been developed too late.

It is clear therefore that the product feature and the market dynamics are strongly coupled in industry platforms, and a methodology for these technical systems must consider both aspects, with a perspective over the entire product lifecycle. The next section will explore the implications of these characteristics on the design process.
Chapter 3  
Methodological framework

2. Implications for product design and development

The model presented in the previous section highlights the peculiarities of industry platforms; now the effects on product development are examined. In traditional product development, the product is designed, manufactured, tested and finally sold in one market; the product specifications and functionalities are determined before the actual use phase. Even in customizable product platform, the customer is able to choose part of the design, but the set of alternatives has already been determined by developers before he/she starts the choice process; furthermore, the choice usually happen before the product has entered the actual utilization phase. The market determines the success or failure of a product and provides feedbacks for future designs, but it is conceptually separated from the product itself.

On the other hand, industry platforms stay in the middle of two, or more, communities and its value is determined by the exchanges between these communities. In the simplest case, the developers’ community creates the modules in order to make profit, while the users’ community buys modules in order to utilize the platform; the developers determine the functions offered by the platform, while the users determine which modules survive in the ecosystem. The design becomes “open”, in that several agents contribute to the final functions and structure of the platform. Moreover, since external entities contribute to the development of the product, control over the design and responsibilities are distributed. Task coordination, information sharing and mutual trust between module developers and platform developers are essential for the prosperity of the ecosystem and the success of the platform as a whole. Innovation is led by a community, and not a network of developers; moreover, the innovation is inherently “open”. Finally, the developers’ community and the module ecosystem are in constant evolution, not only because new developers enter the community and create novel modules, but also because developers in the community update their modules or expand their offer. These features makes industry platforms very similar to engineering systems (de Weck et al. 2011), with which they share the presence of socio-technical issues, the need for flexibility and the “partially designed – partially evolved” nature.

For these reasons, industry platforms are different from internal and supply-chain platforms, and should be designed differently. While internal platforms are designed by a single entity (or a finite consortium of entities) for a single market (the customers), industry platforms are the sum of the central core design and peripheral modules created by an open community, thus generating a two-sided market. Furthermore, even if product platforms can be customized, they usually cannot be modified after the point of sales; on the other hand, industry platforms change over time according to the customer’s preferences and the available modules on the market. Finally, in product platforms, the integrator has a direct, strict control over the modules requirements and specifications; in industry platforms, the developers’ community is open and less directly controlled.

Given all these features, the design process has to face a series of challenges. The design process has to deal with a very high complexity, which in turn require a high mental effort (the “complicatedness”) for the management and the use of information. Furthermore, the presence of multiple developers can decrease the availability and slow the transfer of fundamental information, requiring a high coordination effort. Interface design and communication protocols have to be defined carefully, since they should allow several modules to work seamlessly. Since the platform ecosystem evolves in time, the platform core must be robust enough to accommodate external innovations, while it should provide easy changeability to customers. Finally, since the number of possible platform configurations increases rapidly with the growth of the module catalogue, system reliability, verification and security are under stress. While these issues are serious challenges for designers, some of them are not completely novel, and have already been addressed in engineering design literature, for example in systems engineering, collaborative design or design for reliability.

Among the many challenges, this research focuses on three critical aspects of cyber-physical industry platform design: providing changeability through a well-designed architecture, understanding customers’ needs and preferences and supporting the most promising developers and modules.

A high degree of changeability during the utilization phase is fundamental to insure that new modules can be added to the industry platform. Users must be able to change their configuration easily and rapidly, so that their experience is as seamless as possible. Chapter 3 describes the Changeability Investigation Technique, a method to anticipate the change propagation behaviour of generic system architectures. The Changeability Investigation Technique, which is supported by an independently developed numerical tool called 8AM800, allows selecting the most changeable architecture during the architecting phase, or highlighting the components most sensitive to change propagation if the platform architecture is fixed.
Once the platform architecture has been decided, module development must be prioritized. Users buy and update the industry platform only if the modules provided in the catalogue satisfy their needs. Consequently, the platform is successful only if the right modules are offered. The platform supplier may or may not be willing to design internally the basic modules required by users; however, it still needs to coordinate the developers’ community so that the “right” modules are provided. Chapter 4 advances a value analysis methodology that can highlight the most valuable platform configurations in the eyes of the user. A new definition of value is proposed, and a stated choice model is employed to rank the configuration catalogue.

As mentioned before, industry platforms and two-sided markets are dynamic, in that the number and type of modules, the developers and the users changes over time. The “static” analysis like the one presented in Chapter 4 therefore is not sufficient to insure the correct initiation and success of the two-sided market and, consequently, of the industry platform. In Chapter 5, a socio-technical numerical analysis based on an agent-based model and innovation diffusion is proposed. Platform supporters can take advantage of the results to evaluate several market scenarios and to understand how to maximize the platform diffusion. The two-sided market analysis considers both social and market aspects, like word-of-mouth, choice preferences or pricing, and strategic design aspects, like platform reliability, modules development and modularity penalties.
Cyber-physical industry platforms must be able to accommodate multiple module changes during their lifecycle, otherwise platform users cannot customize their system and developers cannot profit from their work. Changeability, the lifecycle property that allows systems to change or be changed easily, is central to the concept of industry platforms. This chapter focuses on providing changeability in technical systems through architecture design. It defines change abilities according to the system’s change features in order to reduce the ambiguity of the terminology (Section 1); then it proposes a methodology to predict the likelihood that a certain part of the system behaves as an absorber, a carrier or a multiplier (Section 2). Section 3 introduces a numerical tool that supports the methodology and Section 4 offers an initial validation of the results. Finally, Section 5 advances some potential uses of the methodology and the tool. Chapter 7 and Chapter 8 will then show two practical applications of the analysis proposed in this chapter.

1. A classification for change-related abilities

Section 4.3 in Chapter 2 detailed how different ability frameworks captured distinct aspects of engineering changes. It showed how different frameworks consider change abilities from different perspectives and that often, but not always, the definitions overlaps. Before advancing to the change propagation analysis, a method to define consistently change abilities is proposed. The goals of the method are (1) the harmonization of different perspectives on change abilities and (2) the creation of a solid and clear framework for all the stakeholders involved with the system’s design and operation. Changeability is chosen as an umbrella term that summarizes all change-related abilities; then, five out of seven engineering change features allows separating the different flavors of Changeability. The new definitions, and their classification, should not add further ambiguity, but offer a comprehensive and shared view on change-related abilities.

Similarly to (Fricke and Schulz 2005; Ross et al. 2008; de Weck et al. 2011; de Weck et al. 2012), an umbrella term for all the change abilities is chosen. In literature, three proposals were advanced: Changeability, Flexibility and Adaptability. The choice fell on Changeability, as this term is the most neutral among the three. Changeability summarizes all change abilities, in that represents the ability of a system to be changed with different mechanisms with an acceptable consumption of resources by change agents. The system, the stakeholders and the benefits of a change determines the level of acceptability; clearly, the more contained the resources required for a change are, the more likely the change becomes. Increasing Changeability can lead to two outcomes: either reducing the cost for change for a determined set of changes, or opening new opportunities for change. In this respect, it is important also to consider change propagation: the resources must be computed across the whole system, not just for the local change. For this reason, a changeable system must be able to limit change propagation thanks to its change buffers.

However, changeability alone is too general and it should be declined according to the needs of the stakeholders. As mentioned, literature shows various attempts at creating these frameworks, which remain quite ambiguous. The reason is given by the overlapping of several engineering change features, which can refer to several change abilities at the same time. For example, designers may want to reduce the effort in changing a system blueprint (holon), but have to decide what change mechanism and what buffers to use; the outcome of this decision can lead to different abilities and different strategies, even if the lifecycle phase is the same. For this reason, engineering change features must be ranked according to their importance. The most
important change feature will be called key feature, and it will determine the kind of ility required. The other features are called secondary features, and they help refining the ility required, giving more information about the context.

Figure 4.1: change ilities classification methodology

The method to choose and specify the most suitable ility consists of four steps (Figure 4.1). First, stakeholders need to specify what kind of change they want to ease with the change ility. The seven features of engineering allows the stakeholders to orienteer across the different dimensions of a change. Then, the key feature must be selected, i.e. the feature that characterizes the design problem. The third step consists in translating the key feature into a change ility. Finally, secondary change features must be specified. The key features and the relative ilities are mapped in Table 4.1.

Table 4.1: key features and relative ilities

<table>
<thead>
<tr>
<th>KEY FEATURE</th>
<th>VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffers</td>
<td>Behaviour Interfaces</td>
<td>Extensibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reconfigurability</td>
</tr>
<tr>
<td>Lifecycle phase</td>
<td>Design phase</td>
<td>Flexibility</td>
</tr>
<tr>
<td></td>
<td>Utilization phase</td>
<td>Adaptability</td>
</tr>
<tr>
<td></td>
<td>Redesign phase</td>
<td>Evolvability</td>
</tr>
<tr>
<td>Mechanism</td>
<td>Addition/Subtraction</td>
<td>Scalability</td>
</tr>
<tr>
<td>Agent</td>
<td>Internal (system itself)</td>
<td>Self-changeability</td>
</tr>
<tr>
<td></td>
<td>Customer</td>
<td>Customizability</td>
</tr>
<tr>
<td></td>
<td>Maintenance technician</td>
<td>Maintainability</td>
</tr>
<tr>
<td>Resources</td>
<td>Time</td>
<td>Agility</td>
</tr>
</tbody>
</table>

Buffers are key to define Changeability. Once changes are initiated in a system, they propagate until an interface, a component or parts of the system accommodate the original change and the resulting ones. From the literature review, we can specialize Changeability in Reconfigurability if the buffers are the interfaces, or Extensibility, if the buffer is the system’s behaviour.

The second key feature is the lifecycle phase during which the change happens, which also comprehends the object of change. Several lifecycle schemes have been proposed (Haskins 2006; Pahl et al. 2007; Ulrich and Eppinger 2011) and, in theory, it is possible to associate a change ility for each detailed phase. To avoid excessive complexity, here three main phases are proposed: the design phase, during which the system is developed as a holon and manufactured to become an instance, the utilization phase, during which the system operates as an instance, and the redesign phase, during which information from system instances are used to create a new generation of the system holon. As Figure 4.2 shows, an ility was associated to each lifecycle phase. Flexibility is the ability of a system holon to be changed with an acceptable consumption of monetary resources during the design phase; Adaptability is the ability of a system instance to be changed with an acceptable consumption of monetary resources during the utilization phase. Evolvability is the ability of a system holon to be changed with an acceptable consumption of monetary resources during the redesign phase.
The change mechanism is able to differentiate only one change Ility, Scalability, which is associated with addition/subtraction of something. In all other cases, there is ambiguity regarding which change mechanism is employed in order to achieve a certain lifecycle property.

Fourthly, the agents of change are considered. As noted before, agents in literature are mainly divided into two categories: internal and external to the system. Internal agents mean that the system is able to provide autonomous changes, while there may be several external agents that relates to the system. Just to make two examples, if the agent of change is the customer, the Ility involved is Customizability, while if the agent is a maintenance technician, Maintainability is the Ility desired.

The type of resources considered is the last key feature. As mentioned in the previous sections, literature agrees that the Ility limiting the consumption of time for change is called Agility. Agility designates the ability of a system to be changed in an acceptable amount of time. As a note, it is important not to confuse Agility in the technical system with Agility in the product development process, even though the two are related.

Please note that the initiator of change does not seem to characterize any particular Ility. In literature, in fact, only external changes are considered; how Ilities are influenced by change initiator can be a promising research direction.

If change Ilities were completely independent one to another, the third step would conclude the methodology to choose the desired term. However, an engineering change is characterized by all seven features at the same time; therefore, just stating the key features and one Ility can lead to vagueness and misunderstanding. For example, a change can happen during the Utilization phase (Adaptability), by the hands of a system user (Customizability) whose priority is the quickness of change (Agility). When using Ilities therefore, it is fundamental to state clearly all the change features involved; otherwise, misunderstandings can arise. Specifying consistently the secondary features is the goal of the fourth step, which can be performed with the help of Table 4.2.

---

Figure 4.2: relationship between lifecycle phases, object of change and Ilities
Table 4.2: Standard expressions to define secondary features for Ilities.

<table>
<thead>
<tr>
<th>SECONDARY FEATURES</th>
<th>STANDARD EXPRESSION</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffers</td>
<td>“thanks to changeable…”</td>
<td>interfaces/components/subsystems</td>
</tr>
<tr>
<td>Lifecycle phase</td>
<td>“during…”</td>
<td>Design/utilization/redesign phase</td>
</tr>
<tr>
<td>Mechanism</td>
<td>“via…”, “through…”</td>
<td>Modification/Substitution/Addition/Removal</td>
</tr>
<tr>
<td>Agent</td>
<td>“Self-” or “provided by…”</td>
<td>Customer/Technician/Designer/…</td>
</tr>
<tr>
<td>Initiator</td>
<td>“to deal with changes from…”</td>
<td>The system itself/Requirements/Market/…</td>
</tr>
<tr>
<td>Resources</td>
<td>“limiting…”</td>
<td>… consumption</td>
</tr>
</tbody>
</table>

For every secondary feature, an expression with a standard preposition followed by the appropriate name can be used. Not all secondary features must be stated, as some may be implicit; nevertheless, it is recommended to specify all the relevant change features as often as possible.

Some real-world examples can clarify how to use the method. Let us suppose that a marketing research showed that, for a particular type of competitive bicycles, customers want to be able to regulate the stiffness of the suspensions quickly, so that they can adapt it to different tracks (this is actually the example proposed in the previous paragraph). Assuming that the key feature is the Agent, i.e. the customer, the primary Ility is Customizability. Since we know that two secondary features are the lifecycle phase (utilization phase) and the resource (time), a correct definition of Ility is “agile Customizability during the utilization phase” or “Customizability during the utilization phase limiting time consumption”. On the other hand, if the key feature is the lifecycle phase, the Ility can be named as “agile Adaptability provided by customers.”

Another typical example of changeability can be drawn from the design of power plants. Since the design of such a complex system requires several years, at the beginning of the project, it is not clear what the power output request will be. Therefore, the contractor asks for a change Ility strategy in order to minimize the risks. In this example, the lifecycle phase of changes is the design phase. How to cope with changes in requirements (the initiator of change) depends on the other change features. On one hand, it is possible to add or subtract some modular components, so that the final output is closer to the actual market request; on the other hand, it is possible to modify the power generators so that generate less or more energy. In the first scenario, the Ility is “Scalability during the design phase to deal with changes from requirements”, while in the second case, it’s “Flexibility via modifications to deal with changes from requirements.

An automated optimizer for structural automotive frame design is a good example of self-changeability, i.e. changeability where the agent of change is the system itself. In case of requirements change (the external initiator), the optimization program automatically modifies the dimensions of the beams that constitute the frame. If the system holon is particularly changeable, this change does not generate many propagated changes across the system, thus limiting the time and the money for change (the resources involved). The Ility desired can be defined as “Self-Flexibility via modification to deal with changes from load requirements.”

The final example is about the upgrade of a smartphone to a new version, which usually happens every year. In this case, the focus is on improving the performances of the device in order to keep up with the competition; since the final assembler does not produce the electrical components, the upgrade takes place thanks to the substitution of part of the technical system like the screen or part of the embedded circuits. The lifecycle phase is redesign, the buffers are several subsystems, the mechanism is substitution, the agents are the designers, the initiator is the market; both reduced time for development and limited financial expenses are needed. The Ility required can be defined as “Evolvability via substitution of changeable subsystems limiting both time and money for changes”.

2. The Changeability Investigation Technique

Section 1 compared different change Ilities and provided a consistent framework for the different “flavours” of changeability. One feature is in common to all change Ilities: they must reduce the negative effects of
changes, so that the benefits can be exploited more easily and the risks associated to changes are reduced. However, allowing single changes in a system is necessary but not sufficient. In complex systems, an initial change has often consequences in other parts of the systems and triggers subsequent changes, as highlighted in Chapter 2. Change propagation depends on several project management factors as well as numerous technical features and constraints. Unfortunately, the complexity of technical system usually prevents designers from characterizing every technical relationship between components; therefore, a certain degree of abstraction is required. From an abstract point of view, change propagation appears as a subsequent number of modifications in different areas of the system. Changes can be accepted or not; if accepted, they generally require the consumption of temporal and monetary resources to be implemented. After a certain number of changes, patterns start appearing: some components are changed more frequently, others require high resource consumptions to be changed, and some interfaces tend to propagate changes more frequently than others do. These statistical behaviours are the result of complex interactions between knowledge management, resources available, communication between design groups, project management and other contingent factors. However, changes can propagate from one component to the other only if there is a functional, behavioural or structural interface between them (Hamraz et al. 2014). Therefore, the architecture of the system plays a fundamental role in determining the change propagation behaviour and the arrangement of interfaces can influence the number of changes that are generated during the project. For example, if two change multipliers are interfaced together, the total number of change that propagate through the system is likely to be high; on the other hand, isolating a change multiplier with several change absorbers can decrease the total number of changes. A system architecture that limits change propagation is more changeable than one that increases them, therefore it is the ideal candidate for industry platforms.

In this section, the CIT, a new methodology based on the Changeability Assessment Technique (CAT) by Koh et al. (2013), is introduced. The CAT is an evolution of the change prediction model (Clarkson et al. 2004b) that refines some features, as explained in the following paragraph. Its main asset is the use of the Design Structure Matrix, which allows an easy representation of systems’ architecture easily; furthermore, results of the analyses can be applied to all kind of systems, regardless of their specificities. Furthermore, the effects of change propagation are summarized in three numerical indices, the Outgoing Change Risk (OCR), the Incoming Change Likelihood (ICL) and the Incoming Change Impact (ICI).

The Changeability Assessment Technique consists of five main steps (Figure 4.3). First, the system architecture is mapped in the Design Structure Matrix (DSM). According to (Hamraz et al. 2014), a Multi-Domain Matrix (MDM) representing the interactions between functions, behaviours and structures is more suited to represent change propagation; this methodology can be extended to MDMs without loss of validity. Secondly, collection of the data on potential change in a technical system is required. The authors suggests staff interviews or the analysis of change management documents created by the company. Changes in complex systems can be clustered in two main sets: native and propagated. Native changes are desired or undesired modifications of components in the system; propagated changes are undesired changes arising from previous changes. Native changes can be further differentiated in emergent and initiated changes (Eckert et al. 2004). Initiated changes are generated by changes outside of the system boundary usually but not always associated with a requirements change, while emergent changes derive from problems or mistakes discovered during the design activity. The CAT takes into account these differences, as the indices are derived through numerical combination of four kinds of change propagation properties: the likelihood and impact of native
changes and the likelihood and impact of propagated changes between two related components. Likelihood represents the probability that a change takes place in a component \((L_a)\) or propagates from a component to the other \((L_{a,b})\), while impact indicates “the amount of work required to carry out changes”; impact can refer to the change to one component \((I_a)\) or to a change from a component to another \((I_{a,b})\), as well. The planned changes \(L\) and \(I\) are the desired changes in the design, while the propagated changes \(l\) and \(i\) are undesired changes that arise because of structural or functional connections between components.

Then, the data are organized in two matrices similar to DSM, where the column are the initiating component and the rows are the affected system components. The LDSM contains the likelihood of change from a component to another (off-diagonal elements) and the likelihood of planned changes on components themselves (diagonal elements); in a similar fashion, the IDSM contains the impacts of changes between components and planned changes. Figure 4.4 provides a symbolic example of the two matrices.

![Figure 4.4: example of LDSM IDSM, as represented in (Koh et al. 2012)](image)

The LDSM and the IDSM are therefore an extension of the traditional DSM, where the strength of the links represent likelihood of change in the LDSM and the impact of change in the IDSM. In the remainder of the thesis, these two matrices are called Change Propagation Matrices (CPMs).

The third step is the computation of the change indices. This step, based on the Change Propagation Method, is in turn divided into three stages. In the first stage, the single propagation likelihood and impacts are combined together. Combined change propagation likelihood and combined change propagation impact are computed respectively as:

\[
\begin{align*}
L_{k,j} &= 1 - \prod_{z \in Z} [1 - (l_z \times \alpha_z)] \\
l_z &= (l_{k,k-1} \times l_{k-1,k-2} \times \ldots \times l_{j+1,j}) \\
\alpha_z &= (\alpha_{k,k-1} \times \alpha_{k-1,k-2} \times \ldots \times \alpha_{j+1,j})
\end{align*}
\]

\[
\begin{align*}
I_{k,j} &= 1 - \prod_{z \in Z} [1 - (i_z \times \alpha_z)] \\
i_z &= (i_{k,k-1} \times i_{k-1,k-2} \times \ldots \times i_{j+1,j}) \\
\alpha_z &= (\alpha_{k,k-1} \times \alpha_{k-1,k-2} \times \ldots \times \alpha_{j+1,j})
\end{align*}
\]

Where \(j\) and \(k\) are two components, \(j\) being the change-initiating component, \(k\) the last components subject to change; \(z\) is the change propagation path and \(Z\) is the entire set of possible change propagation paths from the components \(j\) and \(k\). \(\alpha\) is the “change reachability”, a scaling factor that limits the propagation of change, as observed in case studies.
Chapter 4

Architecting changeability

The second stage includes the effect of planned changes to the combined change propagation indices. Refined change propagation likelihood and revised change propagation impact are defined as follows:

\[ L_{a,b}^* = L_b \times L_{a,b} \]  
\[ I_{a,b}^* = I_{a,b} \times I_a \]  

(4.3)  
(4.4)

The revised change propagation risk combines the revised change propagation likelihood and the revised change propagation impact:

\[ R_{a,b}^* = I_{a,b}^* \times L_{a,b}^* \]  

(3.5)

This index represents an overall indicator of danger from change. Higher values indicate that the link between the components is critical for the management of the change in the system.

Finally, the third stage summarizes the behaviour of the component with respect to change propagation in three indicators, the Incoming change likelihood (ICL), the Incoming change impact (ICI) and the Outgoing change risk (OCR). The Incoming change likelihood describes “how likely a system component is expected to be changed due to planned changes and change propagation”. It can be obtained from:

\[ ICL = \frac{\sum I_{row entries}^*}{\# \text{ of components}} \]  

(4.6)

The ICI evaluates the effort required to change a component, considering both planned and propagated changes. Equation 4.7 defines the ICI as:

\[ ICI = \frac{\sum I_{row entries}^*}{\# \text{ of components}} \]  

(4.7)

Finally, the OCR express how the component affects other components as a promoter of change propagation. It can be computed as:

\[ OCR = \frac{\sum R_{column entries}^*}{\# \text{ of components} - 1} \]  

(4.8)

Components with high ICL and ICI are likely and hard to change, while high OCR values indicate that the component’s changes will likely affect other parts if the system. The three indicators designates three different, but equally important, behaviours.

In the final step, assessments and recommendations are made. The authors suggest representing ICL and ICI on a Cartesian diagram in order to compare the behaviour of different components (Figure 4.5). Design decisions can be made based on the indicators, so that the system’s changeability can be improved.
The Changeability Assessment Technique and its predecessors have been tested several times and they proved to be a valuable resource in advising designers about possible change propagation. However, it is based on a set of assumptions:

- The interfaces between components are fixed. This allows the use of a unique DSM to represents all changes (Step 1). In the framework provided in Section 1.1, this means that the method is suitable to compute Changeability by modification or substitution of subsystems;
- The system to be designed (or redesigned) is similar to a previous one. In this case, it is possible to retrieve information about previous change propagation patterns from previous projects (Step 2 of the methodology);
- Since the architecture and the change data are based on a single case study, general observations about design rules to limit change propagation are hard to find. This focus on single case studies can limit the comparison between different projects and can ultimately hinder the development of the research field (Blessing and Chakrabarti 2009).

In other words, the Changeability Assessment Technique, like other analyses in the field (Giffin et al. 2009; Siddiqi et al. 2011), is mainly a descriptive method, in that it describes what happened in previous projects and extrapolates the results for future ones. It does not provide prescriptive indications to limit change propagation and increase changeability.

The goal of the thesis is to provide insights for the design of industry platforms. In this chapter, the focus is on how architecture can improve the changeability of a system reducing change propagation. From the assumptions highlighted before, it is clear that the CAT is not suitable for system architecting. In order to generate prescriptive results, a new methodology, called Changeability Investigation Technique (CIT), has been devised. The Changeability Investigation Technique allows designers to simulate several change scenarios for several system architecture and then compare the statistical distribution of the change indices; thus, it is possible to generate several architectures and change statistics, to compare the general distribution of the indices, and finally to select the most desirable architecture. Thanks to the CIT, detailed information about previous projects are not needed, since general statistical distributions can be used as inputs; furthermore, several architectures can be evaluated and directly compared.
Figure 4.6 shows the five steps of the method. Compared to the Changeability Assessment Technique, steps 3 and 4 remain similar, while steps 1, 2 and 5 are modified. As far as the architecture definition is concerned (Step 1), many architectures can be generated from a list or from automated design tools like (Shougarian 2016). Having several architectures allows performing a trade-space analysis between them or exploring the effects of a particular architectural feature across different architectures. The second step now does not require designers to collect actual change data, since they are generated through a Monte-Carlo process. The likelihood and the impact of native and propagated changes are assigned randomly according to input statistical distributions; clearly, the more accurate are the input statistical distributions, the better are the results obtained. The third and fourth steps are the same as the ones in the CAT, but instead of one sample, several samples are analysed, and statistical distributions of ICLs, ICIs and OCRs are generated. For this reason, in the fifth step statistical techniques like distribution comparisons, correlation analyses and linear regressions are used to compare and select architectures.

Monte-Carlo simulations require a software tool that automates the generation and evaluation of the data. Section 3 will detail the functionalities and the structure of this tool.

3. 8AM800, a numerical tool for change propagation

The tool supporting the Changeability Investigation Technique is 8AM800, a modular suite of algorithms for the generation of architectures, the management of change propagation data and the synthesis of change propagation indices. 8AM800 is programmed in MATLAB™; in its latest version (v 3.51), it consists of a main file and three sub-parts, which correspond to the first four parts of the CIT (Figure 4.7).

In the program main file, the input variables and the type of computations required are selected. The main file is also responsible for calling the other modules and saving of the results. In the first sub-part, the architectures are either imported from an external file or generated according to some requirements. In the first case, a list of DSM must be provided so that the software can convert the information into a suitable matrix; in the second case, more parameters must be provided to the internal architecture generator. The second sub-part generates the change data from the input statistical distributions and creates the change propagation matrices. Finally, the third sub-part computes the change propagation indices and other relevant architectural properties, if required by the user. On a machine equipped with third-generation i7 processor and 16 GB of RAM, the algorithm is able to conclude one complete set of computations (from inputs to outputs) in 2.8 s for a DSM with 40 elements; the computational time grows exponentially with the number of subsystems.
3.1 Architectures generation

The architecture generator is a module inside 8AM800 whose function is the creation of random and feasible (i.e. “synthetic”) DSM matrices that respect the constraints imposed by the main file. Other architecture generation methods and tools can be found in literature, for example (Zeidner et al. 2010; Selva 2012; Shougarian 2016), but they need a high characterization of the system to generate the architectures. The 8AM800 architecture generator can create very abstract systems, whose only requirement is the complete connectedness of the network. This high level of abstraction is required only for broad analyses of the architectural features’ influence on the change propagation indices, like in the application in Chapter 7. If the set of feasible architecture is available, 8AM800 can utilize them and begin from the second step, the Change Propagation Matrices Generator.

The architecture generator is composed by two parts. In the first parts, DSM moulds, called “meta-DSMs” are created; in the second part, synthetic DSMs are derived as samples from meta-DSMs. Meta-DSM are matrices that represent a class of feasible architectures. While in DSMs the extra-diagonal elements represent interfaces, in meta-DSMs they represent the likelihood that an interface is present. Therefore, meta-DSM are able to describe a set of possible architectures that respect some constraints imposed by the user in the main file, therefore they can be associated to a “distribution” of possible architectures; synthetic DSM are feasible architectures created from the meta-DSMs, and can be associated to single samples from a distribution. Figure 4.8 shows a meta-DSM and an associated synthetic DSM.

![Figure 4.8: example of a meta-DSM and the associated synthetic DSM](image)

In all meta-DSMs, the elements on the diagonal are null. The elements outside the diagonal represent the likelihood that an interface is present. For example, an interface between subsystem A and subsystem B has the 50% chance to be present, while the interface between D and B exists with a probability of 20%. All meta-DSM are upper diagonal matrices because it is considered that every change is a mutual interaction between subsystems and therefore all DSM are symmetric. If a change can travel from A to B, it can also travel from B to A. This is different from saying that interfaces are bi-directional; fluxes of material, information or energy can have a direction, but changes are not necessarily limited to travel in the direction of the flux. Furthermore, the assumption of symmetry in the DSM does not imply that the LDSM or the IDSM are symmetrical; a change may be more likely or more impactful when it travels in one specific direction.

The synthetic DSM in Figure 3.1.5 is derived from its meta-DSM through a Monte-Carlo process. For each interface, a random number from a uniform distribution is generated by the software; if the number if minor or equal to the interface likelihood, the synthetic DSM gains that interface. For example, in the case shown in
Figure 3.1.5, the interface between A and B was created because a random number smaller than 0.5 was generated; on the other hand, the random number drawn for the interface A to C was greater than 0.2.

The dimensions and content of the meta-DSM depend on the inputs chosen in the main file. The latest version of the generator allows controlling the following architectural features:

- Number of subsystems
- Average density of interfaces
- Number of modules
- Number of subsystems in each module
- Number of bus elements
- Average number of bus elements’ interfaces

The number of subsystems corresponds to the number of elements in the DSM. According to the granularity of the system decomposition, the same system may have a larger or smaller representation. Finding the correct level of characterization is a challenge for every research about change propagation modelling (Chiriac et al. 2011).

Density of interfaces is the average number of interfaces divided by the total number of possible interfaces. Technical systems are usually sparse (Simon 1996); in examples drawn from literature, density varies between 1/3 and 0.07 (Clarkson et al. 2004c; Weck 2007; Yu et al. 2007; Suk Suh et al. 2011; Engel and Reich 2012; Eppinger and Browning 2012; Koh et al. 2013).

Modules can be defined according to different interpretations (Gershenson et al. 2003; Hölttä-Otto et al. 2012). In this research, modules are areas where meta-DSM have higher density of interfaces with respect to the rest of the system. This is consistent with (Hölttä-Otto et al. 2012), which defines a module as “a relatively independent chunk of a system that is loosely coupled to the rest of the system.” Synthetic DSMs are more likely to have an interface inside a module than an interface outside a module because modules in meta-DSMs have a higher interface probability.

Bus elements are subsystems that are connected to a very large number of other subsystems. They are key elements in collecting and distributing the materials, energy and information fluxes that run through a technical system. As for modules, bus elements are characterized as subsystems with a higher-than-average interface likelihood; however, bus elements are single subsystems that are interfaces to other subsystems, while modules are a group of subsystems interconnected one to another. Figure 4.9 shows a graphical difference between the two.
Interface probabilities, number of subsystems, average density, size and number of modules and bus elements are not independent. In fact, density is defined as:

\[
\text{Density} = \frac{N_{\text{int}}}{N^2_{\text{sub}} - N_{\text{sub}}} = \frac{N_{\text{int,mod}} + N_{\text{int,bus}} + N_{\text{int,out}}}{N^2_{\text{sub}} - N_{\text{sub}}} \tag{4.9}
\]

Where \(N_{\text{int}}\) is the number of interfaces in the system, \(N_{\text{sub}}\) is the number of subsystems, \(N_{\text{int,mod}}\) is the number of interfaces inside modules, \(N_{\text{int,bus}}\) is the number of interfaces inside bus elements and \(N_{\text{int,out}}\) is the number of interfaces in the rest of the system. If we assume that the ratio between average density inside and outside modules (or bus elements), a form coefficient \(\lambda\) can be defined as:

\[
\lambda_{\text{mod}} = \frac{p_{\text{mod}}}{\theta} \tag{4.10}
\]

\[
\lambda_{\text{bus}} = \frac{p_{\text{sub}}}{\theta} \tag{4.11}
\]

\(\lambda_{\text{mod}}\) is the form coefficient for subsystems inside modules, while \(\lambda_{\text{bus}}\) is the form coefficients for bus elements. \(P_{\text{mod}}\) is the interface probability inside modules, \(P_{\text{sub}}\) is the interface probability inside bus elements and \(\theta\) is a scaling factor that allows the equations to respect the constraint on average density.

From Eq. 4.9, Eq. 4.10 and Eq. 4.11, it is possible to show that for a given set of form coefficients and a target average density, the scaling factor \(\theta\) must be

\[
\theta = \sqrt{\frac{N^2_{\text{sub}} - N_{\text{sub}}}{\text{Density}}} \left( \sum_{i=1}^{N_{\text{mod}}} N^2_{\text{sub,mod},i} - N_{\text{sub,mod},i} \right) \tag{4.12}
\]

\(\text{Size}_{\text{mod}}\) is the size of all modules, which depends on \(N_{\text{sub,mod},i}\), the number of subsystems inside the \(i\)-th module; \(\text{Size}_{\text{bus}}\) is the size of all bus elements, which can vary if the bus is potentially connected to the entire system or just a subset. \(N_{\text{mod}}\) is the number of modules in the system and \(N_{\text{bus}}\) is the number of bus elements in the system.

The architecture generator takes the inputs from the main file’s design of experiments, then computes the scaling factor \(\theta\) and generates the meta-DSM. Finally, it generates a certain number of synthetic DSMs, as specified in the main file. These synthetic matrices are passed on to the next module, the Change Propagation Matrices generator

### 3.2 Change Propagation Matrices generator

Once the synthetic DSMs have been created, they are passed to the CPMs generator. This sub-part assigns the native change likelihood \(L_i\) and the native change impact \(I_i\) to all components, and the change propagation likelihood \(L_{ij}\) and impact \(I_{ij}\) for each interface. Each parameter is assigned randomly through a Monte-Carlo process.

How to assign these values is one of the most critical aspect of the entire Changeability Investigation Technique. In this work, it is assumed that change parameters either follow a uniform distribution, or a Gaussian distribution. Section 3 compares synthetic CPMs derived from uniform distributions with a CPM obtained from a previous case study (Giffin et al. 2009) and finds a high correlation between ICL and ICI; however, only a comprehensive analysis of change propagation data from different projects can determine the most appropriate distribution.

Thanks to the computational power offered by modern computers, several CPMs can be derived from the same synthetic DSM. This insures that the indices obtained cover a wide range of change scenarios, thus providing robustness to the results. An empirical evaluation clarified that 12 CPMs samples from a single synthetic DSM are enough to stabilize to limit the sampling bias.
<table>
<thead>
<tr>
<th>INPUT NAME</th>
<th>TYPE</th>
<th>PART</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>root folder</td>
<td>String</td>
<td>Main</td>
<td>Path to the 8AM800 folder</td>
</tr>
<tr>
<td>run number</td>
<td>Integer</td>
<td>Main</td>
<td>Simulation identification number</td>
</tr>
<tr>
<td>n_elem</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Number of elements in the architecture</td>
</tr>
<tr>
<td>density</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Average density of the synthetic DSMs</td>
</tr>
<tr>
<td>tot_elem_mod</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Total number of elements in modules</td>
</tr>
<tr>
<td>pr_mod</td>
<td>Binary</td>
<td>Arch. generator</td>
<td>Presence of modules in the architecture</td>
</tr>
<tr>
<td>elem_mod_tab</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Number of elements in each module</td>
</tr>
<tr>
<td>prop_mod</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Proportion of modules sizes if modules have different sizes</td>
</tr>
<tr>
<td>pr_bus</td>
<td>Binary</td>
<td>Arch. generator</td>
<td>Presence of bus elements</td>
</tr>
<tr>
<td>n_bus</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Number of bus elements in the architecture</td>
</tr>
<tr>
<td>dim_bus</td>
<td>Vector</td>
<td>Arch. generator</td>
<td>Dimension of bus elements in the architecture</td>
</tr>
<tr>
<td>lambda_mod</td>
<td>Scalar</td>
<td>Arch. generator</td>
<td>(\lambda_{mod}) (see Eq. 3.10)</td>
</tr>
<tr>
<td>lambda_bus</td>
<td>Scalar</td>
<td>Arch. generator</td>
<td>(\lambda_{bus}) (see Eq. 3.11)</td>
</tr>
<tr>
<td>n_DSM</td>
<td>Integer</td>
<td>Arch. generator</td>
<td>Number of synthetic DSMs generated from a single meta-DSM</td>
</tr>
<tr>
<td>n_CPM</td>
<td>Integer</td>
<td>CPM generator</td>
<td>Number of CPMs generated from a single synthetic DSM</td>
</tr>
</tbody>
</table>

3.3 Change indices evaluator
As mentioned in Section 3, the Changeability Investigation Technique means to suggest the most changeable architectures thanks to three Change Propagation Indices: the Incoming Change Likelihood (Eq. 3.6), the Incoming Change Impact (Eq. 3.7) and the Outgoing Change Risk (Eq. 3.8). This section of the software computes these three indices from the CPMs thanks to the equations explained in Section 2.

3.4 Main file
The main file is responsible for collecting the input parameters and calling the three different sub-parts detailed before. Table 3.2 shows the inputs required to run 8AM800. As it can be noticed, most of the input variables relates to the architecture generator; if a set of architectures is imported from the outside, they are no longer needed.

4. An initial validation
The approach presented in this chapter needs some form of empirical validation. In fact, while the prediction accuracy and the usefulness of CAT has already been demonstrated in previous work, the CIT is based on assumptions that need to be validated. In particular, it is not clear if a random assignment of change likelihood and impact can generate meaningful change propagation patterns. For this reason, this next section will provide a comparison of the numerical results with a real case study.

The Changeability Investigation Technique has been applied to a real case study, based on a previous complex technical system’s development project (Giffin et al. 2009). The project involved the design of a large-scale sensor system and took about eight years to complete. The entire change database contained more than 41500 change records in various subsystems, including hardware, software and documentation. Due to the heterogeneity of the subsystems, we will use the word “area” instead of “component” to designate an element in the system at the current level of hierarchical decomposition. The software was the area where most of the changes occurred, since the hardware was mostly re-used from a prior design. For every change, data about the type of change (native or propagated), parent-child relation, area affected and magnitude of change were recorded.
In order to reduce the scope and simplify the exposition of the results, out of the 41,500 changes a small subset of 87 changes was considered (Figure 4.10). They belong to a critical and well-studied group of changes that occurred during the integration and verification phase and affected 12 areas out of 46 (26% of the entire system). The small subset of the architecture consists of highly coupled areas (the average density of interfaces is 0.22) with one bus element connected to nine other areas.

Figure 3.7 shows the 87 changes, organized by affected area. Most of the changes occurred in areas 1, 3 and 19, where also the major part of the native changes arose. Some areas, namely 11, 23 and 35, did not experience propagated changes, while others were subject only to propagated changes. In this particular example, the algorithm developed for this work did not generate a random DSM from similar architectural features, but it randomly created 300 CPMs based on the given architecture. Furthermore, the native changes likelihood and impact vector were generated based on the actual distribution of the real native changes; propagated changes were assigned randomly from a uniform distribution. The resulting ICL and ICI indices were then mediated and compared to aggregated measures in the dataset, as reported in Table 3.3. It is therefore assumed that the average value of the ICL and ICI distributions leads to the most likely result, as guaranteed by the law of large numbers.

The comparison was carried out by means of Pearson correlation index; furthermore, the no-correlation hypothesis was tested. Three change measures from the real data were compared to the change propagation indices obtained numerically. Change likelihood is the number of changes in the area divided by the total number of changes; change impact is the sum of the change magnitudes in the area divided by the sum of the magnitudes in all the areas.

Table 4.4 shows a strong correlation between the ICL and ICI on one side and change likelihood and change impact on the other, as also confirmed by the small p-values. However, the indices’ prediction accuracy showed to be sensitive to the initial distribution of native changes. In predicting the future changes, it is therefore important to quantify the uncertainty surrounding the systems, converting as many “unknown unknowns” as possible into “known unknowns” (McManus and Hastings 2006).
Table 4.4. Correlation analysis between numerical data and case study statistics

<table>
<thead>
<tr>
<th>SYNTHETIC NUMERICAL INDICES</th>
<th>STATISTICS FROM CASE STUDY</th>
<th>PEARSON COEFFICIENT</th>
<th>P-VAL FOR NULL CORRELATION TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICL</td>
<td>Average number of changes</td>
<td>0.8</td>
<td>0.002</td>
</tr>
<tr>
<td>ICI</td>
<td>Average impact of changes</td>
<td>0.73</td>
<td>0.008</td>
</tr>
</tbody>
</table>

While the proposed correlation cannot be considered an exhaustive validation of the results from the numerical algorithm, the presence of significant correlation between the computed indices and the change indicators from the dataset is considered sufficient as an initial verification. Deeper analyses and characterizations will take place in future works.

5. Potential applications

The Changeability Investigation Technique, and the computational tool associated, can provide a fundamental support during the system architecting phase of a project. Depending on the specific need, CIT can suggest what areas of the design space are worth developing, can help choosing between different architecture proposals and/or can highlight the most critical components in a specific architecture.

If the system architect has a certain degree of freedom in the conceptual design of the technical system, he may be interested in understanding what kind of architectural features provides higher changeability. Typical questions concern the optimal degree of modularity in the system, or the centralisation level for some functions (Holtta-Otto and de Weck 2007; Selva 2012). In this case, 8AM800 is able to generate a wide range of synthetic DSM in order to cover several general architectures and evaluate the effects of modules and bus elements on change propagation. This analysis is performed in Chapter 7, where the influence of several architecture features on change propagation is investigated.

Usually, the design of a system is constrained by previous projects, technology requirements or feasibility rules. In this case, either the concept development is carried out informally, or sophisticated architecture generators are employed (Zeidner et al. 2010; Selva 2012; Shougarian 2016). In both case, the architectures can be easily translated in DSMs and imported inside 8AM800, which can evaluate each architecture in terms of change propagation indices. Global changeability indices like the average or the median value of OCR or ICI can be employed in a tradespace analyses in order to compare changeability against other features, like costs or complexity.

Finally, if the architecture has already been frozen, the CIT can offer the Change Management team an analysis similar to the CAT. Since the DSM is fixed, several CPMs can be derived stochastically, and each subsystem’s indices distribution can provide useful information about the expected change propagation behaviour. In this way, the technical analysis of change propagation can influence both managerial decisions and detailed design specifications, and even create positive synergies between the two. For example, if a certain subsystem shows high level of ICL in all the simulated change propagation scenarios, it will be likely changed several times during the project. On one hand, the change management unit can focus on this subsystem, actively tracking all changes that involves it; on the other, the design team can think of flexible detailed solutions or postpone as much as possible the design freeze. The combined effort can lead to a change-absorbing behaviour, even though the subsystem is subject to several changes. This kind of analysis was performed for Project ARA, a customizable modular smartphone, in Chapter 8.

One of the main assumptions in both CAT and CIT regards the interface layout, which cannot be changed. This assumption is generally violated during the system lifecycle, since interfaces layout is often modified due to foreseen and unforeseen developments. However, the methodology can also be apply in a steady-state approximation: every time an interface is added or removed, a new set of simulations can be run, as if a new system had been created. This steady-state approximation can be useful for quantifying the effects of a new interface layout.
Chapter 5

Selecting valuable platform configurations

Chapter 4 provided a method to assess the changeability of potential architectures for cyber-physical platforms. An industry platform open to change can potentially have a very large variety of customizable configurations, depending on the size of the developers’ community and the diversity of the modules. While in principle this variety is positive and desired, in practice customization is both an opportunity and a challenge.

Customization in industry platforms allows to enter different market niches while preserving economies of scale and obtaining economies of scope (Meyer and Lehnerd 1997); furthermore, the customization process can be highly enjoyable for customers, thus increasing the perceived value of a product (Franke and Piller 2004; Franke and Schreier 2010). However, platforms are complex systems that require a complex design process (Lindemann et al. 2009; Sinha and de Weck 2012) and high front-end investments (Cameron and Crawley 2014); variety require a more sophisticated logistics and a proactive engagement with the market (Wang et al. 2007; Li 2009). Furthermore, the paradox of choice (Schwartz 2004; Piasecki and Hanna 2011) states that the more choice a customer has, the less satisfied he may be, and choosing over an enormous set of options can be burdensome and tedious, or even intimidating. This can result in a counterintuitive situation where the producer has to deal with high costs due to customization, while the customer is dissatisfied.

For the reasons above, the conceptual design of industry platforms and the selection of platform configurations to be offered on the market is not trivial. The costs incurred to create, sustain or use a platform might not be worth the customization benefits: it must be noted that highly modular systems generally show lower performance levels compared to integral ones (Ulrich 1995). This effect, known as modularity penalty, can be attributed to two factors: (1) modular products cannot be optimized as a whole and (2) modules needs resources that cannot be shared in the system as a whole. Moreover, the development of modules must match customer’s preferences, so that the overall platform is more attractive to the market and variety does not increase complexity in vain. Finally, customers often must be guided through the choice process, for example using bundle strategies or configurators (Trentin et al. 2013): knowing the most appreciated configurations can help reducing the choice space.

This chapter introduces a definition of value that is consistent with stated choice models and value engineering; it shows some boundary conditions for the modularity penalty and provides a method to rank platform configurations according to the value they provide to customers. The use of this methodology can bring many advantages. First, the value of a potential platform can be measured against integral products on the market, so that the worth of a platform strategy is assessed before making relevant investments in platform development, as observed in (Cameron and Crawley 2014). Then, after highlighting the most valuable modules combinations, product developers can prioritize modules design and a suitable variety can be offered. Moreover, thanks to the customers-based definition of value, traditional strategies to reduce customers’ burden like bundles can be derived more intuitively. The standard value analysis methodology is tailored to meet the specific needs of platform design. As such, the value analysis presents two peculiarities: it is customer-centered, in that analyses the value provided to customers, and it is applied to product platforms. The research questions of this Chapter are “How to compute customer-centred value for customizable product platforms?” and “How to apply a value analysis to customizable product platforms to select the most valuable
configurations?" In answering these questions, I also follow the suggestions provided in (Simpson et al. 2014), which encourages the use of customer-perceived value instead of cost as an objective in platform design methodologies.

1. Value in customizable platforms

1.1 A definition of value

Conjoint analysis is a well-developed methodology to infer customers’ preferences through questionnaires. In order to quantify these preferences, a mathematical utility model must be employed, which is usually (but not necessary) based on the linear combination of part-worth utilities $b_i$ and product features $u_i$ (Ben-Akiva and Lerman 1985; Rao 2014):

$$ v_{j}^{(h)} = \sum_{i=1}^{N_j} b_{i}^{(h)} u_{i} $$  \hspace{1cm} (5.1)

Where $v_{j}^{(h)}$ is the utility of the product $j$ according to customer $h$, $b_{i}^{(h)}$ are numeric coefficients and $u_{i}$ are dummy variables indicating if the i-th feature among the $N_j$ features is present in the product $j$. It must be noted that price sensitivity is included inside the $b_i$ coefficients. Anyway, several studies (for example (Han et al. 2001)) concluded that price sensitivity is not linear, but it presents thresholds. A more generalized model of utility therefore is:

$$ v_{j}^{(h)} = \sum_{i=1}^{N_j-1} b_{i}^{(h)} u_{i} + u_{i}^{c} \left( \sum_{w} c_{w} \right) $$  \hspace{1cm} (5.2)

Where $u_{i}^{c}$ is the non-linear contribution to utility $v_{j}^{(h)}$ as a function of the sum of the costs of the w components. Independently from its form, utility can be employed to compute the probability that consumer $h$ chooses a certain product among a set of similar products $k$:

$$ P_{a}^{h} = \frac{\exp(v_{a}^{(h)})}{\sum_{k \in M} \exp(v_{k}^{(h)})} $$  \hspace{1cm} (5.3)

Where $P_{i}$ is the probability of choosing the product $i$, $v_{i}^{(h)}$ is the utility given by the customer $h$ to the product $i$ inside a set of choices $M$. Equation (5.3) expresses the probability of choice according to the logit model, a statistical choice model that derives from a logistic distribution (Chandukala et al. 2007).

On the other hand, value in Value engineering is defined as:

$$ V_{j}^{(h)} = \frac{\text{Benefits}(j)}{\text{Costs}(j)} $$  \hspace{1cm} (5.4)

$V_{j}^{(h)}$ being the value given by a customer $h$ to the product $j$. Benefits are generic positive functions provided by the product and costs are a measure of the resource consumptions in order to provide the benefits. There are two possible interpretations of costs in this formulation: (1) costs could be the costs incurred by the product producer in order to provide the benefits or (2) costs could be the price that a user has to pay in order to benefit from the product. Both interpretations are allowed, as they give different perspectives on the same problem: the cost for the producer gives the value, as the product is part of the firm’s system, while the cost for the consumer gives the value as the product is outside of the firm. In order to be consistent with the utility
definition, costs must be the final price paid by customers: in other words, this work is based on a customer-centred view of value. This choice has also the advantage to include part of the pricing strategy inside the framework of the value analysis.

Utility and value seem to be two different answers for the same questions. However, it is possible to bridge the gap between the two fields using an exponential transformation in Eq. 5.2 and considering price sensitivity instead of costs in Eq. 5.4. The result is:

$$\exp(v_j^{(h)}) = \exp\left(\sum_{i=1}^{N-1} b_{ij}^{(h)} u_i + u_c^{(h)} (\sum_u c_u)\right) = \frac{\prod_i \exp(b_{ij}^{(h)} u_i)}{\exp(-u_c^{(h)} (\sum_u c_u))} = v_j^{(h)}$$

(5.5)

Where the numerator is the positive utility provided by the product’s features (the benefits’ utility) and the denominator is the negative utility given by price (the price utility). This definition of value, based on the logit model, will be called Logit value.

Eq. 5.5 is the key to convert the results obtained in a conjoint analysis into a mathematical formulation of value. This formulation of value presents two main advantages: first, it gives a solid theoretical background to value analysis, since Logit value is not derived from designers’ subjective formulations but from a structured analysis of stakeholder’s preferences; furthermore; gives a statistical meaning to value. In fact, the logit model in Eq. (5.3) can now be written as:

$$P_a = \frac{v_a^{(h)}}{\sum_{k \in M} v_k^{(h)}} \quad M \subseteq \{a, b, c, \ldots\}$$

(5.6)

In other words, the probability of choosing a product from a set is given by the ratio between its Logit value and the sum of the Logit values of all the products present in the set.

Finally, a further distinction between the benefits provided by a product must be highlighted. In fact, benefits arise generally from positive functions that bring positive effects with a certain degree of performance or satisfaction. If these two aspects are separated, Eq. 5.5 becomes:

$$V_j^{(h)} = \frac{\prod_i \exp(b_{ij}^{(h)} u_i) \exp(b_{ij}^{(h)} v_i)}{\exp(-u_c^{(h)} (\sum_u c_u))} = \frac{\prod_i U_{F,j}^{(h)} P_i^{(h)}}{U_c^{(h)} (\sum_u c_u)}$$

(5.7)

Where $b_{ij}^{(h)}$ are the part-worth utilities related to functions delivered by the product, while $b_{ij}^{(h)}$ are the part-worth utilities associated to each function’s level of performance.

While the definition of Logit value can be applied to any kind of product, modular platforms Logit value $V_j^{(h)}$ can be at least differentiated into five specific factors:

- The value of primary and secondary functions provided by single modules. Purely modular systems present a one-to-one mapping between modules and main functions, but modular platforms usually show a combination of integral and modular aspects (Ulrich 1995; Ulrich and Eppinger 2011);
- The value of primary and secondary emergent functions, i.e. functions cannot be attributed to a single module (Corning 2002; Crawley et al. 2015);
- Other intangible factors like branding or prestige (Lassar et al. 1995; Vigneron and Johnson 1999);
- The value of customizability (Bharadwaj et al. 2009; Franke et al. 2009);
- The value of product differentiation and uniqueness (Tian et al. 2001; Franke and Schreier 2008; Cheema and Kaikati 2010; Liang and He 2012);

Equation 5.7 therefore can be further differentiated into
Chapter 5

Selecting valuable platform configurations

\[ V_j^{(h)} = \prod_i \left( \frac{U_{F,i}^{(h)} p_i^{(h)}_{md}}{U_c^{(h)} \left( \sum_n c_{w,md} + c_{plat} \right)} \right) \frac{V_0 V_{cust} V_{uniq}}{U_c^{(h)} \left( \sum_n c_{w,md} + c_{plat} \right)} \]  

(5.8)

Where the \( md \) subscript refers to modules, \( emerg \) refers to emergent properties, \( V_0 \) is the value of intangible factors, \( V_{cust} \) is the value given by customizability and \( V_{uniq} \) is the value provided by uniqueness. The price utility in the denominator now takes into account the price for single modules \( c_{w,md} \) and the price of the platform core \( c_{plat} \).

Integral products (Ulrich 1995; Cunningham 1996) have complex function-parts mapping and are limited in customizability and differentiation; on the other hand, they tend to exhibit higher levels of performance (Whitney 1996). In the next sub-section, we apply Logit value to compare modular platforms and integral systems.

1.2 The customizability space

Modular platforms differ from integral systems in two main aspects. The first one is modular changeability during the utilization phase: thanks to a high degree of modularity and careful interface design, it is possible to create systems whose modules can be substituted with low effort. At the same time, imposing interchangeability and standardization requirements on interfaces may lead to a decrease in local performances. A second difference concerns emergent properties. Since a single platform must support several combinations of modules, it is hard to optimize system-level properties for all configurations: a compromise between various performances has to be found (Ulrich 1995). Integral technical systems, on the other hand, generally show a lower degree of modularity and can be designed as a whole: for these reasons, they guarantee higher local and system-level performances (Hölttä et al. 2005).

When choosing between a platform and an integral architecture, therefore, managers and designers need to make a trade-off between increased flexibility and a reduction in performances. The effect of a performance reduction on Logit value ca be described as:

\[ P_{i,mod}^{(h)} = \Delta_{\varsigma} (P_{i,int}^{(h)}) \]  

(5.9)

Where \( P_{i,mod}^{(h)} \) is the contribution to Logit value given by the performances in modular platforms, while \( P_{i,int}^{(h)} \) is the Logit value given by performances if the same function is performed by an integral system.

Performances are not the only factor affected by modularity; cost can vary too. On one hand, product platforms allow modules to be installed on more than one typology of product, generating product families and increasing the economies of scope and the economies of scale (Simpson et al. 2014); on the other, modular products cannot be optimized as a whole, thus increasing the costs for single components. Furthermore, since costs here are meant as costs for customers, they can also be influenced by the pricing strategy. In general, it is possible to write that:

\[ c_{i,mod} = \Delta_{\varsigma} (c_{i,int}) \]  

(5.10)

\( c_{i,mod} \) being the cost of a component as a module, \( c_{i,int} \) being the costs of the same component inside an integral architecture and \( \Delta_{\varsigma} \) being the function that converts the first into the second. Assuming linear or linearized relationships, Eq. 5.9 and Eq. 5.10 become:

\[ P_{i,mod}^{(h)} = (P_{i,int}^{(h)})^{\delta_{\varsigma,\eta}} \]  

(5.11)

\[ c_{i,mod} = \delta_{\varsigma,\eta} (c_{i,int}) \]  

(5.12)
Where $\delta_{p,i}$ and $\delta_{c,i}$ are the proportionality constants between modular and integral performances and modular and integral costs, respectively. $\delta_{p,i}$ will be called modularity performance ratio, while $\delta_{c,i}$ will be addressed as modularity cost ratio. Any $\delta_p$ can vary between 0 and 1, while $\delta_c$ can in theory assume values in the interval $(0; +\infty)$. The Logit value of a modular platform, expressed in comparison to an equivalent integral product, can be written as:

$$ V^{(h)}_{j,mod} = \frac{\prod_j U^{(h)}_{p,j} (P^{(h)}_i)^{\delta_{p,i}}}{U^{(h)}_c (\sum_w c_w)^{\delta_{c,w}}} $$

(5.13)

Product platforms allow a certain degree of freedom in the choice of the functions and performances provided, thanks to a combination of commonality and variety (Thevenot and Simpson 2006). Given Eq. 5.13, it is clear that this customizability is paid in terms of lower performances (the modularity performance ratio) that can or cannot be balanced by differences in costs (the modularity cost ratio).

A question follows: when is a customizable product platform more convenient than an integral product? In other words, when do the advantages of modularity overcome its drawbacks? The customizability space is defined as the subset of the design space where the Logit value of a modular product is equal or greater to a similar integral product. Given a reference integral architecture $V^{(h)}_{i,int}$ and a set of platforms with different module combinations, the customizability space can be found solving the following equation for all the possible $V^{(h)}_{j,mod}$:

$$ V^{(h)}_{i,int} - V^{(h)}_{j,mod} \leq 0 $$

(5.14)

The customizability space allows answering to make clear when a platform design is more convenient than an integral one. Section 2 will provide a mathematical analysis for two strategies that can bring a product platform inside the customizability space.

2. Simplified strategies for platform value enhancement

Among the infinite number of strategies to generate platforms inside the customizability space, two of them can be regarded as elementary:

- Benefits plus strategy: the objective of this strategy is to compensate the decrease in performances with novel and attractive functionalities.
- Costs minus strategy: functionalities bringing low value are removed from the customized product platform, so that the overall product has fewer benefits, but a significantly reduced cost.

In reality, customers personalize their customizable platform adding some features and removing other, thus combining the two strategies presented above. Customers’ Logit value is maximized when the functions provided by the product generate the maximum benefits at the minimum cost. This subsection will show how to derive the customizability space from Eq. 5.8, Eq. 5.13 and Eq. 5.14 after some simplifying assumptions.

2.1 The Benefits plus strategy

In the case of Benefits plus strategy, it is assumed that the customizable platform is able to offer more functions than the integral one. First, a reference integral system is taken as reference; its Logit value is:
Chapter 5

Selecting valuable platform configurations

\[ V^{[b]}_{i,\text{int}} = \prod_{j} \frac{U^{[b]}_{F,j} P^{[b]}_{i}}{U^{[b]}_{c} \left( \sum_{w} c_{w} \right)} \] (5.15)

Which is the same as Eq. 5.8. A customizable platform designed with a Benefits plus strategy must have a Logit value:

\[ V^{[b]}_{j,\text{pl}} = \frac{\prod_{i} U^{[b]}_{F,j} \left( P^{[b]}_{i} \right)^{\delta_{p,i}} \prod_{k} U^{[b]}_{F,k} \left( P^{[b]}_{k} \right)^{\delta_{p,k}}}{U^{[b]}_{c} \left( \sum_{w} \delta_{c,w} c_{w} + \sum_{l} \delta_{c,f} c_{l} \right)} \] (5.16)

\[ V^{[h]}_{j} \], the Logit value of the customizable platform according to customer \( h \), is a combination of the benefits from functions and performance present in the reference integral system (indicated with index \( i \)) and the benefits from novel functionalities (indicated with the index \( k \)). New functionalities bring new costs; therefore, the denominator expresses the price sensitivity of two terms: costs associated to the \( i \)-th functionality are indicated with the letter \( w \), while costs due to novel functionalities are designated by the index \( l \).

Combining the previous equations, the benefits required to bring the customized platform inside the customizability space are:

\[ \prod_{k} U^{[b]}_{F,k} P^{[b]}_{k} \geq \frac{U^{[b]}_{c} \left( \sum_{w} \delta_{c,w} c_{w} + \sum_{l} \delta_{c,f} c_{l} \right)}{U^{[b]}_{c} \left( \sum_{w} c_{w} \right)} \prod_{i} U^{[b]}_{F,i} P^{[b]}_{i} \] (5.17)

Equation 5.17 allows some insights on Logit value. First, the role of the performance decrease ratio \( \delta_{p} \) are essential: if they are close to one, the effectiveness of new features depends just on the ratio between price sensitivities, and therefore on the pricing strategy. Secondly, it is harder to introduce new functionalities that affect the customers’ choice if the integral product has already a high level of benefits or a low cost and the performance decrease ratio is lower than one. Finally, an increase in costs for new functions does not automatically lead to lower Logit value: the effects is mediated by the price sensitivity function \( U_{c} \). If \( U_{c} \) is very steep from \( \sum_{w} c_{w} \) to \( \sum_{w} \delta_{c,w} c_{w} + \sum_{l} \delta_{c,f} c_{l} \), new performances must be very appreciated to compensate for the cost increase; on the contrary, if in the mentioned interval \( U_{c} \) is almost flat, new functionalities easily lead the customizable platform into the customizability space.

Further discussions can be triggered if some simplifying assumption are made. Assuming that:

- The price sensitivity can be approximated by a linear relationship: \( U_{c} \left( \sum c_{i} \right) = \exp \left( \mu_{c0} \sum c_{i} \right) \). While this assumption cannot hold for the entire cost sensitivity function (Han et al. 2001), if the difference between \( \sum_{w} c_{w} \) and \( \sum_{w} \delta_{c,w} c_{w} + \sum_{l} \delta_{c,f} c_{l} \) is small, the function can be linearized;
- The cost ratio between customizable platform components and integral system components is similar for all the components: \( \delta_{c,f} \approx \delta_{c} \forall i \).
Eq. 5.17 then becomes:

\[
\left\{ \prod_k U_{F_k}^{(h)} P_k^{\delta_{F_k}} \geq \left[ U_c^{(h)} \left( \sum_w c_w \right) \right]^{\delta_c (1-1)} \prod_i U_{F_i}^{(h)} P_i^{(1-\delta_{F_i})} \right. \\
\left. \sigma_c = \frac{\sum_i c_i}{\sum_w c_w} \right. 
\]

(5.18)

Where \( \sigma_c \) is the ratio between the costs of the new functionalities in a customizable platform and the cost of the functionalities offered by the integral system, compared without the cost ratio \( \delta_c \). While the comments on the role of functionalities and performances remains the same as Eq. 5.17, the cost term \( \left[ U_c^{(h)} \left( \sum_w c_w \right) \right]^{\delta_c (1-1)} \) now depends on the parameters \( \delta_c \) and \( \sigma_c \). Figure 5.1 shows the effects of these two parameters on the price sensitivity ratio \( \left[ U_c^{(h)} \left( \sum_w c_w \right) \right] \).

![Figure 5.1: Effect of \( \sigma_c \) for different \( \delta_c \) on price sensitivity ratio in Equation 5.18](image)

Figure 5.1: Effect of \( \sigma_c \) for different \( \delta_c \) on price sensitivity ratio in Equation 5.18

It must be remarked that \( \delta_c \) and \( \sigma_c \) model two different effects on costs. \( \delta_c \) is different from one if the modularity of the components affects the platform costs. If it is more convenient to produce a component as a single module, \( \delta_c \) is lower than one; otherwise, it is greater than one. \( \sigma_c \) quantifies the relative cost effect of the introduction of the new functionalities, independently from the design strategy chosen. While both parameters can be influenced by a pricing strategy, \( \delta_c \) is more related to manufacturing aspects and economies of scale, while \( \sigma_c \) is closer to product strategy and market segmentation. Since the cost exponent has a mixed term, high numerical values of both \( \delta_c \) and \( \sigma_c \) must be avoided, as their combination is super-linear. In case
modularity leads to high component costs, the pricing strategy should aim at reducing the ratio $\sigma_c$ as much as possible. Interestingly, if $\delta_c = \frac{1}{(\sigma_c + 1)^{\gamma}}$, costs do not play any role in customers decisions, as the effects on cost due to new functionalities are perfectly balanced by the effects of modularity.

2.2 The Costs minus strategy

Customizable platforms is the strategy opposite to the Benefits plus and envisage removing features that bring little benefits to the customer, thus decreasing the overall cost of the product. The customizability space is defined as still defined by Eq. 5.14, but the Logit value of the platform is:

$$V_{j,pl}^{(h)} = \frac{\prod_i U_{F,i}^{(h)}(P_i^{(h)})^{\delta_{p,i}}}{\prod_k U_{F,k}^{(h)}(P_k^{(h)})^{\delta_{p,k}}(\sum_{w} \delta_{c,w} c_w - \sum_i \delta_{c,i} c_i)}$$  \hspace{1cm} (5.19)

Where $U_{F,i}^{(h)}$ and $P_i^{(h)}$ are the function benefits and the performance benefits of the functions that the platform shares with the integral system of reference, respectively; $U_{F,k}^{(h)}$ and $P_k^{(h)}$ are the benefits of the features that have been removed from the platform. $c_w$ are the costs associated to the i-th function, while $c_i$ are the costs associated with the k-th functions. As before, $\delta_{p,i}$ is the ratio between platform modules performances and the ones from equal integral components (see Eq. 5.11), while $\delta_{c,i}$ is the ratio between the costs as defined in Eq. 5.12.

If the definitions of value from Eq. 5.15 and Eq. 5.19 are inserted into Eq. 5.14, it is possible to find the customizability space as a function of the removed features:

$$\prod_k U_{F,k}^{(h)} P_k^{(h)} \leq \frac{U_{c}^{(h)}(\sum_{w} c_w)}{U_{c}^{(h)}(\sum_{w} \delta_{c,w} c_w - \sum_i \delta_{c,i} c_i)} \prod_i U_{F,i}^{(h)} P_i^{(\delta_{p,i}-1)}$$  \hspace{1cm} (5.20)

Compared to Eq. 5.17, some differences and similarities can be pointed out. The customizability space is defined by an upper bound, instead of a lower bound: features must be removed only if they do not bring enough benefits compared to their cost. As far as costs are concerned, a high reduction in costs can justify the removal of several features. Note that the cost ratios $\delta_c$ play a smaller role compared to the Benefits plus development strategy: removing features can be especially important if modularization leads to high cost compared to integral components. As for Benefits plus strategy, if $\delta_p$ is smaller than one, it reduces the size of the customizability space.

Given the same simplifying assumptions exposed for Eq. 5.18, it is possible to derive from Eq. 5.20 the following relationship:

$$\prod_k U_{F,k}^{(h)} P_k \leq [U_{c}(\sum_i c_i)]^{\gamma-1-\delta} \prod_i U_{F,i}^{(h)} P_i^{(\delta_{p,i}-1)}$$  \hspace{1cm} (5.21)

Where $\gamma$ is the same factor defined in Eq. 5.18 and in this case is generally smaller than one. Further strategic interpretations can be deduced. Costs minus is a defensive strategy: instead of proposing new features, platforms allow to reduce the features and the costs. Developers can affect the performance ratios $\delta_{p,i}$
(increasing the performances of modules) or the cost ratio \( \delta_{c,d} \), but have little influence over \( \sigma_{c} \), which is determined by the market. Therefore, Eq. 5.20 and Eq. 5.21 can provide suggestions about the pool of choices to give to customers, but they have little influence on new modules development. This strategy is particularly useful if target customers have high cost sensitivity and low interest for high performances or advanced functionalities: in this case, the term \( \left[U_i \left( \sum c_i \right) \right]^{\alpha_i-\delta_i} \) would quite big, justifying the removal of several features.

3. Value analysis methodology for customizable product platforms

Section 2 proposed a definition of value consistent with the concept of Utility and derived two major strategies for the development and choice of modules for customizable platforms. However, some simplifying assumption had to be made in order to derive close-form analytical equations that relate market, performance and costs factors. Furthermore, while the analysis of simplified strategies can be useful as general reference, it can provide little insights during actual product development. Real platforms are complex technical systems that can generate thousands of feasible modules combinations; a structured methodology to analyse them is necessary.

A traditional Value engineering job plan consists of five main phases: information gathering, alternatives generation, alternatives analysis, proposal development and presentation/implementation of the proposal. The methodology here presented differs from a standard job plan in three aspects. First, the method can be employed for several goals, from prioritizing modules development to providing suggestions to customers; for this reason, an initial Goal definition phase is required. Secondly, product platforms are complex technical systems that require high cognitive effort during the design: before developing a proposal, proper visualization techniques must be selected. Finally, the value considered is customer-centered, thus it ranks platform configurations from the eyes of the customer. This change allows a change in perspective from traditional value analysis, which tries to minimize the cost of the product for the company; furthermore, the results of the analysis can also be utilized to set a price strategy for modules or to inform customers about the most preferred combinations. The resulting methodology can be appreciated in Figure 5.2.

![Figure 5.2: Scheme of the value analysis methodology for customization-intense platforms](image-url)
All the proposed activities are iterative in nature. Every time a new analysis is performed, the hypothesis and the goals must be laid out clearly, and the results compared within each set of hypothesis. This procedure can be associated with scenario analysis (Armstrong 2001).

3.1 Goal definition
The platform value analysis allows several studies, depending on the life-cycle stage of the system; thus, the first step in the methodology is the choice of one or more goals.

Fundamental insights can be derived before the beginning of the platform’s development. As detailed in Section 2, modularity is a key architectural feature for the mass customization, but it tends to reduce the performances because it prevents system-wise optimization of resource consumption. If the drawbacks of modularity are not balanced by its benefits of customizability, a customizable product may not be the most suitable strategy. This could be the case for performance-demanding markets or markets with uniform customers’ preferences.

After the core architecture has been fixed, designers must evaluate what platform architectures and configurations will be offered on the market. As mentioned before, an optimal number of configurations must be chosen in order to limit the effects of the paradox of choice and to avoid high customizability costs. Value ranking may serve this purpose and the preferable configurations; however, price should also be taken into consideration, as customers’ willingness-to-buy usually shows a price threshold (Han et al. 2001). Customers may decide not to buy very valuable configurations because the expense required is too high.

Another outcome of the analysis is the definition of modules’ technical requirements and modules’ development prioritization. In this case, the focus is on modules functions and performances. Development, manufacturing and logistic costs are another important issue, as they can influence the final price of the configuration. Moreover, if certain modules show synergic effects (Corning 2002), like emergent functions, they can be suitable to form bundles. Customer-based value analyses, being derived from price and not on costs, can be useful to determine the pricing strategy of the platform. Depending on their price, certain modules can be more or less valuable in the eyes of the customer; marketing may prefer to increase the price of modules with high benefits, or to decrease the price of non-optimal combinations. Firms may decide to add a large premium for customizability if the benefits of customizability allows; otherwise, they may reduce profit margins or even subsidize modules. Strategic issues related to platforms and markets can be found in (Gawer 2002).

Finally, if the product platform has already been designed and the modules available set, the information provided by the numerical tool can be integrated in a configuration tool for customers. Previous research discovered that customizability benefits arise only if the choice process is enjoyable (Franke and Schreier 2010) and does not generate frustration (Valenzuela et al. 2009). If individuals’ preferences can be inferred, the analysis results can reduce the total configurations to the most valuable, thus increasing customers’ satisfaction and reducing the potential frustration.

3.2 Information gathering
Information gathering collects the data required for the computation of Logit value as described in Section 3. Information gathering means to find information inside the firm as well as outside, and focuses on three main areas of investigation: the technical system, the market and the costs.

As far as the technical system is concerned, the system must be defined and a functional analysis must be performed. System definition requires designer to detail the list of all the potential modules actually or potentially available. Functional analysis is necessary in order to specify the functions offered by the system and the relative performances. Several functional modelling approaches are offered in several branches of Engineering design (Eisenhart et al. 2013; Crawley et al. 2015). Every function should have one or more technical parameter specification, which are needed as a proxy of the performance evaluation by customers. The detail of the analysis must be a compromise between completeness and the complexity, as functions will be the input of a successive customers’ preference investigation. A principal function must be associated to every module, but secondary functions and/or product behaviours can also be included. Furthermore, system-level function must be inferred from the combination of the modules: as mentioned earlier, emergent properties are usually the ones associated to the highest benefits.

Market provides two important class of information: customers’ preferences and competitors. Customers’ preferences are computed through a conjoint analysis. While preferences about modules, performances,
emergent properties and brand are quite easy to obtain, inferring the benefits of customizability and the value of uniqueness or price sensitivity can be challenging. Usually, a cluster analysis (Louviere et al. 2000; Hastie et al. 2009) is performed on results, so that average customers profile can be inferred from the dataset. A technical and functional analysis can be performed also on the competitor, so that competing products value can be quantified; this step is required to inform the decision about whether or not a modular platform concept can be successful against integral products on the market (Section 4.3).

The last part of the Information gathering phase is the platform cost modelling. Cost modelling can be considered more an art than a science, but a wide literature can guide designers through this task (Stewart et al. 1995; Weustink et al. 2000; Tu et al. 2007).

3.3 Alternatives generation and Alternatives evaluation

Once all the information is collected, a list of feasible modules combinations must be generated, either manually or automatically. In the first case, designers’ explicit and tacit knowledge is employed; if previous configuration database exist, they can provide the basis for the list.

Since the number of combinations grows with the number of modules as a factorial, automated tools for the generation of feasible combinations are suggested. Some examples can be found in (Zeidner et al. 2010; Selva 2012; Shougarian 2016). Once the combinations have been generated, their Logit value can be computed according to Eq. 5.8. While the value associated with the modules’ main function and emergent properties is relatively easy to compute, the value of intangible aspects, the value of customizability and the value of uniqueness may be more challenging, as they require market insights and a close designer-user interaction.

3.4 Alternatives visualization and Proposal development

In order to advance a proposal, designers must understand which are the most valuable configurations and why. Visualization can be a powerful tool to understand complex systems and design them. While several visualization techniques can be used (Ware 2008), three alternatives are suggested here:

- A tradespace is a Cartesian coordinate system where alternatives are described by two or more performance variables and are compared one against the other. It is a powerful visualization tool when there is a large set of combinations to be contrasted (Figure 5.3a). Tradespaces allow an easy representation of the Pareto front, the set of points in which it is not possible to improve a performance index without worsening another index. For example, a tradespace can compare function utilities with price utility or Logit value with price;

- Spider charts (also known as radar charts) represent performance indices on several graded axes; the area of the polygon is a measure of the overall worth of the combination (Figure 5.3b). Spider-plots allow a more fine-grained comparison of the combinations, as the axes can indicate specific performance levels for modular or emergent functions; for this reason, they are suitable for detailed comparison of a small set of configurations;

- Bi-directional bar charts are useful representation for a small sample of configurations. One direction of the bar chart represents the functions utilities, the other the price utility, so that configurations that provide great benefits at high costs can be differentiated from the ones that provide small benefits at low cost. This representation can be used for market differentiation of the platform’s configurations.
The Proposal development phase consists in a decision-making process helped by the analyses results. Depending on the goals defined in Step 1, it can address different aspects of the platform development, like the platform architecture, the module offered to customers or the pricing strategy. This step can benefit from the involvement of different units in the firm; iterations between the first four steps may be required either to increase the robustness of the proposal, or to inquire novel aspects of the analysis.

3.5 Presentation of results to stakeholders and implementation

Once the proposal has been formulized, it needs to be presented to relevant stakeholders and implemented (Freeman and McVea 2001; Garvare and Johansson 2010). The value analysis can bring positive effects only if it is implemented correctly; results presentation is not enough. As the project development proceeds, information become less uncertain (McManus and Hastings 2006; de Weck et al. 2007; Wynn et al. 2011); at the same time, market preferences change and the competition landscape evolves; analyses update therefore is fundamental, especially if the final objective is to support the customers’ choice process.
The previous chapter presented a “static” analysis, which takes the preferences of customers and extrapolates the potential choices over a set of platform configurations, assuming that all the modules are present at the same time and all the choices are made together. As mentioned in Chapter 3, industry platforms and their business ecosystem change over time, therefore a static analysis can provide biased results. Like other engineering systems, industry platforms are partially designed and partially evolved (de Weck et al. 2011), therefore designers must also look at the whole system lifecycle to insure that the technical specifications of the platform are well devised.

As explained in the model of Chapter 3, every industry platform is associated to an ecosystem, which can be modelled as a multi-sided market. This market strongly depends on network externalities, therefore it is hard to initiate (Evans and Schmalensee 2010) and has to reach a minimum number of customers in order to sustain itself. While literature in economics can provide theoretical indications and innovation management can illustrate what practices are beneficial qualitatively, in order to design an industry platform from a technical perspectives detailed, quantitative information is necessary.

In this chapter, a systemic socio-technical simulation of the industry platform and its ecosystem will be presented. Section 1 and Section 2 describes a numerical tool to study the market dynamics associated with industry platforms, while Section 3 focuses on the elicitation of the numerous inputs to the simulations. In the next part, Chapter 8 will show the application of this tool to a real case study.

1. AbMX, an agent-based model for industry platforms

In this section, the main features of AbMX (Agent-Based Model for Cross-side network effect) will be explained. AbMX is an agent-based model that means to reconstruct the time evolution of the industry platform and the associated environment, so that the strategic design decisions can be informed by numerical scenario analyses. The model consists of four main elements: Modules and Platforms are passive objects, while Users and Developers are active agents. Modules are objects that can be created by Developers and bought by Users; Platforms are feasible combinations of Modules. Users are agents who can buy products on the market. They are able to purchase an entire platform configuration, some modules (if they already have the platform core) or a product from competing sellers. Developers are agents who can enter the ecosystem offering one or more modules, exit the ecosystem, retire modules or update them to a new version.

The design of the agent-based model followed a recent empirical methodology called Characterization and Parametrization Framework (CAP), as described in (Smajgl and Barreteau 2014). In that work, the authors claim that the methodology is able to provide a sound model characterization and parametrization, which are fundamental to insure the model robustness and increase the reliability of the results. The methodology consist of six sequential steps (Figure 6.1). During the step M1, the general model features are sketched; then, the specific attributes and behaviours have to be investigated (steps M2 and M3). Generally, agents with similar attributes and behaviours are grouped together (step M4). There may be iterations from step M2 to step M4, as the agents are characterized better and better and previous assumptions are revisited. Finally, since the actual
population is larger than the simulated population, scaling methods are specified. Model assessment verifies and validates the results of the simulations and provide feedbacks for the entire modelling process.

In particular, AbMX can be included in the Case 2 model types, which describe a large population of agents, but can be populated thanks to surveys, expert knowledge and census data. Figure 6.2 shows the specific characterization and parametrization process adopted.

The model sketch was derived from surveys, expert knowledge or direct observation; previous literature also offered a contribution in this phase. M2 characterizes the agents through surveys or expert opinion, while agents’ behaviour was derived from literature or from direct observation. In the AbMX, all agents in the same class behave the same, but they have different attributes (Step M4a). Finally, scalability was provided with direct multiplication based on Census data.

The following sub-section will detail the characterization process for each of the four model elements, starting from step M1 to step M5. Then, Section 2 illustrates how simulations run and what is the output of the model. Finally, Section 3 describes how to derive the actual parameters in the model from external sources of information.
Figure 6.2: the specific Characterization and Parametrization Framework for AbMX; adapted from (Smajgl and Barreteau 2014)

2. Model characterization

The characterization described in this section was derived from direct observation, surveys, experts’ suggestions, statistical techniques and literature reviews. The model is the result of several iterations and it has been shaped by several discussions with project managers, designers and managers working on the case study in Chapter 8. However, even though the specific form of the model is based on the case study, it is generic enough to be applied to a wide range of cyber-physical systems, from production plants and cities to modular electronic devices and robots. AbMX simulates the interaction between technical elements (the modules and the platforms) and human elements, the Users and the Developers. Users and Developers have been modelled human-written, perfectly rational empirical agents (Chen 2012) which want to maximize their use value and their monetary value, respectively.

In this section, each element will be described in terms of behaviours and attributes. How to determine the parameters inside the model and how to assess the robustness of the results will be the object of Section 3.

2.1 Modules

Modules are passive objects without behaviours; they can be conceptually separated into classes and instances. Module classes are abstract representations of modules that share similar attributes, instances are distinct realizations of module classes; two modules may belong to the same class, but they are independent instances.

In this research work, classes are determined by their main technical properties; in particular, modules’ class can be based on the modules’ functions, working principle or structure; a trade-off exists between the granularity of the analysis and the complexity of the mathematical model. In order to account for the missing technical information and differentiate one module instance from the other, the $MD_{SF}$ index is included, representing all the relevant features that do not determine module classes. As it will be explained, it is assumed that $MD_{SF}$ grows over time due to novel technical knowledge and the pursuit of differentiation.
Classes determine financial attributes, which are further differentiated into permanent or modifiable. Permanent financial attributes are bound by external conditions and cannot be changed by agents; for example, the investments required to design the module or its production cost. Modifiable financial attributes can be changed by Developers according to their behaviour: this category includes the price of the module or the amount of debt gathered in order to start the module’s production.

Status properties are used to indicate if a certain module instance is on the market and for how long it has been available. Furthermore, status properties univocally determine which Developer produces the module.

2.2 Platform
Platforms are feasible combinations of modules, as interconnected by the platform core. As described in Chapter 3, platforms have two different abstraction layers. A platform architecture is the technical system representation where modules have a class, while a platform configuration represents the platform as a set of module instances; platform configurations are the instantiations of generic platform architectures. As it will be detailed in the next subsections, this differentiation is required to reduce the complexity of the computations and affects the Users’ choice model. Platforms have classes, technical properties, financial properties and status (Table 6.2).

The most relevant technical properties are the platform structure, i.e. the module classes that the platform architecture is made of, and the emergent properties, which are the system-level characteristics that do not belong to a single module, but are borne from the interactions between components. The number and kind of emergent properties depends on the specific cyber-physical system under evaluation. Among the several emergent properties, one is particularly important: reliability. Cyber-physical system testing is still a critical part of the design process, and sometimes it cannot be performed until the actual system is operative. The binary attribute $PL_{MF}$ indicates if a platform architecture is composed by modules that do not work well together.
Financial properties consists in core price and the total platform price to Users. The price of a platform can be determined only when the platform is represented as a configuration, since different module instances can have different prices. Finally, status includes a binary variable that states whether a certain configuration is feasible or not with the modules present on the marketplace.

### Table 6.2: Main Platforms properties

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>BELONGS TO…</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PL_{struct}$</td>
<td>Module classes that constitutes the platform</td>
<td>Technical</td>
<td>Platform configuration</td>
</tr>
<tr>
<td>$PL_{emerg}$</td>
<td>Emergent properties attribute</td>
<td>Technical</td>
<td>Platform configuration</td>
</tr>
<tr>
<td>$PL_{MF}$</td>
<td>Platform malfunctioning behaviour</td>
<td>Technical</td>
<td>Platform configuration</td>
</tr>
<tr>
<td>$PL_{core,price}$</td>
<td>Price of the core</td>
<td>Financial</td>
<td>Platform configuration</td>
</tr>
<tr>
<td>$PL_{price}$</td>
<td>Price of the platform configuration</td>
<td>Financial</td>
<td>Product configuration</td>
</tr>
<tr>
<td>$PL_{feas}$</td>
<td>Platform feasibility given the current modules on the market</td>
<td>Financial</td>
<td>Platform configuration</td>
</tr>
</tbody>
</table>

#### 2.3 Other products

Before addressing the characterization of Users and Developers, it is important to mention that the socio-technical system characterization would not be complete without competition. Competitive products are modelled as integral or modular technical system with properties comparable to the cyber-physical industry platform. In this version of AbMX, only integral products have been considered, but future works may include other industry platforms, so that Developers can choose to design modules across one or more platforms. This dynamics can played an important role in other industry platforms (Gawer 2014).

#### 2.4 Users

Modules and Platforms are passive elements inside the agent-based model, since they do perform any task. Users and Developers, on the other hand, are proper agents: they can actively modify their attributes inside the virtual environment in order to accomplish a certain goal. Users have preferences and monetary resources; they want to maximize the use value deriving from a product in exchange of monetary value. In other words, the Users side of AbMX is modelled according to a utilitarian approach (Kiesling et al. 2012).

In AbMX, all Users have the same behaviour, which is based on the logit choice model and the value definition of Section 5. Value is defined as:

$$V_j^{(k)} = \frac{\prod_i (U_{PF,i}^{(k)} P_{PF,i}^{(b)})_{md} \prod_i (U_{SF,i}^{(k)} P_{SF,i}^{(b)})_{md} \prod_i (U_{SF,i}^{(k)} P_{SF,i}^{(b)})_{emerg} V_0 V_{cust} V_{uniq,i}}{U_{e}^{(k)} (\sum_w c_{w,md} + c_{plat})}$$  \hspace{1cm} (6.1)

As is Eq. 5.8, the total value is given by the product of the baseline value with modules, emergent properties, customizability, and uniqueness value, divided by the price sensitivity of the agent. As mentioned before, module characterization inevitably suffers from features approximation; for this reason, the value provided by modules function $\prod_i (U_{PF,i}^{(k)} P_{PF,i}^{(b)})_{md}$ has been subdivided into two components. The first one, labelled with the letters $PF$, is the value of the primary functions with a certain performance level; the other, designated with the letters $SF$, is the value of all the other features that have not been specified in the module class. Modules
in the same class have the same primary functions and/or main features, but they differ by their secondary functions and/or features.

The value of customizability is proportional to the number of modules in the market, mediated by the part-worth utility for the module class:

\[ V_{\text{cust}} = \prod_i \left( U_{PP,i}^{\text{h}} P_{PP,i}^{\text{h}} \right) \exp(b_{\text{cust},i} \sqrt{N_{\text{mod},i}}) \]  \hspace{1cm} (6.2)\

Where \( b_{\text{cust},i} \) is the part-worth utility associated to customizability and \( N_{\text{mod},i} \) is the number of modules in the market for class \( i \). The relationship is assumed sub-linear, because the benefits of choice are more evident when the set of alternatives is small.

The value of uniqueness is a function of the number of users who possess the same product; in general, this relationship is ambiguous and depends on the individual. In this version of AbMX, it is assumed that a linear relationship exist:

\[ V_{\text{uniq},j} = b_{\text{uniq}} N_{\text{arch},j} \]  \hspace{1cm} (6.3)\

\( b_{\text{uniq}} \) is the part-worth utility of uniqueness according to agent \( h \), while \( N_{\text{arch},j} \) is the number of architectures \( j \) on the market.

As briefly mentioned before, the value a certain customer gives to a product also depends on the opinion that her social network gives to the product (Janssen and Jager 2003; East et al. 2007; East et al. 2008; Cheema and Kaikati 2010; Goldenberg et al. 2010; Peres et al. 2010; Moldovan et al. 2011). Therefore, the choice of a product in the agent-based model is based on a variable called perceived value, which also accounts for other users’ evaluations. The model was inspired by (Delre et al. 2010), where the buy/not buy selection is given by the model:

\[
\begin{align*}
U_{i,t} &= \beta x_{i,t} + (1 - \beta) y_{i,t} \\
x_{i,t} &= \frac{\sum_j w_{i,j}}{\sum_j a_{i,j}} \\
y_{i,t} &= \frac{q_g^\gamma}{q_g^\gamma + p_t^\gamma}
\end{align*}
\]  \hspace{1cm} (6.4)\

Where \( U_{i,t} \) is the total utility for agent \( i \) at time \( t \), \( \beta \) is the sensitivity to word-of-mouth, \( x \) is the social utility, while \( y \) is the individual utility that the agent \( i \) gives to the product. \( x \) is given by the fraction of neighbours that have adopted the product, while \( y \) depends on a “quality factor” \( q \), the individual preference \( p \) and a sensitivity parameter \( \gamma \). If the total utility exceeds a threshold factor \( L \), the agent buys the product.

AbMX differs from Eq. 6.4 in two aspects. First, it uses a more consistent model to quantify the individual value, as described in Section 5. Then, it further details the effects of social utility. On one hand, social influence depends on the individual’s attitude towards uniqueness and other people’s choices and may not be strictly linear and positive (Tian et al. 2001; Liang and He 2012); on the other, the valence of word-of-mouth can influence either positively and negatively the perceived value (East et al. 2008). The former aspect has been included in Eq. 6.1, while the latter is considered in the definition of perceived value \( PV \):

\[ PV_j^{\text{h}} = (1 - \beta) V_j^{\text{h}} + \beta V_j^{\text{WOM}} \]  \hspace{1cm} (6.5)
Chapter 6  
Initiating and sustaining industry platforms

Where \( b \) is again the sensitivity to the word-of-mouth, \( V_j^{[h]} \) is the value given by agent \( h \) to the product \( j \) and \( V_j^{WOM} \) is the average value given by neighbours to product \( j \) (Eq. 6.6).

\[
V_j^{WOM} = \sum_{z \in (n,p)} \frac{V_j^{[z]}}{N_{n,p}}
\]

(6.6)

\( N_{n,p} \) is the number of neighbours that own the same product or the same platform, therefore \( V_j^{WOM} \) is the average value given by the neighbours informed about the real value of the product.

It is now possible to associate a likelihood of choice to the perceived value, as in Section 1.1 of Chapter 5:

\[
P_a = \frac{PV_j^{[h]}_{a}}{\sum_{k \in M} PV_j^{[h]}_{k}}
\]

(6.7)

The likelihood of choice is again given by the value that agent \( h \) gives to the product \( a \), divided by the sum of the values that \( h \) gives to the products in the market; in AbMX, products can be alternatively platform configurations of the cyber-physical platform, or products offered by competition. AbMX is therefore product-based, as it always considers the utilities of entire configurations; this choice improves the performances of the computational tool and allows comparing integral and modular products consistently.

However, a full comparison of all possible product configurations would be impossible not only for the computers running the simulations, but also for the customers choosing their configuration. In fact, industry platforms can offer hundreds of platform configurations and several thousands of product configurations. Since the choice process can be modelled as a hierarchical process where customers filter out general options (the module classes) and then choose specific options (the module instances inside the class), a more refined choice representation would be the nested logit model (Ben-Akiva and Lerman 1985; Louviere et al. 2000):

\[
P_{m,a} = \frac{\exp \left[ \lambda_m v_{m} + \frac{\lambda_m}{\lambda_a} \log \sum_{a' \in A^a} \exp(\lambda_a v_{a'm}) \right]}{\sum_{m \in M} \exp \left[ \lambda_m v_{m} + \frac{\lambda_m}{\lambda_a} \log \sum_{a' \in A^a} \exp(\lambda_a v_{a'm}) \right]} \cdot \frac{\exp(\lambda_a v_{a'm})}{\sum_{a' \in A^a} \exp(\lambda_a v_{a'm})}
\]

(6.8)

Where \( a \) is the platform configuration and \( m \) is the platform architecture; \( a' \) is an index indicating all the platform configurations inside the platform nest \( M \), \( m' \) indicates all the platform configurations and \( \lambda_a \) and \( \lambda_m \) are shape coefficients, which model the relative weight between the architecture-level choices and the configuration-level choices. In Eq. 6.8, the first term represents the likelihood of choice of a certain platform architecture and is usually called “lower model”; the second term is the likelihood of choice of a certain platform configuration and is addressed as “upper model.” In this case, instead of comparing all the possible configurations, the model would need the part-worth utilities of the module classes and the conditional probabilities of the module instances with respect to the class they belong.

This model has three weaknesses. Even if the computational complexity has been reduced, this model still needs many parameters in order to compute the choice process. This issue is relevant both for the computational resources required to run the tool, and for the assessment of these parameters. As it will be described, the choice model has to be informed through conjoint analyses questionnaire, whose length affects the quality of the responses. A model with a large number of parameters requires long questionnaires, which in turn generate noisy responses. Furthermore, emergent properties cannot be easily decomposed into modules’ properties, and therefore are hard to model in a nested logit model. Finally, a nested choice model does not take into consideration the feasibility of the platform architectures.
For these reasons, a simplified choice heuristic has been employed, which is still consistent with the logit model, but assumes that customers make two decoupled, sequential choices. In the first choice, users select one module instance inside each class (if available); in the second choice, they pick a platform configuration, with the chosen module instances or another product.

Equation 6.1 then becomes:

$$
P_{n,m} = \frac{V_n^{(h)}}{\sum_{h \in M_n} V_i^{(h)}} \sum_{h \in M_n} \sum_{h \in M_n} PV_{h}^{(i)}$$

$$V_n^{(h)} = \prod_{i} (U_{SF,i}^{(h)} P_{SF,i}^{(h)})_{md} \prod_{i} (U_{PF,i}^{(h)} P_{PF,i}^{(h)})_{md} \prod_{i} (U_{F,i}^{(h)} P_{F,i}^{(h)})_{emerg} V_{c}^{(h)} (\sum_{n} c_{n,md} + c_{plat})$$

$$PV_{mn}^{(h)} = (1 - \beta)V_{mn}^{(h)} + \beta V_{mn}^{WOM}$$

With respect to Equation 6.9, the value of customizability is absent, the cost of the phone is independent from its components and it is supposed that integral phones have no malfunctioning.

As mentioned in Table 6.3, Users have a certain amount of money that can be spent on the platform each month. Every time they make a purchase, they have to wait a certain time interval, given by the purchase cost divided by the monthly budget. In this version of the model, Users cannot make trade-offs between a cheap purchase at present versus an expensive purchase in the future; this aspect will be improved in the future.

Finally, the computational resources in modern computer usually do not allow the simulation of the entire User community. A proportional scaling method (Step M5) is introduced; one virtual agent represents therefore the choice of more than one user. If the agent population size is large enough, the sampling error should be negligible; however, it is possible that the social network effects are biased for high scaling factors.

Users are typified by class, status attributes, preferences, financial attributes and social attributes (Table 6.3). While real users are different one from the other, in the agent-based model they must be clustered in order to
reduce the simulation complexity (Step M4). Each class represents a certain average type of customer, which is often defined as “Persona” in marketing literature (Aulet 2013). A class determines the average preferences and the other agent’s attributes; it can be considered a meta-attribute that determines all the other. In AbMX, all Users have the same behaviour; hence, their class determines the attributes.

As modules and platforms, also Users have status attributes. If a User joins the community, his status becomes active and he is referred to as Adopter. Furthermore, each agent owns a competitors’ product or a certain product configuration, which was chosen among the set of possible products thanks to the perceived value (see Eq. 6.5).

Table 6.3: Users’ main attributes

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_class</td>
<td>User class</td>
<td>Class</td>
</tr>
<tr>
<td>U_status</td>
<td>User presence in the ecosystem</td>
<td>Status</td>
</tr>
<tr>
<td>U_{PV,a}</td>
<td>Perceived value for configuration a</td>
<td>Status</td>
</tr>
<tr>
<td>U_{PLC}</td>
<td>Product configuration or product owned</td>
<td>Status</td>
</tr>
<tr>
<td>U_{b,i}</td>
<td>User part-worth utilities for property i</td>
<td>Preferences</td>
</tr>
<tr>
<td>U_budget</td>
<td>User monthly budget</td>
<td>Financial</td>
</tr>
<tr>
<td>U_{buy}</td>
<td>Time interval before the next purchase</td>
<td>Financial</td>
</tr>
<tr>
<td>\beta</td>
<td>Influence of word-of-mouth</td>
<td>Social</td>
</tr>
<tr>
<td>U_net</td>
<td>Neighbours in the social network</td>
<td>Social</td>
</tr>
</tbody>
</table>

Preference attributes include the part-worth utilities for the modules and platforms features. These attributes correspond to the various value components in platforms, as described in Section 1.1. Users are also defined by financial attributes, like the amount of money that they save every month for the platform, which determines the frequency of their purchases.

In order to consider social interactions, Users are bounded together by a social network, which has the property of a small-world network (Watts and Strogatz 1998) and symbolise the social interactions of real customers. Users have a certain sensitivity towards other people’s opinion and their purchasing behaviour is affected by the word-of-mouth, as it was highlighted in Equation 6.5.

2.5 Developers

Developers represent the second side of the ecosystem around the industry platform. Developers mean to maximize their monetary value by developing and selling modules to the Adopters in the ecosystem. Their behaviour reflects the decision tree in Figure 6.3: depending on their status, Developers can introduce new modules, retire them or update them to a new version. The Developers side of AbMX can be considered an econometric agent-based simulation of innovation diffusion (Kiesling et al. 2012).

If a Developer is not part of the community, it can evaluate whether he/she wants to start the development of a new module or not. For each module associated to the Developer, a simplified financial model is computed to find the net present value (NPV) of the investments in engineering and manufacturing.

\[
NPV(MD_i) = \sum_{t \in T_{devel}} \frac{ECF_{devel}(i)}{(1 + D)^t} + \sum_{t \in T_{setup}} \frac{ECF_{setup}(i)}{(1 + D)^t} + \sum_{t \in T_{market}} \frac{ECF_{market}(i)}{(1 + D)^t} \quad i \in D_{md}^{(1)}
\]  

(6.11)
The financial estimate takes into consideration three macro-phases: the first one refers to the product development and engineering; the second one is about the manufacturing facilities set-up, while the third is the actual market phase, when the product is produced and sold. The NPV for the module $i$ belonging to the set of modules $D_{md}$ of Developer $j$ is the sum of the estimated cash flows in the three different phases. $D_r$ is the discount rate for the Developer $j$.

The development phase considers just the Non-recurrent Engineering costs, divided by the total length of the development process (Eq. 6.12); the cost for product development is scaled by the number of previously developed modules $D_{md, market}$ multiplied by the learning curve coefficient $LC_{devel}$ (Stewart et al. 1995).

$$ECF_{devel}(t) = -\frac{MD_{NRE}}{T_{devel, fin} - T_{devel, init}} \cdot \frac{1}{LC_{devel} D_{md, market}}$$

(6.12)

- **Developer $j$**
  - **Supporter?**
    - **Yes**: Cash flow $> 0$?
      - **Yes**: Module update?
        - **Yes**: New version of the modules
          - **Yes**: No more modules on the market?
            - **Yes**: Developer leaves the ecosystem
              - **No**: No action
            - **No**: Price update
              - **Yes**: No action
              - **No**: New module on the market
        - **No**: Retire module
      - **No**: New module on the market
        - **Yes**: No action
        - **No**: Price update
          - **Yes**: No action
          - **No**: Developer leaves the ecosystem
    - **No**: Introduction of new module?
      - **Yes**: Developer joins the ecosystem
        - **No action**
      - **No**: Can the Developer introduce a new module?
        - **Yes**: Is a module price update convenient?
          - **Yes**: Module update?
            - **Yes**: New version of the modules
              - **Yes**: No more modules on the market?
                - **Yes**: Developer leaves the ecosystem
                  - **No**: No action
              - **No**: Price update
                - **Yes**: No action
                - **No**: New module on the market
            - **No**: Retire module
          - **No**: No module update
            - **Yes**: New module on the market
              - **No action**
            - **No**: Price update
              - **Yes**: No action
              - **No**: No module update

Figure 6.3: Decision tree for Developers
Equation 6.13 defines the costs for the manufacturing plant start up as:

\[
ECF_{\text{startup}}(t) = -\frac{MD_{\text{SUC}}}{T_{\text{startup,fin}} - T_{\text{startup,init}}}
\]

(6.13)

Where \(MD_{\text{SUC}}\) is an acronym for manufacturing set-up costs and the denominator is the length of the set-up phase.

The market phase expected cash flows are given by:

\[
ECF_{\text{market}}(t) = (MD_{\text{price}} - MD_{\text{C,man}} - MD_{\text{C,log}} - MD_{\text{C,fees}})N_{\text{md,est}} + MD_{\text{C,op}}
\]

(6.11)

Where \(MD_{\text{price}}\) is the price of the module, \(MD_{\text{C,man}}\) are the manufacturing costs, \(MD_{\text{C,log}}\) the logistic costs and \(MD_{\text{C,fees}}\) are the fees that the platform owner may require from the Developer for being part of the ecosystem. \(N_{\text{md,est}}\) is the number of module that the forecasting model estimates from the market history and \(MD_{\text{C,op}}\) are the costs for operations. In the current version of AbMX, the forecast is based to a polynomial interpolation of the previous six months, other methods like machine learning may be included in the future.

The price of the module is not fixed a priori, but it is determined by the agents thanks to an optimization algorithm computing the expected market potential as a function of the price. In fact, the potential revenues increase as the price increase, but the number of potential customers decrease with the increase in price due to the price sensitivity (Eq. 5.7). The constrained optimization process, based on interior point algorithms, solves the problem:

\[
\max_{t \in \text{md}} \left( \sum_t ECF_{\text{market}}(t) \right) \quad \text{such as } MD_{\text{price}} \in (0.01, MD_{\text{price,lin}})
\]

(6.12)

The upper price limit \(MD_{\text{price,lin}}\) has been introduced so that the optimizer does not converge to infeasible solutions. In order to estimate the market potential \(N_{\text{md,est}}\) inside the optimization process, average part-worth utilities have been employed.

If a new module is introduced for the first time in the ecosystem, the Developer becomes a Supporter. If the Developer is already a Supporter, every 12 months he/she computes the cash flows of the previous year to decide what to do. The cash flow model in Eq. 6.13 computes the net cash flows \(NCF\) as defined in (Crundwell 2008).

\[
NCF = (1 - T)[(MD_{\text{price}} - MD_{\text{C,man}} - MD_{\text{C,log}} - MD_{\text{C,fees}})N_{\text{md,est}} - MD_{\text{C,op}} - I_{\text{md}}]
\]

(6.13)

\(T\) is the income tax rate taken by the government (if the taxable income is positive) and \(I_{\text{md}}\) are the interest payable on loans.

The net cash flows can be either positive or negative. If they are positive, the agent can either change the price of her modules on the market or introduce new modules. In the former case, Equation 6.12 is employed and a new price is set; in latter, the entire decision process from Equation 6.8 is performed again. On the other hand, if the net cash flows are negative, the Developer can either invest his cash and design a new version of the same module, or it can remove the module from the market. Once again, the agent decides what to do forecasting the potential revenues thanks to Equation 6.11. If all the Developer’s module are retired, the Developer leaves the ecosystem permanently.
Developers are characterized by four types of attributes (Table 6.4), the most important of which is the class. There are two classes of Developers: Legacy Developers and Start-ups. Legacy Developer can benefit from economies of scale and economies of scope from previous projects; therefore, the costs they have to sustain are lower than the Start-up’s.

The remaining attributes are clustered status, modules and financial types. Status attributes determine if a Developer is part of the community, when is time for him to evaluate the cash flows and how many modules he/she has already introduced in the market. Module attributes are the interface between modules and the Developer: they show which modules are available and which modules are on the market. Financial attributes include the Developers discount rate, the Developer’s debt (needed to calculate $I_{md}$ in Eq. 6.14) and the cash flows.

### 2.6 Overall simulation

AbMX code consists of four main parts. In the first part, the simulation parameters are defined; in the second part, the socio-technical system is initialized, the inputs are transferred to the simulation variables and the random parameters are drawn. In the simulation main body, the Users and the Developers interact until the time limit has been reached or the market collapses. Finally, all the variables of interest are stored in a file.

The input variables include the attributes described in the previous sections, as well as general simulation parameters like the maximum simulation length, the time from which the Developers can start generating modules, the maximum and minimum value for the attributes that are assigned stochastically to agents, and the initial conditions.

During the system initialization, all the attributes are assigned to elements in the simulation. Attributes are allocated to agents randomly, according to the statistical data that have been collected from the real socio-technical system; a small random variation around the mean value of the attribute is allowed to consider the variability of the agents.

The main part of AbMX lies in the Monte-Carlo simulation of the platform ecosystem. At each time-step (representing a month), the Users that replenished their savings from the previous purchases can change the product they own according to choice model in Eq. 6.11. Users first choose a module inside each available class; then they evaluate the different product configurations. A platform configuration (or a competitive product) is chosen according to the stated choice model; it is then the turn of the Developers side. As mentioned in Section 2.5, Developers behave depending on their status and their financial evaluations: they can enter, remain in or exit the ecosystem, and can introduce new modules on the market or retire them. In this case, the randomness of the process lies in the market forecast, which can vary according inside the projections’
confidence bands. Every time a new module is introduced into the market, the parameter $MD_{SF}$ is assigned to represents all the features excluded from the module class definition; newest modules have higher numerical values of $MD_{SF}$ in order to represent technological advancements, differentiation from previous modules and the appeal of novelty. Integral products have their own rate of improvement, which increase their baseline value $V_0$ every 12 months. As it will be highlighted in Chapter 8, the market rate of improvement plays an important role in the evolution of the ecosystem.

In the last part, the data from the time simulation are rearranged to find the Users’, Developers’ and modules’ birth-rate, death-rate and affinity curve (Chapter 2, Section 2.3). Finally, all the variables are saved in a file so that they can be retrieved later and compared with other simulations. Given the stochastic nature of the model, the results must be averaged across several simulation conditions, in order to reduce the noise, or outliers must be studied more carefully to understand the origin of the anomalous behaviour.

3. Model parametrization

As described in Section 2, the AbMX requires several parameters for a robust characterization. The Characterization and Parametrization framework proposes five steps to synthesize properly agents-based models (Figure 6.2): Model characterization, Attribute elicitation, Behaviour elicitation, Typologies definition and Scaling methods; in this section, the attention will be focused on the model parametrization, i.e. the definition of the parameters that serve as an input to the model. Some steps of this process are similar to the Information gathering phase described in Section 3.2 of Chapter 5; if the value analysis has already been performed, the information collected can be utilized also to parametrize several elements inside AbMX.

3.1 Module

Two different types of attributes have to be parametrized in modules: technical features and costs.

As mentioned in Section 6.2, the most important technical attribute in modules is the classes. A module is a part of the system that has a well-defined primary function (Ulrich 1995; Ulrich and Eppinger 2011); for this reason, functions are well-suited to be the discriminating factor for modules. Furthermore, a second level of characterization can refer to market segments or the average performance levels. The definition of module classes is an iterative process that involves both the user’s side (through the preferences assessment), the developer’s side and the platform supporters; furthermore, this classification can change in time according to the ecosystem evolution and must be updated, if necessary.

Once the classes have been defined, the financial attributes must be evaluated. Cost modelling is more an art than a science, but an incorrect cost estimation can seriously harm the entire product development process (Stewart et al. 1995; Weustink et al. 2000); cost estimation is not trivial in mass customized products (Tu et al. 2007). For very novel projects, reference (previous) projects are taken as a reference and rescaled according to the new circumstances. Proportionality indexes include learning curves and scaling factors, as well as complexity evaluations (Stewart et al. 1995). Since agent-based models allow a certain variability in the agents attributes, it is possible to include the uncertainty of cost estimation inside the simulation algorithms.

3.2 Platforms

As far as platforms are concerned, two main features have to be modelled: the number of feasible platforms and the module classes that they are made of, and the emergent properties.

For industry platforms, a very wide set of platform configuration can be potentially generated, and assessing the feasibility of each one can be a very demanding task. The use of automatic tools can help designers navigating in this complex design space and increase the reliability of the results. In this research work, and externally developed architecture synthesizer has been employed, based on a grammar of feasible interfaces and a convex hull algorithm (Shougarian 2016).

After the feasible configurations have been listed, their emergent properties must be assessed. Depending on the number of feasible configurations and the presence of data from similar projects, testing, statistical inference or simulations are preferred. Testing with actual platforms or mock-ups is the most comprehensive and reliable method to assess if the behaviour of a complex system is the same as the expected one. Unfortunately, industry platforms can have so many variants that the complete testing of all configurations is impossible. Alternatively, it is possible to sample a limited set of configurations and extrapolates the results over the entire feasible design space. In a similar fashion, previous similar systems can provide the data
required. In both cases, statistical methods like regression modelling or machine learning evaluate the product variants that have not been physically tested. Finally, simulations tools can be developed ad hoc.

The evaluation of emergent properties and system-level testing is one of the most critical aspects in the design of cyber-physical systems (CPS Steering Community 2008; Sztipanovits et al. 2012; Energetics Inc. 2013; Gunes et al. 2014).

3.3 Users

Users’ parametrization focuses on preferences and budget. Users choice preferences can be evaluated through conjoint analysis (see Chapter 5). Several types of conjoint analysis exist: the granularity of the model and the availability of respondents can guide the choice of one type or another. The evaluation of the part-worth utilities relative to technical features are relatively easy to assess, while uniqueness and customizability utilities are harder to assess through a questionnaire. Users’ budget can be inferred thanks to market research as well.

3.4 Developers

Since the behaviour of Developers is determined by financial evaluations, the Developers’ parametrization must evaluate the financial attributes of the agents. In this case, either direct interviews with developers or questionnaires can be employed. The parametrization should investigate not only the specific financial parameters, but also the costs for the development and the production of modules (Section 3.1).

Obtaining information about Developers and modules in industry platforms is more difficult than in product platforms. In product platforms, the number of external developers is limited to the product’s supply chain and strong relationships exists between integrators and vendors. In industry platforms, modules are created by a wide community whose ties with the platform supporter can be weak or even partially conflicting. Valid assumptions based on previous projects or freely available information can provide the missing information if these cannot be retrieved in any other fashion.
Part 3.0

Strategic design analyses application
Chapter 7

Influence of architectural features on change indices

This chapter means to broadly investigate how change propagation is influenced by architectural features, such as degree of modularity, system’s structural complexity or subsystems’ betweenness. The goal is to understand if and how architectural features increase or decrease the change propagation indices, so that system architect can anticipate change propagation and mitigate the negative effects of changes.

To this end, the Change Investigation Technique and the software tool 8AM800 (Chapter 3) are employed. Section 1 exposes the design of the experiments that generated the synthetic Design Structure Matrices and the Change Propagation Matrices. Section 2 focuses on the effect of modules and bus elements on the change propagation behaviour; Section 3 describes the set of metrics chosen to represent architectural features. Finally, Section 4 explores more in details the correlation between architectural features and change propagation indices and Section 5 summarizes the results of the chapter and provides suggestions for limiting change propagation through system architecting.

1. Generating architectures and changes

This section will explain how the synthetic architectures (in the form of DSMs) and changes (in the form of CSMs) have been generated numerically with a Monte Carlo process. As illustrated in Section 3.1 of Chapter 3, 8AM800 is equipped with an internal architecture generator that can create feasible architecture models from a design of experiments. The inputs to the architecture generator are provided in Table 7.1.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>PARAMETER IN THE DSM</th>
<th>VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of components</td>
<td>Size of synthetic DSM</td>
<td>40; 60</td>
</tr>
<tr>
<td>Density of interfaces</td>
<td>Average density of synthetic DSM</td>
<td>1/7; 1/5</td>
</tr>
<tr>
<td>Total number of subsystems in modules</td>
<td>Relative number of subsystems that belong to one module</td>
<td>6/10; 9/10</td>
</tr>
<tr>
<td>Modules combinations</td>
<td>Combination of clusters’ size</td>
<td>See Table 6.2</td>
</tr>
<tr>
<td>Number of bus elements</td>
<td>Number of highly connected components</td>
<td>0; 1; 2</td>
</tr>
<tr>
<td>Size of bus elements</td>
<td>Relative number of interfaces for bus elements</td>
<td>1/2; 1</td>
</tr>
</tbody>
</table>

The number of components is adequate for representing a second-level decomposition of a technical system. In fact, it is usually suggested to subdivide a system into approximatively seven subsystem for every level of decomposition, following the empirical rule of “the magical number seven” (Miller 1956).

The density of interfaces was derived from other case studies (Eppinger and Browning 2012) that make use of DSMs. The selection was slightly biased towards high-density matrices, since for very low densities the meta-DSMs generator shows some numerical instabilities.
Modules, which are defined as clusters of subsystems with higher density of interfaces (Hölttä-Otto et al. 2012), are generated according to two sets of parameters: the total fraction of subsystems in the system that belong to modules and the relative size of modules. The algorithm fixes the total fraction of subsystems inside modules first, and then assigns one out of nine combinations to the meta-DSM (Table 7.2). The number of modules in each combination depends on the number of module sizes. If all module are uniform, the number of modules is given by the rounded total number of subsystems in modules divided by the module size, so that the number of modules is maximized; otherwise, the ratio between two sizes of modules is given, in order to avoid unfeasible architectures.

The number of bus elements varies between zero and two; each bus element can be potentially connected either to the entire system or to half of it.

### Table 7.2: Combinations of modules

<table>
<thead>
<tr>
<th>COMBINATION</th>
<th>NUMBER OF MODULES</th>
<th>MODULE SIZE A</th>
<th>MODULE SIZE B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Total number of subsystems in modules</td>
<td>1/3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Total number of subsystems in modules</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Total number of subsystems in modules</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Total number of subsystems in modules</td>
<td>1/9</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Total number of subsystems in modules</td>
<td>1/8</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Total number of subsystems in modules</td>
<td>1/6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2 Modules A and 3 Modules B</td>
<td>1/3</td>
<td>1/9</td>
</tr>
<tr>
<td>8</td>
<td>2 Modules A and 4 Modules B</td>
<td>1/4</td>
<td>1/8</td>
</tr>
<tr>
<td>9</td>
<td>1 Modules A and 3 Modules B</td>
<td>1/2</td>
<td>1/6</td>
</tr>
</tbody>
</table>

Once the meta-DSMs are generated, 12 synthetic DSMs are assembled automatically, assigning interfaces from a uniform random distribution. After a preliminary sensitivity analysis, it was chosen to generate 12 samples of CSMs from each DSM (Section 3.2 of Chapter 3).

The numerical experiments generated a total number of 13,824 synthetic DSMs and 165,888 CSMs. Out of those, 27,948 CPMs (the 16.8% circa) had to be excluded from the dataset because violated constraints on density. In fact, interfaces are assigned randomly and it is not possible to control deterministically the average density inside synthetic DSMs. A 15% variability was accepted; all the DSM samples outside this threshold were removed. Furthermore, some architectures were not connected and therefore had been removed. This brought the total number of valid samples to 137,844.

### 2. Comparison of change propagation indices distribution for integral, modular and star architectures

The 137,844 synthetic architectures were subdivided into four groups. If more than 50% of the components were part of modules, the architecture was called “Modular”. If there was at least one bus element, the architecture was named “Star”. “Modular-star” architectures were the ones with more than 50% of their components in modules and at least one bus element. All the remaining architectures were considered as “Integral”. Since the Integral architectures were randomly generated, it is possible that some of them are implicitly Modular or Integral due to accidental creation of groups of clustered components. However, the randomness of the generation process should have assured that the properties of integral architectures are overall different from the modular and integral ones.

In this section, the analysis of the indices distribution and the influence that the control variables have on such distributions is given. This is the rationale: if there are significant differences between the distributions of change indices, it is possible to show that architectural decisions have an impact on change propagation. All
Chapter 7  Influence of architectural features on change indices

the Figures in this section display the histograms of the ICL, ICI and OCR indices normalized to represent probability density functions. To simplify the notation, the terms “probability density function” and “(probability) distribution” will be used interchangeably.

Figure 7.1 shows the probability density function of the total population of ICL, ICI and OCR indices in all the 137,844 cases.

![PDF](image)

**Figure 7.1. Probability density functions for three change propagation indices ICL, ICI and OCR**

All indices have asymmetrical distributions, with a highly positive skewness and a long right tail, even though the CPMs contained only parameters coming from uniform (and therefore symmetric) distributions. Long-tail and fat-tail risk distributions have already been observed not only in change propagation data (Siddiqi et al. 2011), but also in other fields, like for example in financial transactions (Farmer and Geanakoplos 2008). There are profound differences between the indices, as well. ICL is the most dispersed distribution, with values ranging from 0 to 1, and a near linear decrease of the initial value of probability. The ICI index has an almost flat distribution until the value of 0.1, then it falls off quickly, even though some components still present high values. Interestingly, the ICL and ICI distributions show almost the same likelihood of having a component with index close to zero. The OCR distribution is much more condensed, with a high peak and a rapid decay.

Some observations can be made regarding the prediction of change propagation as predicted by the model. These distributions imply that the number of changes in a technical system can vary considerably, probably depending on the topological characteristics of the architecture. On the contrary, costs and time required for change are mainly condensed in a small subset of components that dominate the entire system.

In order to test if different architecture types have an effect on the probability density functions of the indices a set of parametric and non-parametric numerical tests were carried out. Analysis of Variance (ANOVA) would have been the most intuitive tool to this end. Unfortunately, all the resulting distributions are right-skewed, so the results of ANOVA could be biased. As a box-cox transformation on samples did not provide normal distribution in the results, the Kruskall-Wallis non-parametric test was chosen and carried out. The Kruskall-Wallis test is a non-parametric alternative to ANOVA that shows if two samples derive from the same population or from different populations with same distribution. For each of the indices, the samples from different architecture types were compared; since the four samples were not homogenous after removing the architectures with biased density, tests on random samples were performed, all of which rejected the null
hypothesis. The result was very conclusive. According to the test, for all indices and all architecture types, the samples do not come from the same population, as their p-values are well below the standard threshold of 0.05 and actually close to the minimum value computable by the statistical software used. Therefore, it can be said conclusively that the presence of modules or bus elements (hubs) does affect the statistical behaviour of change propagation. The remainder of the section will show graphically how the samples differ by architecture type. Figure 7.2 shows the samples of ICL divided according to the type of architecture they come from.

Two main different distribution shapes can be noted. Integral and Star architectures have a probability decreasing linearly with the ICL score, while Modular and Modular-Star architectures have a hyperbolic distribution shape, with an higher number of components in the left part of the diagram. Furthermore, the average and median values decrease significantly in the presence of modules.
ICI distributions are represented in Figure 7.3. As in the previous figure, there is a clear difference between Integral- and Star-based ICI and the other ICI. Compared to ICL, though, the dissimilarity is less striking. Modular and Modular-Star architectures are characterized by a higher peak in the low values and a steeper decrease of components with bigger ICI values. Integral and Star architectures have similar averages and medians, like Modular and Modular-Star architectures.

Finally, OCR probability density functions are presented (Figure 7.4). As happened for the previous indices, also in Figure 6.4 the presence of modules in the architecture substantially changes the shape of the distribution. In this case, Modular and Modular-Star samples have a lower mean value and are more positively skewed. Furthermore, among those samples there are more components with values close to zero. The presence of modules decreases the average and median values by almost 30%.

In the Changeability Assessment Technique, the last step in the evaluation of subsystems’ level of criticality is the comparison of ICL versus ICI for the entire system. In the Changeability Investigation Technique, the entire ICL and ICI datasets can be contrasted, in order to highlight patterns and correlations. Figure 7.5, Figure 7.6, Figure 7.7 and Figure 7.8 provide this comparison for Integral, Modular, Star and Modular-Star architectures, respectively. Data are shown as scatterplots with density colour-code (heatmaps) in order to highlight the density in data points with colours.
Chapter 7

Influence of architectural features on change indices

Figure 7.5: ICL vs. ICI for all architectures; lateral bar indicates the colour-code for density of data points; red line is the dataset least-squared line.

Figure 7.6: ICL vs. ICI for Integral architectures; lateral bar indicates the colour-code for density of data points; the red line is the dataset least-squared line.
Figure 7.7: ICL vs. ICI for Modular architectures; lateral bar indicates the colour-code for density of data points; the red line is the dataset least-squared line.

Figure 7.8: ICL vs. ICI for Star architectures; lateral bar indicates the colour-code for density of data points; the red line is the dataset least-squared line.
Chapter 7  
Influence of architectural features on change indices

Figure 7.9: ICL vs. ICI for Modular-Star architectures; lateral bar indicates the colour-code for density of data points; the red line is the dataset least-squared line.

Comparing the figures above, some trends in the data appear. First, data points are scattered in most of the figures, even if ICL covers the spectrum from 0 to almost 1 consistently and ICI rarely has values over 0.5. This is consistent with the distributions shown above: ICL has a long-tailed probability density function, while ICI drops significantly in a short interval of values. Secondly, there are not clear patterns that relate ICL with ICI, but many combinations of ICL and ICI are feasible for all the subsystems and all architecture types. However, as shown in Table 7.3, there is a significant, weak linear correlation between the indices, both in the general comparison and in the comparison for specific architecture types.

Table 7.3: Pearson correlation between ICL and ICI for different architecture types

<table>
<thead>
<tr>
<th>ARCHITECTURE TYPE</th>
<th>PEARSON CORRELATION COEFFICIENT</th>
<th>PEARSON P-VAL</th>
<th>SPEARMAN CORRELATION COEFFICIENT</th>
<th>SPEARMAN PVAL B</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>0.1591</td>
<td>&lt; 0.001</td>
<td>0.1285</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Integral</td>
<td>0.1441</td>
<td>&lt; 0.001</td>
<td>0.1169</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Modular</td>
<td>0.1589</td>
<td>&lt; 0.001</td>
<td>0.1304</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Star</td>
<td>0.1658</td>
<td>&lt; 0.001</td>
<td>0.1336</td>
<td>&lt; 0.001</td>
</tr>
<tr>
<td>Modular-Star</td>
<td>0.1656</td>
<td>&lt; 0.001</td>
<td>0.1344</td>
<td>&lt; 0.001</td>
</tr>
</tbody>
</table>

If the architecture presents a bus element, there is a stronger correlation between ICL and ICI, meaning that if a subsystem has a high risk of being changed, this also implies a high impact of the change. On the other hand, integral architectures have the lowest correlation index. The difference between the architecture types is quite limited, as the Pearson correlation coefficient minimum value stays within the 13% of the maximum value. For the Spearman correlation coefficient, the difference is even more contained, meaning that the correlation between the indices is well represented by linear relationships.

### 3. Choice of architectural indices

Section 2 compared the effect of macro-level decisions in system architecting. However, the design of technical systems in not limited to the choice of modules and bus elements; more fine-grained analyses can guide designer more effectively. To this end, an additional 8AM800 part is needed, which will be called Network
evaluator. The Network evaluator is responsible for analysing global and local topological properties of architectures.

The indices selected for this analysis can be clustered into three groups: network metrics, system architecture metrics and change propagation metrics. Overall, 20 independent variables are considered (Table 7.4), three of which refer to local (i.e. subsystem’s) properties and seventeen to global (i.e. system’s) properties. The code to compute these indices was either designed independently, or integrated from open-source software toolboxes (Gleich 2008; Rubinov and Sporns 2010; Lev Muchnik 2013)

A brief description of the metrics that have not been introduced yet is offered below; for a detailed description of the indices and the mathematical formulation, the reader is referred to the literature cited in Table 6.4.

- **Degree of a node:** the number of connections with other nodes. In directed networks, it is possible to distinguish in-degree and out-degree; for the reasons already explained in Chapter 3, synthetic DSM are symmetrical, therefore the relative network is undirected, and only one degree measure exists. Since changes propagate through interfaces, degree indicates how exposed a subsystem is to other subsystems’ changes.

- **(Local) clustering coefficient:** the measure about how much a node is connected to nodes in the same cluster, i.e. local communities of nodes. It can be computed as the ration between the number of actual links from one node to its neighbours and the possible number of links in the neighbourhood. A high clustering coefficient means that changes can propagate in cluster easily because of the abundance of interfaces.

- **Betweenness (or betweenness centrality):** the measure about the number of shortest paths between nodes in the network that necessarily pass through the node. A shortest path is the minimum set of nodes crossed going from one node to another. If a subsystem has a high betweenness, many changes have to pass through the subsystem to reach other parts of the system.

- **The number of interfaces:** the total number of connections in the synthetic DSM. A higher number of interfaces can lead to more changes propagating across subsystems.

**Table 7.4: Network metrics computed in the Network properties evaluator**

<table>
<thead>
<tr>
<th>METRICS</th>
<th>REFERENCES</th>
<th>APPLIES TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>(Newman 2010)</td>
<td>Subsystem</td>
</tr>
<tr>
<td>Clustering coefficient</td>
<td>(Watts and Strogatz 1998)</td>
<td>Subsystem</td>
</tr>
<tr>
<td>Betweenness</td>
<td>(Newman 2010)</td>
<td>Subsystem</td>
</tr>
<tr>
<td>Number of subsystems</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Number of interfaces</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Average density</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Diameter</td>
<td>(Newman 2010)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Number of modules</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Total number of subsystems in modules</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Number of bus elements</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Average number of bus elements’ interfaces</td>
<td>---</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Assortativity</td>
<td>(Newman 2010)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>MS modularity</td>
<td>(Huang and Kusiak 1998)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Energy</td>
<td>(Gutman and Zhou 2006)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Number of component loops</td>
<td>(Browning and Sosa 2010; Sosa et al. 2011)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Structural complexity</td>
<td>(Sinha and de Weck 2013)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Cost</td>
<td>(Clarkson et al. 2004c; Koh et al. 2013)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Uncertainty</td>
<td>(Clarkson et al. 2004c; Koh et al. 2013)</td>
<td>Whole architecture</td>
</tr>
<tr>
<td>Closeness centrality</td>
<td>(Newman 2010)</td>
<td>Subsystem</td>
</tr>
<tr>
<td>Algebraic connectivity</td>
<td>(Fiedler 1973)</td>
<td>Whole architecture</td>
</tr>
</tbody>
</table>
- The average density: the density given as input to the architecture generator. Since interfaces are assigned stochastically through a Monte-Carlo process, the actual density of synthetic DSMs can vary from the average value. As for the number of interfaces, a denser architecture is likely to suffer more from change propagation.
- A network diameter: the length of the longest path connecting two nodes in the networks.
- Assortativity: the measure about how likely a node is connected to nodes with similar degrees.
- MS modularity: a measure of the architecture modularity. It compares the number of interfaces inside modules to the total number of interfaces outside modules. As seen in the previous Section, modular architectures should be less sensitive to change propagation phenomena.
- The energy of the network: the sum of the eigenvalues of the synthetic DSM.
- Number of component loops: a component loop: is “a subset [of subsystems] whose dependencies form a complete circuit” (Browning and Sosa 2010). The number of component loops in the architecture has a deep influence on the quality of software products and may influence change propagation too.
- Structural complexity: a fundamental property of the system that depends on both subsystems and interfaces structural characterization. In this research, the value suggested in (Sinha et al. 2013) have been employed. Previous literature suggests that complexity is positively correlated with the change propagation indices.
- Uncertainty: the proxy for native changes likelihood, which in real projects can be only estimated. In a similar fashion, Cost represents a measure of the impact for native changes in the subsystem. In this Chapter, they are equal to the actual values of native changes likelihood and impact.
- Closeness centrality: a measure about the distance of the node from the other nodes whose shortest path crosses the node under consideration. A subsystem with high closeness centrality can be reached by many native changes that are borne in other subsystems.
- Algebraic connectivity: the second-smallest eigenvalue of the Laplacian matric of a network. It measures how well a network is connected; therefore, it should have a positive correlation with the change propagation indices.

### 4. Correlation analysis

Section 4 will carry out a series of correlation analyses between the indices and will try to explain the relationship between architectural features and change propagation indices. The scatterplots for all the correlation analyses in this Section can be found in Appendix A.

The first correlation analysis confronts the three change propagation indices, ICL, ICI and OCR. The correlation between ICL and ICI has already been investigated in Section 2, and showed a weak but significant correlation between the two. Now, Table 7.5 presents the general correlation analysis.

<table>
<thead>
<tr>
<th></th>
<th>ICL</th>
<th>ICI</th>
<th>OCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICL</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICI</td>
<td>0.1591**</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OCR</td>
<td>0.4570**</td>
<td>0.3097**</td>
<td>1</td>
</tr>
</tbody>
</table>

*: p-val < 0.05; **: p-val < 0.01

OCR is significantly correlated to both ICL and ICI. In particular, for the architectures generated in this research, OCR correlates more to ICL than to ICI, suggesting that subsystems likely to be changed can also generate further changes in the system. Since OCR is the combination of both change propagation likelihood and change propagation impact (Equation 3.8), it clearly share similarities with ICL and ICI, which are derived from change propagation likelihood and change propagation impact, respectively.

The second correlation analysis compares the change propagation indices with the architectural features in Table 6.4. As it can be noticed from the scatterplots in Appendix A, the data are very noisy, and it is usually difficult to notice a clear linear correlation between the variables. Furthermore, the change indices are often non-homoscedastic, i.e. their variance increases as the architectural index grows. For this reason, not only the Pearson (linear) correlation coefficients are reported, but also the Spearman (non-parametric) correlation
coefficient. The first ones can be found in Table 7.6 and Figure 7.10; the second ones in Table 7.7 and Figure 7.11.

Table 7.6: Pearson coefficients of correlation between the change propagation indices and the architectural features

<table>
<thead>
<tr>
<th></th>
<th>ICL</th>
<th>ICI</th>
<th>OCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>0.417**</td>
<td>0.313**</td>
<td>0.751**</td>
</tr>
<tr>
<td>Clustering coefficient</td>
<td>0.137**</td>
<td>0.060**</td>
<td>0.196</td>
</tr>
<tr>
<td>Betweenness</td>
<td>0.118**</td>
<td>0.169**</td>
<td>0.317**</td>
</tr>
<tr>
<td>Number of subsystems</td>
<td>0.293**</td>
<td>0.093**</td>
<td>0.359**</td>
</tr>
<tr>
<td>Number of interfaces</td>
<td>0.435**</td>
<td>0.183**</td>
<td>0.595**</td>
</tr>
<tr>
<td>Average density</td>
<td>0.379**</td>
<td>0.201**</td>
<td>0.566**</td>
</tr>
<tr>
<td>Diameter</td>
<td>-0.375**</td>
<td>-0.170**</td>
<td>-0.514**</td>
</tr>
<tr>
<td>Number of modules</td>
<td>-0.012**</td>
<td>-0.009**</td>
<td>-0.019**</td>
</tr>
<tr>
<td>Total number of subsystems in modules</td>
<td>0.014**</td>
<td>0.000</td>
<td>0.025**</td>
</tr>
<tr>
<td>Number of bus elements</td>
<td>0.030**</td>
<td>0.020**</td>
<td>0.058**</td>
</tr>
<tr>
<td>Average number of bus elements' interfaces</td>
<td>0.052**</td>
<td>0.033**</td>
<td>0.094**</td>
</tr>
<tr>
<td>Assortativity</td>
<td>0.011**</td>
<td>-0.009**</td>
<td>-0.001**</td>
</tr>
<tr>
<td>MS modularity</td>
<td>-0.019**</td>
<td>-0.012**</td>
<td>-0.019**</td>
</tr>
<tr>
<td>Energy</td>
<td>0.385**</td>
<td>0.147**</td>
<td>0.504**</td>
</tr>
<tr>
<td>Number of component loops</td>
<td>0.450**</td>
<td>0.228**</td>
<td>0.679**</td>
</tr>
<tr>
<td>Structural complexity</td>
<td>0.445**</td>
<td>0.194**</td>
<td>0.618**</td>
</tr>
<tr>
<td>Cost</td>
<td>0.001</td>
<td>0.815**</td>
<td>0.000</td>
</tr>
<tr>
<td>Uncertainty</td>
<td>0.757**</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Closeness centrality</td>
<td>-0.033**</td>
<td>0.074**</td>
<td>0.048**</td>
</tr>
<tr>
<td>Algebraic connectivity</td>
<td>0.413**</td>
<td>0.192**</td>
<td>0.582**</td>
</tr>
</tbody>
</table>

*: p-val < 0.05; **: p-val < 0.01

Figure 7.10: Pearson coefficients of correlation between the change propagation indices and the architectural features
### Table 7.7: Spearman coefficients of correlation between the change propagation indices and the architectural features

<table>
<thead>
<tr>
<th></th>
<th>ICL</th>
<th>ICI</th>
<th>OCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>0.440**</td>
<td>0.257**</td>
<td>0.771**</td>
</tr>
<tr>
<td>Clustering coef.</td>
<td>0.185**</td>
<td>0.080**</td>
<td>0.297**</td>
</tr>
<tr>
<td>Betweenness</td>
<td>0.171**</td>
<td>0.149**</td>
<td>0.337**</td>
</tr>
<tr>
<td>Number of subsystems</td>
<td>0.280**</td>
<td>0.078**</td>
<td>0.371**</td>
</tr>
<tr>
<td>Number of interfaces</td>
<td>0.417**</td>
<td>0.160**</td>
<td>0.623**</td>
</tr>
<tr>
<td>Average density</td>
<td>0.344**</td>
<td>0.169**</td>
<td>0.573**</td>
</tr>
<tr>
<td>Diameter</td>
<td>-0.360**</td>
<td>-0.154**</td>
<td>-0.559**</td>
</tr>
<tr>
<td>Number of modules</td>
<td>-0.015**</td>
<td>-0.008**</td>
<td>-0.019**</td>
</tr>
<tr>
<td>Total number of subsystems in modules</td>
<td>-0.004**</td>
<td>-0.006**</td>
<td>0.001**</td>
</tr>
<tr>
<td>Number of bus elements</td>
<td>0.026**</td>
<td>0.009**</td>
<td>0.039**</td>
</tr>
<tr>
<td>Average number of bus elements’ interfaces</td>
<td>0.042**</td>
<td>0.016**</td>
<td>0.063**</td>
</tr>
<tr>
<td>Assortativity</td>
<td>0.011**</td>
<td>-0.003**</td>
<td>0.010**</td>
</tr>
<tr>
<td>MS modularity</td>
<td>-0.015**</td>
<td>-0.009**</td>
<td>-0.015**</td>
</tr>
<tr>
<td>Energy</td>
<td>0.406**</td>
<td>0.154**</td>
<td>0.603**</td>
</tr>
<tr>
<td>Number of component loops</td>
<td>0.432**</td>
<td>0.169**</td>
<td>0.658**</td>
</tr>
<tr>
<td>Structural complexity</td>
<td>0.414**</td>
<td>0.158**</td>
<td>0.618**</td>
</tr>
<tr>
<td>Cost</td>
<td>0.001</td>
<td>0.865**</td>
<td>0.000</td>
</tr>
<tr>
<td>Uncertainty</td>
<td>0.792**</td>
<td>0.000</td>
<td>-0.001</td>
</tr>
<tr>
<td>Closeness centrality</td>
<td>-0.002**</td>
<td>0.089**</td>
<td>0.116**</td>
</tr>
<tr>
<td>Algebraic connectivity</td>
<td>0.386**</td>
<td>0.157**</td>
<td>0.586**</td>
</tr>
</tbody>
</table>

*: p-val < 0.05; **: p-val < 0.01

![Spearman coefficients of correlation between the change propagation indices and the architectural features](image)

The two correlation indices give similar answers, even though differences exist. Most of the architectural features proposed are significantly correlated to all three indices. Notable exceptions are Cost, Uncertainty and
the total number of subsystems in modules (but only for the Pearson coefficient). Spearman coefficients differ significantly from the Pearson coefficients for:

- Clustering coefficient
- Total number of subsystems in modules
- Number of bus elements
- Average number of bus elements’ interfaces
- Assortativity
- Cost
- Closeness

In all cases, the correlation is not clear from the visual examination of the data (see scatterplot in Appendix A); for these relationships, the form of the bond is highly non-linear. However, the two types of correlation indices are in accord about the rejection of the null hypothesis in all cases.

Table 6.4 also highlights the strength of the correlation between the indices. Three levels are defined:

- Weak correlation (correlation coefficient between -0.25 and 0.25); highlighted in green;
- Medium-strength correlation (correlation coefficient between -0.75 and -0.5 or between 0.5 and 0.75); highlighted in orange;
- Strong correlation (correlation coefficient between -1 and -0.75 or between 0.75 and 1); highlighted in blue.

Just three exhibit strong correlations with the change propagation indices: Uncertainty for ICL, Cost for ICI and degree for OCR. Degree, number of subsystems, number of interfaces, average density, diameter, energy, number of component loops and algebraic connectivity can be used as early detectors of Incoming Change Likelihood or Outgoing Change Risk. ICI, on the other hand, is weakly correlated with anything but degree.

Among the three change propagation indices, OCR is the one more sensitive to architecture, as on average it is more correlated with the architectural indices and it is not affected by native changes likelihood or impact (i.e. Uncertainty and Cost). The remaining architectural indices show weak correlations; in particular, modules and bus elements correlates very weakly with the change indices.

While most of the architectural features are positively correlated with change propagation indices, the network diameter and the measure of modularity can decrease them. Systems with a larger diameter have lower risk of change propagation because changes have to travel more to reach other components, while modules trap changes within confined areas of the system. The negative correlation between MS and the change propagation indices confirm the results of the analysis in Section 2. Closeness centrality is negatively correlated to ICL, but positively correlated to ICI and OCR; however, the correlation is so weak and the relationship with ICL is so non-linearity that the coefficient of correlation can be biased by the noise in data.

Several attempts have been made in order to find a valid linear regression that could characterize better the relationship between architectural features and the change propagation indices. Both traditional methods like stepwise regressions or lasso regression, and modern methods like symbolic regression (Schmidt and Lipson 2009) have been employed; unfortunately, no acceptable functions were found.

5. Conclusions and design recommendations

This Chapter showed how architectural elements (like modules and busses) or architectural features (like the centrality of a subsystem or the structural complexity of the system) can affect change propagation in complex technical systems.

As far as the impact of architectural elements is concerned, the results of the Kruskall-Wallis test clearly indicate that module and bus elements play an impactful role on change propagation behaviour. In general, the indices’ probability density functions have a positive skewness and present a long tail, as predicted by previous literature. In particular, the presence of modular elements has a profound impact on the indices distributions, since it reduces the average and increase the asymmetry. Modules decrease the average Incoming Change Likelihood by 26%, the Incoming Change Impact by almost 12% and the Outgoing Change Propagation index by the 30% circa. The bus elements’ presence, even though relevant, has smaller effects than modules’ presence.

The second part of the case study focused on the correlation between architectural features (measured by 20 architectural indices) and the change propagation indices. First of all, the analysis confirmed the relevance of
architecture in change propagation, as all architectural indices are significantly correlated to at least one change propagation index. However, the noise in the data and the high variance also indicates that architectural features are able to justify only partially the values of the change propagation indices. Furthermore, some change propagation indices are more sensitive to architecture than others and some architectural features are more important than others.

The Outgoing Change Risk is the index with the highest average correlation with the architectural indices. This means that the effects of changing a component are significantly influenced by the interface layout of the system itself. If the designers’ goal is to provide changeability during subsequent lifecycle phases, he/she must take architectural decisions about it. On the contrary, Incoming Change Impact is weakly correlated to all the architectural indices proposed but degree and native changes’ costs. This can indicate that the cost caused by changes in a system is dominated by the cost of changing the subsystem and cannot be reduced significantly with architecting. This implies that agile methods can be applied only in system whose components costs are low. Finally, Incoming Change Likelihood is in between OCR and ICI. It is influenced significantly by several architectural indices, but the likelihood of native changes is preponderant.

As far as the architectural features are concerned, degree is the most important one, since it is strongly correlated to OCR and is well correlated to ICL and ICI. This is obvious, as engineering changes can only propagate through interfaces. The same observation can be made for other architectural indices that are related to the number or density of interfaces, like the algebraic connectivity. The second most relevant predictor of change propagation is the number of component loops, i.e. the number of closed feedback loops between subsystems. This can provide a useful suggestion to system architects: while the number of interfaces is often constrained by functional and structural requirements, the reduction of component loops can be achieved just by rearranging the system interfaces. Finally, also structural complexity is very well correlated with ICL and OCR. This confirms the general perception that complex system are more sensitive to change propagation.

These results of both analyses confirm the beneficial role of modules for changeability, as reported in previous literature, but also characterize more precisely how architectural choices affect change propagation. Modularity had already been qualitatively associated with relative ease of change in the engineering design literature, without measuring its impact on change propagation quantitatively. For example, (Fricke and Schulz 2005) consider modularity and encapsulation adoption as a basic principle for changeability implementation. In (Saleh et al. 2009), the authors state that modularity and platform design are relevant fields for the study of flexibility in product design, while (Holttta-Otto and de Weck 2007; Gu et al. 2009) takes modularity into consideration in the development of a design method for adaptability. Finally, (Engel and Reich 2012) propose the use of modules as building blocks to evaluate real options at the architecture level. At the same time, the MS measure of modularity is weakly correlated to the change propagation indices; therefore, the benefits of modularity for change propagation are not straightforward. These benefits may lie more in the modules interfaces standardization than in the interface layout alone. On the other hand, many measures of modularity can be found in literature, therefore a different metric can lead to stronger correlations. In-depth research on actual systems may support one of these hypotheses.

As far as cyber-physical platforms are concerned, the results highlight both positive and negative aspects. This kind of technical systems generally has a high number of interfaces and several subsystems, and the communication network is very central and has a high degree, therefore it can be predicted that it is very sensitive to change propagation. On the other hand, the network protocol usually follows international standards and remains stable through time; moreover, their intrinsic modularity of subsystems and their large diameter have positive effects on changeability. The changeability of cyber-physical systems can be further improved by reducing the number of component loops and taking precautions on highly interconnected subsystems, like for example databases.
Chapter 8

Strategic design of a customizable smartphone

This chapter presents the main case study of the thesis, focused on the strategic design of a customization-intense modular cyber-physical device. On one hand, the Chapter further clarifies the application of the analyses shown in Part 2; on the other, it shows the analyses’ applicability and usefulness thanks to a concrete example. Furthermore, the results provide insights on the design and behaviour of a specific industry platforms, which can be generalized to many other cyber-physical systems.

After a short description of the device (Section 1), the entire set of analyses will be applied: Section 2 focuses on change propagation for different platform architectures, Section 3 concentrates on the value analysis of the initial set of modules and Section 4 studies the socio-technical dynamics of the platform and its ecosystem. Section 5 summarizes the conclusions and provides suggestions to improve the design of industry platforms.

1. Project ARA, the modular customizable smartphone

The case study refers to a modular, customization-intense smartphone called Project ARA. Project ARA is an electronic device made of a central platform core, called “endoskeleton” or simply “endo,” and several modules containing the electronic components that allow the technical system to function. The modules are available on a marketplace similar to the one already available for software applications; they are developed by independent designers, who are free to join the platform ecosystem. The distinctive feature of the product is the empowerment of smartphone owners, who can select the most suitable smartphone configuration and update their device module after module, without the need to buy a completely new product. With respect to the classification in (Stucheli and Meboldt 2013), Project ARA is an Internal network cyber-physical system that can easily become a Pervasive Network cyber-physical system.

Project ARA is currently being developed by the Google ATAP (Advanced Technologies and Programs) division, which was firstly inspired by Phoneblocks (Hakkens 2015), a modular smartphone concept created by Dave Hakkens in order to reduce e-waste (see Chapter 1, Section 1). Usually, mobile phones and other electronic devices are substituted even though most of the phone’s components can still function correctly; thanks to modularity, Phoneblocks owners could change only part of the phone and maintain the rest. This concept has generated great interest in the media and the public (Peckham 2013) and has inspired several other devices, like other smartphones or smartwatches.
What distinguishes ARA from other Phoneblocks-inspired devices is the presence of a multi-sided market. Project ARA is a proper industry platform, in that it connects at least three communities: the customers, who can enjoy a customizable phone, the module developers, who can profit from the creation of the electronic components, and software developers in the software platform ecosystem. The changeability of the modules and the scope of the project make ARA a milestone in the history of industry platforms and can deeply influence the next generation of electronic devices.

The key feature supporting this potential is the internal UniPro bus, which allows any generic module to communicate with the other components in the platform. The UniPro bus and the endoskeleton are respectively the cyber and physical part of the platform core, whose synergy allows the changeability of the phone. For this reason, Project ARA can be considered a cyber-physical industry platform, whose hardware (the endoskeleton and the modules’ electronics), software (the phone operating system and the modules firmware) and network (the UniPro bus and its interfaces) have to be designed together in order to allow system-level capabilities (customizability). Furthermore, the technical system alone is necessary but not sufficient, since the community of developers has a fundamental role in providing the components necessary to the device: for these reasons, Project ARA must be studied as a socio-technical system made by a set of technical features and people that evolve in time.

While these features are the foundations for the potential success of the device, they also generate a high level of complexity. The smartphone can have hundreds of module instances, and thousands of possible platform configurations; this makes the choice of the module to be prioritized and tested not trivial. Furthermore, socio-technical interactions imply that the design and the management of the platform must be considered together, adding a new layer of complexity to the system. Finally, since the system is dynamic, the entire lifecycle of the platform and the community must be taken into consideration. Other challenges originates from the novelty of the technical system. Project ARA is one of the first electronic devices that is designed as an open, modular platform; therefore, there is a high level of uncertainty surrounding both the technical and the market features.

For these reasons, the Google ATAP team sponsored a research project at the Massachusetts Institute of Technology in order to scrutinize the strategic design challenges of Project ARA. This Chapter is the result of the individual research efforts carried out inside that larger research framework and means to provide guidance.
in the strategic design decisions about Project ARA. The research was developed in a period of 15 months, from September 2014 to November 2015; the collaboration with Google ATAP group was fundamental in order to understand the difficulties in the project and to collect precious information about it. The continuous support of the Google team was also fundamental in the evaluation of the results of the analysis and the feedbacks provided contributed significantly to the development of the research. Given the involvement with the project development team, this case study can be considered an example of embedded research (Lewis and Russell 2011) in industry platform design.

This thesis will focus on the Spiral 2 version of the device, which Google ATAP intended to release in Puerto Rico as a first market pilot. The technical architecture is based on the Module Development Kit version 0.2 (Google ATAP 2015); the modules and the Developers data have been collected from actual developers during the second Developers conference, which took place in February 2015. The conjoint analysis data were collected in April 2015.

The case study is centred on technical aspects about the hardware part of the system. While the interaction between software and hardware is fundamental in cyber-physical systems, the Android software community is already large and mature, thus it is supposed that the software part of the platform will be able to adapt to the hardware customization. For this reason, the platform architecture will not consider the operating system and the software apps, while the ARA ecosystem will be simplified into a two-sided market, the users and modules developers.

2. Case study characterization

In this section, all the relevant information about Project ARA and its community are collected. For the sake of clarity, different data collection processes taking place in the three analyses have been grouped together: platform architectures were generated at the beginning of the Change Investigation Technique (Chapter 4, Section 2); Users preferences and the modules financial data were obtained according to the Information gathering phase in the Value Analysis (Chapter 5, Section 3.2), while AbMX was parametrized according the Characterization and Parametrization framework in Chapter 6.

Project ARA architectures are bounded by the endoskeleton architecture and the module compatibility. The endoskeleton architecture can be represented on different abstraction layers highlighting the set of structural, energy or information interfaces. According to the latest Modules Development Kit (MDK) (Google ATAP 2015), the energy and information architecture can be further differentiated into seven layers, but their specifications have already been fixed. The structural level, on the contrary, can have different layouts; in this thesis, the reference layout is the medium one, consisting of two frontal module slots and eight back slots (Figure 8.2). One of the modules on the front is bounded to be a screen, while the slots are able to contain any kind of electronic component, as long as it respects the platform requirements and specifications.

The geometry of the modules on the back is fixed, as dictated by the aesthetic rules given by the industrial design team, and can be subdivided into little squares. Four modules have size two by one, two modules are two by two, and two modules are one by one. The remaining module on the front is as large as the endo, but very slim.
These slots can be filled by customers with sensors, batteries, processing units and screens. The list of module classes was collected from informal discussions with the project management team and from questionnaires answered by the ARA developers’ community. Table 8.1 shows the 21 classes, plus the endoskeleton.
Table 8.1: List of modules for Project ARA

<table>
<thead>
<tr>
<th>TYPE</th>
<th>PRIMARY FUNCTION</th>
<th>PERFORMANCE LEVEL</th>
<th>PRICE TO CUSTOMER [$$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endoskeleton</td>
<td>Supporting and connecting the modules</td>
<td>---</td>
<td>125</td>
</tr>
<tr>
<td>Processing unit</td>
<td>Processing data</td>
<td>---</td>
<td>24</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>Displaying information</td>
<td>High resolution</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Receiving inputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen (basic)</td>
<td>Displaying information</td>
<td>Mid resolution</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Receiving inputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>Generating sounds</td>
<td>Hi-fi quality</td>
<td>16</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>Generating sounds</td>
<td>Standard quality</td>
<td>12</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>Communicating data</td>
<td>Optimal reception everywhere</td>
<td>34.5</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>Communicating data</td>
<td>Good reception</td>
<td>26</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>Taking pictures</td>
<td>Professional quality</td>
<td>29</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>Taking pictures</td>
<td>Amateur quality</td>
<td>22</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>Connecting to other devices</td>
<td>Fast data transfer rate</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Connecting to electric plugs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>Connecting to other devices</td>
<td>Slow data transfer rate</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Connecting to electric plugs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>Monitoring environmental conditions</td>
<td>---</td>
<td>20</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>Capturing health data</td>
<td>---</td>
<td>20</td>
</tr>
<tr>
<td>Security sensor</td>
<td>Preventing unauthorized accesses</td>
<td>---</td>
<td>20</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>Provide inputs to videogames</td>
<td>---</td>
<td>20</td>
</tr>
<tr>
<td>Memory (advanced)</td>
<td>Storing data</td>
<td>256 GB</td>
<td>56</td>
</tr>
<tr>
<td>Memory (intermediate)</td>
<td>Storing data</td>
<td>64 GB</td>
<td>45</td>
</tr>
<tr>
<td>Memory (basic)</td>
<td>Storing data</td>
<td>16 GB</td>
<td>34</td>
</tr>
<tr>
<td>Battery (basic)</td>
<td>Storing and providing energy</td>
<td>250 mAh</td>
<td>6.5</td>
</tr>
<tr>
<td>Battery (intermediate)</td>
<td>Storing and providing energy</td>
<td>500 mAh</td>
<td>8.5</td>
</tr>
<tr>
<td>Battery (advanced)</td>
<td>Storing and providing energy</td>
<td>750 mAh</td>
<td>10.5</td>
</tr>
</tbody>
</table>

Each module has a well-defined primary function and performances, as it can be noticed in the last columns of Table 8.1. The list of available can be extended, but a compromise between completeness and complexity had to be found. Furthermore, a comprehensive list of performance indices can be derived in a detailed case study; here, generic “performance levels” were imagined, each one clustering several detailed technical parameters. Some modules, like the most advanced battery or the highest-capacity memory module cannot be found on the market, and are here utilized as potential research and development directions.

The cost of the components was derived from freely available cost breakdowns in integral phones (AppleInsider Staff 2015; Fairphone 2015; Keizer 2015). The cost of components was increased by a factor of
20% in order to consider the increment in complexity when the components become independent modules; then, the various financial figures in Table 8.2 were derived based on the cost model of Fairphone (Fairphone 2015), another modular smartphone on the market. Cost forecasting is a complex activity, especially for completely new concepts. Actual financial entries may be different from the costs proposed, which have been independently obtained; future research will benefit from actual markets data. The cost considered is the one-time payment price from independent retailers, not the recurrent price given by subsidized contracts with wireless providers.

Table 8.2: Costs of ARA modules

<table>
<thead>
<tr>
<th>MODULE CLASS</th>
<th>NON-RECURRING ENGINEERING COSTS</th>
<th>MANUFACTURE SET-UP COSTS</th>
<th>OPERATIONS</th>
<th>PRODUCTION COST (SINGLE MODULES)</th>
<th>SALES COSTS (SINGLE MODULES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen (basic)</td>
<td>920061.84</td>
<td>434938.33</td>
<td>697016.55</td>
<td>56.06</td>
<td>0.75</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>1226749.12</td>
<td>579917.77</td>
<td>929355.40</td>
<td>74.75</td>
<td>1.00</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>122674.91</td>
<td>57991.78</td>
<td>92935.54</td>
<td>7.48</td>
<td>0.10</td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>163566.55</td>
<td>77322.37</td>
<td>123914.05</td>
<td>9.97</td>
<td>0.13</td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>204458.19</td>
<td>96652.96</td>
<td>154892.57</td>
<td>12.46</td>
<td>0.17</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>204458.19</td>
<td>96652.96</td>
<td>154892.57</td>
<td>12.46</td>
<td>0.17</td>
</tr>
<tr>
<td>Security sensor</td>
<td>204458.19</td>
<td>96652.96</td>
<td>154892.57</td>
<td>12.46</td>
<td>0.17</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>263751.06</td>
<td>124682.32</td>
<td>199811.41</td>
<td>16.07</td>
<td>0.22</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>351668.08</td>
<td>166243.09</td>
<td>266415.21</td>
<td>21.43</td>
<td>0.29</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>204458.19</td>
<td>96652.96</td>
<td>154892.57</td>
<td>12.46</td>
<td>0.17</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>61337.46</td>
<td>28995.89</td>
<td>46467.77</td>
<td>3.74</td>
<td>0.05</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>81783.27</td>
<td>38661.18</td>
<td>61957.03</td>
<td>4.98</td>
<td>0.07</td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>220814.84</td>
<td>104385.20</td>
<td>167283.97</td>
<td>13.46</td>
<td>0.18</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>294419.79</td>
<td>139180.26</td>
<td>223045.30</td>
<td>17.94</td>
<td>0.24</td>
</tr>
<tr>
<td>Memory (16 GB)</td>
<td>343489.75</td>
<td>162376.98</td>
<td>260219.51</td>
<td>20.93</td>
<td>0.28</td>
</tr>
<tr>
<td>Memory (64 GB)</td>
<td>457986.34</td>
<td>216502.63</td>
<td>346959.35</td>
<td>27.91</td>
<td>0.37</td>
</tr>
<tr>
<td>Memory (256 GB)</td>
<td>572482.92</td>
<td>270628.29</td>
<td>433699.19</td>
<td>34.88</td>
<td>0.47</td>
</tr>
<tr>
<td>Battery (250 mAh)</td>
<td>64404.33</td>
<td>30445.68</td>
<td>48791.16</td>
<td>3.92</td>
<td>0.05</td>
</tr>
<tr>
<td>Battery (500 mAh)</td>
<td>85872.44</td>
<td>40594.24</td>
<td>65054.88</td>
<td>5.23</td>
<td>0.07</td>
</tr>
<tr>
<td>Battery (750 mAh)</td>
<td>107340.55</td>
<td>50742.80</td>
<td>81318.60</td>
<td>6.54</td>
<td>0.09</td>
</tr>
<tr>
<td>Processing unit</td>
<td>245349.82</td>
<td>115983.55</td>
<td>185871.08</td>
<td>14.95</td>
<td>0.20</td>
</tr>
</tbody>
</table>

All the platform architectures were assembled thanks to an externally developed architecture topology enumerator (Shouqargerian 2016). The tool elaborates the modules specifications and the architecture feasibility rules in order to compute the entire set of feasible architectures. The enumerator is composed by four parts: encoding, grammar, search algorithm and evaluation. In the encoding phase, system architectures are transformed into DS2M, a generalization of Design Structure Matrices that consider all the set of possible interfaces in a group of components. Grammar removes from the infeasible interfaces between components directly; unfeasibility may arise because of system constraints or requirements. To navigate very large design spaces, the tool uses a convex hull intersection algorithm. Finally, the various designs can be ranked according to performance indices in order to highlight the most desirable architectures. A total number of 21,168 feasible ARA architectures were generated.
While Table 1 presents the modules’ primary functions, the system is characterized by at least two emerging or system-level performances: battery life and responsiveness. Battery life is the maximum time a device can be used from full charge state to full discharge state. It is a function of the battery storage capacity, the number and type of the components in the device, the battery management software and the use profile, i.e. what the phone is asked to do. Responsiveness it the time lag between an input and an output. It depends on the processing unit, the available memory, the set of components in the device, the software manager and the use of the device. These two features cannot be associated to a single module, even though battery and processing unit have a large impact on the two emergent properties, respectively. Both emergent properties are relevant for the user experience and therefore for the value of the device, but while computational power is not a critical constraint in electronics, battery life is. It is not hard to find harsh customers’ reviews about devices that offer high performances in computations, screen brightness and sensors’ accuracy, but fail to last more than a day without being charge. For this reason, in the remainder of the case study the attention will be focused on battery life alone.

Battery life was estimated thanks to a simulation tool developed by other members of the research team involved in Project ARA. The simulator assembles a virtual platform architecture and evaluates the components power consumption during typical usage modes like talking, web browsing or texting. An average use profile was taken from an international survey (Informate 2015), and the average battery life was derived for all the platform architectures. The granularity of the model and the lack of physical prototypes prevented from examining specific platform configurations power consumption; it is therefore assumed that all modules in the same class consume the same energy in the same time. The battery model was verified by a comparison with actual smartphones data; the error between real and simulated battery lives was smaller than 15% in all the usage modes assessed. As mentioned in Chapter 5, modular systems usually show lower performances compared to integral ones. Sections 4 and 5 will highlight the effects of modularity penalty on the device’s value and the ecosystem evolution, respectively. However, a Project ARA architecture can have up to two batteries a part from the internal battery, all of which can work in parallel to increase the overall battery life. Since in the Module Developers Kit there is no indication about the size of the battery internal to the endoskeleton, a 1000 mAh battery was assumed.

As far as Users’ preferences are concerned, a Conjoint analysis questionnaire was developed (Appendix C). Customers’ part-worth utilities were investigated through a self-explicated conjoint analysis (Rao 2014). Self-explicated conjoint analyses have the advantage of being based on an intuitive and relatively short questionnaire compared to other types of conjoint analysis; furthermore, they have demonstrated a high degree of robustness (Srinivasan and Park 1997). The main drawback of this type of conjoint analysis lies in the absence of mixed terms: the effect of synergies between multiple modules cannot be evaluated and it can actually generate biased results. Moreover, asking respondents an evaluation on technical facts may be confusing. Previous research showed that the combined use of House of Quality with Conjoint analysis data can be very effective (Kazemzadeh et al. 2009); however, it is believed that the modules classes detailed in Table 8.1 are generic enough to be easily understood also by respondents without a detailed technical knowledge.

The questionnaire was administrated to 200 people in a Puerto Rican mall through in-person interviews. The sampling was based on sex and age. 49% of the respondents are male, the rest female; 36% of the respondents are younger than 30, 33% is aged 30 to 40 and 31% is 45-years-old to 64-years-old. The questionnaire consisted of 35 questions and took approximatively 15 minutes to be completed. The results have then been clustered into five groups thanks to Ward’s algorithm on customers’ part-worth utilities. Details about the five cluster’s characteristics are given in Table 8.3. Since battery life and price are continuous variables in the model, they have been interpolated thanks to an exponential model and a linear model, respectively.

As it can be seen in the part-worth numeric values in Figure 8.3, basic-functionalities represents people who like basic phone features like long battery life, screen modules and audio modules, but are not interested in novel sensors or high-resolution cameras. Price-sensitive is the cluster defined by high price sensitivity and a low maximum price threshold. Performance premium comprises respondents who like advanced modules like high-definition displays, hi-fi loudspeakers and are not particularly price-sensitive. The Balanced group groups respondents who do not present particular preferences, but like advanced sensors and have a large gradient in price sensitivity. Finally, the Enthusiast likes every aspect of a smartphone and is willing to pay a high price to have one.
Table 8.3: clusters of customers from conjoint analysis

<table>
<thead>
<tr>
<th>CLUSTER NUMBER</th>
<th>CLUSTER NAME</th>
<th>DISTINCTIVE FEATURE(S)</th>
<th>PERCENTAGE OF CUSTOMERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic-functionalities</td>
<td>Main interest in feature phone functionalities</td>
<td>17%</td>
</tr>
<tr>
<td>2</td>
<td>Price sensitive</td>
<td>Extremely sensitive to price</td>
<td>8%</td>
</tr>
<tr>
<td>3</td>
<td>Performance premium</td>
<td>High preference for high-performance modules</td>
<td>8%</td>
</tr>
<tr>
<td>4</td>
<td>Balanced</td>
<td>No distinctive feature</td>
<td>34.5%</td>
</tr>
<tr>
<td>5</td>
<td>Enthusiast</td>
<td>High utility for most of the modules, low price sensitivity</td>
<td>33.5%</td>
</tr>
</tbody>
</table>

Developers were modelled according to the answers collected from the participants of the second developers’ conference. Developers were classified according to their competences, their role in their work organization and the participant’s company size. A part from general characterization parameters, two important pieces of information were obtained: the number of intended modules and the probability that a certain module class was developed, which both informed the agent-based model. The data showed that most of the participants were part of small companies or start-ups, that more than half of the participants intended to design on or two modules and that the most popular modules are antennas, modules for fun and games and processing units.
Figure 8.3: Part-worth utilities for the five clusters of respondents
3. Change propagation evaluation for ARA architectures

This section will apply the Changeability Investigation Technique to the 21,168 feasible ARA architectures generated by the architecture enumerator tool. In this case, the goals are different from the general analysis in Chapter 7, as 8AM800 will be employed to highlight the module classes’ sensitivity to change propagation. Once the overall set of change propagation indices has been generated, the indices are classified according to the module class they belong. For each architecture, 20 CPM samples are created; as far as the propagated changes likelihood and impact are concerned, they are generated from a uniform random distribution. The likelihood and impact of native changes were assigned according to two different perspectives.

Following the change Ilities classification in Chapter 4, Project ARA’s changeability can be declined into two more specific life-cycle properties. On one hand, new endoskeletons and modules can be designed by developers to provide new functions, improve performances or remove defects. This Ility can be called “Flexibility by designers” or simply “Flexibility.” At the same time, the physical instances of the platform are changed by users every time they buy a new module. In this case, modules are substituted by users during the utilization phase: the Ility can be defined as “Adaptable Customizability.” The different in these two perspectives is not only semantic, as it affects the numerical values of native changes’ likelihood and impact (Table 8.4).

Table 8.4: Native changes parameters for the Project ARA changeability assessment

<table>
<thead>
<tr>
<th>MODULE CLASS</th>
<th>FLEXIBILITY</th>
<th>ADAPTABLE CUSTOMIZABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIKELIHOOD</td>
<td>IMPACT</td>
</tr>
<tr>
<td>Endoskeleton</td>
<td>0.0375</td>
<td>0.7500</td>
</tr>
<tr>
<td>Battery (250 mAh)</td>
<td>0.1125</td>
<td>0.0161</td>
</tr>
<tr>
<td>Battery (500 mAh)</td>
<td>0.1125</td>
<td>0.0161</td>
</tr>
<tr>
<td>Battery (750 mAh)</td>
<td>0.1125</td>
<td>0.0215</td>
</tr>
<tr>
<td>Screen (basic)</td>
<td>0.3000</td>
<td>0.0215</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>0.3000</td>
<td>0.0268</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>0.3000</td>
<td>0.0268</td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>0.3000</td>
<td>0.2300</td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>0.3000</td>
<td>0.3067</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>0.3000</td>
<td>0.0307</td>
</tr>
<tr>
<td>Security sensor</td>
<td>0.3000</td>
<td>0.0409</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>0.3000</td>
<td>0.0511</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>0.3000</td>
<td>0.0511</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>0.3000</td>
<td>0.0511</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>0.3000</td>
<td>0.0659</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>0.3000</td>
<td>0.0879</td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>0.3000</td>
<td>0.0511</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>0.3000</td>
<td>0.0153</td>
</tr>
<tr>
<td>Memory (16 GB)</td>
<td>0.3000</td>
<td>0.0204</td>
</tr>
<tr>
<td>Memory (64 GB)</td>
<td>0.3000</td>
<td>0.0552</td>
</tr>
<tr>
<td>Memory (256 GB)</td>
<td>0.3000</td>
<td>0.0736</td>
</tr>
<tr>
<td>Processing unit</td>
<td>0.1125</td>
<td>0.0859</td>
</tr>
</tbody>
</table>

The native likelihood for Flexibility was assigned on an estimate of the likelihood of a change in design, while the native changes impact is proportional to the non-recurring engineering costs of the project. The native changes likelihood and the impact for adaptable customizability were derived from the expected numbers of modules substitutions and the modules costs, respectively.
Chapter 8

Strategic design of a customizable smartphone

The two changeability analyses also differ about the propagated change distribution that generates the Change Propagation Matrices. Module designers can create completely new modules, with novel features at functional, structural or behavioural level; for this reason, a wide range of changes can occur and generate change propagation.

On the contrary, users can only substitute the modules approved by the platform supporter. For this reason, it is assumed that change propagation is less likely to happen and less impactful. For the Flexibility analysis, the Change propagation matrices were derived from a uniform distribution ranging from 0 to 1; for the Adaptable Customizability analysis, the propagated change parameters $pcp$ were randomly generated from:

$$pcp = N(0.3, 0.3/8)$$

(8.1)

Figure 8.4 to Figure 8.6 show the distribution of the indices for the Flexibility analysis for four different module classes, which were taken as reference.

The four ICL distributions in Figure 8.4 are very different one to the other. The endoskeleton has a very thin distribution with a very small variance; battery life has an initial flat PDF, followed by a rapid decay. The processing unit has a slightly asymmetrical distribution centred on the value 0.06, while the audio module (which has a distribution very similar to all the other modular sensors) has a very large variance. Compared to ICL distributions, ICI distributions (Figure 8.5) show a similar variety, but dissimilar shapes. In this case, the endoskeleton distribution is the one most widespread, while battery modules have very small ICI indices. Processor modules’ and audio modules’ sensitivity to change impact is between the endoskeleton’s and battery’s. Finally, Figure 8.6 highlights how the Outgoing Change Risk is distributed according to the module class. The endoskeleton and the processor have a very similar distribution, while the battery OCR distribution resembles an exponential distribution. Finally, Audio OCR has an asymmetrical distribution with a long right tail. To sum up, all distribution are asymmetrical, with a high skewness and kurtosis; a synthetic description of the distributions must consider not only the mean, but also the standard deviation and the distribution skewness.
As far as ICL is concerned (Figure 8.7), three groups of modules can be highlighted: batteries and endoskeleton, the processing unit and the other modules. The last group contains the elements that are most likely to be changed; batteries and endoskeleton are the ones with the lowest average and median indices, while the processor has an ICL equal to 0.058. ICL is very sensitive to the likelihood of native changes (Chapter 7, Section 4); however, the endoskeleton has a significant tendency to be changed even though the native changes
likelihood is low; this effect can be attributed to its high degree. Furthermore, the skewness of endo and batteries distributions is 30% higher than the skewness of the other components, indicating highly asymmetrical distributions.

Figure 8.7: mean, standard deviation (left axis) and skewness (right axis) of ICL for the Flexibility analysis

Figure 8.8: mean, standard deviation (left axis) and skewness (right axis) of ICI for the Flexibility analysis
Figure 8.9: mean, standard deviation (left axis) and skewness (right axis) of OCR for the Flexibility analysis

Figure 8.8 shows the mean, standard deviation and skewness of the ICI distributions. In this case, the high investments required to develop a complex component like the endo dominate much of the other architecture. Other components with a significant ICI are the screen modules, the antennas, the memory modules and the processing unit.

OCR is the index most related to changeability, in that it measures the likelihood that a change in a component propagates to other components. As Figure 8.9 underlines, central components are very likely to propagate changes: in this case, the endoskeleton and the processing unit have a very high OCR compared to the rest of the system. Batteries, on the other hand, have the lowest tendency to propagate changes, on average; their distribution is also very right-skewed, as it can also be noticed from Figure 8.6.

A similar analysis can be carried out also for the Adaptive customizability of the ARA devices. In Figure 8.10 the ICL statistics are shown. The components with highest likelihood of change is the processor, while the endoskeleton is rarely changed. The other modules have ICL indices slightly smaller than the processor ICL, with the exception of the batteries, whose ICL values are half.

ICI statistics (Figure 8.11) emphasize that any change in the platform configuration can have a high impact on the endo or the screens; this pattern is probably influenced by the native changes impact. The ICI indices are very low, highlighting that this modular device require low effort to be changed.

The OCR distributions indicate a clear difference between highly central components (the endoskeleton and the processor) and peripheral components. The high standard skewness indicates that there may be a very high variability in the change propagation behaviour, depending on the specific platform architectures and the type of native change.
Figure 8.10: mean, standard deviation (left axis) and skewness (right axis) of ICL for Customizability analysis.

Figure 8.11: mean, standard deviation (left axis) and skewness (right axis) of ICI for Customizability analysis.
Figure 8.12: mean, standard deviation (left axis) and skewness (right axis) of OCR for Customizability analysis

The two changeability analysis have similar aspects and differences. They are similar in the shape of the change propagation indices distributions, which have long right tails and a relatively high standard deviation. These characteristics, which were found also in Chapter 7, make change propagation a serious threat to the design and use of the device, because the change propagation likelihood, impact and risks are not limited to few, well-defined intervals. The standardization of interfaces and a careful software design can mitigate the risks associated to this behaviour. At the same time, ICL, ICI and OCR distributions strongly depend on the type of Changeability considered. This supports the introduction and use of the change-relatedilities framework proposed in Chapter 4: different Changeability characterizations lead to different change features, which in turn generate different change propagation patterns. It is imperative therefore that the change lity under evaluation is clearly determined before the analyses are conducted.

From the Flexibility analysis emerges that the endoskeleton has a very low likelihood of change, but its change is very expensive in terms of resources; moreover, changing the endoskeleton can have severe consequences on the rest of the platform architecture. Likewise, the processing unit is central to the architecture and its change can trigger a cascade of other modifications; however, its mean ICL is very high, indicating that this component will be changed frequently due to native or propagated changes. The combination of high OCR and high ICL makes the processing unit a critical component, whose changes should be monitored carefully by developers. On the opposite end of the spectrum, battery modules have a low mean ICL, a negligible mean ICI and a small average OCR, therefore they are components less sensitive to change propagation inside the Product ARA architectures. The rest of the modules have a very high incoming change likelihood, a low incoming change impact and a great variability in the indices. It is possible to deduce that these modules can be a potential threat to change propagation, but that depends on the specific platform architecture considered.

The results of the adaptive Customizability analysis indicate that there is a net difference between the two most central modules (the endoskeleton and processor) and the rest of the modules. The endo has a low likelihood of change, but a very high ICI; the processor has the highest ICL and an average ICI; both have a high values of OCR that contrasts with the rest of the module classes. Since Customizability is a key feature of the cyber-physical platform and the OCR is the index most related to change propagation, the results suggest integrating the processing unit into the endoskeleton; furthermore, the software part of the endoskeleton must be particularly adaptable in order to accommodate the modules substitutions.
4. Value analysis

In this section, the value analysis detailed in Chapter 5 will be applied to Project ARA. The platform value analysis is structured in five steps: goal definition, information gathering, alternatives generation and evaluation, alternatives visualization and proposal development and presentation of results to stakeholders. The following subsections will describe how these steps allowed informing the platform developers about the most valuable Project ARA architectures and the most vital modules in the ecosystem.

4.1 Goal definition

The Project ARA value analysis has two objectives, as requested by the Google ATAP team: the discovery of the most valuable platform architecture among the set of feasible architectures and the determination of the modules that form the most valuable architectures. The list of most valuable architectures can prioritize the modules that will be available at the market launch, or it can inform the bundle strategy; furthermore, it allows testers to start the verification and validation phase from the most valuable architectures. Once the “best” architectures have been highlighted, the development of the modules inside the architecture must be supported, either through incentives or through a dedicated fee policy, so that these modules are always available in the market.

This case study also provides the means to highlight some conceptual aspects that have been introduced in Chapter 5: some example about the benefits plus or cost minus development strategies will be provided.

4.2 Information gathering

Most of the information presented in Section 2 were collected during the Information gathering phase of the Value analysis. As explained in Chapter 5, this phase has three main purposes: collecting the part-worth utilities of the platform through a conjoint analysis, clarifying the technical and economic features of the modules and the core and computing the Logit value of competitors.

For this case study, a self-explicative conjoint analysis was administrated to 200 respondents in the capital of Puerto Rico (Appendix C). The answers were then clustered into five groups, according to the part-worth utilities of the respondents (Figure 8.3).

The technical features of modules were derived from actual electronics components present in integral smartphones, the opinion of the Project ARA development team and the planned design projects of the developers’ conference participants. As already mentioned, there is a trade-off between the granularity of the modules description and the complexity of the analyses; after several iterations and debates, the module classes in Table 8.2 were selected. The economic parameters were derived from the cost of actual electronic components and the cost model of another modular phone (Fairphone 2015).

For this case study, the proposed competition analysis was only sketched. Inside the Puerto Rico conjoint analysis questionnaire, one section is dedicated to the evaluation of the utility due to different brand of famous smartphone producers. Unfortunately, the importance of branding is not as high as expected, and it does not discriminate between different smartphone in a significant way. It was learned that without a proper competition analysis, the customizability space cannot be defined; future research will focus on this aspect, which is more decisive than first expected.

4.3 Alternatives generation and evaluation

The feasibility of the modules combinations and their emergent properties were evaluated thanks to two numerical tools designed and programmed by other members of the research team. As explained in Section 2, the architectures were generated thanks to algorithms based on a feasibility rules and a convex hull exploration of the design space. The total number of architectures generated is 21,168. In this case study, the system-level property evaluated is battery life, which depends on both the battery module capacity, the classes of modules inside the architecture and the use profile. A numerical simulation tool was able to predict the number of hours every architecture could last without being charged.

The Logit value of the architectures was computed thanks to Equation 5.8. Since at this level of abstraction the value of customizability and the value of uniqueness cannot be computed, they are not considered in this value analysis; as Section 5 will demonstrate, they will be important to capture the value of Project ARA in a
dynamic market. Moreover, since only Project ARA platforms will be compared, the baseline value $V_0$ is the same for all the samples and can be neglected. Equation 5.8 then becomes:

$$V_j^{(h)} = \frac{\prod_i \left(U_{F,i}^{(h)} P_i^{(h)} \right)_{md} \prod_i \left(U_{F,i}^{(h)} P_i^{(h)} \right)_{emerg}}{U_c^{(h)} \left( \sum_w c_{w,md} + c_{plat} \right)}$$

(8.1)

As Table 8.3 showed, there are five average user profiles in the conjoint analysis data; therefore, the Logit value defined in Equation 8.1 must be computed five times for all the feasible platform architectures.

4.4 Alternatives visualization and Proposal Development

The Logit value computed in the previous step of the methodology must be visualized in a comprehensive, but understandable, illustration, so that the meaningful proposals can be advanced. First, the statistical distributions of Logit value are shown (Figure 8.13).

Cluster 1, Cluster 3 and Cluster 4 have similar distributions, even though the mean and the maximum are different. Cluster 2, composed of respondents particularly price-sensitive, has a very asymmetrical distribution, with a rapid decay in the right tail; Smartphone enthusiasts, on the other hand, have the most widespread distribution, the highest mean and the largest preferences for Logit value. In all clusters, there are few ARA architecture that dominate the others, which means that with the current module pricing the customizability of the device is not fully exploited. Table 8.5 summarizes the main features of the best ARA architectures according to the Logit value model, while Table 8.6 shows the modules inside the most valuable architectures for each of the five clusters.
Table 8.5: summary of the most valuable ARA architectures for each cluster

<table>
<thead>
<tr>
<th>INDICES</th>
<th>CLUSTER 1</th>
<th>CLUSTER 2</th>
<th>CLUSTER 3</th>
<th>CLUSTER 4</th>
<th>CLUSTER 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture code</td>
<td>12706</td>
<td>18178</td>
<td>18178</td>
<td>18106</td>
<td>17404</td>
</tr>
<tr>
<td>Logit value</td>
<td>5.7183</td>
<td>1.9130</td>
<td>4.6410</td>
<td>4.9839</td>
<td>9.4418</td>
</tr>
<tr>
<td>Estimated price [$]</td>
<td>377</td>
<td>379</td>
<td>379</td>
<td>388</td>
<td>428</td>
</tr>
</tbody>
</table>

Table 8.6: Modules inside the most valuable ARA architectures for each cluster

<table>
<thead>
<tr>
<th>MODULE CLASS</th>
<th>CLUSTER 1</th>
<th>CLUSTER 2</th>
<th>CLUSTER 3</th>
<th>CLUSTER 4</th>
<th>CLUSTER 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Security sensor</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory (16 GB)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Memory (64 GB)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory (256 GB)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Battery (250 mAh)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Battery (500 mAh)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Battery (750 mAh)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Processing unit</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Logit value alone can be a very useful indication; however, the trade-off between benefits and price can be further explored thanks to the use of trade-spaces. Figure 8.14 and Figure 8.15 show two alternative trade-space analyses for the platform architectures evaluated according to Cluster 4 part-worth utilities. Figure 8.14 compares the benefits value (the numerator in Eq. 8.1) with the price value (the denominator in Eq. 8.1). Since the exponential function amplifies the differences in the utilities, the architectures can be compared more easily as in Figure 8.15, where the benefits utility is drawn against the price utility (Equation 5.2). In the former comparison, two architectures have the same likelihood of being chosen if the ratio between benefits and price
values are the same; in the latter comparison, two architectures are chosen with the same probability if the difference between benefits and price utility is the same. In Figure 8.14, the utopia is in the low-right corner, while in Figure 8.15 it is located in the upper-right corner. In both cases, there are just 53 Pareto-front architectures, i.e. architectures that offer the maximum benefits at the minimum price; they represent the 0.2% of the total number of feasible architectures.

The architecture on the trade-space Pareto-front do not possess the same Logit value; they simply are the best trade-offs between price and benefits. For example, Table 8.7 compares five different architectures from different areas of the Cluster 4 Pareto-front. From the comparison, it is clear that, even though architecture belong to the Pareto-front, they may have very different Logit values. For example, the architectures considered vary from a total Logit value of 4.3924 to a total Logit value of 0.1925. Furthermore, the set of architecture highlight how the reduction of value due to price can be compensated by the increase in value thanks to emergent and modules utility. For example, Architecture 5494 is cheaper than architecture 3490, but the latter has an advanced interface and an advanced antenna (Table 8.8) that increase the value of the phone by 19%.

Table 8.7: summary of five Cluster 4 Pareto-front architectures

<table>
<thead>
<tr>
<th>INDICES</th>
<th>ARCH. 1</th>
<th>ARCH. 2</th>
<th>ARCH. 3</th>
<th>ARCH. 4</th>
<th>ARCH. 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture code</td>
<td>5494</td>
<td>3490</td>
<td>3184</td>
<td>7402</td>
<td>11925</td>
</tr>
<tr>
<td>Logit value</td>
<td>3.5762</td>
<td>4.3924</td>
<td>2.0121</td>
<td>1.0435</td>
<td>0.1925</td>
</tr>
<tr>
<td>Estimated price [$]</td>
<td>373</td>
<td>394</td>
<td>434</td>
<td>423</td>
<td>498</td>
</tr>
</tbody>
</table>

Table 8.8: Modules inside five Cluster 4 Pareto-front architectures

<table>
<thead>
<tr>
<th>MODULE CLASS</th>
<th>ARCH. 1</th>
<th>ARCH. 2</th>
<th>ARCH. 3</th>
<th>ARCH. 4</th>
<th>ARCH. 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Security sensor</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Memory (16 GB)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Memory (64 GB)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Memory (256 GB)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Battery (250 mAh)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Battery (500 mAh)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Battery (750 mAh)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Processing unit</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
This comparison can also be represented through a spider chart (Figure 8.16). In the chart, the benefits have been subdivided into battery life utility, multimedia modules (screen, audio and game interface) utilities, sensors (camera, medical, environmental and security modules) utility and essential modules (interface, processor and internal memory) utility. Even if all the selected architectures are the best trade-off between price and benefits, they are evaluated very differently. Architecture 5494 is the cheapest one, but also the one with the least number of features. On the contrary, Architecture 11925 has the highest benefits, thanks to the multimedia modules and the sensors utility; these rich features are counterbalanced by the lowest price utility, which is a non-linear function of price. Finally, Architecture 3184 is the most balanced one, thanks to a small increase in price and a small increase in modules utilities.

The value analysis also permits to compare the Logit value of architecture in different clusters. Figure 8.17 represents the same trade-space shown in Figure 8.15, but it contains the architecture evaluations according all the five clusters.
Figure 8.17: Tradespace utility analysis for ARA architectures evaluated according to all cluster’s part-worth utilities.

Even though the platform architectures are the same, they are evaluated dissimilarly by the five clusters. Cluster 5 is the one with highest evaluations and the mildest price sensitivity function, since the price value decreases very slowly with the increase in price. On the contrary, Cluster 2 is the most difficult to satisfy, due to a very steep price sensitivity function, which decreases the platform Logit value quickly. Cluster 1, cluster 3 and cluster 4 have similar Pareto Frontiers, even though the architectures contained vary. For example, Table 8.9 and Table 8.10 compare three different Pareto-front architectures that have the same Logit value according to different clusters’ evaluations.
The three architectures have some modules in common: the security module, a high-capacity antenna, an advanced interface for quick data transfer, and a high-resolution camera. Cluster 1 architecture also presents a low-resolution screen, one high-capacity battery, a 64 GB memory and an additional, low-resolution camera; Cluster 3 prefers the high-resolution screen to the low-resolution one, it trades the medical sensor for an environmental sensor, while it maximizes the battery life thanks to two high-capacity battery modules. The architecture chosen by Cluster 4 features a high-resolution screen, a professional-quality camera and two battery modules. Even though these three architectures appear quite different, the Logit value that the different clusters attribute is the same.
Once the Pareto-front architectures have been found, it is possible to scrutinize what are the modules inside them. Figure 8.18 shows the relative number of modules inside the Pareto-front architectures of the five clusters combined.

![Figure 8.18: percentage of module classes inside the five clusters’ Pareto-front architectures](image)

The most popular modules are the entry-level audio module, the low-resolution screen, the high-speed interface, the security sensor, and the 750 mAh battery. The modules selected are only partially correlated to the part-worth utilities in Figure 8.3, because the choice process is mediated by the feasibility of platform configuration and the utility of emergent properties. For example, displays are not very popular across the five clusters, but every ARA architecture needs one screen to function. Hi-fi audio modules are in theory more valuable than traditional audio modules, but they consume more energy to work; the combination of power usage and the highest price make them very unpopular. In other words, Figure 8.18 indicates that just considering the customers’ interest in certain module classes can lead to biased decisions and it justifies the value analysis methodology explained in Chapter 5.

The shape of the Pareto-frontiers and the architecture inside them can be influenced by modules, system-level functions and modules pricing. In this case study, how these factors can influence the Logit value of the architectures set has not been investigated; however, future research might include a pricing optimization algorithm to maximize the number of ARA architectures inside the Pareto-frontiers or a core topological optimization to find the best physical architecture.

Finally, trade-space analysis permits to highlight the importance of modules or emergent properties in the Logit value, which in case of Project ARA is the battery life. As it can be seen in Figure 8.19, battery life depends not only on the total battery capacity, but also on the modules in the architecture; for example, the variation in battery life for architectures with 2500 mAh amounts to 8 hours. Since battery life is associated to very high part-worth utilities (Figure 8.3), the value analysis can investigate whether the positive contribution of modules is counterbalanced by the decrease in battery life.
As Table 8.11 shows, not all Clusters have significant correlation between optimal architectures and battery life: Cluster 2 evaluations are probably influenced by the price sensitivity, while Cluster 5 modules part-worth utilities well compensates for the decrease in battery life.

<table>
<thead>
<tr>
<th>INDICES</th>
<th>CLUSTER 1</th>
<th>CLUSTER 2</th>
<th>CLUSTER 3</th>
<th>CLUSTER 4</th>
<th>CLUSTER 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pearson correlation</td>
<td>0.52</td>
<td>0</td>
<td>0.36</td>
<td>0.36</td>
<td>0.2167</td>
</tr>
<tr>
<td>Correlation p-value</td>
<td>2.00E-04</td>
<td>0.99</td>
<td>0.0123</td>
<td>7.20E-03</td>
<td>0.11</td>
</tr>
</tbody>
</table>

4.5 Presentation of results to stakeholders
The results of the analysis were presented to key members of Project ARA development team and positive comments were collected. In particular, the stakeholders were interested in supporting the most valuable modules, programming the testing process efficiently, quantifying the impact of battery life and in assessing the most profitable module pricing strategy. The value analysis shown in this Section supports the development effort: Figure 8.18 predicts the modules that are most chosen by customers, the Pareto-frontier architectures are the ones that must be selected most carefully, because they will likely be the most popular ones, while Table 8.7 indicates that not all customers will base their choice on battery life alone. The very small fraction of optimal ARA architectures suggests the pricing model described in Section 2 may be financially feasible for developers, but it could be improved in order to maximize the platform differentiation. Future research will address this challenge.
5. Ecosystem simulation

As already mentioned in Chapter 3, the design of an industry platform cannot avoid taking into consideration the dynamical evolution of the coupled multi-sided market. In this section, a set of scenarios will be advanced, and, thanks to AbMX, the influence of socio-technical features on the achievement of the critical mass, the two-sided market equilibrium point and the maximum market size is explored.

Among the large number of variables that could be investigated, this research work focuses on 15 input variables:

- Total number of potential adopters: the total size of the users’ side;
- Total number of potential supporters: the total size of the developers’ side;
- Number of Users: number of agents simulating users’ behaviour; the number of Users is generally lower than the number of potential adopters, therefore linear scaling has been adopted. A sensitivity analysis proved that over 10,000 Users, the simulation outputs are only marginally affected by the scaling factor;
- Number of Developers: number of agents behaving like developers; the number of developers is smaller than the number of users, so there is no need for scaling;
- Number of initial Adopters: number of Users that are inside the ARA ecosystem at the beginning of the simulation;
- Number of initial Supporters: number of Developers that are inside the ARA ecosystem at the beginning of the simulation; for the sake of simplicity, each initial Supporter is associated to just one initial module;
- Number of initial modules: number of modules that are inside the ARA ecosystem at the beginning of the simulation. A very wide range of potential scenarios could be imagined for the market launch; in this research, three sets of initial modules are considered: (1) a very limited initial set of modules, which resembles the components contained in the feature phones; (2) a mid-sized market launch, with 15 modules that represent the typical electronics contained in modern smartphones, and (3) a large initial choice of 20 modules, which contains innovative modules that no competitor offers. Table 8.12 details the three scenarios. Please note that in some scenarios the most valuable modules highlighted in Section 4 are not present at market launch.
- Maximum number of architectures in the configurator: in the real platform ecosystem, users have to choose between several platform architectures and configurations; this index is the maximum number of feasible architectures that a User considers when making a choice; it represents the effectiveness of the configurator in guiding users in the customization process. Architectures are ranked according to their value;
- $\beta$ interval: $\beta$ represents the influence that word-of-mouth has on the evaluation of a platform configuration (see Equation 6.5); inside the agent-based model, $\beta$ numeric values within a range are assigned randomly to Users;
- Users sensitivity to customizability: the numerical value of the parameter $B_{\text{cust}}^{(h)}$ in Equation 6.2;
- Percentage of malfunctioning architectures: the relative number of architectures that are believed to be feasible, but they are not. Each time a User chooses a malfunctioning architecture, the perceived value of the ARA platform is reduced by the numerical value of Value loss for each malfunctioning architecture;
- Value loss for each malfunctioning architecture: the loss in perceived value that all Project ARA architectures withstand when a malfunctioning ARA architecture is chosen (Equation 6.9); in this research, all cluster experience the same value loss $V_{\text{mal}}$;
- Resistance to innovation: the resistance of the market to novel modular electronic devices. Resistance factors to innovation are related to people’s resistance to change and can be classified according in functional barriers and psychological barriers (Claudy et al. 2015). For Project ARA, potential resistance factors are for example usage barriers related to the choice effort or risk barriers related to malfunctioning configurations. This aspect of the users behaviour has been modelled with a bonus on the baseline value $V_{0}$ of all integral smartphones;
- Virtual prototyping reduction of development time: the use of virtual prototypes, model-based systems engineering and design automation can reduce the development time of systems and the investments required for the engineering design (de Weck 2012). This scaling factor reduces the average development time linearly to understand the benefits of a speedup in engineering design;
- Virtual prototyping reduction of NRE costs: relative reduction of non-recurring engineering costs thanks to the use of virtual prototyping tools.
Table 8.12: number of module classes available at market launch according to three different scenarios

<table>
<thead>
<tr>
<th>MODULE CLASS</th>
<th>12 MODULES MARKET LAUNCH</th>
<th>16 MODULES MARKET LAUNCH</th>
<th>21 MODULES MARKET LAUNCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Screen (advanced)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Audio (advanced)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Environmental sensor</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Medical sensor</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Security sensor</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Antenna (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Antenna (advanced)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Gaming interface</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Interface (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Interface (advanced)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Camera (basic)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Camera (advanced)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Memory (16 GB)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Memory (64 GB)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Memory (256 GB)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Battery (250 mAh)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Battery (500 mAh)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Battery (750 mAh)</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Processing unit</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Each AbMX run simulates 8 years of ecosystem evolution; in this time arc, it is supposed the market reaches or approaches closely its equilibrium. Competition was modelled as ten integral smartphones, ranging from premium to basic, and inspired by real integral phones on the market. These parameters will be varied systematically in the sensitivity analysis described in Section 5.2 and Section 5.3; before that, Section 5.1 details the behaviour of the socio-technical system for a baseline case study.

5.1 Baseline simulation

Given the uncertainty related to Project ARA specifications, the complexity of users’ and developers’ behaviour, and the difficulty in predicting unprecedented events in markets over long periods of time (Taleb 2010), the results of the agent-based model are employed as explorative scenarios, rather than market forecasts. For this reason, the core of the research lies in the sensitivity analyses, which are able to determine whether a certain factor influences the ecosystem dynamics. However, the two-sided market dynamics is interesting per se, because it clarifies the interactions between the two sides and shows stylized facts that support the validation of the model. For this reason, the evolution of the main variables of the model is shown for a baseline case (Table 8.13).
For the baseline case study, a market launch in Puerto Rico is simulated with 10,000 Users and 100 Developers, simulating 2.44 million potential adopters (the scaling factor is 1 to 244) and 100 potential supporters (no scaling required). The initial adopters are 20, while the initial developers are 21, providing 21 initial modules (the third column in Table 8.8). The ARA configurator is able to suggest to customers the 750 “best” feasible platform architectures, the influence of word-of-mouth is centred on an average value of 0.5 and the sensitivity to customizability is set to a generic low value. There are no malfunctioning architectures, the resistance to innovation is low and there are no virtual prototyping tools helping modules developers in their design activity. Finally, the modularity penalty on battery life is to 20%. These would be the most likely variables for an actual market launch in Puerto Rico.

Figure 8.20 shows the Users side dynamics, Figure 8.21 displays the Developers side dynamics, while Figure 8.22 illustrates the number of modules in the market for each time step. The results of three simulations run have been averaged, and the 95% confidence bounds are provided.
Chapter 8

Strategic design of a customizable smartphone

Figure 8.20: number of Project ARA adopters over time for baseline simulation

Figure 8.21: number of Project ARA supporters over time for baseline simulation
Four phases can be highlighted. In the first year, the only modules available are the ones provided by the initial Developers, since the remaining part of the community requires time to launch its modules. However, a large number of initial adopters buys the platforms; this peculiar phenomenon has to be attributed to the high preferences expressed by Cluster 5, which forms most of the initial adopter’s community. Between month 13 and month 20, both the Users and the Developers side of the market grow. At month 20, the Project ARA initial adopters are saturated, then, from month 21 to 25 the followers who did not buy the platform at the market launch enter the community. At month 20, the Developers reach a market equilibrium. Unfortunately, the total size of the User market in Puerto Rico is too small to support the Developers community, therefore after two years the Developers community shrinks, until a new equilibrium point is reached; meanwhile, the number of ARA adopters diminishes slowly. The final number of Supporters is 14, so there are fewer Developers than the initial market; however, each Developer provides more than one module to the ARA market, as Figure 8.20 shows.

The baseline case study offers some means for an initial validation through stylized facts. Figure 8.18 and Figure 8.19 are the combination of several S-curves. S-curves are well-known functions emerging from other innovation diffusion models (Schilling 2013). Furthermore, the positive feedback relationship between two or more market sizes is central to the literature on multi-sided markets (Chapter 2, Section 2.3). In the simulation outputs, this relationship is clear and well characterized, and might provide insights on the characterization to models like the one suggested in (Kumar et al. 2010).

From the general observation of the baseline scenario and other scenarios (which present similar features), three conclusions can be inferred. First, the two-sided market simulated is not very efficient, in that several Developers and Users leave the ecosystem after an initial rapid inflation. This may be a consequence of the simple behavioural characterization of the Developers or it may be a general tendency of two-sided markets. Secondly, the two-sided market presents two equilibrium points, one unstable, the other stable. As a development strategy, the platform developer may decide to upgrade the core to a new version as soon as the unstable equilibrium is reached. AbMX is currently not able to simulate a multi-platform market; therefore, it is unclear whether this strategy can lead to more successful market equilibrium points or whether the cannibalization has detrimental effects. Finally, Puerto Rico is a very small community, which cannot sustain a large developers’ community; the ecosystem dynamics might be so heavily influenced by the number of potential adopters that the conclusions drawn may be biased. The next subsections focuses on this issue.
5.2 Potential market size influence on simulation outputs

The market analysis for the case study was conducted in Puerto Rico, which has a potential market of 2.44 million people; furthermore, the Developers conference suggested that about 100 developers are seriously intentioned in designing a module for ARA. However, the size of the market can influence the two-sided market dynamics, so that the results found in the analyses for Puerto Rico cannot be generalized to other industry platforms. To this end, a full factorial design of experiments was created, with the number of initial modules, the total number of potential adopters and the total number of potential supporters as inputs (Table 8.14). The total number of potential adopters varies from the population size of Puerto Rico, to the population size of Mexico to the population size of the United States of America. The number of potential adopters ranges between 100 and 200, while the initial modules varies according to Table 8.8. For each combination of parameters, three samples were synthesised numerically and averaged; the maximum community size and the final community size were collected (Figure 8.21).

Table 8.14: Design of experiments simulation inputs

<table>
<thead>
<tr>
<th>Simulation Input</th>
<th>Variant #1</th>
<th>Variant #2</th>
<th>Variant #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of potential adopters</td>
<td>2.44 millions</td>
<td>122 millions</td>
<td>300 millions</td>
</tr>
<tr>
<td>Total number of potential supporters</td>
<td>100</td>
<td>150</td>
<td>200</td>
</tr>
</tbody>
</table>

Figure 8.23: Influence of different market sizes on percentage of Users and Developers; the first index in the labels refers to the total number of potential Adopters variant, the second index refers to the total number of potential Developers variant from Table 8.14

Figure 8.23 shows the size of the two sides of the market when the markets reached an equilibrium (blue dots) and when they have reached the maximum size. The data labels are a reference to the full-factorial design of experiments: the first digit indicates the reference to total number of potential adopters (Table 8.10, first row), while the second digit indicates the total number of potential supporters (Table 8.10, second row). The Puerto Rico data-point are the ones beginning with the digit 1. Both the market equilibrium and the market maximum...
size are strongly affected by the number of potential adopters, but there is a sharp difference between Puerto Rico and the other two countries, indicating that the relationship is not linear. The number of potential Supporters seems to be negatively correlated to the final and maximum number of Supporters, probably because competition decreases the probability of Developers’ survival.

A formal correlation analysis, supported by the Pearson’s correlation coefficient, scrutinizes these relationships (Table 8.15). The analysis indicates that the number of potential adopters is more impactful than the number of potential supporters, as it influences four parameters in the ecosystem dynamics out of six, while the number of potential adopters is significantly correlated to just one factor. The total number of potential adopters is significantly correlated to the Users’ and Developers’ side equilibrium point and it influences the Users’ community maximum size, both in terms of magnitude and in terms of time. The number of potential supporters is correlated significantly only with the maximum size of the Developers’ community.

The findings have implications for market launch of the platform. Puerto Rico is a very small potential market that can supports only a limited community of developers; while there may be strategic reason for choosing a market launch in the US territory, the long-term sustainability of the platform may be seriously compromised by this choice. At the same time, the high number of smartphone enthusiasts can be a unique Puerto Rican feature; as such, it would be incorrect to use the Puerto Rican part-worth utilities as representative of other countries. Overall, Puerto Rico is a suitable test market for the ARA platform, but it is financial feasible for a limited amount of time; it can be considered a potentially successful beachhead market.

Table 8.15: Pearson correlation coefficients between total number of agents and ecosystem dynamics

<table>
<thead>
<tr>
<th></th>
<th>Users Community Size at Equilibrium</th>
<th>Developers Community Size at Equilibrium</th>
<th>Users Community Maximum Size</th>
<th>Developers Community Maximum Size</th>
<th>Time of Users Community Maximum Size</th>
<th>Time of Developers Community Maximum Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of potential adopters</td>
<td>0.76*</td>
<td>0.69*</td>
<td>0.76*</td>
<td>-0.12</td>
<td>0.67*</td>
<td>0.12</td>
</tr>
<tr>
<td>Total number of potential supporters</td>
<td>0.30</td>
<td>0.38</td>
<td>0.29</td>
<td>0.97**</td>
<td>-0.04</td>
<td>-0.37</td>
</tr>
</tbody>
</table>

*: p-val < 0.05; **: p-val < 0.01

5.3 Sensitivity analysis

The previous subsections highlighted that, even though some limitations are present, Puerto Rico can be a potentially successful initial market for Project ARA. In this subsection, hypotheses about the effect of socio-technical parameters will be tested thanks to a sensitivity empowered by AbMX.

As underlined in Chapter 3, the main thrust of the community built around an industry platform are the cross-side network externalities, i.e. the positive feedback loops that connect the sides of the multi-sided market. From the perspective of the platform developer, initiating this dynamic correctly is crucial; the initial modules should be interesting enough to attract the initial adopters and initiate the market dynamics. The prioritization of modules has already been addressed in the previous section, but that analysis had two assumptions: (1) it was static, in that all configurations where chosen at the same time and (2) the Developers side was not considered. The first assumption implies that customers choose the most desired configuration first and then they keep the platform frozen for the rest of the time; on the contrary, in a dynamical market customers can choose one configuration first and then slowly upgrade it until they reach the most desired one. This pattern can relax the need of having all the best configurations available from the start, as non-Pareto optimal configurations can be good enough to attract the initial set of adopters and escape the null equilibrium attraction. The second assumption neglects the fact that module developers have to profit; otherwise, they leave the ecosystem. For these reason, the first model input under evaluation is the number of initial modules, as
presented in Table 8.8. Literature showed that variety can be perceived by users as a source of frustration; however, in the current version of AbMX it is considered to be a positively correlated with value (Equation 6.2). Therefore, it is expected that:

**H1a:** an increase in the number of initial Supporters and initial Modules leads to a larger Users side equilibrium point and a larger Users side maximum size

On the other hand, having too many modules from the beginning (when the User market is still small) increases the competition between the initial Developers and therefore can reduce the growth of the developer’s side of the market.

**H1b:** an increase in the number of initial Supporters and initial Modules leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

Battery life is one of the most important features in the choice of a smartphone for many Users clusters (Figure 8.3). However, as mentioned several times in Chapter 4, modular systems tend to have worse performance than integral architectures. Since the battery life simulator is tuned on integral phones’ battery life, the modularity penalty can be set a posteriori, and its detrimental effect can be assessed in this sensitivity analysis.

**H2a:** an increase in the modularity penalty on battery life leads to a smaller Users side equilibrium point and a smaller Users side maximum size

**H2b:** an increase in the modularity penalty on battery life leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

The third sensitivity analysis input is the maximum number of architectures in the configurator. Configurators are fundamental to provide a pleasant customization experience (Fogliatto et al. 2012; Trentin et al. 2013). This parameter models the effectiveness of the configurator in guiding users across the large choice options offered by the industry platform; an increase in this parameter should lead to more satisfying Users choices.

**H3a:** an increase in the maximum number of architectures in the configurator leads to a larger Users side equilibrium point and a larger Users side maximum size

**H3b:** an increase in the maximum number of architectures in the configurator leads to a larger Developers side equilibrium point and a larger Developers side maximum size

Word-of-mouth influences Users’ choices through the parameter $\beta$. As shown in (Delre et al. 2010; Goldenberg et al. 2010), if the Users population is very sensitive to other agents’ opinion, the innovation spreading process is damped, since the positive opinion of satisfied initial adopters has to spread across the social network before the major part of the population takes the novel product into consideration.

**H4a:** an increase in the average importance of word-of-mouth leads to a smaller Users side equilibrium point and a smaller Users side maximum size

**H4b:** an increase in the average importance of word-of-mouth leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

The value definition in Equation 6.1 comprises the value of customizability and the value of uniqueness, which are evaluated according to Equation 6.2 and Equation 6.3. The last two equations contain two part-worth utilities ($b_{\text{cust}}^{(h)}$ and $b_{\text{uniq}}^{(h)}$) that cannot be easily computed with traditional conjoint analyses. For this reason, this sensitivity tries to quantify the effect of these two inputs on the model outputs.

**H5a:** an increase in the Users’ sensitivity to customizability leads to a larger Users side equilibrium point and a larger Users side maximum size

**H5b:** an increase in the Users’ sensitivity to customizability leads to a larger Developers side equilibrium point and a larger Developers side maximum size

**H6a:** an increase in the Users’ sensitivity to uniqueness leads to a larger Users side equilibrium point and a larger Users side maximum size

**H6b:** an increase in the Users’ sensitivity to uniqueness leads to a larger Developers side equilibrium point and a larger Developers side maximum size
Every time an innovation is introduced on the market, users have to adapt to novel features, novel usage modes or break with some traditions. This generates a resistance to innovation, which can have impactful consequences on the diffusion of the innovative product, especially in mature populations (Laukkonen et al. 2007). It is therefore assumed that:

**H7a:** an increase in the Users' resistance to innovation leads to a smaller Users side equilibrium point and a smaller Users side maximum size

**H7b:** an increase in the Users' resistance to innovation leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

Project ARA offers to the Users the freedom to choose the electronic device that is most suitable to their needs. However, one of the main challenges of this project, and of cyber-physical systems in general, is the verification and validation of hundreds of thousands of possible platform configurations (Gunes et al. 2014; Wolf and Feron 2015). Traditional testing methods based on physical prototypes cannot be scaled for such a huge variety and new testing methods are still under development in the research community (Suh et al. 2013). At the same time, innovative products diffusion can suffer from negative word-of-mouth, which can determine the failure of a product rapidly, even if the malfunctioning is fixed soon after the market launch (East et al. 2007; East et al. 2008). AbMX can randomly choose a certain fraction of the total architectures database and make them “malfunctioning”: every time a User chooses a malfunctioning architecture, her perceived value about all the ARA architectures is reduced by a certain value loss.

**H8a:** an increase in the number of malfunctioning ARA architectures leads to a smaller Users side equilibrium point and a smaller Users side maximum size

**H8b:** an increase in the number of malfunctioning ARA architectures leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

**H9a:** an increase in the Users' value loss for malfunctioning architectures leads to a smaller Users side equilibrium point and a smaller Users side maximum size

**H9b:** an increase in the Users' value loss for malfunctioning architectures leads to a smaller Developers side equilibrium point and a smaller Developers side maximum size

Finally, innovation literature has underlined how toolkits for innovation can foster users innovation (von Hippel and Katz 2002). As the success of a customizable platform also depends on the variety and the originality of the modules available (Section 4.2), platform supporters may want to consider the development of these toolkits for users. The investments required for the toolkit design may be repaid by the increase in the number of potential modules developers. Furthermore, project META (de Weck 2012) showed that the use of model-based engineering tools for the virtual prototyping of complex vehicles reduced the engineering design time by a factor of 3.8, compared to traditional systems engineering. It would be interesting to assess the effect of virtual prototyping tools on industry platforms evolution. The increase in speed and the decrease in Non-recurring Engineering costs is expected to lead to more attractive platforms, since novel modules appear more frequently:

**H10a:** a reduction in the modules technical development time leads to a larger Users side equilibrium point and a larger Users side maximum size

**H10b:** a reduction in the modules technical development time leads to a larger Developers side equilibrium point and a larger Developers side maximum size

**H11a:** a reduction in the modules non-recurring engineering costs leads to a larger Users side equilibrium point and a larger Users side maximum size

**H11b:** a reduction in the modules non-recurring engineering costs leads to a larger Developers side equilibrium point and a larger Developers side maximum size

The sensitivity analysis was conducted thanks to a linear regression model (Saltelli et al. 2008). Five samples from an orthogonal design of experiments (see Appendix D) were computed, for a total number of 135 simulations. For each sample, the maximum market size and the market size after 96 months were collected as indicators of the market transient and the market regime; then, a linear regression model was fitted on the data-points from normalized inputs. These linear models indicate whether a coefficient is significant (i.e. if it influences the ecosystem dynamics) and allow a comparison of the importance of the significant coefficients. Each linear model was verified according to the guidelines in (Weisberg 2005); after having assessed that the residuals are not normally distributed for some the linear models, a robust fitting method was employed to all models, so that the coefficients are not influenced too heavily by the outliers. The complete models and the
test results are available in Appendix D. The simplicity of the linear models tested can be criticized, as in a complex systems the interaction terms can play a relevant role; however, these “simple” models can already explain most of the outputs variations, as their high R-square indices indicate.

Figure 8.24: Significant linear model coefficient on Users side equilibrium and Users side maximum size

Figure 8.24 presents the linear model significant coefficients for the Users side market. The Users market equilibrium is positively affected by the Users average sensitivity to uniqueness and time saving in the engineering phase, but it is negatively influenced by the importance of word-of-mouth in agent’s choices, the resistance to innovation and the number of malfunctioning architectures. The maximum size of the Users community can be increased thanks to a higher number of initial modules, a larger set of architectures that can be chosen in the configurator and a decrease in the importance of word-of-mouth. Furthermore, sensitivity to uniqueness, resistance to innovation, the percentage of malfunctioning architectures and the engineering time reduction thanks to virtual prototyping are other significant coefficients. Comparing the effect of the inputs on the outputs, the maximum market size is affected by a larger number of input variables; however, uniqueness and time saving have a larger effect on the market regime.

The Developers side is affected by the input variables differently (Figure 8.25). The market equilibrium is influenced by the number of initial modules (positive relationship), the word-of-mouth (negative relationship), the sensitivity to uniqueness (positive relationship) the resistance to innovation (negative relationship), the percentage of malfunctioning architectures (negative relationship) and the engineering time reduction (positive relationship). Similar relationships can be found for the maximum community size, but in this case, only the value loss for malfunctioning architectures is not significant. As in the Users side, all input variables have a similar effect on the outputs, but the maximum market size is influenced by more input variables than the market equilibrium.

The comparison between the two sides highlights how the input variables have asymmetrical effects on the ecosystem dynamics. On one hand, all the input coefficients affect the market dynamics with the same sign; a variable that increases the Users side also increases the Developers side, and vice-versa. This confirms the presence of cross-side network externalities. On the other hand, the strength of this influence and the number of significant variables are different. In particular, the battery penalty and the sensitivity to customizability are significant for the Developers side, but not for the Users side.
The initial number of modules is positively correlated to Users side maximum, the Developers side market regime and the Developers side maximum. However, it is not correlated to the Users market equilibrium. Thus, the H1a hypothesis is partially supported by the data, while H1b is completely supported by the data. The absence of evidence to support the direct influence of the initial modules on the Users side equilibrium indicate that the spontaneous growth of the Developers side in the long run can compensate the lack of initial modules; however, the largest size of the community can be achieved only with a large initial module variety.

![Figure 8.25: Significant linear model coefficient on Developers side equilibrium and Developers side maximum size](image)

The modularity penalty on battery life is significant only for the Developers community maximum size. This might indicate that a higher battery penalty shifts the Users choices towards battery modules with higher capacities, thus reducing the number of battery modules Developers. However, Project ARA configurators with large battery capacity are still able to compete against integral smartphones competition, thus the Users side dynamics is not affected by modularity penalty. H2a is not supported, while H2b is only partially supported.

The number of architectures that can be easily reached inside the configurator affects the community maximum size, but not its long-term equilibrium. Hypotheses H3a and H3b are therefore only partially supported. The unexpected lack of correlation can justified considering that in a small market like Puerto Rico only the most popular modules can survive in the ecosystem. The Users choices filter out the modules that are most appreciated, thus reducing the need for a wide selection of architectures. A market with a wider variance on preferences might be more sensitive to this variable.

Sensitivity to word-of-mouth is the most influencing input variable in the model, even if it is not significant for the maximum size of the Developers community. The linear model supports hypotheses H4a entirely and H4b partially. The lack of sensitivity of the maximum number of Developers is quite difficult to explain; it might imply that the initial Developers dynamics is only partially coupled with the number of Users, and it is more influenced by the number of modules that Adopters buy when they are part of the community.

Hypotheses H5a is not supported, while H5b is only partially supported: Users’ sensitivity to customizability affects only the maximum size of the Developers community. Since customizability is one of the most valuable features of Project ARA, this result highlights the need to find a better characterization of the value of customizability.
As far as sensitivity to uniqueness is concerned, hypotheses H6a and H6b are confirmed by the sensitivity analysis. Uniqueness is another exclusive feature of industry platforms, and further research is needed to quantify its relevance in stated choice models.

Resistance to innovation is the third most influencing variable for both side’s dynamics. Its effects are similar to word-of-mouth, in that it does not alter the Developers maximum dimension; this peculiarity strengthen the hypothesis that the initial Developers side growth depends on the number of changed modules on already bought platforms.

Malfuctioning architectures have detrimental effects on the Project ARA diffusion in the two markets, but while H8a and H8b are supported, there is no evidence that the value loss affects the socio-technical system. These results suggest that even small value losses due to malfuctioning architectures are able to reduce the Project ARA market penetration, therefore the testing strategy is key to the success of the cyber-physical system.

Finally, the sensitivity analysis confirms only two of the four hypotheses on the benefits of virtual prototyping tools. H10a and H10b are supported by the empirical evidence, while H11a and H11b are not. In other words, the provider of the virtual prototyping tools should focus on speeding up the engineering process, and not on reducing the engineering costs.

5.4 Practical implications
The AbMX simulations clarified many unknown cause-effect relationships in the design and management of industry platforms. First, it characterized the ecosystem dynamics as made of inflation and deflation phases; then, it highlighted the effect of market size and finally it supported some relevant hypotheses about the sensitivity of the model to strategic design decisions.

From a practical perspective, it is concluded that platforms designers and managers should not focus exclusively on the market launch strategy, but they must also choose carefully the platform substitution strategy. Most of the literature on industry platforms concentrated on initiating the two-sided market that supports the platform ecosystem; however, the agent-based model always shows an initial inflation followed by a deflation. This pattern is particularly significant in small communities, like Puerto Rico, but it can be found also in larger country simulation; furthermore, it may be attributed to the high fraction of smartphone enthusiasts present in the conjoint analysis. However, it does underline that substituting the core with a new generation when the market is close to the maximum size may be more fruitful than waiting for the market to reach the final equilibrium.

Choosing the initial market for an industry platform like Project ARA is not an easy task. On one hand, a small community allows the platform development team to monitor the launch more carefully; on the other hand, the two-sided market dynamics is particularly sensitive to the number of potential adopters. Surprisingly, the number of potential supporters is not as relevant as the other side of the community. For this reason, if a zig-zagging strategy is pursued (Evans 2009), it is more beneficial to start with the Users side of the market, because when the number of adopters grows, the other side follows.

The sensitivity analysis in Section 5.3 provides several insights about how to design and manage a platform. The most important factors influencing the ecosystem dynamics are the number and variety of initial modules on the market, the sensitivity to word-of-mouth and the resistance to innovation. In industry platforms, the first months are essential to reach the critical mass; AbMX suggests that the initial development strategy should follow “go big or go home” mind set: the platform should offer multiple modules choices that satisfy different market segments, increase the word-of-mouth about the product and clearly show the benefits of the device with respect to the competition. The presence of marquee modules, as suggested in (Evans 2009), is encouraged. Moreover, to win the scepticism about novel modular devices, the platform and the configuration process must be very user-friendly, so that the cyber-physical system is not limited to the tech-savvy community.

A part from word-of-mouth and resistance to innovation, other three parameters proved to be significant in the model. The number of architectures that can be chosen in the configurator is positively correlated to the maximum community size, thus the configurator user experience must be a key point in the design strategy. The value of customizability and the value of uniqueness are two peculiar aspects of customization-intense platforms that need to be defined better; in particular, the value of uniqueness can be an important asset.
From a technical perspective, AbMX underlined the importance of a careful verification and validation strategy of the platform configurations, since the likelihood that a device malfunctions affects both sides of the community negatively. Thanks to the high number of battery configurations, the modularity penalty on battery life affects the community only slightly. On the contrary, the development of virtual prototyping tools that quicken the design of novel modules should be a key element in development strategy. The simulations in fact showed that a time-to-market reduction is the fourth most influencing input variable, while development costs reduction does not perturb the community size significantly.

5.5 Model validation

Agent-based model are complex mathematical representations that involve many parameters and functions; this can lead to a lack of rigour in the results if the algorithms are not properly tested, verified and validated. In this research work, the use of the CAP framework supported the modelling effort; nevertheless, proper testing, verification and validation are crucial to guarantee that the results are consistent with reality. (Rand and Rust 2011) provides a set of guidelines for agent-based model verification and validation. Verification shows that the implemented model corresponds to the conceptual model, i.e. the implemented model computes the results according to the intentions; validation demonstrates that the implemented model corresponds to the real world, i.e. can predict the real course of the events. Rigor during verification can be achieved through three steps: documentation, programmatic testing and test cases and scenario. Documentation registers the model assumptions, use and modifications; programmatic testing evaluates parts of the code singularly, while test cases and scenarios considers special cases where the output of the conceptual model is known.

Validation can be conducted rigorously thanks to four processes. Micro-face and macro-face validation insure the attribute characterization of agents corresponds to attributes of real agents. Empirical input validation ascertains that the inputs of the model corresponds to the real world, while empirical output evaluation confronts the output of the implemented model and empirical data. Clearly, the last evaluation is the core of model validation, but it can be performed only if the real world data exist: at the time of writing Project ARA is under development and has not been released to the market yet, thus this validation method cannot be exploited. Alternatively, it is possible to confront the model with stylized facts, which are general notions that have been observed regularly in a certain class of systems, or to cross-validate the model with other models, which have been validated before. At the time of writing Project ARA is under development and has not been released to the market yet. For this reason, it is not possible to compare the simulation analyses with real-world data. Still, other validation methods can be applied. Micro-face and macro-face validations were insured by the strict application of the CAP framework and by the use of surveys, experts’ opinion and trusted literature in populating the model. Several stylized facts and cross-validation examples can be found. As far as two-sided market are concerned, both cross-sided externalities and market equilibria can be found, as predicted by (Rochet and Tirole 2003). Then, literature on innovation diffusion often showed that novel products spreading over time is shaped as S-curves (Schilling 2013); Figure 8.19 and Figure 8.20 can be decomposed in a series of increasing and decreasing S-curves. Furthermore, numerous hypotheses based on literature were confirmed in Section 5.3; for example, the chilling effect of word-of-mouth on innovation diffusion was also observed in (Delre et al. 2010), while the negative impact of malfunctioning architectures (which in turn generate negative word-of-mouth) resembles the findings in (East et al. 2007; East et al. 2008).
Chapter 9

Conclusions and future research directions

1. Summary of the activities

This thesis proposed the use of a set of numerical analyses to investigate the effects of strategic design variables on cyber-physical systems, modelled as industry platforms. To this end, three numerical analyses have been devised. The first one investigates the relationship between change propagation and platforms’ architecture, the second one makes use of a statistically-sound definition of value to highlight the most interesting modules combinations according to customers; the third one simulates an industry platform and its community to understand how the socio-technical system can reach a stable, positive market equilibrium.

Part 1 provides the framework of the research activity. Chapter 1 introduces the importance of cyber-physical systems design and models cyber-physical systems as industry platforms. This model suggests, by leveraging changeability and open innovation, cyber-physical system can potentially tackle market differentiation needs, increase the variety of innovations and enhance products sustainability. Moreover, it shows how engineering design literature has not explored relevant topics about industry platform design yet, and it describes the goals of the three strategic design analyses.

Chapter 2 exposes the State of the Art. First, a brief introduction on technical systems design is given, followed by an analysis of the features, applications and challenges for cyber-physical systems. Then, product platform design is reviewed, the peculiarities of internal and external product platforms are highlighted and the main features of the two-sided market dynamics is explained. After providing a general characterization of innovation, Section 3 focused on open innovation, the role of innovation networks and communities and the main models that describe the diffusion of innovation. Furthermore, the features of engineering changes are disserted, and nine frameworks for the definition of change-related llities are compared. Finally, a multi-disciplinary review about the concept of value is given.

Part 2 consists of four Chapters that describe the theory behind the numerical analyses. Chapter 3 provides a general framework about cyber-physical systems as industry platforms and explains how the design of the platform components is strongly coupled with the community associated to the industry platforms. From this socio-technical model, it emerges that technical, market and cognitive aspects must be considered together during the design of industry platforms.

Chapter 4 studies how cyber-physical platforms, and in general complex technical systems, are subject to a system lifecycle property called Changeability. After providing a framework to define and typify Changeability, it remarks how this property is related to change propagation phenomena, which can be studied with the Changeability Investigation Technique (CIT) and its supporting tool, called 8AM800. The CIT means to predict the likelihood, impact and risk of change propagation when previous projects change management cannot inform designers about change statistics. This may happen when the product is very innovative, when the company does not have a formal change management unit or when multiple proposals are under evaluation during the concept generation phase. The CIT is inspired by a previous mathematical characterization of change propagation, called Changeability Assessment Technique, with which shares the statistical
Chapter 9  Conclusions and future research directions

classification of change propagation and the general rationale. However, the inputs and outputs are novel: instead of utilizing architectures from actual systems and change statistics from previous projects, it analyses a broad range of alternative system architectures and uses Monte-Carlo methods to generate a wide variety of possible change statistics. This allows designer: (1) to understand what are the most advantageous architectures, comparing changeability with other system features, and (2) to characterize components with change propagation indices distributions.

Chapter 5 describes a product platforms value analysis methodology that prioritizes modules according to customers’ preferences. Industry platforms prosper when the customizeability they can offer is capitalized with a large number of modules variants, but constraints on financial resources and time limit the number of modules available at the product launch. Furthermore, verification and validation procedures may be hardly scaled up to check hundreds or thousands of variants. The Value analysis is composed of five steps: goal definition, information gathering, alternatives generation and evaluation, alternatives visualization and proposal development, and presentation of results to stakeholder. With respect to a traditional value analysis, greater importance has been given to the formal definition of value, the generation of the platform variants and the visualization of results. The definition of value used in this research, called Logit value, is consistent with stated choice models: the likelihood of choosing a platform architecture is equal to the architecture Logit value divided by the sum of all architectures Logit value. This not only allows a robust and unambiguous evaluation of the platform architectures, but also the description of the necessary conditions for a successful development of modules.

Chapter 6 is focused on AbMX, an agent-based model that simulates the platform ecosystem dynamics. The agent-based model is composed of four elements: modules and platforms represent the technical part and possess technical and economic features, while Users and Developers are the social part. Users choose module instances and architectures according to the value definition exposed in Chapter 5, while Developers behave according to financial models that aim at maximizing their profit. In other words, Users try to maximize their Use value, while Developers try to maximize their monetary value. The Chapter describes in detail the attributes and behaviours of the elements in the agent-based model following a recent methodology that provides robustness to agent-based models.

Part 3 shows two application of the strategic design analyses. Chapter 7 employs the Changeability Investigation Technique to study the influence of architectural features on change propagation. The rationale is the following: if changes propagate through interfaces, the functional and structural interface layout can influence the system changeability. To prove this hypothesis, a large number of feasible architectures were stochastically generated through Monte-Carlo processes; then, each synthetic architectures was assigned a certain number of possible change propagation statistics. The results show that systems architecture does influence change propagation; in particular, modular architectures are in general less sensitive to change propagation than integral or star architectures. Furthermore, a correlation analysis quantifies how relevant are specific architectural features for the prediction of change propagation likelihood, impact and risk. The four most influencing variables are the degree of the subsystem, the number of component loops in the entire architecture, the maximum distance between subsystems (i.e. the diameter of the architecture) and the structural complexity of the architecture.

Chapter 8 presents the main case study of this research work, the strategic design of a modular, customizable cyber-physical device called Project ARA. The case study applies all three analysis in order to highlight opportunities and threats in the product development of the device. The Changeability Investigation Technique applied to the Project ARA feasible architectures highlighted that Changeability indices can vary if the change is performed by users or modules developers. This finding supports the Changeability definitions framework provided in Chapter 4. Furthermore, it was found that the Endoskeleton and the CPU modules are the subsystems most sensitive to change propagation. The value analysis underscored that only few of the 21,168 possible ARA architectures are optimal, i.e. belong to the Pareto frontier of the Benefits versus Costs tradespace. The analysis revealed that the five Puerto Rican users cluster have different part-worth utilities, and so their choices are quite diverse. Furthermore, the dataset presents a weak correlation between the modules value and the modules inside in the Pareto-front architectures: this finding imply that the module value alone is a poor indicator of the potential success of modules, and that a holistic value analysis like the one presented is required. Finally, a hypothetical product launch in Puerto Rico was simulated inside the AbMX. The results indicates that Puerto Rico can be an interesting beachhead market, but that the small number of the potential adopters has negative effects on the long-term sustainability of the platform ecosystem. This observation can motivate the substitution strategy of the platform core over time and inform about the
most suitable time for the substitution. As general results, the sensitivity analysis displayed that the most relevant social factors are the influence of word-of-mouth and the market resistance to innovation; the product development team must try to increase the variety of initial modules and monitor carefully the number of malfunctioning architectures. Teams supporting the Developers with virtual prototyping tools should focus their effort on speeding up the design process, because the socio-technical system is very sensitive to the modules time-to-market.

2. Research results and contributions

As mentioned in the Introduction, this thesis meant to answer the following research questions:

- How can Changeability be precisely defined and measured?
- How does system architecture influence change propagation?
- How can platform architectures be ranked according to their potential success?
- What socio-technical factors can foster an industry platform and help reaching the platform critical mass?

The three analyses provided answers to the research questions. Changeability can be measured in terms of resources required for change part of the technical system and it is related to change propagation. However, Changeability cannot be uniquely determined: the Project ARA changeability assessment shows that depending on the type of change performed, a system may be more or less changeable. However, a classification scheme like the one provided in Chapter 4 can specify what type of change is under evaluation and reduce ambiguity.

Architecture does have an influence on change propagation. The general analysis in Chapter 7 underlines the importance of modules in reducing the sensitivity to change propagation, but the degree of modularity is not among the most impactful variable. In order to reduce change propagation, system architects should avoid designing central subsystems interfaced with many other subsystems and architectures with a high number of components loops. As literature claims, complex systems are more sensitive to change propagation, while the positive effect of the interface network diameter has never been observed. As far as cyber-physical systems are concerned, the network topology must be carefully designed, as they present some critical features that increase change propagation. Overall, the Changeability Investigation Technique fills a gap in literature, quantifies the influence of architecture on change propagation and provides practical suggestions about how to interface components in complex technical systems.

The architectures were ranked according to Logit value, a definition of value that is at the same time objective (since it can be inferred from questionnaires) and practical (since it is related to statistical choice models). Thanks to Logit value, it is possible to prioritize modules development, incentivize creative novel solutions for the most popular modules, inform the generation of bundles, and focus the testing phase. The Project ARA Value analysis for example reveals that anticipated price strategy is not very efficient in exploiting the device customizability: less than the 0.5% of the total configurations are optimal. The analysis also shows that the most successful modules are high-speed interfaces, low-quality loudspeakers, standard-resolution screens and security sensors, like fingerprint readers or retina scanners.

Finally, the agent-based model supports various hypotheses about the influence of socio-technical factors on industry platforms communities. It is discovered that two-sided markets are susceptible to inflation and deflation phases, and that the initial market phase may not be as important as the sustainment of the community through the years. The total number of potential Adopters influences the ecosystem dynamics deeply, while the total number of potential Developers only affects the number of developers when the developers’ side reaches its maximum. The most influencing factors that can dump the spread of innovation are the importance of word-of-mouth on Users’ choices and the resistance to innovation, while the variety of modules offered at market launch is the most prominent variable increasing market penetration. Other significant factors are the modules development average time, the number of malfunctioning architectures and the importance of uniqueness.

This thesis provides both theoretical and practical contributions. It provides a framework to categorize change-relatedilities according to literature on engineering changes, so that the ambiguity in the field is reduced. It exposes a method to anticipate and study change propagation when there is high uncertainty about the system
Chapter 9

Conclusions and future research directions

architecture or its change propagation, as it happens for innovative cyber-physical system. Particular attention was dedicated to value, in particular how it affects customers’ choices and how it can be computed and utilized; the outcome is a sound and objective definition that highlights platform development strategies and ranks the importance of platform architectures and configurations holistically. AbMX is the first agent-based model that describes an industry platform ecosystem and one of the first numerical model quantifying the influence of socio-technical features on the two-sided market dynamics.

From the application of the analyses, it is possible to draw practical suggestions for the design of cyber-physical industry platforms. The correlation analyses between Change propagation indices and architectural features can support the designing of novel platforms or warn designers about what subsystems should not be modified, if the system already exist. Furthermore, the CIT can be applied to specific systems, so that change propagation can be anticipated; for example, it was discovered that the CPU module in Project ARA is very susceptible to undesired changes and therefore should be integrated into the endoskeleton.

The Value analysis can identify the most important modules in the feasible platform architecture dataset. This can lead to a consistent modules selection process or to the development of novel modules. Furthermore, it can inform about the architectures that have to be verified more carefully and discover if the customization strategy adopted is well-crafted. For example, the value analysis based hypothetical Project ARA features pointed out that the customizability of the device has not been exploited well, as very small fraction of the module combinations dominates the entire set of combinations. Inside this set, the most critical modules are large-capacity batteries, standard-resolution screens, low fidelity audio modules and security sensors.

Finally, the AbMX provides important practical insights for the design of a cyber-physical platform, as the Puerto Rico market launch exemplifies. Puerto Rico is a potentially successful beachhead market in the short term, and it might be profitable in the long term with an adequate core evolution strategy. The most critical social challenge is engaging the users’ community effectively, explaining how to use the device and what are its benefits; configurator design and user experience are two areas that can help achieving this goal. On the technical side, the diffusion of the innovative device would be supported by a wide initial choice of modules, especially the most novel ones (called by (Evans 2009) “marquee modules”). The results of the value analysis can inform systematic innovation strategies or technology forecasting methods about the most urgent R&D directions. Another relevant factor is module combination testing. Validating all the physical variants of the cyber-physical platform can be a colossal task; virtual validation is a potentially successful alternative. It is not recommended to release all possible configurations on the market and wait for the market to discover the malfunctioning, since it can determine the failure of the platform. Furthermore, the use of virtual prototyping tools should be encouraged among the developers, as the reduction of engineering time can be very beneficial to the growth and sustainment of the ecosystem.

It can be concluded that the three strategic design analyses answer the research questions and provide important contributions to the body of literature on technical systems design in general and to industry platform design literature in particular.

3. Limitations and future developments

All the three design analyses have limitations. As far as the architecture generator inside Changeability Investigation Technique is concerned, the synthetic DSMs have constraints on the interfaces, but may not represent real system architectures, as the interface layout can be mathematically sound, but logically flawed or incorrect. The use of more advanced architecture generators like (Zeidner et al. 2010; Selva 2012; Shoungarian 2016) can increase the robustness of the results. Moreover, the statistical distributions used to generate the change statistics in the Change Propagation Matrices can be further validated thanks to new and larger change management databases.

8AM800 can be employed to study more deeply the relationship between complexity (Sinha and de Weck 2013), modularity and change propagation, or to support previous design guidelines (Fricke and Schulz 2005). Dealing with changes in technical systems, designers can either favour changes through Changeability, or resist change thanks to Robustness. Currently, it is not clear when one strategy is more convenient than the other; the CIT can be adapted to investigate on this topic. Finally, a complete socio-technical simulation of the change agents and the system architecture can provide a complete overview over changes in technical system and the strategies to cope with change, like suggested in (Pasqual and de Weck 2011). 8AM800 can be interfaced either with an agent-based representation of the project team, or to AbMX.

165
The definition of value employed derives from a linear combination of part-worth utilities. Other models considered that preferences have saturation effects and can be subdivided into “must-be”, “one-dimensional” and “attractive” (Kano et al. 1984; Matzler and Hinterhuber 1998; Shen et al. 2000; Wang and Ji 2010). The Kano model may be integrated into the value definition in order to provide a better characterization of the system attributes through time. Furthermore, module instances have been generically grouped into module classes; this reduction of model granularity may simplify excessively the module characterization and hide the very successful or unsuccessful outliers inside the module class. Finally, module pricing can be optimized to have the highest number of Pareto-optimal architectures, so that a larger number of platform architectures becomes attractive.

Over-simplification is a potential threat to the agent-based model results. The Users’ choice model is a simplification of the nested choice model and may be inaccurate to some extent. Developers’ behaviour is based on rational, financial computations; real developers may be partially rational or may consider non-financial motivations to join the ecosystem, like prestige, personal satisfaction or fun (Huff et al. 2013). Moreover, AbMX is modelled in a framework that can change for long-term scenarios; for example, it may be possible that after some years new modular smartphones appear on the market and completely change the ecosystem dynamics. It is also possible that customers change their preferences as the market evolves. In order to reduce these uncertainties, a validation of the model outputs against the real ecosystem dynamics, once the device enters the market, would be very beneficial.

The conjoint analysis performed in Puerto Rico tried to balance complexity and completeness: in order to reduce the length of the questionnaire, a self-explicated method was employed. For this reason, the interaction terms in the Logit value model could not be assessed. Moreover, the surveys was administrated in a single location in Puerto Rico, thus results can suffer from sampling bias. Future research may benefit from the use of data mining techniques to determine customers’ preferences from the actual ARA configurator (Liechty et al. 2001) or through other recommender systems (Park et al. 2012). Many other Project ARA data are uncertain, as the device was under development when the research was carried out. Once Project ARA enters the final design stages, the input data can be improved: battery life can be assessed better, architecture feasibility rules can be evaluated consistently, and pricing models can be validated on actual costs.

Future potential developments for the case study include the synthesis of birth-rate, death-rate and affinity curves in the discrete-time dynamical model proposed by (Sinha et al. 2015). This would allow a reduction of the computational complexity and the cross-validation of the agent-based model. In addition, the current version of AbMX does not compute the platform owner revenues; in the future, it will be possible to compute the optimal financial strategy that maximizes the community benefits, like the two-sided social surplus (Economides and Katsamakas 2006). As mentioned previously, a potentially interesting development involves the simulation of more industry platform in the market, and their competition over the users’ and the developers’ sides.

4. Cyber-physical systems innovation

Cyber-physical systems are not only changing competition (Porter and Heppelmann 2014; Porter and Heppelmann 2015), but they are also influencing product design. While Section 3 detailed the short-term development of the analyses, here a sketch about the future of innovation in cyber-physical systems is given.

The research proposal takes advantage of the fluid information fluxes inside and across cyber-physical systems to combine three themes: cyber-physical-social systems design, massive customization and fluent engineering design. The design of Cyber-physical-social systems (Xiong et al. 2015) couples the technical design of hardware, software and networks with social factors like user experience, human choices and cognition, social behaviour, management and economics. As cyber-physical systems enter people’s life, the interactions between the technical system and the customer must be understood more deeply, as well as the interplay between the technical system and society as a whole: cyber-physical systems have an influence on the community, but also the community shapes the cyber-physical system, and the whole system is partially designed, partially evolved (de Weck et al. 2011).

Massive customization takes advantage of cyber-physical systems’ ability to sense the environment and collect data in order to provide customization with minimum choice effort. As remarked in the thesis, customization can be detrimental because in order to make a choice, people need to know what they want and how to obtain it. While removing entirely the choice process can be technically and ethically dangerous, data about the use
of the cyber-physical devices can provide guidance in the customization process. As the Project ARA case exemplifies, cyber-physical systems are potentially so changeable that they can not only become unique, but also evolve over time to meet the users’ changing needs.

The potentialities of cyber-physical systems are exploited if the development process takes advantage of their connectivity and changeability. Data collected can inform developers about the devices use, performances, health, and users’ satisfaction. Then, designers can exploit the systems’ changeability to update their modules more quickly and effortlessly, like now happens in software systems. From a designer’s perspective, this implies that there is a continuous flux of incoming and outgoing information; agile development methods can take advantage of this by continuously improving the product. Instead of discrete generations of products, the design process would provide continuous improvements, updates and fault fixes, becoming more fluid. This fluent engineering design paradigm is strengthen if there is a community of developers that offer, update and retire cyber-physical modules independently from the platform developer, like it happens in other industry platforms.

The development of the three pillars depends on several key research areas, which have also influenced this research work. Changeability and Open innovation have been explored in depth in this thesis, but further research efforts are needed to synthesize a robust and comprehensive set of guidelines to avoid the propagation of changes and to understand how community can integrate seamlessly their products inside cyber-physical platforms. Furthermore, the incentive structure and the formalization of the roles inside the developers’ community is worth further research. Other key areas are (1) data mining and artificial intelligence, which support massive customization by reducing the choice effort, (2) bottom-up verification and validation, which have to predict malfunctioning without physical testing of the whole set of variants, (3) privacy and data security, which is paramount to gain the trust of customers that should handle personal data to the system, and (4) agile development methods, which are needed to continuously update the modules.

These three pillars are means to increase customers’ contentedness, while reducing the material wastes. Customized devices not only can satisfy individual needs, but they provide satisfaction through the customization process; at the same time, changeability can be employed to substitute only the parts of the system that have to be updated, leaving the rest unchanged. This strategy increases the efficiency of material consumption and can promote the decoupling between economic growth and material consumption, if associated with the right set of policies. Furthermore, the “smartness” and changeability of some cyber-physical systems can also increase sustainability by allowing the system to change according to the environment. At the same time, the modularity penalty can lead to systems that consume more resources than integral alternatives; the actual sustainability of modular industry platforms must be verified carefully.

All in all, the research is still long and unfathomable, but the fruits along the path can be worth the journey.
References


References


169
References


de Weck, Olivier L. 2012. “FEASIBILITY OF A 5X SPEEDUP IN SYSTEM DEVELOPMENT DUE TO META DESIGN.” In Proceedings of the ASME 2012 IDETC/CIE.


References


Informate. 2015. *International Smartphone Mobility Report*.


Klein, N. 2014. This Changes Everything: Capitalism vs. the Climate. Simon & Schuster.


Lee, Jay, Hung-An Kao, and Shanhu Yang. 2014. “Service Innovation and Smart Analytics for Industry 4.0


References


References


References


References


UN System Task Team. 2012. *Realizing the Future We Want for All. Report to the Secretary General.*


References


References


Appendix A

A definition of complexity

This Appendix aims at building a definition of the word “complexity” that is consistent with previous literature and is able to clarify some aspects of the practice in the engineering design of systems. The objective is to focus on key aspects that must be taken into consideration when developing theoretical work regarding systems design, both in the analysis of current practices and in the synthesis of new methods and tools. The definition proposed is not meant to be imposed as the final truth on the topic, but rather it would foster a useful debate to clarify different perspectives on systems design and to pursue a better understanding of it. The ultimate goal is to use such definition as a reference to classify models, methods and tools in the field through a homogeneous framework.

1. A synthetic literature review

It is evident that complexity has a quantitative aspect in its roots. For example, in the field of complex systems engineering, (Bar-Yam 2003) identifies the complexity of a system as the number of possible states in a system, and (Norman and Kuras 2006) indicates the volume of a “characterization hyper-space”. From the field of engineering design, (Weber 2005) counts directly some entities in the design process, like number of components or number of variants, while (Magee and de Weck 2004) and (Simon 1996) state that a system is complex only if it is composed by “numerous components” or by a “large number of parts”. Another relevant trait in the definition of complexity is the relationship with the external observer of the system. Several works relate complexity to a perceived “difficulty” in understanding the global properties or behaviours of a system from the properties and behaviours of the parts. This is evident in Edmond’s definition of complexity (Edmonds 1999), but also in (Magee and de Weck 2004) and in (Simon 1996). It is relevant to mention that the duality between the system and the perception of the system is a key aspect also in the definition of Emergence (Deguet et al. 2006; Kim 2006), which is related the concept of “complexity” according to Complexity Science and Complex systems engineering (Ronald and Sipper 2001; Frei et al. 2012).

Both size and observer’s perception lead to the idea that complexity cannot be separated from a modelling language. The concept is central in the thesis of Edmonds (Edmonds 1999), when he claims that complexity is linked to a language or framework, but it is also assumed in (Bar-Yam 2003), where “states” imply a state-space description of the system, and in (Suh 1999), where functional requirements (FR), Design parameters (DP) and their relationships are all elements of a modelling language.

If complexity is related to the adopted language, then it can be quantified by the information content of the description. Algorithmic complexity (Kolmogorov 1965), one of the most famous definition of complexity in Computer Science, associates complexity with the minimum amount of information necessary, defined as “the size of the smallest program of an optimal universal Turin machine generating that string”. Complexity can be also related to the length of a schema (Gell-Mann 1995) or to the information required to achieve the FRs of a design in Axiomatic design (Suh 1999).

Finally, complexity can be further classified according to several criteria or dimensions of characterization. (Weber 2005) proposes the use of five dimensions to evaluate the complexity in the design field. Interestingly, the author does not consider only features linked to the technical system, but also organization type. Another classification is provided by (Suh 1999), who distinguishes six different kinds of complexity, according to time dependence and designer’s knowledge.
After examining five key concepts about complexity, a new definition of complexity is proposed. Before that though, it is important to distinguish between the aim of design, what is commonly called “system”, and its abstract representation, the “holon”. Complexity is associated to the information content of a “holon”. Given that the systems design is concerned with the interaction with system, complicatedness is introduced as the observer’s effort in processing the information due to complexity. Finally, this Appendix proposes a characterization of different aspects related to complexity also through an illustrative example of a railway vehicle suspension.

2. Systems and holons

“System” in engineering design can boast a wide characterization. A system (in engineering design) has been defined as “a combination of interacting elements organized to achieve one more stated purposes” (Haskins 2006); technical artefacts can be treated as technical systems (Hubka and Eder 1988), which can be divided into subsystems and has input/output relationships with the environment (what is outside the system’s boundary).

Engineering design usually assumes that systems are ontological aspects of reality: the world outside the designer is made up by different systems related in different ways. This paradigm is usually called “Hard systems thinking” and dates back to General Systems Theory (von Bertalanffy 1968) and Cybernetics (Wiener 1948).

To deal with systems in social sciences, Checkland and other authors changed this paradigm in a new theory, Soft Systems Thinking (Checkland 2000). According to their movement, a system is an epistemological concept ascribable to an observer. In order to avoid misunderstandings, Checkland proposed the use of the word “holon” instead of system; “holon” must be used “whenever we refer to the abstract concept of a whole or build a model of a holon (models being always descriptions of holons which might or might not map onto some bit of real-world complexity)” (Checkland 1988). In social sciences, the use of holon allows agents to realize that the perception of system can or cannot be shared by other agents; therefore, it is important to consider each single stakeholder’s perspectives about individual holons before taking common action.

The concept of holon can be very useful also in engineering design, since it shows how systems design is carried out through partial representations of the system itself. “Systems” can now refer to the objective of design, the technical artefacts resulting from the design process. Every system can have one or more technical holons, defined as the abstract concept of a technical system. Technical holons can be mathematical models, functional models, structural architectures, technical drawings, CAD models… Holons can be used both to study the behaviour of systems (analysis), to synthesize the final design of systems (synthesis) and to choose between different alternatives (choice). Every technical holon describes the (technical) system partially, and different holons may be combined to achieve a better overall representation. There are at least four evident aspects regarding the technical holon:

1. Every technical holon is a representation of both the technical system and its relevant interactions with the environment;
2. Every technical holon is created to achieve a certain objective;
3. Every technical holon has a specific language;
4. Every technical holon has specific assumptions, both declared and latent. Generally, these assumptions are determined by knowledge and/or resources available.

The four features are relevant in understanding why a technical holon is chosen and why it is different from other technical holons of the same systems.

As an example, the secondary suspension subsystem of a railway vehicle is considered. The suspension is composed by a series of mechanical organs that connects the bogie to the carbody of a vehicle. Many technical holons can describe this system, depending on the objective of the description:

- Suspension can be represented in a functional modelling language. Several languages can be used, like EMS (Energy-Material-Signal) functional modelling, SysML (Systems Modelling language), IDEF0
A suspension can be seen as a dynamical system with inertia, stiffness and damping. In this case, the language of description is mathematics and several assumptions can be made according to the goal of the technical holon. If the aim is to provide a rough analysis, a linear model of a three degree-of-freedom vehicle can be sufficient, while if the complete behaviour at low frequencies is the matter of interest, a dynamical non-linear model with rigid bodies is more appropriate. On the other hand, if the range of frequencies of interest extends to the ones involved with passengers’ comfort, not only the springs but all parts must modelled as a deformable body with its own stiffness and damping.

A suspension can be described as a mechanism to be produced. In this case, the relevant aspects are the quality and the cost of production; the environment of the technical holon, i.e. the representation of the manufacturing system, can impose some constraints and requirements, for example on the shape of the components;

A suspension can be depicted as a bill of materials, i.e. the parts it is composed of;

A suspension can be drawn in a technical representation to communicate the shape and the dimensions of the assembly among several engineers.

3. Complexity

The introduction of the technical holon is the first step to define complexity. Being an abstract representation, a holon is defined by its language and consists of interconnected information. It is assumed that complexity is related to the holon, not to a system. Complexity is here defined as the information content of the holon itself. The definition is coherent with the five key points illustrated in the previous section. The information content is related to the “size” of the system described; technical holons are representations of the system and are depends on the representation language used. Furthermore, as stated in literature, each system has different aspects that contribute to the definition of complexity: this is reflected in the use of different holons in the description of a system. Some further insights concerning the role of the observer in the definition of complexity will be proposed in the following sub-section.

This definition, therefore, agrees with the five key aspects underscored in Section 2.1, but it also allows further reflections.

The information content assessment can be performed in accordance with the specific language of the holon and it is related to the “size” of the system itself. There are several measures of complexity in literature (Edmonds 1999; Lloyd 2001) that can be used in this regard, but new measures can also be designed according to the specific holon of interest, provided that they are consistent with the definition. The reason why there are so many proposed definitions regarding complexity can be explained by the presence of many different holons in the scientific and technical world.

Complexity depends on the characteristics of the technical holon: its objective, its language and its assumptions. Engineers might feel uncomfortable with this definition, because it seems subjective in nature. The same system can have different complexities according to the technical holon provided, therefore complexity depends on how single engineers model the system. This aspect should not be perceived as a drawback, since defining complexity as a property of representations and models can lead to a clearer discussion about which aspect of a system is “complex” and why. This is considered one of the main improvements that the proposed definition of complexity can bring.

4. Complicatedness

It has been noted that a relevant component of complexity is given by the relation between the observer and the system. In the proposed definition of complexity this aspect has been deliberately avoided, since complexity has been associated with the information content of a representation, the technical holon. The notion of the observer’s perspective anyway retrieved through complicatedness.

(Sinha et al. 2013) describes complicatedness “an observer-dependent property that characterizes an actor’s / observer’s ability to unravel, understand and manage the system under consideration”. More generally,
Complicatedness is defined here as the effort required by an observer in order to process information about a technical holon.

Complicatedness is a function of complexity, but also it depends on the observer’s (the designer’s) own characteristics. As an example, (Sinha et al. 2013) cite novelty of an application and cognitive bandwidth of the designer or group of designers. More insights on how designers deal with complicatedness are given in section 3, which analyses the System Engineering design process.

Complicatedness can be evaluated only indirectly. There can be two main classes of measures: one refers to quantifiable characteristics of the design process, the other on mental workload.

Development cost and time are two examples from the first class. Development time has been proved to increase exponentially with complexity both in (Sinha et al. 2013) and in (Bashir and Thomson 1999). In the first case, the time required to assemble a chemical structure is compared to a network-based definition of complexity; in the second case, the time required to develop a product is associated to the depth of a functional analysis.

Mental workload evaluation can be carried out either dynamically during an experience (Rouse et al. 1993) or statically at the end of the experience. If it is evaluated dynamically, measures of different body parameters like temperature and eye movement are required, while if it is gauged statically, many evaluation methods can be found in literature (Hart and Staveland 1988; Reid and Nygren 1988; Rouse et al. 1993).
Correlation plots from change propagation analysis

This Appendix shows the correlation plots, also known as scatterplots, between the change propagation indices and the architectural indices. Given the very high number of samples, a density plot is required; a colour bar on the right inside of the Figures clarifies the densities of samples associated to each colour.

1. ICL
2. ICI
3. OCR
Hello respondent,

We are honoured that you have decided to dedicate part of your time to answer this short survey. We are a group of young researchers in product design and we are currently working on a new mobile phone that has the potential to change the way you can use your handset. We will ask you some information about who you are, then we proceed to some question about what you like about mobile phones. All data is absolutely anonymous and will be used for internal research purposes. Thank you very much again!

1. Section one: let’s break the ice

1. How old are you?

…………………………..

2. Are you male or female?
   □ Male
   □ Female

3. What kind of mobile phone do you currently own?
   □ I don’t have a phone
   □ Feature phone (just calls and texts)
   □ Smartphone (Internet and apps)

4. When did you last change your phone?
   □ I don’t have a phone
   □ Less than 6 months ago
   □ 6 months up to 1 year ago
   □ 1 to 2 years ago
   □ 2 or more years ago

5. What are you using your mobile phone for? Pick the three options that represent you most.
   □ Phone calls
   □ Text messages
   □ Web surfing
   □ Email checking
   □ Professional/work
   □ Social networks
   □ Reading books/newspapers
   □ Taking pictures
☐ Games
☐ Sport
☐ Other (specify): ..........................
2. Section two: Tell us what you like about mobile phones

We would like to ask you to comment about your tastes about phones and what you like about them. For every question, you will find the description of a phone feature or component, followed by a scale that indicates how much you consider it important. Please cross the highlighted bar in the point that represents more accurately your judgement on the feature; some numbers below the scale can help you compare the alternatives. Detailed description of the terms are provided before each question, in case you have doubts about what we are talking about.

2.1 Battery life

Battery life measure the total number of hours you can actively use the phone before you must recharge it.

a. How much do you like a phone with a battery life of 5 hours?

I really dislike it | I dislike it | I slightly dislike it | I'm indifferent | I slightly like it | I like it | I really like it
-3 | -2 | -1 | 0 | 1 | 2 | 3

b. How much do you like a phone with a battery life of 9 hours?

I really dislike it | I dislike it | I slightly dislike it | I'm indifferent | I slightly like it | I like it | I really like it
-3 | -2 | -1 | 0 | 1 | 2 | 3

c. How much do you like a phone with a battery life of 16 hours?

I really dislike it | I dislike it | I slightly dislike it | I'm indifferent | I slightly like it | I like it | I really like it
-3 | -2 | -1 | 0 | 1 | 2 | 3
2.2 Display
The display shows the information on a screen. Brightness indicates how much light the display is capable to emit, so that it can be seen easily even when you are outside under the sun; resolution indicates how realistic and clean the image appears on the screen.

a. How much do you like a screen with medium brightness and resolution?

![Rating Scale for Medium Brightness and Resolution]

b. How much do you like a screen with very high brightness and resolution?

![Rating Scale for Very High Brightness and Resolution]

2.3 Audio
Here we are talking about loudspeakers to listen to music or to the people you are calling. A higher quality allow you to appreciate more subtle details in the music, which can result in a more enjoyable experience.

a. How much do you like a loudspeaker with very high quality?

![Rating Scale for Very High Quality]

b. How much do you like a loudspeaker with medium quality?

![Rating Scale for Medium Quality]
2.4 Branding
We will now show you four brands of phones. Please tell us how much you like the products they produce based on previous personal experiences or what other people said about them.

a. How much do you like the products of this brand?

-3 -2 -1 0 1 2 3

b. How much do you like the products of this brand?

-3 -2 -1 0 1 2 3

c. How much do you like this brand?

-3 -2 -1 0 1 2 3

2.5 Sensors
Environmental sensors measure the environmental conditions like temperature, humidity or the quality of the air. More advanced modules could act like weather stations and even predict the future weather.

a. How much do you like an environmental sensor?
Appendix C

Medical sensors measure your health and your body status. For example, blood pressure and pulses are useful to compute the health condition of your heart or how many calories you spent during sport activities. Or you can think about breath sensors, able to detect if the alcohol in your blood is too much for driving.

b. How much do you like a medical sensor?

![Scale](image)

Security components help you restrict the access to your phone, thus protecting your phone from thieves. For example, a fingerprint reader scans your fingers and recognizes the fingerprint and a retina scanner can recognize your eyes and unlock information on your phone.

c. How much do you like a security sensor?

![Scale](image)

An antenna is the component in your phone that captures the radio signal and gets you connected to the operator. A better antenna allows you to have more reliable signal, thus you can connect in remote areas or download content more quickly.

d. How much do you like an antenna that connects well in all locations, even the most remote?

![Scale](image)

A game controller is a set of buttons you can use for videogames on the phone.

e. How much do you like a game controller in a phone?

![Scale](image)

Interfaces are the components that allow you to connect your phone to a computer, the socket or other electronic devices. Slow interfaces require a lot of time to transfer music, pictures and other files from a computer to a phone and vice-versa.

f. How much do you like an interface that allows you to transfer large amount of data very quickly?

![Scale](image)
2.6 Camera
The camera is the sensor that captures the pictures in your mobile.

a. How much do you like an average camera?

b. How much do you like a professional quality camera?

2.7 Price
Price is an important feature of a phone. And of course we all like cheap phones. We now ask you how critical the following prices for phones are in your opinion.

a. How much do you like a phone that costs 200 dollars?

b. How much do you like a phone that costs 300 dollars?

c. How much do you like a phone that costs 400 dollars?
Appendix C

d. How much do you like a phone that costs 600 dollars?

I really dislike it               I dislike it               I slightly dislike it       I’m indifferent         I slightly like it
I like it               I really like it
-3                      -2                      -1                      0                      1                      2                      3

e. At what price would you consider a phone to be so expensive that you would not consider buying it?

……………………………

f. At what price would you consider a phone to be priced so low that you would feel the quality couldn’t be very good?

…………………………

g. At what price would you consider a phone starting to get expensive, so that it is not out of the question, but you would have to give some thought to buying it?

…………………………

h. At what price would you consider a phone to be a bargain—a great buy for the money?

…………………………

2.8 Internal memory

Internal memory is the part of the phone that stores your data. A bigger memory allows you to have more photos, music or applications on your phone. A gigabyte (GB) is a measure of the memory in electrical components; one gigabyte contains 3000 songs or 4000 pictures.

a. How much do you like a memory of 16 GB?

I really dislike it               I dislike it               I slightly dislike it       I’m indifferent         I slightly like it
I like it               I really like it
-3                      -2                      -1                      0                      1                      2                      3

b. How much do you like a memory of 64 GB?

I really dislike it               I dislike it               I slightly dislike it       I’m indifferent         I slightly like it
I like it               I really like it
-3                      -2                      -1                      0                      1                      2                      3
c. How much do you like a memory of 256 GB?

-3 -2 -1 0 1 2 3
I really dislike it I dislike it I slightly dislike it I’m indifferent I slightly like it I like it I really like it
3. Section three: Rank your priorities

Ok, last effort! We kindly ask you to rank the categories of features you have evaluated few minutes ago. You are given 80 points you can distribute to the 80 categories; the more important the category is for you, the more points you should give it. On average, every category receives 10 points, but you can decide to give as many points as you like, as far as the points sum up to 80.

**TOTAL NUMBER OR POINTS: 80**

<table>
<thead>
<tr>
<th>Category</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery life</td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td></td>
</tr>
<tr>
<td>Branding</td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td></td>
</tr>
<tr>
<td>Price</td>
<td></td>
</tr>
<tr>
<td>Internal memory</td>
<td></td>
</tr>
</tbody>
</table>
Sensitivity analysis for ecosystem simulations

This Appendix shows the inputs and outputs of the sensitivity analysis performed on the AbMX in order to understand what socio-technical factors influence a platform ecosystem dynamics.

The Chapter is structured as follows: Section 1 exposes the orthogonal design of experiments that generated the 135 samples used in the sensitivity analysis; Section 2 shows the linear regression coefficients and the linear model statistics for the Users’ side equilibrium regression. Furthermore, the normal probability plot, the relation between residuals and fitted values and the lagged residuals plot are provided. Section 3, Section 4 and Section 5 are similar to Section 2, but they are related to the Developers’ side equilibrium, the Users’ side maximum dimension and the Developers’ side maximum dimensions, respectively.
1. Inputs of sensitivity analysis

Table D1: Orthogonal design of experiments for sensitivity analysis

<table>
<thead>
<tr>
<th>EXPERIMENT CODE</th>
<th>INITIAL MODULES</th>
<th>BATTERY PENALTY</th>
<th>CONFIGURATOR MODULES</th>
<th>CUSTOMIZABILITY SENSITIVITY</th>
<th>UNIQUENESS SENSITIVITY</th>
<th>INNOVATION RESISTANCE</th>
<th>MALFUNCTIONING VALUE LOSS FOR MALFUNCTIONING TIME SAVING</th>
<th>NRE SAVING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>0.1</td>
<td>1000</td>
<td>0.5</td>
<td>0.001</td>
<td>0.005</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.3</td>
<td>750</td>
<td>0.3</td>
<td>0.001</td>
<td>0.05</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>0.2</td>
<td>1000</td>
<td>0.7</td>
<td>0.000</td>
<td>0.005</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>0.3</td>
<td>1000</td>
<td>0.3</td>
<td>0.002</td>
<td>0.005</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>0.3</td>
<td>1000</td>
<td>0.7</td>
<td>0.001</td>
<td>0</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>0.1</td>
<td>750</td>
<td>0.7</td>
<td>0.001</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>0.2</td>
<td>500</td>
<td>0.5</td>
<td>0.000</td>
<td>0.05</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>0.1</td>
<td>500</td>
<td>0.5</td>
<td>0.002</td>
<td>0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>0.3</td>
<td>500</td>
<td>0.7</td>
<td>0.002</td>
<td>0.05</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>0.1</td>
<td>500</td>
<td>0.3</td>
<td>0.001</td>
<td>0.05</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>0.3</td>
<td>750</td>
<td>0.7</td>
<td>0.000</td>
<td>0.05</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>0.3</td>
<td>750</td>
<td>0.5</td>
<td>0.002</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>0.3</td>
<td>1000</td>
<td>0.5</td>
<td>0.000</td>
<td>0.05</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>0.2</td>
<td>750</td>
<td>0.3</td>
<td>0.000</td>
<td>0.05</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>0.1</td>
<td>500</td>
<td>0.7</td>
<td>0.000</td>
<td>0.005</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>0.1</td>
<td>1000</td>
<td>0.7</td>
<td>0.002</td>
<td>0.05</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>0.1</td>
<td>750</td>
<td>0.3</td>
<td>0.002</td>
<td>0.005</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>0.3</td>
<td>500</td>
<td>0.5</td>
<td>0.001</td>
<td>0.005</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>0.2</td>
<td>1000</td>
<td>0.5</td>
<td>0.002</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>0.2</td>
<td>1000</td>
<td>0.3</td>
<td>0.001</td>
<td>0.05</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>0.1</td>
<td>1000</td>
<td>0.3</td>
<td>0.000</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>11</td>
<td>0.1</td>
<td>750</td>
<td>0.5</td>
<td>0.000</td>
<td>0.05</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>15</td>
<td>0.2</td>
<td>750</td>
<td>0.7</td>
<td>0.002</td>
<td>0.05</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>20</td>
<td>0.2</td>
<td>500</td>
<td>0.7</td>
<td>0.001</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>11</td>
<td>0.3</td>
<td>500</td>
<td>0.3</td>
<td>0.000</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>20</td>
<td>0.2</td>
<td>500</td>
<td>0.3</td>
<td>0.002</td>
<td>0.005</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>15</td>
<td>0.2</td>
<td>750</td>
<td>0.5</td>
<td>0.001</td>
<td>0.005</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
2. Users side equilibrium point

Table D2: Linear regression model statistics

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Standard error</th>
<th>t-statistics</th>
<th>p-val for t-statistics</th>
<th>p-val for ANOVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>2399345.8</td>
<td>220716.6</td>
<td>10.8707</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>Initial modules</td>
<td>192343.5</td>
<td>130754.3</td>
<td>1.4710</td>
<td>0.1439</td>
<td>0.1439</td>
</tr>
<tr>
<td>Battery penalty</td>
<td>75020.2</td>
<td>87680.6</td>
<td>0.8556</td>
<td>0.3939</td>
<td>0.3939</td>
</tr>
<tr>
<td>Configurator architectures</td>
<td>119786.1</td>
<td>116939.2</td>
<td>1.0243</td>
<td>0.3078</td>
<td>0.3078</td>
</tr>
<tr>
<td>WOM</td>
<td>-1824849.4</td>
<td>104420.3</td>
<td>-17.4760</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Customizability sensitivity</td>
<td>-129765.6</td>
<td>77815.7</td>
<td>-1.6676</td>
<td>0.0980</td>
<td>0.0980</td>
</tr>
<tr>
<td>Uniqueness sensitivity</td>
<td>447131.0</td>
<td>54062.6</td>
<td>8.2706</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Innovation resistance</td>
<td>-1053729.8</td>
<td>148996.8</td>
<td>-7.0722</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>% Malfunctioning architectures</td>
<td>-181573.6</td>
<td>53361.0</td>
<td>-3.4027</td>
<td>0.0009</td>
<td>0.0009</td>
</tr>
<tr>
<td>Value loss for malfunctioning</td>
<td>-44020.8</td>
<td>73977.0</td>
<td>-0.5951</td>
<td>0.5529</td>
<td>0.5529</td>
</tr>
<tr>
<td>Time saving</td>
<td>202469.0</td>
<td>59276.7</td>
<td>3.4157</td>
<td>0.0009</td>
<td>0.0009</td>
</tr>
<tr>
<td>NRE saving</td>
<td>-54116.1</td>
<td>58879.3</td>
<td>-0.9191</td>
<td>0.3599</td>
<td>0.3599</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.798</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>0.779</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>2.74 $10^6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p-val of F-statistic vs. constant model</td>
<td>&gt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson-Darling test p-val</td>
<td>0.1108</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson test p-val</td>
<td>0.5793</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure D1: Residuals normal probability plot for Users side equilibrium linear regression

Figure D2: Residuals of Users side equilibrium linear regression
Figure D3: Lagged residuals analysis of Users side equilibrium linear regression
3. Developers side equilibrium point

Table D3: Linear regression model statistics

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Standard error</th>
<th>t-statistics</th>
<th>p-val for t-statistics</th>
<th>p-val for ANOVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>32.1</td>
<td>3.0</td>
<td>10.7580</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Initial modules</td>
<td>16.0</td>
<td>1.8</td>
<td>9.0588</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Battery penalty</td>
<td>-1.9</td>
<td>1.2</td>
<td>-1.6352</td>
<td>0.1047</td>
<td>0.1047</td>
</tr>
<tr>
<td>Configurator architectures</td>
<td>2.4</td>
<td>1.6</td>
<td>1.4852</td>
<td>0.1402</td>
<td>0.1402</td>
</tr>
<tr>
<td>WOM</td>
<td>-21.3</td>
<td>1.4</td>
<td>-15.0774</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Customizability sensitivity</td>
<td>0.0</td>
<td>1.1</td>
<td>0.0332</td>
<td>0.9735</td>
<td>0.9735</td>
</tr>
<tr>
<td>Uniqueness sensitivity</td>
<td>2.7</td>
<td>0.7</td>
<td>3.7121</td>
<td>0.0003</td>
<td>0.0003</td>
</tr>
<tr>
<td>Innovation resistance</td>
<td>-14.4</td>
<td>2.0</td>
<td>-7.1197</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>% Malfunctioning architectures</td>
<td>-2.2</td>
<td>0.7</td>
<td>-3.1044</td>
<td>0.0024</td>
<td>0.0024</td>
</tr>
<tr>
<td>Value loss for malfunctioning</td>
<td>-1.3</td>
<td>1.0</td>
<td>-1.3409</td>
<td>0.1825</td>
<td>0.1825</td>
</tr>
<tr>
<td>Time saving</td>
<td>5.8</td>
<td>0.8</td>
<td>7.2575</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>NRE saving</td>
<td>0.3</td>
<td>0.8</td>
<td>0.3178</td>
<td>0.7512</td>
<td>0.7512</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.798</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>0.779</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>$2.74 \times 10^5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p-val of F-statistic vs. constant model</td>
<td>&gt;0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson-Darling test p-val</td>
<td>0.6473</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson test p-val</td>
<td>0.7851</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure D4: Residuals normal probability plot for Developers side equilibrium linear regression

Figure D5: Residuals of Developers side equilibrium linear regression
Figure D6: Lagged residuals analysis of Developers side equilibrium linear regression
4. Users side maximum

Table D4: Linear regression model statistics

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Standard error</th>
<th>t-statistics</th>
<th>p-val for t-statistics</th>
<th>p-val for ANOVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>2611237.2</td>
<td>214765.7</td>
<td>12.1585</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>Initial modules</td>
<td>1101674.2</td>
<td>127229.0</td>
<td>8.6590</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Battery penalty</td>
<td>-69100.5</td>
<td>85316.6</td>
<td>-0.8099</td>
<td>0.4196</td>
<td>0.4196</td>
</tr>
<tr>
<td>Configurator architectures</td>
<td>272975.3</td>
<td>113786.3</td>
<td>2.3990</td>
<td>0.0180</td>
<td>0.0180</td>
</tr>
<tr>
<td>WOM</td>
<td>-2027053.2</td>
<td>101605.0</td>
<td>-19.9503</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Customizability sensitivity</td>
<td>-51339.5</td>
<td>75717.7</td>
<td>-0.6780</td>
<td>0.4991</td>
<td>0.4991</td>
</tr>
<tr>
<td>Uniqueness sensitivity</td>
<td>217972.5</td>
<td>52605.0</td>
<td>4.1436</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Innovation resistance</td>
<td>-1589971.3</td>
<td>144979.6</td>
<td>-10.9669</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>% Malfunctioning architectures</td>
<td>-209108.7</td>
<td>51922.3</td>
<td>-4.0273</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Value loss for malfunctioning</td>
<td>-28916.6</td>
<td>71982.5</td>
<td>-0.4017</td>
<td>0.6886</td>
<td>0.6886</td>
</tr>
<tr>
<td>Time saving</td>
<td>166492.7</td>
<td>57678.5</td>
<td>2.8866</td>
<td>0.0046</td>
<td>0.0046</td>
</tr>
<tr>
<td>NRE saving</td>
<td>65981.5</td>
<td>57291.9</td>
<td>1.1517</td>
<td>0.2518</td>
<td>0.2518</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.849</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>0.835</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>2.67 10^6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p-val of F-statistic vs. constant model</td>
<td>&gt; 0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson-Darling test p-val</td>
<td>0.005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson test p-val</td>
<td>0.5149</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure D7: Residuals normal probability plot for Users side maximum linear regression

Figure D8: Residuals of Users side maximum linear regression
Figure D9: Lagged residuals analysis of Users side maximum linear regression
### 5. Developers side maximum

Table D5: Linear regression model statistics

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>Standard error</th>
<th>t-statistics</th>
<th>p-val for t-statistics</th>
<th>p-val for ANOVA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>36.7</td>
<td>1.5</td>
<td>25.1306</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>Initial modules</td>
<td>14.0</td>
<td>0.9</td>
<td>16.1524</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>Battery penalty</td>
<td>-1.9</td>
<td>0.6</td>
<td>-3.1963</td>
<td>0.0018</td>
<td>0.0018</td>
</tr>
<tr>
<td>Configurator architectures</td>
<td>1.7</td>
<td>0.8</td>
<td>2.1814</td>
<td>0.0311</td>
<td>0.0311</td>
</tr>
<tr>
<td>WOM</td>
<td>-0.6</td>
<td>0.7</td>
<td>-0.8068</td>
<td>0.4214</td>
<td>0.4214</td>
</tr>
<tr>
<td>Customizability sensitivity</td>
<td>-1.4</td>
<td>0.5</td>
<td>-2.7508</td>
<td>0.0069</td>
<td>0.0069</td>
</tr>
<tr>
<td>Uniqueness sensitivity</td>
<td>0.8</td>
<td>0.4</td>
<td>2.3167</td>
<td>0.0222</td>
<td>0.0222</td>
</tr>
<tr>
<td>Innovation resistance</td>
<td>-0.6</td>
<td>1.0</td>
<td>-0.5650</td>
<td>0.5731</td>
<td>0.5731</td>
</tr>
<tr>
<td>% Malfunctioning</td>
<td>-0.8</td>
<td>0.4</td>
<td>-2.2218</td>
<td>0.0282</td>
<td>0.0282</td>
</tr>
<tr>
<td>architectures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value loss for</td>
<td>0.3</td>
<td>0.5</td>
<td>0.5993</td>
<td>0.5501</td>
<td>0.5501</td>
</tr>
<tr>
<td>malfunctioning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time saving</td>
<td>5.1</td>
<td>0.4</td>
<td>12.9304</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>NRE saving</td>
<td>0.6</td>
<td>0.4</td>
<td>1.6112</td>
<td>0.1098</td>
<td>0.1098</td>
</tr>
<tr>
<td>R-squared</td>
<td>0.800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjusted R-squared</td>
<td>0.782</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMSE</td>
<td>1.81</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p-val of F-statistic vs. constant model</td>
<td>&gt; 0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anderson-Darling test p-val</td>
<td>0.1626</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Durbin-Watson test p-val</td>
<td>0.5309</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure D10: Residuals normal probability plot for Developers side maximum linear regression

Figure D11: Residuals of Developers side maximum linear regression
Figure D12: Lagged residuals analysis of Developers side maximum linear regression