Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato di Ricerca in Ingegneria dell'Informazione

Energy-aware service-based Information Systems

Doctoral Dissertation of:
Alexandre Mello Ferreira

Advisor:
Prof. Barbara Pernici
Tutor:
Prof. Letizia Tanca
Supervisor of the Doctoral Program:
Prof. Carlo Fiorini

2012 – XXIV

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32 I-20133 — Milano

To my wife.

Acknowledgements

I would like hereby to express my gratitude to all that have made suggestions and offered help towards the successful completion of this work, and to all that have supported me from different aspects during the past years.

First of all, my full gratitude to my advisor Prof. Barbara Pernici, who guided me with patience, encouraged me and provided me with precious suggestions. I also thank my tutor Prof. Letizia Tanca for her important support when I needed most.

I am also grateful to all my colleagues for their continuous support and enlightening discussions, in particular Pierluigi Plebani, Cinzia Cappiello and Monica Vitali. Special thanks to Kyriakos Kritikos for his support and friendship.

I would also to thank all people involved in the GAMES project, from who I got many useful information that guided my research in different aspects.

Finally, thanks to my wife who accompanied me and helped me taking care of many issues so that I could concentrate on the work.

Abstract

The problem of Information Technology energy consumption has gained much attention recently due to the always increasing use of IT both for business and for personal reasons. In particular, data centers are now playing a much more important role in the modern society, where the information is available all the time and everywhere. In parallel with this scenario, governmental institutions have launched many international programs and regulations in order to measure and to reduce energy consumption in many areas, including Information and Communication Technology. In this context, the aim of this thesis is to study energy efficiency issues within data centers from the Information System perspective. The proposed approach integrates the application and infrastructure capabilities through Business Process co-design, in which the enactment of adaptation mechanisms is aligned with the business process requirements. Based on both energy and quality dimensions of service-based applications, we propose a model based approach and formulate a new constrained optimization problem that takes into consideration over-constrained solutions where the goal is to obtain the better trade-off between energy consumption and performance. These ideas are combined within a framework in which energy and performance do not always represent opposite objectives, but they are context dependent. Such dependencies are represented as a goal-based model, in which time-based analysis allow the identification of potential system threats and drive the selection of adaptation actions improving global quality and energy indicators. In addition, the framework includes an evolution mechanism that is able to evaluate past decisions feedback in order to adjust the model according to the current underlying environment. Finally, the benefits of the approach are analyzed in an experimental setting.

Riassunto

Il problema del consumo energetico nell'Information Technology ha guadagnato molta attenzione recentemente a causa dell'uso sempre crescente dell'IT, sia a livello imprenditoriale che nella vita quotidiana. In particolare, i data center stanno giocando un ruolo sempre più importante nella società moderna, in cui l'informazione è disponibile ovunque e in ogni momento. Parallelamente a questo scenario, le istituzioni governative hanno lanciato molti programmi e regolamenti internazionali al fine di misurare e ridurre il consumo energetico in molti settori, tra cui l'Information and Communication Technology. In questo contesto, l'obiettivo di questa tesi è quello di studiare i problematiche legati all'efficienza energetica all'interno dei data center dal punto di vista dei sistemi informativi. L'approccio proposto integra le funzionalità delle applicazioni e delle infrastrutture attraverso il Business Process co-design, nel quale l'adozione di meccanismi di adattamento è allineata ai requisiti del processo di business. Basandosi sulle grandezze di energia e qualità delle applicazioni basate sui servizi, proponiamo un approccio basato sui modelli e formuliamo un nuovo problema di ottimizzazione che prende in considerazione soluzioni over-constrained in cui l'obiettivo è quello di ottenere il miglior compromesso tra consumo energetico e prestazioni. Queste idee sono combinate in un framework in cui energia e prestazioni non sempre rappresentano obiettivi opposti, ma sono dipendenti dal contesto. Queste dipendenze sono rappresentate tramite un modello basato sugli obiettivi, in cui analisi temporali consentono di individuare potenziali minacce al sistema e guidare la selezione delle azioni di adattamento per migliorare gli indicatori globali di qualità ed energia. Inoltre, il framework comprende un meccanismo di evoluzione che è in grado di valutare il risultato di decisioni passate per regolare il modello in base ai cambiamenti dell'ambiente. In fine, i vantaggi del metodo vengono analizzati in un dispositivo sperimentale.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Over the last years, managing the energy efficiency of ICT (Information and Communication Technology) has dramatically emerged as one of the most critical environmental challenges to be dealt with. Energy consumption and energy efficiency of ICT centers became priority due to high computing demand, scarcity of resources (which leads to electricity prices increase), and emergence of new environmental regulations.

Energy consumption issues of ICT has been tackled for years aiming to create more autonomous mobile devices, i.e., longer battery autonomy and to reduce produced heat. However, the target is not only mobile devices anymore, but what we have in the background: the data centers. Data centers are more important than ever, in particular with cloud computing paradigm where users are always connected through different devices. The Greenpeace 2012 year report [55] states that "data centers are the factories of the 21st century Information age" which can consume as much electricity as 180,000 homes. Due to the decreasing costs of computing resources, digitization of business processes, and the broader use of ICT, data centers are growing fast from both computational and

energy consumption aspects. The 2011 Data Center Industry Census [1] estimates 31GW of global consumption in 2011 with an increase of 19% in 2012.

In order to drop the electricity load of ICT equipment, Green IT [172] (Information Technology) in its first wave provided solutions to save energy from hardware (such as processors able to scale up and down their computational capacity) and software (like virtualization) layers. In the second wave, which is called as Green ICT/IS (Information System), the solutions are extended to the entire equipment life-cycle, such as eco-friendly procurement and recycling. To cope with these emerging challenges, data centers are adopting Service Oriented Architectures (SOA) [137], in which the available computing resources are shared by several different users or companies. In such systems, the software is accessed as-a-service [165] and computational capacity is provided on demand to many customers who share the same pool of IT resources. In this way, "services and their composition, both at the providers' side (to provide new value-added services), and at the users' side (with mash-ups of services composed by the users themselves), are becoming more and more widespread in a variety of application domains. Hence, since the service-oriented approach is steadily increasing for many application domains, its impact on data and service centers will become more and more significant." [25]

Despite the efforts in delivering more energy efficient IT equipment, the problem of data centers energy consumption is far from an ultimate solution. Recently, a *New York Times* article [2] pointed out the current difficulties in maximizing the facility energy efficiency taking an environmental sustainable approach.

This thesis focuses on efficient Information System aspects, which we believe has not been entirely addressed. There is a gap in considering all the interrelations between the different IS architectural layers (business/applications and infrastructure), which can be expressed by trade-offs like quality versus energy requirements. While most of the approaches have concentrated on the design of such IS systems, very few approaches have proceeded with the integration of both design and runtime phases and models to support energy-aware self-adaptation mechanisms. Moreover, detailed monitoring systems also produce considerable overhead within the infrastructure. Thus, managing such huge amount of data and making this information useful for reasoning purposes are

[1] http://www.datacenterdynamics.com/research/market-growth-2011-2012

[2] http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html

problems addressed in this thesis.

1.2 Research Context

As mentioned, data centers are huge electrical power consumers and the proposed thesis aims to improve energy efficiency aspects by focusing on the business/application layer. To do so, it is necessary to understand the main components of this environment in order to properly place our objectives. A data center can be defined as a facility that contains a high number of electronic equipment used for data processing, data storage, and communication [166], which operates at high availability, reliability, and performance rates, achieving almost fault-free operation on service level.

The facility can be divided into three main sub-system [112] depicted by dashed lines in Figure 1.1: i) Heating Ventilation & Air Controlling, ii) Power Infrastructure, and iii) IT Equipment. The first two provides the minimal conditions [32] so that IT equipment can properly run under desired risk levels. All three sub-systems are followed described.

Heating Ventilation & Air Controlling (HVAC). The HVAC sub-system is responsible to maintain the ambient air temperature and/or humidity of spaces that contain data center IT equipment, reaching up to 50% of the total facility power consumption. Typically, Computers Room Air Conditioning (CRAC) units pump cold air into the raised floor, which escapes through perforated tiles in front of server racks while warm air gets out from the back [20], called hot-aisle/cold-aisle. Improvements in the cooling system can be achieved by optimizing the layout and the airflow and investing in low-energy cooling technologies. Malone et. al [110] compare new blade servers chassis designs which reduce significantly the system airflow rate, and therefore, reduce the data center cooling infrastructure power requirements. Their experiments have shown that blade designs fans consume around 15% less of system power than 1U rack-mount, and that, the CRAC supply air temperature could be increased by 8 ^{o}C while keeping similar internal temperatures.

Power Infrastructure. Before reaching the IT equipment, the power that comes from an outside transformer has to pass first through a primary electrical switchgear, to scale the voltage down, and then through Uninterruptible Power Supply (UPS) systems, which provide power until generators come online when the utility power fails. Then the power

flows into Power Distribution Units (PDU) to be distributed to multiple racks of IT equipment [20]. Along this path, significant amount of energy can be lost and the careful design of UPS systems and on-site power self-generation can produce significant savings [127].

IT Equipment. The IT Infrastructure is the data center's core and it is composed basically by servers, network equipment, and storage devices. Research has been performed on all these three components aiming to reduce energy consumption and increase efficiency. Virtualization, data deduplication, and energy-efficient Ethernet are examples of common technologies nowadays available for server, storage, and network respectively. Concerning the hardware layer, eco-labelling performed by EPEAT [3] and Energy Star [4] are helping buyers to recognize equipment that fulfill their energy standards through a classification according to their energy consumption. Server-specific efforts have concentrated on processor dynamic voltage and frequency scaling (DVFS) and server virtualization. The first technique adapts the CPU power when it is not running during critical periods, saving significant amount of energy with little performance reduction [103]. Due to the high computational capacity of modern servers and to the fact that servers typical utilization rates remain between 10 and 50 percent [19], it is possible through virtualization to increase data center computational capacity (by hosting multiple virtual servers) without increasing, at least not proportionally, energy demands and floor space.

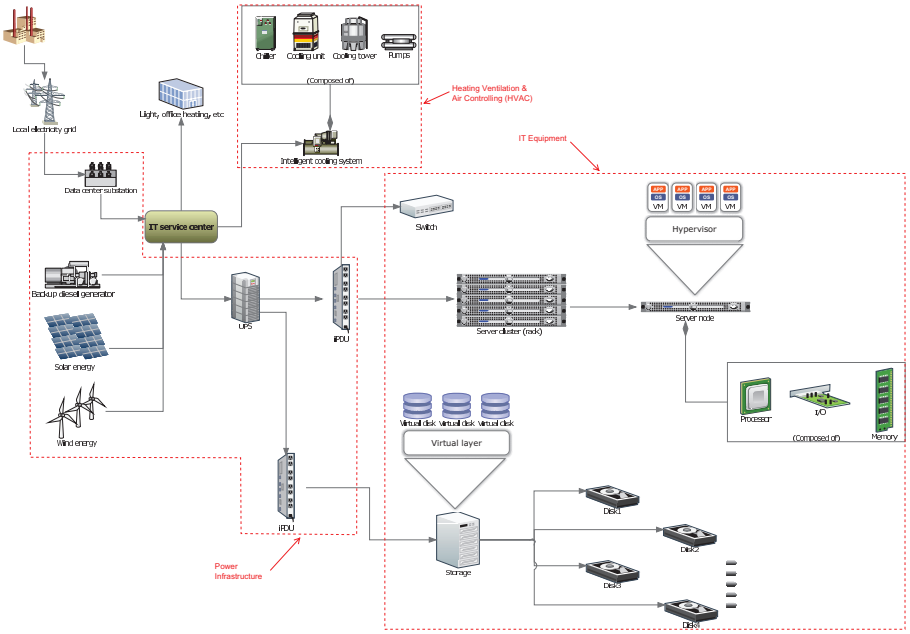
In the data center, all these IT equipment are available for service-based applications (SBAs) in a loosely coupled manner. As this thesis concentrates on how these applications make use of this environment, we divide the business/application layer into two phases: design and execution. We emphasize how the application design can significantly improve the data center efficiency through energy-aware mechanisms that make a better use of the infrastructure. These mechanisms are based on the application and infrastructure detailed information, which is used to compute established business objectives. The identification of energy inefficiencies shall trigger application self-adaptation actions in order to restore desired levels of satisfaction. Such elasticity is only possible when there is a comprehensive view of the system, in which the diverse elements relationships are properly stipulated.

It is worth noting that our approach does not deal with the area of Green IT that is focused on the optimization of algorithms to efficiently

[3] `http://epeat.net`

[4] `http://energystar.gov`

Figure 1.1: Data Center components overview



manage IT facilities in order to save energy [172]. We consider software components as black boxes, hiding implementation details but including relationships to the virtualized execution environment which specify how infrastructure devices consume energy. Consequently, we perform our analysis at the level of energy-aware software components of the BP. Our solution is based on the analysis of the structure of the business processes and the functional and non-functional properties included in the Service Level Agreement defined between the user of the provider.

1.2.1 Research Challenges

1. Considering SBAs, is it possible to identify the minimal resources required to satisfy quality constraints? What metrics can be used? Can the application energy consumption be estimated based on the usage of these resources?

We argue that the application layer holds high potential contribution to the data center energy efficiency. For instance, an application could require less resources for its execution or adapt its behavior to the current context in order to reduce global energy consumption by exploiting their elasticity regarding minimal requirements and/or running modes when

emergency situations such as power shortages occur. Another aspect to consider is that, while in the current literature there is a focus on power (and in particular on peak situations), it is the global energy consumption that has an impact on operational costs and therefore its reduction can be beneficial to both provider and requester

Focusing on the design of SBAs, energy consumption can be constantly monitored by specific indicators called *Green Performance Indicators (GPIs)* [115, 93]. The aims is to guarantee the satisfaction of energy requirements, specified on these GPIs, together with more traditional functional and non-functional (i.e., QoS) requirements. In order to properly design energy-aware applications, it is fundamental to consider the relationship holding between the structure of an application and the energy consumed by the underlying infrastructure for executing this application. On this basis, we propose an approach for designing *energy-aware business process* (E-BP). *Energy-awareness* is given as an extension of the typical BP conceptual model, which contains elements that are able to capture the energy consumption of the involved business tasks.

2. Once metrics are properly calculated, are there adaptation mechanisms able to improve the application energy efficiency? What is the role played by the application design in selecting and enacting the best adaptation actions? How to model and analyze adaptation triggering mechanisms?

The monitored information is the main element to drive *energy-aware adaptation*. It shall have the capability of recognizing business process properties in order to enact specific strategies to adapt its execution or structure in case energy consumption needs to be lowered or energy leakages have been identified. For the selection of the proper actions to be taken, we use the goal-based risk model introduced by [15], in which the relationships between the resources available in a data center and the applications running on it are represented. The model provides information regarding the adaptation triggering mechanisms and the overall impact of one adaptation throughout other indicators, events and/or actions. However, spotting a system threat is not straightforward and, for this reason, we propose a timed analysis module that is able to quickly reason over a bunch of continuous monitored data, represented as streams.

3. How the model can capture the current underlying environment based on the monitoring data analysis? What is the impact of unforeseen situations throughout the model?

Starting from an initial identified version of the proposed goal-based model, which contains the relationships between indicators, events and adaptation actions, the system should be able to change itself in the basis of the analysis of historical monitored data. In this way, the energy-aware adaptation decisions are taken based on a model instance that properly represents the underlying environment. The system learns with unexpected situations in order to evolve the model with unforeseen relationships. This feature is possible due to the analysis of historical data, in which mistaken relations shall produce undesired behavior. The key point of the analysis is to create a set of rules that are defined as patterns within a data mining process. These rules aim to identify situations in which indicators are not fulfilled and, if so, backtrack the problem in order to find relations with past taken decisions.

1.3 Thesis Contribution

By being able to compute both quality and energy metrics for each service, a service-based process is designed using a new constraint-based quality and energy-aware service composition algorithm that advances the state-of-the-art by: i) taking into account not only single (average or minimum) values of independent quality and energy metrics but a range of values, while also dependency functions are used in order to express trade-offs between these metrics; ii) producing a concrete execution plan even if the minimal requirements are over-constrained by accepting a given degree of violations; iii) allowing for the use of non-linear functions in the optimization (QoS-based Service Composition) problem to be solved.

These metrics are represented by both Key Performance Indicators (KPIs) and GPIs. However, the comparison of their results should be made with caution, since the values are calculated in different ways or measured following different methods and equipment. To solve that, we formally define an indicator and propose an aggregation system, which contribute to: i) the identification and classification of both KPIs and GPIs in order to enable their aggregation within meaningful clusters called green index functions; ii) the normalization of indicators values considering their four different boundaries dimensions, which are represented by warning and alarming thresholds; and iii) the aggregation metric based on weights and defined in terms of risk management index, which identify and prioritize the most relevant indicators against the system non-functional goals fulfillment.

Once indicators are defined and their value calculated, the system

shall be able to recognize which ones represent system threats, i.e., can harm the system like exceeding the expected energy consumption of an application. The identification of such system threats introduces new complexity boundaries to the model, which needs to be selective in mining the monitored data in order to identify relevant data to support the decision making mechanisms.

Finally, the proposed framework makes sure that the considered model can adapt itself, evolving its elements and relationships according to new input data gathered from the monitoring system. In this way, it is possible to create different instances of the same model that fits within different applications purposes, like eBusiness and high performance computing. Moreover, the model can also evolve its source structure after a deeper analysis through historical data using data mining techniques. This mechanism ensures that the proposed approach is not strictly dependent on the system manager expertise, but can identify unforeseen relationships among the diverse considered indicators and application adaptation actions.

1.4 Thesis organization

This thesis is organized as follows:

- *Chapter 2* provides a survey of the state-of-the-art with respect to the main concepts involved in this thesis. First the main issues and advances in Green IT/IS are presented. Then we describe SOA characteristics, metrics and adaptation mechanisms that are used to measure the business process greenness and to adapt service-based applications. Finally, we describe goal-driven models that can support this adaptative behavior.

- *Chapter 3* presents our metric, called SBA Energy Efficient Indicator (SBA-EEI), which drives our constraint-based quality and energy-aware service composition. In addition, we also present an indicator aggregation system that aims to identify and classify indicators in order to enable their aggregation within meaningful clusters called Green Index Functions (GIFs).

- *Chapter 4* describes an approach for designing Energy-Aware Business Process (E-BP) extending the typical business process conceptual model to capture the energy consumption of the involved business tasks. Since the E-BP energy consumption depends on their deployed virtual environment, we introduce a 3-layers meta-model

to support business process co-design and energy-aware adaptation.

- *Chapter 5* defines our goal-based model used to represent the several relationships between indicators, events and designed adaptation actions. This chapter introduces our proposed framework and its modules, and focuses on the event identifications ones that represent system threats.

- *Chapter 6* details the adaptation selection mechanisms, which are supported by the goal-based model and a historical data analysis. The objective of this analysis is to evolve the considered model according to monitored data.

- *Chapter 7* describes the architecture that supports our framework by providing the monitoring system and the assessment tool. Also, we demonstrate initial results of the proposed framework using real testbed data provided by GAMES project.

- *Chapter 8* draws conclusions for this thesis, describes ongoing work, and presents future directions.

2 State-of-the-art

This chapter presents an overview of the most relevant approaches with respect to the topic of this thesis. Energy related issues are brought to the business process level through specialized metrics. In addition, trade-off mechanisms that analyze the interrelationships of involved components are studied in order to support decision-making process in different dimensions. The first section introduces green IT/IS main concepts and presents the main approaches towards greener service-based applications within data centers. The main characteristics of such applications are described in the next section, in which business process management techniques are used to support green analysis at higher level of abstraction. The service-based application adaptation section describes current efforts towards dynamic and autonomous adaptation in order to enable application aspects improvement without harming performance. Such investigation is driven by the identification and better understanding of the involved elements interrelationships. This is investigated through the adoption of goal-driven models, in which impact propagation algorithms provide ways to perform qualitative and quantitative analyses.

2.1 Green IT/IS

The role of IT in modern society has become highly complex and its benefits and damages more visible. Considering IT environmental aspects, green IT initiatives are product-oriented and focus on reducing energy consumption, GHG [1] emission and e-waste of data centers. San Murugesan [122] defines green IT (also called green computing) as "the study and practice of designing, manufacturing, and using computer hardware, software, and communication systems efficiently and effectively with no or minimal impact on the environment". The author suggests four directions to be followed in order to make IT green: *green use*, such as reducing data center energy consumption; *green disposal*, which regards to refurbish and reuse of old server; *green design*, which aims to design energy-efficient components; and *green manufacturing*, which consists in manufacturing electronic components with minimal environmental impact. However, this is seen as a first wave by [73]. The authors suggest that a second wave, called sustainable IT, has to go beyond energy use in IT operation and embraces corporate sustainability and social responsibility efforts. It involves considering IT as a set of services that drive the corporate business strategy based on ecological, regulatory, ethical and economic factors.

Such alignment between the underlying infrastructure and the corporate business goals is identified by Watson et. al [174, 175] as green IT and green IS. According to the authors, green IS enlarges the green scope as it contributes to sustainable business process by, for instance, supporting consumers to make green choices with relevant information. As in sustainable IT, the green IS approach involves not only internal factors such as energy efficiency or GHG emission, but external ones that are represented by stakeholders type (suppliers, consumers, and governments) and general eco-goals (eco-efficiency, eco-equity, and eco-effectiveness). Eco-efficiency goals are in line with corporations goals as they look for greater profits. On the other hand, eco-equity goals focus on the social responsibility aspect by bounding the excessive consumption of resources and limiting the organization behavior. Finally, eco-effectiveness goals are represented by the ultimate solution in designing things properly. For instance, in order to reduce servers power consumption eco-effectiveness means to design new servers with autonomous power management capabilities while eco-efficiency means to power off old servers.

The multilevel framework proposes by [82] aims to identify how green IT/IS relates to sustainability within organizations. Within the frame-

[1] Greenhouse gas is any kind of gas that contribute to global warming.

work, the authors identify four types of existing environmental sustainability initiatives. In type zero (image-oriented), organizations announce intentions towards green IT/IS that are never implemented. In type one (eco-efficiency), organizations make efficient use of their existing resources in order to prevent and control negative environment impact. In type two (eco-equity), involves reducing environmental throughout product life-cycle. Finally, in type three (eco-effectiveness) organizations business goals and environmental issues are aligned together in a complementary manner. In a similar way, Headman and Henningsson [75] also introduce three strategies towards green IT such that *storefront* can be mapped as eco-efficiency, *tuning* as eco-equity, and *redesign* as eco-effective.

2.1.1 Green Computing

In order to develop and use computer resources efficiently, software and hardware techniques have been proposed in order to provide better interaction among components and promote the analysis from a green perspective. Essentially, Green Computing (GC) aims at a sustainable computer resources development and usage through using less hazardous materials, maximizing energy efficiently, and promoting recyclability. By extracting the main idea of sustainability and applying it into the Information System field, GC can be defined as an arrangement of all IS resources (assets and capabilities) in order to achieve accumulation, generation, and deployment and therefore reaching market advantages [160]. GC techniques for saving energy can be applied into different levels with different implementation costs. EPA report [167] suggests three energy efficient scenarios: *improved operations* (30% of IT improvement and 20% of saving); *best practice* (70% of IT improvement and 45% of saving); and *state-of-the-art* (80% of IT improvements and 55% of saving). However, most of the approaches take a very generic view or focus only on the infrastructure layer, especially on hardware issues. In addition, many of them focus on non-IT equipment such as cooling and power systems, which correspond to approximately 50% of total data center power consumption [89]. Although Kant [89] tackles the problem of data centers power consumption in IT and non-IT equipment, the approach does not deal the problem from the application level and the existing interconnections between application and resources usage.

Aiming to involve all existing elements of a modern data center, Maturity Model Frameworks have being proposed in the literature [57, 64, 162]. The IT-Capability Maturity Framework (IT-CMF) [57] defines fives maturity levels, from initial to optimized. In each level, managing

strategies are categorized in four macro-processes: *managing IT budget* that involves service level adjustment and supplier negotiation for cost reduction, *managing IT capability* that deals systematically with IT assets and their business value, *managing IT for business value* such as return-on-investment (ROI) measures, and *managing IT like a business* which uses professional business practices within the IT functions such as ERP [2] for IT operations. Although the framework does not focus on sustainable issues directly, it contains many key characteristics to lead to efficient data centers. Such extension is presented by [64], in which the Sustainable ICT Capability Maturity Framework (SICT-CMF) exploits the underlying technology to achieve sustainability gains. The framework aims to align what SICT is actually achieving and what the business want. To do so, the approach describes four key actions:

1. *Scope and goals definition:* It represents a preliminary definition of the organization view about sustainability and what are its objectives. In order to achieve these objectives, the authors argue that they must be clear defined within the organization's business objectives.

2. *Current IT maturity level understanding:* Having the objectives defined, the organization has to assess its current maturity level as starting point. It adopts the five maturity levels of IT-CMF [57]. This is done through specific metrics applied at different levels of the organization. CO2 emission per employee metric is an example at the overall level, while FLOPS (floating point operations per second) and SWaP (space, wattage, and performance) are examples at the data center level.

3. *Capability building blocks management:* At this stage, the goal is to assess the organization's capabilities towards SICT. It is composes by nine capability building blocks grouped into four categories:

    a) Strategy and planning: Specific objectives of SICT and their alignment with organization strategies;

    b) Process management: Sourcing, operation and disposal of ICT systems;

    c) People and culture: Common language throughout the organization;

    d) Governance: Policies that comply with regulations and legislation.

[2] Enterprise Resource Planning

4. *Evolution management:* It consists in assessing and managing SICT progress over time in order to create a roadmap and an action plan. The assessment shall highlight small gaps between current and desired maturity, such that short-term opportunities can be prioritized.

Focusing on the data centers, the Data Center Maturity Model (DCMM) proposed by The Green Grid consortium [162] aims to improve data center energy efficiency and sustainability across many aspects. As in SICT-CMF, the proposed maturity model is divided into levels that goes from minimal/no progress (level 0) to visionary (level 5). Levels 1 and 2 represent data centers that partially or fully use current best practice techniques. Levels 3 and 5 indicates future capabilities direction in which the industry should move within the next five years. These levels are analyzed according to facility and IT perspectives. The facility category involves mainly power, cooling and management issues while the IT category accounts for compute, storage and network issues. Differently from other models, the DCMM provides very detailed and technical information about sub-categories in terms of existing metrics, i.e., quantitative manner. For example, in the compute sub-category (IT category) the metric `utilization` is described as following throughout the levels: *Level 0*, not measured; *Level 1*, average monthly and peaks are measured; *Level 2*, average monthly is great than 20%; *Level 3*, average monthly is great than 35%; *Level 4*, average monthly is great than 50% and application compute power use understanding; *Level 5*, average monthly is great than 60% and management of spare compute capacity. By doing that, it also allows the use of the maturity model to benchmark operations and monitor progress of the entire data center or individual categories and sub-categories.

Cloud Computing

With the growth of pervasive and ubiquitous computing, modern data centers have adopted cloud computing models in order to deliver computing and storage capacities as services that are available in a pay-as-you-go manner [22]. Such services are divided into three fundamental models from bottom to up: Infrastructure (IaaS) that involves servers and VMs, Platform (PaaS) that involves the application container, and Software (SaaS) that represents the SBAs. Generally speaking, cloud computing environments provide two main techniques to tackle data center energy efficiency at the middleware level: server consolidation and Virtual Machine (VM) migration [27]. Berl et al. [24] point out the key issues

in energy-saving techniques applied to cloud computing environments. The focus relies on the capability of the cloud environment in providing scalable and virtualized resources in order to maximize hardware utilization without degrade service level agreements. Considering the fact that "many services often need only a small fraction of the available computational resources" [24], the utilization rates of virtualization (hardware sharing) techniques directly increase the hardware usage rate. Keeping the hardware at high utilization levels is one of the basic principles to increase overall energy efficient within data centers. An ideal efficiency value comes when the total capacity of the server is been used to compute the "useful work", i.e., the work performed by application services which aim to satisfy business objectives. Although virtualization can bring significant energy savings, it also requires additional management and, therefore, additional costs. The authors argue that the major challenge is to explore the relationships among all involved components and find out an optimal balance between performance and energy consumption. Such balance is exploited by an energy efficient self-management mechanism that decides, from a holistic perspective, runtime adaptation actions over the underlying environment such as VMs migration, copying, creation, and deletion; and unused hardware equipment turned off (or hibernated).

In [113], the authors also start from the fact that data center hardware equipment remain underutilized for long periods of time. They claim that the use of VMs and their common capabilities such as migration can lead to significant data center energy-savings. Similar to the approach described by [24], the aim is to reduce energy consumption without reducing application quality parameters. In order to make appropriate decisions about VMs migration actions, the approach proposes a service request prediction model, which predicts future request patterns based on historical service data. These patterns are refined based on the Service Level Agreement (SLA) violation rate feedback. Controllers are in charge of evaluating migration and hibernate actions, of which a centralized one, called Cloud Controller, is responsible to manage the application throughout their deployment, execution and disposal. Other three local controllers, Node, Cluster and Storage, are used to manage more fined information about single VM activities, VMs running on set of nodes, and storage systems attached to VMs. For each cluster, an energy optimization algorithm evaluates the current number of active servers and the number of required servers for duration $t + t_{opt}$, where t_{opt} represents the "minimum time period for which energy consumption while running a server in idle mode equals the energy consumption for hibernating and waking up the server". Based on that,

the controller tries to maximize server utilization rate by migrating VMs among server/clusters and putting underutilized ones in hibernate mode.

A different approach for resource allocation problem in cloud environments is presented by Beloglazov et. al [22]. Although the basic principle of reducing energy consumption while keeping agreed Quality of Service (QoS) requirements holds, the authors present a Green Service Allocator that introduce new capabilities. It is composed by negotiator, service scheduler, VM manager, pricing, service analyzer, consumer profiler and a new component called energy monitor. The energy monitor component gathers the energy consumption values of VMs and physical servers in order to help the VM manager to make energy efficient decisions. The energy-aware resource allocation is composed by two sub-components: VM placement and VM selection. For the VM placement, the authors apply the Best Fit Decreasing (BFD) algorithm with some modifications in order to choose the most power-efficient server nodes first. The role of the VM selection is to optimize current VM allocations. To do that, all VMs that need to be migrated are selected and, therefore, the Modified Best Fit Decreasing (MBFD) algorithm is executed. The upper and lower utilization thresholds of the server node and the single VM determine the VM selection for migration. For instance, if the utilization of a server node falls below the lower threshold, all VMs are migrated to another server and the original node is switched to a sleep mode. Such migrations are guided by three policies: (a) Minimization of Migration, which selects the minimum number of VMs to be migrated; (b) Highest Potential Growth, which is used to migrate the VM with the lowest usage of CPU with respect to its defined maximum capacity; (c) Random Choice, which randomly selects the VM to be migrated according to a uniformly distributed discrete random variable. Simulated experiments, performed using CloudSim toolkit, validate the approach against static resource allocation techniques.

Kim et al. [92] extend the approach of [22] by focusing on power-aware management techniques of real-time Cloud services, in which real-time services are modeled as a real-time VM requests. In the proposed Cloud service framework, the authors classify real-time services in hard and soft services. The terms hard and soft denote different provider penalty functions when there is a constraint violation. In the hard model, the provider receives some penalty while in the soft model a penalty function, such as linear decreasing function, is used. Moreover, hard real-time services use power-aware schemes based on dynamic voltage frequency scaling (DVFS) and soft real-time services use a power-aware profitable VM provisioning scheme. In the first case, i.e. power-aware hard real-time cloud service, the algorithm trades the dynamic scaling of the pro-

cessor speed such that: higher processor speed allows more VMs sharing it, but power consumption is increased; lower processor speed results in lower power consumption, but less number of VMs. Three schemas are proposed for VM provisioning: i) Lowest-DVFS, which consists in adjusting the processor speed to lowest level in which the VMs still meet their requirements; ii) δ-Advanced-DVFS, in which the processor speed is over-scaled by δ% of the lowest speed in order to guarantee quality requirements during service request variation periods; and iii) Adaptive-DVFS, which makes use of the advantage of knowing the service arrival rate in advance and adjusts the processor speed as a function that involves average values of service arrival rate, service rate, response time.

In the second case, i.e. power-aware soft real-time cloud service, a profitable delay analysis is conducted when quality requirements are not fulfilled. A refund mechanism based on the service value is used to reduce the user's service cost that is proportional to the service delay. For soft real-time services, the VM provisioning algorithm is initially similar to the hard real-time services (looking for the minimally priced resource). However, the difference relies on the fact that the priority is to maximize profit even if some service delays are expected. A profit calculation function evaluates when it is better to provision the VM in a specific server even when the service cost is reduced. As in [22], simulated experiments demonstrate that, for hard real-time services, the best VM provisioning algorithm is δ-Advanced-DVFS with respect performance and power consumption.

2.1.2 Energy-aware Applications

Another direction to take to reduce energy consumption is the utilization of metrics and techniques at the software level. Calculate the exact among of power that an application is consuming is not an easy task although many approaches consider the application as the origin of all other layers energy consumption. Thus, the goal is to provide quantitative mechanisms that enable analyzing the applications regarding to their energy efficiency aspects from design to disposal phases. An efficient software shows good proportionality between resource utilization and the amount of work done (considering both functional and non-functional application facets) [159]. One of the first energy metrics for software systems was introduced by [47] which extracts information from the flowgraph model to improve the application design independently from the underlying hardware. The considered flowgraph is the representation of control flows (*if* and *loop* statements) as a direct graph. The authors consider two software sources of power consumption: pro-

cessor and memory (storage is not considered as the approach focuses on portable computing). A pair of hierarchical energy measures is presented in order to quantify the number of executed instructions and the number of data memory accesses. For the number of executed instructions, it is calculated based on the instruction level of processor power models. However, one of the main drawbacks of the approach is regarding to the flowgraph annotation in which the number of loop iterations have to be estimated. The analysis of software energy consumption based on the application source code is also the goal proposed by [42] where a set of software complexity and quality metrics are used to estimate energy consumption as a top-down approach. The application is assessed from a logical perspective by evaluating how much energy is required by a single bit status commutation applied on a given number of bits (thermodynamic depth) for a given number of operations (complexity). Therefore, analysis is performed in order to identify the most representative metrics according to physical power measurements. The obtained results are used as the basis of a tool capable of extracting a set of energy-related metrics by looking at the application code.

Kansal and Zhao [86] also present an energy profiling metric for applications at design phase. An automated tool is described to profile the energy consumption of resource components used by an application and to properly guide design decision-making process. The energy consumed by an application is divided into two parts: the performance required for the application and the system resources used by the application. Through these components, the application consumes power during its activation (i.e., the energy consumed to run the application and underlying system), waiting states (when there is a subsystem powered up but the application is using another subsystem), and idle periods (the application is not doing any work at all). Hence, the application energy profiling approach considers how energy consumption is spanned across the involved resources such as CPU, disk, memory and network interfaces during each state. In active states, an energy profiler component traces resources related events in order to map application energy usage across various system resources. By using events logs details generated from resources utilization, the component identifies the correspondent usage of an application regarding to CPU usage and disk activity only. The current version of the tool does not provide the energy consumption of the memory or network interfaces yet. However, the major drawback of the tool is regarding to homogeneous equipment, which makes it very difficult to be used in existing data centers where hardware is likely heterogeneous.

In the same direction, the “pTop” tool developed by [62] aims to pro-

file the energy consumption at process-level applications. The idea is to estimate the application energy consumption based on its resource utilization through indirect measurements. It provides an easier and real-time way to get the application power consumption through application programming interface (API). Moreover, the approach enables runtime adaptation actions to be taken towards energy reduction. As it is offered as a service in the system, the computational requirements of "pTop" are less than 3% of CPU and 0.15 % of memory. The resources taken into consideration are processor, network interface and hard disk (memory is not considered). Due to validation purposes, the authors developed an adaptation framework to interact between the tool API and the user's applications. Resource monitor, demand predictor, and adaptation manager components compose the framework. The last one is responsible to decide which application has to be adapted based on the information provided by the other two components. Unfortunately, the authors do not present detailed information about the application adaptation actions available nor the policies used to decide for one or another adaptation action.

Real-time application energy consumption estimation is also proposed by [56] through a methodology that tackles the problem by using indirect performance counters measurements (such as CPU time per second, process migrations per second and instructions per second), application process information (such as percentage of CPU used by a single task, percentage of CPU used by a single task within the VM, amount of bytes tasks read/write from disks, and amount of bytes in memory tasks use) and server node data (such as number of packets send/received during last second, average time to complete a writing/reading operation and process executing in kernel model). Using simple linear analysis of all considered elements (metrics and processes), the authors define the power consumption of a single application process.

2.2 Business Process Management

In cloud computing environments SBAs are composed in order to fulfill expected business goals. Such goals are represented in terms of business process, which understands a set of abstract coordinated activities. Business Process Management (BPM) is responsible for "supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information" [168]. This section aims to clarify what are the main issues involved when green as-

pects are used to drive decision-making process at the business process level. Considering green IS approaches, several attempts have been made to extend BPM methodologies towards green BPM [77, 70, 163, 128].

In [77] and [70] the authors propose a new class of methodologies to identify areas of improvement by understanding the relationships among the BP elements and performing GHG emission reduction actions throughout the entire BPM life-cycle. Each process activity is annotated (using BPMN [3]) with the amount of GHG emitted by the activity when executed. This is done by evaluating two functional aspects of the activity: consequence and consuming resources. An example of activity consequence is the impact of cutting a tree and an example of resource consumed is the impact of traveling by car. The approach adopts direct acyclic graphs to model usage-cost relationships. The BP activity impact analysis is also the objective of [163], in which the authors identify the set implementation that satisfies temporal constraints. Having that, it is possible to choose the activity implementation that produces the lowest amount of CO_2 emission and, at the same time, keep the BP temporally compliant. A decision support system for evaluating and selecting efficient IT investments is described in [128]. The goal is to narrow the gap between IT and BP layers based on a trade-off between minimal quality requirements and resources operation costs. The term green BPM is also exploited by [130, 131], in which the main characteristics and differences with respect to traditional BPM are highlighted and compared.

2.2.1 SOA

In order to enable to execute and to manage BPs, Service-oriented architecture (SOA) represents "the logical way of designing software system to provide services either to end-user applications or other services distributed in a network, via published and discoverable interfaces" [135]. With SOA it is possible to coordinate the several distributed BPs through linked web services [94]. Typically, BPs running on SOA are composed by several invocations of different components (services) in order to satisfy the whole BP goal. By using SOA we implicitly define services as "self-contained modules that provide standard business functionality and are independent of the state or context of other services" [137]. They commonly adopt cloud computing under the utilization of Web services standards. Web services components are hosted within services containers that interface business services and infrastructure services. The main concepts of SOA involved in the presented thesis are [136]:

[3] Business Process Modeling Notation.

- Service composition: It consists in aggregating multiple services into a single composite service. Such aggregation is coordinate by orchestration and choreography mechanisms, which describe how services interact at the message level. An important issue relies on identifying each single service quality level defined in the service-level agreement.

- Service management and monitoring: It spans from deployment to collecting data for metrics calculation in order to verify both composite services and single independent services implementation. Such information is analyzed in order to report possible constraints violations (performance and energy) or to support the decision and enactment of adaptation strategies such that existing or upcoming problems are solved.

- Service design and development: The challenges involved here rely mostly on engineering methodologies to model the business environment, such that performance and energy indicators are mapped into the service design phase. In addition, service governance issues are involved, in which services are effectively guided towards functional and non-functional (including performance and energy) requirements satisfaction.

The Vienna Runtime Environment for Service-Oriented Computing (VRESCo) proposed by [117] aims to cover the complete publish-find-bind-execute of SOA-based applications. The approach enforces the usage of service metadata to store information about services and QoS models in order to enable suitable service mediation. The VRESCo metadata model is represented as blocks of concepts that capture the service functionalities details. The main concept blocks are represented by feature concepts (concrete actions that implement the same functionality), data concepts (entities), and predicate concepts (data flow required or produced by the feature their valid states after invocation). A specific language, VERSCo Query Language (VQL), is used to query this metadata model. The VRESCo Mapping Framework (VMF) does the mapping between the service concrete operations (metadata model) and the service abstract features (service model) in order to perform service mediation. In the mapping process services are grouped into categories according to their available features. Therefore, services operations are mapped to concept features such that one operation implements one feature. In this way, the mediation follows the service invocation and it does not need the direct service input parameters, which may vary from one service to another, but the high-level VRESCo feature representation.

Finally, the environment supports 5 different service-rebinding strategies for dynamic biding: fixed (never), periodically (time interval), on demand (upon client request), on invocation (binding is checked before the invocation), and on event (upon events notification). However, the described mapping between service metadata model and service model does not support service composition yet.

2.2.2 Service Composition Issues

Service composition is one of the basis and most important functionalities intrinsic in SOA. SBAs are built from other services, usually at runtime, when the user's requirements are issued to a broker or service composition engine. The composite service construction is separated into two sequential phases: a) the creation of an abstract execution plan; b) service selection for each abstract task of the execution plan. Various techniques have been proposed for automatically or semi-automatically creating an abstract execution plan of a composite service based on user's functional needs and functional capabilities of available services in the system's registry. In the second phase, which is based on the abstract execution plan, functionally-equivalent services are selected as candidate services for each abstract service, such that they differ in their non-functional characteristics, i.e, performance and energy. The functional selection of these candidate services can be solved with approaches like [141]. The final goal of this phase is achieved by solving the well-known Service Concretization (SC) or QoS-based Service Composition problem. According to that, the best service available at runtime has to be selected among all candidate ones for each abstract service, taking into consideration the global and local quality (and energy) constraints defined in the SLA.

There exist various approaches to solve the SC problem, which are separated into two main classes: *local* [116, 183, 13] and *global* approaches [183, 81, 38, 14]. Local approaches select services one at a time by associating the abstract service to the candidate service that better supports it by satisfying the local quality constraints. In this way, global constraints are not taken into account at all. On the other hand, global approaches try to solve the SC problem by selecting those services that satisfy both the local constraints imposed on the corresponding (local) abstract service and the global constraints that affect the whole composite service. In order to guarantee the fulfillment of global constraints, these approaches use optimization techniques like MIP [183, 14, 119] or evolutionary algorithms such as Genetic Algorithms and Cultural Algorithms [161, 38, 95]. However, most of these approaches usually consider the worst or most common case scenario for the composed service,

which concerns to the longest or the hottest execution path, respectively [183, 38] or they satisfy the global constraints only statistically by reducing the number of loops to a single task [81]. Therefore, they are either conservative or not very accurate.

One approach that guarantees that all possible execution paths of an execution plan satisfy all global constraints is analyzed in [14]. However, even this approach presents the following disadvantages: i) it does not allow non-linear constraints; ii) it does not produce any solution when the requirements of the user are over-constrained; iii) all execution paths have to satisfy the global constraints – even the longest ones that are not so probable have to satisfy all the global constraints; iv) it takes into account only the minimum or average value of a metric for each service while it also regards that all metrics are independent; v) it does not take into account any energy metrics. The latter two disadvantages are common for all SC approaches.

With respect to dynamic environments, there is not a consolidated approach that takes into account service quality parameters into an aggregated function to satisfy end-to-end business process requirements. Such environments can be dynamic regarding to QoS constraints and/or IT performance capabilities. Cao et al. [40] propose a new negotiation protocol which acts as a neutral mediator to trade-off service QoS when the concrete composite service execution path may fail due to non-functional requirements. The model has several domain managers to negotiate single services attributes and a centralized coordinator agent that takes care of the total negotiation process to find "win-win" agreement based on time-sensitive negotiation strategy. Considering the IT infrastructure changeability, [161] uses genetic algorithms implemented inside a Linux kernel to guide runtime DVFS towards SLA meeting. However, the approach does not take into consideration workload prediction or VM migration/reconfiguration.

2.2.3 Business Process Metrics

The definition and usage of software metrics is not a simple task. Starting from general software engineering metrics, Kaner and Bond [84] demonstrate and justify the many dimensions that must be taken into consideration when defining or simple using a software metric. The authors delineate a framework for evaluating software metrics regarding to their purpose, scope, calculation formula, value meaning, and their relationships. In their approach, a metric is generally defined as "the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of

describing them". Distinctively from software engineering metrics, the following introduced metrics evaluate BP design attributes and mashup applications performance and energy parameters during execution time.

We divide BP measurements into groups: design-time and runtime. Design-time metrics are closer to software engineering metrics and they measure the BP by considering BPEL description such as source code. Static analysis are performed looking into the BP internal objects, control-flow components, and expected quality parameters [149]. On the other hand, runtime metrics can only be calculated if additional information about the environment and its inter-related elements is available. For instance, the response time of a BP instance may depend on the flow path taken during its execution. The monitoring of the BP execution is typically made through a Business Activity Monitoring (BAM) system which provides real-time values of individual BP components mapped into their running environment [85]. Therefore, these monitored values are translated into Process Performance Metrics (PPMs) or Process Performance Indicators (PPIs) in order to provide proper significance for the collected single values. Finally, these PPIs are taken as input parameters for KPIs calculation, which definition is based on business goals. KPIs are influenced by numerous PPIs but also technical parameters such as Quality of Service (QoS) [178].

Design-time metrics are calculated based on the designer background knowledge and the process internal attributes, such as size, structural complexity, cohesion, coupling and length. Despite the vast literature on software measurement, business process metrics field was recently introduced by Reijers and Vanderfeesten [149] and Cardoso [43] in which, inspired by existing software engineering metrics, new process metrics regarding to cohesion, coupling and complexity were introduced in order to analyze workflows. Vanderfeesten et al. [171] describe internal attributes in details and how they can be used to drive more efficient process models design. With respect to energy consumption, it relies on the fact that less complex BP consumes fewer resources to perform the same tasks, i.e., more efficient resource utilization. In addition, less complex BP also increase its flexibility and, in particular, the set of adaptation actions available at runtime.

Cardoso [44] proposes a process control-flow complexity (CFC) metric to analyze process complexity based on the process interactions described in BPEL [4], which is a standard language for specifying Web process. The CFC metric of a process is actually the CFC metric of each activity within the process. The approach defines activities as basic or

[4]Business Process Execution Language.

structured. Basic activities are weighted as 1 and are divided in communication (call and receive messages), control (wait or terminate), and data manipulation (assign). Structured activities are formed by control-flow activities, i.e., they control the steps that shall be executed in the process. Their complexity calculation depends on the control-flow type. They are five: i) *sequence*, activities are invoked one after another; ii) *switch*, represent optional branches; iii) *while*, a set of activities are executed repeated according to a Boolean condition; iv) *flow*, represent branches that are executed in parallel; and v) *pick*, the occurrence of an event starts the execution of an activity. The CFC value interpretation is done by using McCabe's cyclomatic complexity thresholds, in which the BP is classified as simple, slightly complex, complex, and untestable. In [152] the authors analyze the CFC metric and validate it using several processes described in BPMN against a set of common derived measurements, such as the total number of events of the model. However, the approach focuses on pure control-flow dimension and just scratch the process data dimension, without considering in detail the other dimensions of the process model.

A similar approach is taken by Sanchez-Gonzalez et al. [158], which focus on the early phases of process design in order to minimize costs and to facilitate future business process evolution. The approach delineates positive and negative correlations in a set of process structural metrics, which includes: number of nodes, coefficient of connectivity, sequentiality, concurrency, and others. Their goal is to identify correlations among a fixed set of structural metrics in terms of understandability and modifiability. Using regression analysis to identify these correlations, the authors investigate their impact with respect to time, accuracy and efficiency dimensions. The obtained results demonstrate how design-time metrics can be used as business process quality predictors models. Model-driven business process metrics are also described in [185, 155]. The authors in [185] define the set of metrics used to measure business operations within an *observation meta-model* which is described according to information organization and operation. The first corresponds to the process entities to be monitored while the second regards to relevant information extraction and metrics calculation. A more efficient method is proposed in [155] to create specific metric meta-models according to Model Driven Architecture (MDA) specifications. In short, the presented meta-model describes the relationships and the attributes among the business process elements and pre-defined metrics converted from business logic models.

Considering that runtime metrics represent the minimum expected quality requirements defined at design-time, which are monitored at

runtime and additional information regarding to the process execution environment is needed. In BP, such performance requirements can be expressed in terms of PPI defined as a special type of KPI that focuses exclusively on BP activities. However, the definition of such metrics and their monitoring system are complex. The approach proposed by [120] deals with this issue by presenting a SOA platform-independent monitoring and controlling. The solution focuses on BP monitoring where PPIs specifications are described through meta-models. Such meta-models are linked to the concept of process at the same level of abstraction of BPMN, which enables the identification of appropriate measuring points. Each PPI is defined as a basic indicator or an aggregation of set of PPIs. For that purpose, three values attributes are specified: current value, target value thresholds and alarm values thresholds. The current value calculation is handled by the *PPIMonitor* component which is part of the PPI monitoring model. The model defines in which process object the indicator operates. All monitored process objects and their respective attributes are described within the PPI monitoring model. The monitoring model instrumentation is basically performed by means of *EventProbe*, which is responsible to gather information about monitored objects and tasks.

Aiming to map and to describe the relationships among PPIs and BP elements (represented in BPMN), [60] make use of an ontology. The approach integrates PPIs into the entire BP life-cycle, i.e. from design and analysis (metrics are modeled together with the BP) to the evaluation phase (identification of correlations based on monitored information). Each PPI is specified by a measure element that contains values, thresholds, and the process instance object under measurement taken from the BPMN. The measures are classified as base, aggregated or derived. Base measures understand time (duration between two tasks), count (number of the instances verification within a time period), condition (check if the instance conditions are met), and data (measure certain object properties) measures. Aggregated measures are values obtained from the calculation of a certain aggregation function on a set of single measures values. Finally, derived measures are obtained through mathematical functions that combine two or more measures (base or aggregated). Considering these types of measures represented within an ontology, the main advantage of the approach relies on the specification of the PPIs dependencies through the *isCalculated* relation, which can have positive or negative impact. For instance, taking the PPI $z = \frac{x}{y} * 100$ then we have the relations: $isCalculatedPositively(?z, ?x)$ and $isCalculatedNegatively(?z, ?y)$. Considering these relations other two are derived that correspond to direct and inverse dependencies re-

spectively. Therefore, inference rules are applied to propagate the dependencies throughout all measured PPIs. An example of inference rule to find out (x, z) dependency can be: $isCalculatedNegatively(?x, ?y)$, $dependsInverselyOn(?y, ?z) \rightarrow dependsDirectlyOn(?x, ?z)$. Despite the advantages of inference rules within the ontology to find relationships at design-time, the approach is limited by the rules themselves and the degree of the indicators interconnections (heterogeneity).

There is another level of metric relationships that is regarding to KPIs. Wetzstein et al. [178] advocate that PPIs themselves do not represent useful business measurement. Instead, they are used as input for a higher-level metric, i.e. KPI, which combines one or more PPI and Quality of Service (QoS) metrics. PPIs are metrics based on business events (e.g., OrderReceivedEvent) while QoS parameters are metrics used to measure IT characteristics (e.g., web-service response time). The authors depict a framework for dependency analysis in order to discover the main factors that influence a specific KPI. Such elements are therefore presented as a decision-tree. The framework is divided in three layers, which are: *process runtime*, *monitoring*, and *analysis*. In the first layer, process runtime, a WS-BPEL process is executed, activity events are collected and sent to the second layer, monitoring. In the monitoring layer the information is used to calculate the PPI (which can be atomic or composite), the QoS (web-services compositions on top of SOA), and the KPI. A database stores all information for further analysis. The layer also receives as input a set of potential influential factors for each KPI from the user that will be further evaluated. Getting historical data from the database, the decision tree algorithm generates a dependency tree that shows KPIs impact factors. The information is then presented in the BAM dashboard to the users for BP optimization.

The analysis of influential factors in the last layer is divided into four main phases. In the first phase the user has to specify the input parameters to be analyzed, which are: KPI target value, analysis interval period, number of process instances, set of PPIs and QoS parameters considered as potential influential factors, and decision tree algorithm selection (in their experiments the authors used the well-know C4.5 [146] and ADTree [67] techniques). The training set is created during the second phase, in which the database is mined in order to extract the information with respect to the parameters defined in phase one. Hence, the KPI is evaluated and classified as "KPI fulfilled" or "KPI violated" based on its target value parameter. The third phase executes the selected algorithm and provides as output the dependency tree relating PPI/QoS with the KPI. Finally, phase four displays the obtained results within the BAM dashboard. The entire framework is strongly based on initial parameters

provided by the user, which may not have the proper knowledge to guide the framework analysis direction. Moreover, the identified KPI violation causes are not used to adjust the BP adaptation behavior in order to avoid upcoming indicators violation. In addition, there is no prediction mechanism to prevent the violation occurrence.

In order to automatically identify the KPI relations and extract their potential influential factors, the work proposed by Popova and Sharpanskykh [142] formalizes the concept of performance indicator and their internal and external relationships. Internal relations represent the relationships between indicators and external relations represent the relationships between indicators and other concepts such as goals, processes and roles. First each performance indicator is defined according to the following characteristics: name, definition, type (continuous or discrete), time frame (evaluation time internal), scale, min and max values assumed, source (where the information is extracted), owner, threshold (cut-off values that indicate the degree of influence between indicators), and hardness (qualitative or quantitative). After that, the indicator is associated with an expression in order be evaluated afterwards. A model is therefore created using a variant of the first order sorted predicate language in which indicators attributes are represented by predicates with arguments (e.g., **has_attribute_value**: *indicator* x *attribute* x *value*). The model defines three initial relations: **causing**, **correlated**, and **aggregation_of**. The first, **causing**, defines that one indicator causes changes to a second indicator. Changes can be positive or negative over a scale from very positive to very negative. Very positive impact means having small changes in one indicator such that it causes big positive changes in the second indicator. This relation is formally described using Temporal Trace Language. The **correlated** relation states that two indicators are positively or negatively correlated based on the previous relation. Finally, **aggregation_of** relation states that the first indicator is an aggregation of the second indicator. This means they hold the same measure, but at different levels. Having an aggregated relation implies the indicators have a positive correlated relation and the same attributes (type and unit). Based on these three relations, in particular **causing**, inference rules are used to discover new relationship based on the transitivity property, i.e., $((p \rightarrow q) \bigwedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$, and to validate the performance indicator structure.

In order to relate performance indicators with other concepts such as goals, tasks, and roles some predicates are defined as well. Examples of these predicates are: **is_based_on** (Goal x Indicator), **measures** (process x indicator), and **environmental_influence_on** (IT environment characteristics x indicator x {positive,negative}). However, the link

with external concepts (such as process) does not specify the process instance. Runtime variables may cause ambiguous values interpretation of the same process having two or more instances. This problem is partially solved by Rodriguez et al. [151] approach that quantifies relationships in the performance measurement system context (QRPMS). This is done by defining the relationships among KPIs and mapping them with Performance Measurement System (PMS) in order to create cause-effect relations at business goals level. The QRPMS is divided into four phases, which are: i) design and analysis of the PMS; ii) initial data treatment; iii) identification and mapping of KPI relations; and iv) result analysis. Focusing on the third phase, it receives as input a data matrix with all detailed information about the indicators. The relationships between KPIs are identified by applying two mathematical techniques over the data matrix. The first, principal component analysis (PCA), recognizes cause-effect relations based on each indicator description. The KPIs that contain cause-effect relations are named Business Drivers Key Performance Indicators (BDKPI) because of their high factor of impact with respect the others. The second technique, partial least squares models, quantifies the importance degree of each identified cause-effect relation. These models are represented by typical regression equations that predict effect(s) from cause(s) variable(s), called PLS models. It makes the approach to be highly based on the designer expertise in order to develop accurate PLS models.

2.2.4 Green BPM

Taking into consideration the green aspects of BP, Nowak et al. [130] introduce the green Business Process Reengineering (gBPR) methodology in order to tackle existing SBAs and energy consumption issues from a holistic approach within modern data centers. In order to identify the interrelationships the authors introduce Key Ecologial Indicators (KEIs), which are special types of KPIs to measure up business process greenness. The maximization of these metrics leads to a better use of virtual and physical infrastructure resources. The approach uses common BP optimization techniques, such as dynamic binding and control-flow and data-flow management, in order to find out possible optimal scenarios that fulfill quality and energy constraints. It relies on green IS approach, which focuses on the outcome of the process execution of the real business actions. For instance, to reduce the KEI "CO_2 emission" of a shipping process, the solution adopted is to reduce the number of times per day the shipper picks up the charge from three to one. Instead, the "CO_2 emission" with respect to the IT resources used to execute the

BP is not taken into consideration. A more comprehensive view of KEIs that also includes IT resources usage is presented in [131]. Therefore, KEIs consist in both green IS and green IT aspects, which are placed throughout the BP life-cycle in order to provide ecologically-aware process design and resource selection. The BP life-cycle is supported by a green BPM architecture depicted in Figure 2.1. The *input data* layer represents the runtime environment where sensors provide the necessary data to determine the indicators values. Having this information, the *event infrastructure* layer calculates KEIs and KPIs, in which process instances and IT resources usage are mapped. The *management components* layer extracts relevant indicators data (extraction, transformation, and loading - ETL service) in order to allow stakeholders to react on KEIs deviations. The enactment of stakeholder's decision is supported by the *adaptation* layer, in which the BP is effectively modified. Examples of BP adaptations are: adding or removing activities, change of data or control flow, and activities exchanging. Moreover, the figure highlights the components that need to included or extended in order to support green BPM in comparison with conventional BPM architectures.

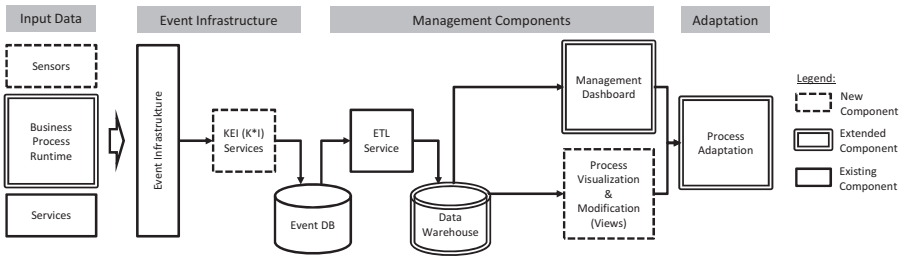


Figure 2.1: Green BPM architecture proposed by [131]

The usage of runtime metrics to identify the application greenness level without changing its current implementation is more likely if we consider the number of existing applications [35]. In this way, local and global controllers make decisions based on collected runtime data in order to pursue both desired performance and energy levels. The approaches proposes by [7, 98] consider self-optimization of application energy consumption at the Service-as-a-Software (SaaS) level, in which an application is defined as a set of services orchestrated by the business process, called as green SBAs. Lago and Jansen [98] state that green awareness from process, service and people aspects is the main issue to be solved. The authors propose a framework, called service greenery, which makes available environmental strategies and green metrics as-a-service (GaaS). The framework presents a similar approach with the

green BPM architecture of [131] with respect to monitoring and measuring and adaptation strategies enactment. The difference is regarding to the metric availability and the separated approach for green metrics with respect process, service, and people.

In [7] a set of event-condition-actions periodically checks if the SBA is using the best services from both quality and energy points of view. The optimization problem is modeled as a Constraint Satisfaction Problem (CSP) and, similar to other previous mentioned work, the single service energy consumption calculation is based on its associated resources. The approach takes into consideration processor, disks and memory power consumption, but the models used to estimate the energy consumption of a single service within shared environments is omitted. Instead, the authors focus on finding the best service for each BP abstract task such that global constraints (quality and energy) are optimal. To do that, the approach incorporates GPIs as Quality of Service (QoS) parameters. For example, during the enactment of service substitution adaptation mechanism, the first GPI considered is the service energy consumption per invocation (i.e., the ratio of energy consumption to the number of invocations of a service). Section 2.3 describes the most relevant service-based application adaptation approaches.

2.3 Service-based Applications Adaptation

Service-based Applications are characterized by independent services that, when composed, perform desired functionalities [156]. In general, these services are provided by third parties and are utilized by different applications. Thus, SBAs operate in a heterogeneous and constantly changing environment which includes both internal and external factors. To cope with that, they have to be able to constantly modify themselves in order to meet agreed functional and quality constraints in face of a raised problem, an identified optimization or an execution context change [90]. In the ambit of service oriented computing (SOC) and cloud computing, application adaptive features play an even more important role, in which adaptation mechanisms need to be performed in an automated or semi-automated manner. According to Di Nitto et al. [61], SBA self-adaptivity is achieved when the application "automatically and autonomously adapt their behavior to respond to evolving requirements, changes in its context, as well as failures of component services". The key aspects of self-adaptivity rely on when and why an adaptation should be performed which is applied throughout the three service-based applications layers: service infrastructure, service composition and coordination,

and finally, business process management layer.

Focusing on autonomous SBA adaptations, Bucchiarone et al. [36] delineate the effect of application design principles (design-time) within its execution (runtime) recovery capabilities. Thus, a SBA life-cycle that focuses on adaptation was created in order to cover both requirements changes and performance issues. Figure 2.2 depicts the proposed life-cycle where design-time (right side circle) and runtime (left side cycle) are represented together such that they support each other and allow the evolution of the application. In the figure, boxes represent monitoring and adaptation actions while hexagons represent the artifacts of which adaptation actions are based on. A set of these actions together composes an adaptation strategy. Adaptation strategies are triggered according to application degradation (functional/non-functional) or context shift that involves both environmental and stakeholders' requirements changes. Suitable adaptation strategies are selected based on the adaptation *scope* and *impact*, which may vary depending on the strategy trigger. For that purpose, the authors propose three SBA design-time adaptation modes: i) *built-in adaptation*, in which the adaptation needs and configurations are known in advance and fixed such as service re-execution; ii) *abstraction-based adaptation*, in which the adaptation needs are fixed but the configuration depends on runtime parameters such as service re-composition; and iii) *dynamic adaptation*, in which the adaptation needs and configuration are both defined at runtime and, therefore, specific mechanisms are provided for adaptation strategies enactment in a autonomous or semi-autonomous manner.

Due to the importance of context identification in supporting the selection the most suitable adaptation strategies, Ortiz and Prado [134] emphasize client-context models for SBAs. Not only the client-context is important, but server-context is crucial as well. Aiming to join both context models, [37] extend the proposed life-cycle. In the approach, context models are used to define dynamic adaptation triggers and to identify relevant data from the monitoring system. The context model depicts the application current situation according to six dimensions: *time* (access period), *ambient* (access mode), *user* (user's role and preferences), *service* (concrete services in the composition), *business* (business factors), and *computational* (software and hardware characteristics). Based on these dimensions, the approach maps the dimensions changes into the application available adaptation strategies. This is done through the identification of the adaptation triggers and their requirements, which are represented as "context-driven adaptation reasoners". Although the proposed context-based adaptation approach is quite complete from its context definition aspect, it provides limited analysis with respect to the

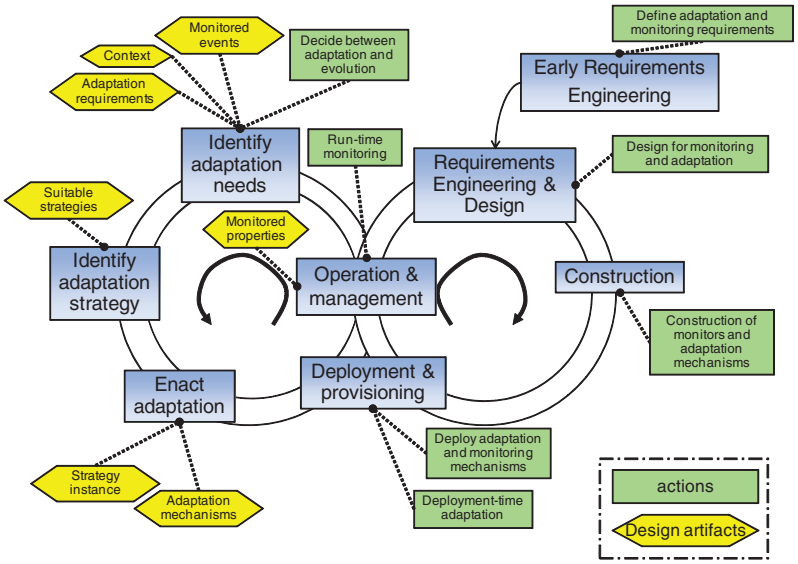


Figure 2.2: SBA adaptation life-cycle [36]

impact of the selected adaptation within other context dimensions that were not identified by the monitoring system before.

In the survey [157] the authors start from the identification of the adaptation triggering causes as *self* (regarding to the application layers) or *context* (regarding to the environment layer that affects somehow the application) which are viewed as closed feedback loops. Thus, adaptation strategies are triggered against *self* or *context* changes by enacting suitable adaptation actions with respect to costs and timing. In order to do so, a self-adaptation taxonomy is presented based on six aspects: *where*, layer and level of granularity that the adaptation is needed; *when*, temporal aspects of the adaptation enactment; *what*, attributes and artifacts that can be changed through the adaptation actions; *why*, set of goals of the self-adaptive system; *who*, autonomic adaptation level; and *how*, set of adaptation actions and their sequence and costs. The described self-adaptation taxonomy involves four main branches that are represented by the object to adapt, adaptation realization issues, temporal characteristics, and interaction concerns.

The first branch, object to adapt, deals with the *where* and *what* aspects and aims to identify the application layer (i.e., between the bottom middleware to upper business process), the granularity of the modules due to change, and the adaptation impact and cost. The impact depends on the adaptation scope while cost depends mostly on adaptation dura-

tion time, required resources and level of complexity. Based on that, each adaptation action is categorized from weak/low cost/limited impact, which are parameters changing actions, to strong/high cost/extensive impact which are components insertion, removal or modification actions. Although such categorization roughly introduces the basics towards adaptation actions inter-relationships recognition, it is still too genetic to effectively compute all possible side-effects of an action and its positive and negative influence all over application desired goals such as performance indicators.

The second branch, adaptation realization issues, deals with the *how* aspect by identifying all single steps to be performed in order to successfully accomplish the adaptation. Therefore, issues related to the selected approach, such as static or dynamic decision-making and internal or external mechanisms, and adaptation type, such as close or open adaptation and model-based or free adaptation, are described in details. The third branch, temporal characteristics, deals with the *when* aspect of the adaptation. It is sub-divided into reactive/proactive adaptation (i.e., after or before the change occurrence) and continuous/adaptive monitoring (i.e., the monitoring system either concerns the entire environment or adjusts the monitoring scope according to its needs). Finally, the interaction branch deals with the *who* aspect and regards to the level of human involvement, reliance of past situations, and interoperability support between application, middleware and infrastructure layers.

In order to identify the most critical facets of self-adaptive systems, Cheng at al. [49] propose a roadmap that is detailed into four analyses dimensions: modeling, requirements, engineering, and assurances. Focusing on modeling dimension, the authors have identified four distinct groups named *goals*, *causes*, *mechanisms*, and *effects*. *Goals* represent the objectives the system is expected to achieve and, therefore, they guide the selection of appropriate adaptation actions. The group *causes* contains the internal (self) or external (context) changes that trigger the adaptation actions. *Mechanisms* represent the adaptation actions enactments towards the desired changes and, finally, the *effects* group captures the impact of the adaptation within the system under consideration. Table 2.1 depicts each group's attributes with a short description.

2.3.1 Service Composition Adaptation

In this subsection relevant self-adaptive approaches are presented regarding to service composition adaptation within SOA. One of the most common adaptation action is service replacement, also called dynamic reconfiguration, which consists in replacing a concrete service at runtime

Table 2.1: Essential view of self-adaptation: Modelling dimensions from [49]

Group	Attribute	Description	Range
Goals	Evolution	Goal change lifetime	statical, dynamic
	Flexibility	Regard to the goal expression	rigid, constrained, unconstrained
	Duration	Validity of the goal	temporary ↔ persistent
	Multiplicity	Adaptation number of goals	
	Dependency	Capture how goals are related to each other	independent, dependent
Causes	Source	Identifies the origin of the change	external, internal
	Type	Nature of change	functional, non-func., technological.
	Frequency	How often the changes occurs	rare ↔ frequent
	Anticipation	Capture whether the change can be predicted	foreseen, foreseeable, unforeseen
Mechanisms	Type	Relation of the adaptation with the system	parametric, structural, both
	Autonomy	Degree of adaptation external influence	autonomous ↔ assisted
	Organization	Performed by single or multi components	centralized, decentralized
	Scope	Level of components involved	local, global
	Duration	How long the adaptation lasts	short, medium, long
	Timeliness	Capture the adaptation execution behavior	best effort ↔ guaranteed
	Triggering	How to start the adaptation	event-trigger, time-trigger
Effects	Criticality	Impact case of adaptation failure	harmless, mission-critical, safety-critical
	Predicability	Adaptation consequence prediction	non-deterministic, deterministic
	Overhead	Negative impact upon performance	insignificant ↔ failure
	Resilience	Ability to justify the provided resilience	resilient ↔ vulnerable

in an automatically or semi-automatically way. The approach proposed by [38] uses late-binding mechanism in order to perform composite service replanning during execution. Therefore, service composition is not based on estimated QoS values, but instead, current monitored information. The approach assumes that for each workflow abstract service there is a list of functional equivalent concrete services that may vary regarding to delivered quality parameters. The late-binding mechanism selects the concrete services that best contribute to maximize QoS constraints agreed in the SLA. The optimal solution is found applying an optimization problem in which it maximizes a fitness function while meeting the defined constraints. According to the authors, a genetic algorithm approach is the best strategy to handle the non-linear aggregation formulae and the high number of concrete services for each abstract service. On the other hand, integer programming is better for simpler situations. The proposed service composition replanning strategy defines three key points: i) *when* the replanning is triggered, ii) *where* is the workflow slice to be replanned, and iii) *how* the replanning will be performed.

The replanning is triggered if one of the quality constrains deviates from the expected value over a given percentage or the monitored value actually violates the quality constraint. After the identification of the problem, the approach selects which are the abstract services to be replanned in the workflow by considering the last concrete services executed. Depending on their position in the workflow control structure, the scope of the service replanning is adjusted. Thus, the fitness function maximization is applied only within the considered scope. By doing that the proposed algorithm reduces quality violations based on updating monitoring values and early replanning activation.

A similar approach is also presented by [102] in which concrete services are replaced according to their non-functional characteristics. The aim is to identify what and how many services need to be replaced according to their critical factors such that desired performance levels are achieved. The critical factor states the level of importance of one service within the entire application. Services with higher critical factors shall be replaced firstly in order to minimize the number of replacements and reduces possible side-effects generated from services dependencies. The service replacement is performed until a satisfactory solution is found or there are no more attempts to be performed. The critical factor is defined by the contribution the QoS of each service against the QoS of the entire SBA. Such contribution depends on the flow pattern of the service within the workflow, which is associated to a specific weight: Sequence = 1; Loop = 0.75; Parallel = 0.5; Selection = 0.25

He at al. [74] also consider the business process internal logic (workflow

patterns) and the impact of the adaptation on the other services in order to perform replacement and also service QoS re-negotiation. The adaptation model receives as input the services that need to be adapted and then calculates which adaptation action, replacement or re-negotiation, has higher profit based on cost. The optimal solution contains the adaptation action that brings the minimum Value Of Changed Information (VOC) recovery cost, the calculation of which is domain-specific. The approach takes into consideration nine flow patterns individually and also six combinations arranged by splits and joins pairs. In each case, the service role within the workflow has a different impact over the VOC computation. However, VOC calculation generally implies high costs within highly dynamic environments. In order to evaluate the impact of an environmental change within the service composition execution, Na et al. [124] propose Change Impact Probability (CIP) which calculates the probability of such impact through a quantitative mathematical model. The aim is to avoid the enactment of costly adaptation actions and focus on cost-effective ones. To do so, QoS values are represented as several levels according to their degree of influence on the SLA. To support the adaptation it is necessary to trace the service composition execution dynamically and to identify each single service flow pattern. This is done through a composite execution state model which is represented as tree structure where leafs are concrete service invocations and parent nodes are control flows. Although the workflow internal logic provides initial hints about possible services dependencies, it does not fully support non-functional analysis dependencies during the adaptation action selection phase.

Narendra [126] proposes to use Aspect-Oriented Programming (AOP) for specifying and relating services non-functional properties in order to enable on-demand service composition adaptation. Two main mapping relations are used: (i) non-functional requirements into service composition crosscutting aspects and (ii) composite services into individual services. In fact, the last relation enables to recognize the propagation of modification aspects from high (composite level) to low level (service level). In face of runtime non-functional requirement changes, the approach firstly determines what are the needed changes and, through a "concern integration" relation, identifies the role of individual services within the composition. Having the single services identified, it analyses each single aspect of the involved services that need to be modified. Finally, the approach maps and applies the changes from the service composition into each single service asking them to "reweave" (i.e., readjust) their behavior according to the new requirements.

Composite service adaptation is also the aim of the approach proposed

by [101], in which the adaptation is supported by a monitoring system. It uses a reactive approach that recognizes unhealthy service composition behavior regarding to the network environment or the single service malfunctioning. In fact, the authors focus on the identification of network environment events, such as CPU utilization threshold violation, that reflect QoS variations in the service composition model, such as increase the execution time of a specific service. The approach makes use of BP neural networks to perform the calculation impact of the network change into the composite service. Although the authors fill an important gap, they do not provide mechanisms to avoid the identified quality degradation nor prevent them to occur. A more comprehensive approach is proposed by [14], in which service selection is performed dynamically according to different users preferences enabling adaptive service compositions. The approach is formulated as mixed integer linear programing problem and makes use of MAIS framework [140], where services are dynamically selected at runtime. Through the usage of peeling techniques to solve loop iterations, the authors guarantee that all possible workflow execution paths satisfy all local and global QoS constraints even the most severe ones. In the case of over constrained quality requirements, the approach exploits negotiation techniques which autonomously re-negotiate quality parameters with the broker up to a max number of iterations or negotiation deadlines.

2.3.2 SBA self-adaptation frameworks

Considering more comprehensive approaches, SBA adaptation frameworks deal with large different types of service adaptations and, in particular, propose an integrated view from the infrastructure to the application layer [12, 91, 145]. These frameworks dynamic adapt SBAs from both structural and behavioral aspects by taking into account software and hardware changes.

The PAWS (Process with Adaptive Web Services) framework proposed by [12] divides service adaptation issues in process design and execution phases. The importance of process design phase is emphasized as it actually enables autonomous service adaptation at runtime. Three modules compose each phase. At design-time, the *advanced service retrieval* module matches all candidate services able to perform each task from the business process. Eligible candidate services present similar interface descriptions regarding to functional aspects and attend non-functional constrains at local level, i.e., at the task level. The functional matching is performed by the *mediator configurator* module which allows the *mediation engine* module to enact adaptation actions at runtime. The

last module at design-time, *SLA generator*, supports the framework to automatically negotiate QoS properties between the user and the service providers at local level. The module parses two different configuration policies. The first contains all QoS dimensions expressed as a vector of weights (user's policy) and, the second contains the candidate service-pricing model (provider's policy). At runtime, the *process optimizer* module selects one service among the candidates' ones in order to satisfy the previous defined local and global constraints. For that purpose, it uses mixed-integer linear programming optimization models based on runtime environment conditions (as seen in the previous sections). The *mediation engine*, previously configured at design-time, sets the connection between the deployed service invocations and the selected concrete services. In case of service execution failures, the *self-healing* module is triggered. The module is responsible to enact appropriate semi-automated recovery actions such as service retry (re-execution), redo (re-execution with different input parameters), substitute (use of another candidate service), and compensate (compensation actions to restore the previous state).

An extended version of PAWS framework can be identified in another framework proposed by [11]. The Discorso (Distributed Information Systems for Coordinated Service-Oriented Interoperability) framework provides a holistic view for SBAs that involves not only managing business process, but also their specification in an enriched way. Flexible business processes are defined as processes that "can break into independent elements and then recompose on demand" through self-validation and self-adaptation techniques. The design environment enriches the business process with annotated information regarding to automated services (abstract tasks) and manual tasks (human-based tasks as web interfaces). This information contains both functional requirements and QoS constraints, which are associated to supervision rules. The rules are responsible to monitor the underlying business execution environment and to trigger corrective adaptation actions at runtime when needed. Abstract services are matched with concrete ones at runtime through late-biding mechanisms that are based on mixed-integer linear programming optimization models.

Focusing on the interrelationships among the adaptation actions, [91] present a cross-layer SBA adaptation framework. The aim of the approach is to align the adaptation actions and monitored events from the different layers in order to obtain more effective adaptation results. Three layers are taken into consideration: BPM (business process management composed by the business process workflow and KPIs), SCC (service composition and coordination composed by service compositions

and process performance metrics - PPM) and SI (service infrastructure composed by service registry, discovery and selection mechanisms). The framework is divided into two groups that converge to service adaptation requirements. On the first group, monitoring mechanisms are able to raise events that shall trigger adaptation actions. On the other group, adaptation mechanisms are realized in terms of adaptation strategies which are composed by adaptation actions. The authors propose several adaptation actions suitable for each layer and their related triggering events. These actions are described in terms of requirements and mechanisms while monitoring events are described in terms of subject and mechanisms. At SCC layer, for instance, service replacement is available through dynamic biding in face of SLA violation. Service re-composition and control/data flow changes are also adaptation actions at SSC layer that are triggered by process model changes or KPI violations. They are enacted through automated compositions and model-driven transformations mechanisms.

The key point made by the authors regarding to their framework is that it takes into consideration the dependencies and effects of such actions within the three different layers. First, they tackle the lack of alignment of monitored events such that events and events mechanisms have to be related in a cross-layer way. It enables their correlation and aggregation. Second, the lack of adaptation effectiveness is filled by providing a centralized mechanism able to aggregate and coordinate different adaptation actions that are triggered by the same event. It also checks out different adaptation actions in order to find conflicts and interrelationships. Third, the lack of adaptation compatibility, which means to identify adaptation necessities across layers by identifying the source of the problem that generated the event. Finally, the lack of adaptation integrity is dealt in terms of foreseeing results. It means to ensure if the selected adaptation actions are enough to achieve the desired results and how many times they need to be enacted. Although the authors point out relevant characteristics from a holistic perspective, internal details of the framework are not fully described.

Mirandola and Potena [118] framework also considers dynamic service adaptation based on optimization models in order to minimize adaptation costs and enforce QoS aspects. The necessity for adaptation is assessed through a context-aware self-adaptation mechanism that captures required data about the environment and triggers appropriate adaptation actions. The novelty of the framework relies on the fact that it handles both software and hardware adaptation from functional and non-functional requirements perspectives. In addition, the framework optimization model is flexible as it is independent from adopted methodology

or architectural model. It deals with the following adaptation actions, called as adaptation plans:

- Hardware:
  1. Service's hardware deployment specification changes;
  2. Embed hardware resources into system where the service is deployed;
  3. Modify the characteristics of the underlying hardware resources in terms of server component such as CPU and memory.
- Software:
  1. Embed into the system one or more new services;
  2. Replace existing services with functionally equivalent ones;
  3. Modify the service dynamics by adding or removing service interactions.

Six main modules compose the framework. The *Software Models Creator* module generates system models based on the system current implementation. These models are inputs for the *Generator and Evaluator* module, which is subdivided into builder (handles the incoming adaptation requests and format them to the solver) and solver (processes the optimization models received from the builder and produces the results). Moreover, the *Executor* module implements the adaptation actions suggested by the *Generator and Evaluator*. Adaptation requests can be generated from users (*Users Requests Manager* module) or from the monitoring system (*Monitor* module). The *Users Requests Manager* module is also responsible to receive from the user what are the sets of adaptation plans to be considered for each new adaptation requirement. The last module, *Provider Info*, provides the necessary information about the service providers in terms of available hardware changes, which are used to cope with hardware adaptation actions. Figure 2.3 depicts the overview of the framework and the relations between the described modules.

Focusing on ubiquitous computing characteristics, the framework proposed by [188] enforces the dynamic aspects of SBA self-adaptation. The proposed approach tackles the problem when several adaptations are enacted concurrently. In this way, the framework acts as an adaptation coordinator in order to ensure the system correctness during and after the adaptation. The approach separates the different adaptability concerns in different modules. For instance, the actuation module enables the adaptation management module to act on the distributed application module, which represents the SBA components. Assuming adaptation

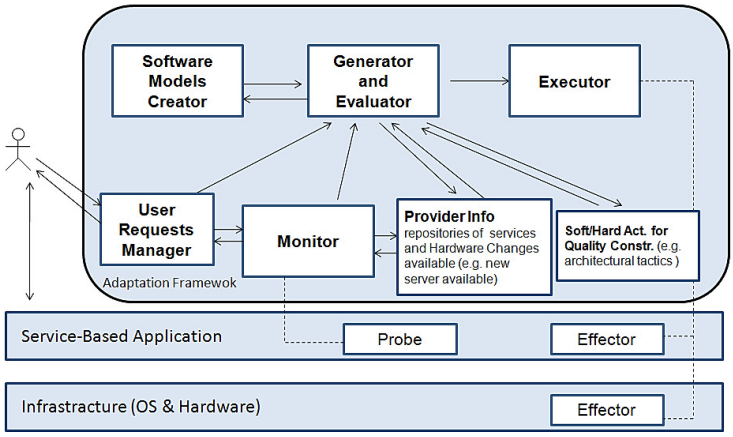


Figure 2.3: Mirandola and Potena [118] adaptation framework.

managers are autonomous, their interests may conflict with each other. This issue is solved by the decision coordination process, which "brings some deciders into a common decision-making to ensure non-conflicting and complementary decisions". It is based on a global satisfaction driven mechanism that can follow three different patterns: master-slave, strategy publishing or negotiation. In the first case, master-slave, one adaptation manager behaves as master and he is the only one who reason about context information. The master also binds the other managers to change their actions. In the second case, strategy publishing, the master manager stays, however the other managers are able to react against it by sending their context information. In the last case, negotiation, there is no master manager and all managers agree about global strategies. Conflicting situations are solved by giving different priorities to proposed adaptation strategies or adaptation managers. Although is the most effective, it becomes unmanageable in face of high number of adaptation managers.

Psaiser et al. [145] present VieCure framework. The framework focuses on unpredictable and faulty behavior of service into a mixed system of Service-based Systems (SBSs) and Human-provided Services (HPSs). The *monitoring and adaptation layer* is the framework interface that contains the mixed system environment. Feedback loop functions are used to provide the framework self-adaptation and behavior monitoring features through a MAPE-k cycle (Monitor, Analyze, Plan, Execute, and Knowledge). The monitoring components are responsible to gather and

to store information about different systems, mixed systems, regarding to the infrastructure, application activities, and QoS. The aggregation of such information is therefore presented as events that trigger the diagnosis and analysis component. These components define the required recovery actions by analyzing historical failure data sources. The framework focuses on unpredictable and flawed execution of services, which is identified through service behavior observation. Unhealthy situations are recognized through the monitoring of two types of tasks delegation behavior: *delegation factory* and *delegation sink*. In the first case, one node a accepts and delegates, without performing, large amount of tasks to another node b. In the second case, one node a accepts more tasks from other nodes than it is able to handle. Both cases lead to performance degradation issues. Once the failure source is identified, the framework triggers appropriate adaptation actions in order to compensate the effects of the service misbehavior. The framework implements three recovery actions (control capacity, add channel, and redirect delegations) that are selected through a rule-based approach. A simulated environment is used to validate the approach. In a more recent paper [144], the authors integrate the VieCure framework into the G2 SOA testbed, which provides real environment that enables dynamic adaptation. Obtained results confirm the satisfactory results of the healing framework algorithm considering a reactive approach.

Proactive self-adaptation mechanisms are presented in the PROSA (PRO-active Self-Adaptation) framework [76], which uses online testing techniques to detect undesired behavior before they actually occur. By online the authors mean that the test is performed at runtime instead of design-time. During the testing, four different cases are identified, which answer the questions *why*, *when* and *what* to test. Two major strategies are described with respect to what to test: individual services instances and service composition. For testing services instances, WSDL description is exploited while for service composition the framework exploits BPEL through regressing testing techniques. Despite the advantages of proactive self-adaptation, the use of online testing is fault sensitive once fault services can be invoked during the test. In order to compare the aforementioned frameworks, Table 2.2 highlights their main characteristics according to [49] roadmap previous described. The attributes of each group (column) are separated by comma.

Table 2.2: SBA self-adaptation approaches

Approach	**Goals**	**Causes**	**Mechanisms**
Canfora et al.[38]	static, constrained, persistent, and independent	internal, non-functional, and foreseeable	parametric, semi-autonomous, local, and event-triggered
Li et al. [102]	static, constrained, and independent	internal, non-functional, and foreseen	parametric / structural, autonomous, centralized, local, and event-triggered
He at al. [74]	static, constrained, and independent	internal, non-functional, and foreseeable	parametric, semi-autonomous, centralized, local / global, and event-triggered
PAWS [12]	static, constrained, and independent	internal, non-functional, and foreseeable	parametric / structural, semi-autonomous, centralized, local / global, and event-triggered
Cross-layers [91]	dynamic, constrained, and dependent	external / internal, functional / non-func., and foreseeable	structural, semi-autonomous, decentralized, global, and event-triggered
VieCue [144]	dynamic, constrained, and independent	external / internal, functional / non-func., and foreseeable	structural, semi-autonomous, decentralized, global, and event-triggered
PROSA [76]	dynamic, constrained, and dependent	internal, functional, and unforeseen	parametric, autonomous, decentralized, global, and event-triggered
SAFDIS [188]	static, constrained, and dependent	external / internal, functional / non-func., and foreseen	parametric / structural, autonomous, centralized, local / global, and event-triggered

2.4 Goal-driven models

Goal-driven models have been widely used in Software Engineering field in many different ways and, especially in Requirements Engineering (RE), where the objective is to focus on *why* systems are constructed instead of *what* features the system has to comply with [10]. Goal-Oriented Requirements Engineering (GORE) provides richer and higher-level abstraction models from which reasoning techniques are used to answer *why*, *who* and *when* questions during early software development phases. The goal-driven models allow the designers to specify the system goals and their relations such that they are aligned with the system requirements. Van Lamsweerde [99] defines the goal-based model as "annotated AND/OR graph showing how higher-level goals are satisfied by lower-level ones (goal *refinement*) and, conversely, how lower-level goals contribute to the satisfaction of higher-level ones (goal *abstraction*)". From GORE, two groups of frameworks emerged independently with common characteristics during early nineties: KAOS [59, 169] and the family of frameworks that includes *i** [181, 180], NFR Framework [123, 51] and Tropos [33]. These frameworks were pioneered in supporting early RE analysis through goal-driven models, where system non-functional requirements play key roles in designers decision-making supporting. Chung et al. [52] widely discuss about non-functional requirements definition, classification and representation. In particular, the authors highlight that goal-driven models provide the mechanisms to justify design-time decisions through the analysis of available alternatives.

KAOS [59] (Knowledge Acquisition in autOmated Specification or Keep All Objects Satisfied) emphasizes goal satisfaction through formal and semi-formal reasoning about agents and goals behavior. The framework has three main components: i) *conceptual model*, which provides both functional and non-functional requirements within a conceptual graph representation where nodes are abstractions (goals) and edges are structured links; ii) *acquisition strategy*, which represents domain models for traversing the conceptual model through a 7-steps approach; and iii) *acquisition assistant*, which automates the acquisition strategy component. The conceptual model is represented as a three levels model and described by meta level, domain level, and instance level. The meta level is composed by meta-concepts (such as "agent" and "actions"), meta-relationships (such as "performs" and "isA"), meta-attributes (such as "postCondition" and "cardinality") and meta-constraints that connect goals, actions and agents. While all meta elements are domain-independent abstractions, the domain level instantiate the meta level elements into the system domain. Considering a library system exam-

ple [59], the "agent" can be represented by the "borrowerName" attribute and the "action" can be the "bookISBN". Finally, the instance level represents instances of the domain level elements, i.e., the attributes values. In the example, the "borrowerName" can be represented by the value "Alex" while "bookISBN" by "978-88-7488-549-7". Concerning to the conceptual model, goals are declarative statements that represent objectives of which the system has to achieve and, based on their behavior pattern, are classified into: achieve (cease), maintain (avoid) and optimize. Goals are further specialized into *satisfaction*, goals that satisfy agent requests; *information*, goals that provide information about system objects states to agents; *robustness*, goals that represent recovering features; *consistency*, goals that look for maintaining the consistency between the system parts; and *safety*, goals that maintain agents within safe states. Goals are refined through AND/OR decomposition until be realizable by singular system actors (agents). At this point, they are formalized in terms of operational constraints over objects states that are monitorable and controllable by agents. In the library system example, the goal *EnoughCopies*, which is a subgoal of *BookRequestSatisfied*, can be "operationalized" by the *LimitedBorrowingAmount* constraint, formally defined as:

$$\begin{aligned} &(\forall lib : Library, bor : Borrower, bc : BookCopy) \\ &\pounds\,[\#\{bc \mid Borrowing(bor, bc) \wedge bc \in lib\} Max(bor)] \end{aligned} \tag{2.1}$$

where the function $Max(bor)$ defines the upper bound for the number of copies that borrower *bor* can borrow and the symbol £ indicates the *maintain* behavior of *EnoughCopies* subgoal.

Although KAOS models represent both functional and non-functional requirements, there is no clear difference among them within the model specification. However, in opposition to functional characteristics, non-functional requirements do not need to be completely satisfied. To deal with that, Letier and van Lamsweerde [100] propose quantitative reasoning techniques in which weighted contribution links are used for quantifying the many design alternatives impact on high-level goals. Thus, the degree of satisfaction of the goal (a non-functional one) is represented by the weighted average of its subgoals satisfaction. On the one hand, weighted links support better understanding of the model and provide more accurate decision-making process. On the other hand, the definition of such weight is not trivial and relies on stakeholders subjective criteria. In order to soften this issue, the authors adopt *quality variables* (domain-specific, goal-related variables) and *objective functions*

(domain-specific, goal-related values that are maximized or minimized). Looking to identify the proper quality variables and objective functions, five heuristics are proposed: i) to identify quality variables from conditions constrained by goals, ii) to specify new quality variables according to parent goals, iii) to specify quality variables from required level of analysis, iv) to identify objective functions from domain standards, and finally v) to identify objective functions from goals categories/patterns.

During the early nineties, John Mylopoulos, Eric Yu, Lawrence Chung and Brian Nixon [182, 123] introduced the initial concepts and representations of *i**/NFR which, together with KAOS, were milestones in RE field. Tropos [33] methodology appeared later on as an *i**-based modeling with enriched domain semantics. However, all approaches focus on non-functional requirements for early-phase RE support. The models distinguish functional and non-functional requirements as hard-goals and softgoals, respectively. The *soft* characteristic of softgoals relies on their subjective aspect, in which there is not only one way to achieve the goal, but one might be better than other. Therefore, they cannot be tackled in a clear-cut sense such as goals satisfaction. Thus, the NFR Framework [123, 51] introduces the concept of *satisficing* in which requirements are not absolutely satisfied but acceptable under certain limits. A softgoal is *satisficed* when there is sufficient positive and little negative evidence towards the non-functional requirement achievement. The opposite is valid when the softgoal is *denied*. These evidences are propagated by several types of labeled contributions links that exploit the softgoals interdependence. The labels MAKE (++) and HELP (+) are used to represent respectively fully and partially positive contribution of one offspring softgoal (sub-softgoal) into its parent softgoal. In the opposite way, BREAK (--) and HURT (-) represent fully and partially negative contributions respectively. Such positive and negative contributions are analogous with weighted links in KAOS. The central idea is that actors depend on each other through goals/softgoals satisfaction/satisficing. Thus, both goals and softgoals are refined into subgoals and sub-softgoals through AND/OR decomposition. Focusing on softgoals, an AND link indicates that all offspring softgoals are needed to address the parent softgoal achievement while an OR link indicates that any offspring softgoal is needed. The approach also identifies priority softgoals, which are used to make appropriate trade-offs and to define possible softgoals *operationalizations*. Softgoals operationalizations, normally represented by the most refined nodes, represent development techniques that "say how" to provide the desired level of quality. All this information is visually represented in the *Softgoal Interdependence Graph* (SIG), where explicit and implicit interdependencies are

also represented, i.e., positive/negative contribution links. As an example, Figure 2.4 shows the SIG of a system adapted from [51] that manages customer accounts. The softgoal "good performance" is refined into two sub-softgoals ("response time" and "space") through an AND decomposition. Operationalizations are represented by the actions "use indexing" and "use uncompressed format" which have positive contribution on "response time". An implicit negative contribution of "use uncompressed format" can be inferred as compressed formats are more space-efficient.

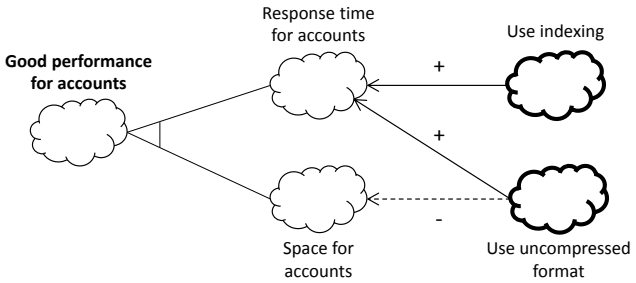


Figure 2.4: NFR Framework - SIG example adapted from [51]

It is clear from the figure that the operationalization softgoal "use uncompressed format" is conflicting with two parent softgoals. At this point, an early decision has to be made by choosing or denying operationalizations. This decision will further be evaluated when, using a bottom-up approach, operationalization softgoals propagate satisficed or denied evidences into higher-level softgoals (qualitative reasoning). Positive contribution propagates the offspring label up to the parent, while negative contribution propagates the inverted offspring label up to the parent. Assuming the operationalization "use uncompressed format" and "use indexing" are satisficed, therefore "response time for accounts" shall be labeled as satisficed and "space for accounts" shall be labeled as denied. Such decision is later justified by linking operationalization softgoal with functional requirements.

The *i** [180] framework focuses on actors autonomous behaviors. In the model, an actor is defined as "an active entity that is capable of independent action" and they can be humans, hardware, software or a combination of them. The actor behavior is described in terms of motivational and intentional attributes. The author argues that, modeling actors intentions provide richer expressiveness, such as *why* an actor takes a certain decision or action. Intentional description can be derived from NFR and KAOS frameworks as well, but they do not care about the social interactions among actors. In *i** actors depend on each

other, in a beneficial or harmful way, to achieve their objectives. Such dependencies are represented in the *Strategic Dependency* (SD) model, which represents actors external relationships in a high abstraction level. In the SD model, one actor (*depender*) depends on another (*dependee*) for something (*dependum*). Considering the previous library example, the *borrower* depends on the *library system* in order to *borrow a book*. The dependum is divided into four types: task dependency (activity), resource dependency (entity), goal dependency (assertion), and softgoal dependency (quality). With this type of model is possible to analyze opportunities and vulnerabilities of actors within a global view. However, actors internal details are needed in order to justify external relationships. The *Strategic Rationale* (SR) model represents goals, softgoals, tasks, and resource attributes within the actor scope. In SR models *means-end* links between tasks and goals indicate ways to achieve the goals. Finally, task decomposition links represent subgoals, subsoftgoals and subtask while contribution links indicate positive (MAKE/HELP) and negative (BREAK/HURT) through the quality achievement. Therefore, in order to improve the system operation, analyses are carried out from each actor perspective reflecting its dependencies. The SR model provides the available paths that actors can take while the SD model provides their implications throughout the system. A comparison made by [23] between KAOS and *i** identified that KAOS provides more description support of tasks while *i** enforces the tasks relationships. In addition, the risks analysis of KAOS is more advanced than *i** through the concept of obstacles. Such drawback is solved and enhanced by the *i*-risk* [109] and the Tropos-based Goal-Risk [15] frameworks.

Tropos [33], an *i**-based modeling methodology, is an agent-oriented software development methodology that copes with all main development phases, from early requirements to the software implementation. In particular, the methodology focuses on the software early requirements phase which is not well consolidated as the other phases. Besides the intentional approach proposed in *i**, Tropos adopts Agent Oriented Programming (AOP) concepts and flexible characteristics. Tropos makes use of agents and their mentalistic notion throughout all development phases, which is based on Belief, Desire, and Intention (BDI) agent architectures [148]. With respect to early requirements, the methodology introduces the goal analysis, which describes goals in terms of a non-deterministic concurrent algorithm. The algorithm starts with an initial set of actors and goals. Then, each root goal is broken from its actor's perspective in order to derive new subgoals that are delegated to other actors. The process goes on until all actors' desires be satisfied by goals and subgoals. Moreover, Tropos provides a new specification language,

Formal Tropos [68], that brides the gap between *i** requirement models and formal methods. The language enriches the domain semantics by adding constraints, invariants, and pre/pos conditions to the graphical models, which can be validated using model-checking techniques.

2.4.1 Risk analysis in goal-driven models

In order to identify the goals role throughout the model and to provide mechanisms in which decision-making algorithms can reason about benefits and drawbacks of goals fulfillment, risk management techniques are used. The use of such techniques into software development process was introduced by Barry Boehm [28, 29] framework. The framework, composed by two primary steps and six sub-steps refining the primary ones, defines the basic activities for software risk analysis in Software Engineering, which are adopted by many following frameworks. Liu et at. [104] present a systematic literature review of how Software Process Simulation Modeling (SPSM) can address software risk management. In [132] the authors present an empirical study that highlights the main difficulties in identifying and monitoring software risks. In particular, they deal with the intrinsic risk complexities and the lack of an ultimate approach to manage subjective calculations. Fuzzy Cognitive Map (FCMs) are used for risk assessment in [26] due to its capability of modeling complex, uncertain and subjective information. These maps are graphical representation of a squared matrix where each cell represents a weighted relationship between two risks factors (input and output). The weight of a relationship means its impact risk and is calculated through fuzzy membership functions. The input function parameter is calculated through IF-THEN fuzzy logic rules over five linguistic values that are provided by experts and from "Very Low Risk" to "Very High Risk" range. The identification and classification of software development risk factors is the aim of [53], in which a 3-phases 10-steps framework is proposed. The authors have systematically identified 170 sub-factors that are grouped into 44 situational risk factors. Such factors are classified into eight categories that aim to cover the entire software development life-cycle. This influential factor classification is used as basis for several quantitative software analyses, like risk analysis, in order to support weighing techniques.

The framework proposed by Roy and Woodings [154], named ProRisk, provides a comprehensive approach for risk analysis of software development projects. It involves the identification of factors that can impact on quality highlighting the high risk ones and enacting strategies that mitigate such risks. The approach organizes, in a tree structure, 194 fixed

quantifiable risk factors that may impact on the project risk at different levels, which characterize the *risks perspectives*. The risk value is calculated by measuring the impact of raised unwanted factors, called *events*. The event impact is calculated through the product of the likelihood of the event and its impact cost. The likelihood is represented as a typical probability distribution, which is a triangular function of optimistic (left corner), most likely (top corner), and pessimist (right corner) values. On the other hand, the impact costs are weights defined by the expert at each level of the tree (subjective judgment). Aggregated values at higher levels are obtained by summing up offspring events impact values, where nodes are mutually independent. Although the approach does not support dependency relationships with simple impact propagation, it provides fundamental concepts for risk analysis in goal-driven models.

All the goal-driven frameworks previous introduced have their own way to support the designer decision-making process in order to provide higher rewards. In [170] the author demonstrates how KAOS methods and supporting toolset can be used towards risk analysis within goal-based models, called *obstacle analysis*. An obstacle is defined as "precondition for non-satisfaction of some goal, assumption, or questionable domain property used in the goal model". This analysis aims to create a risk tree (risk-based models) linked with each *achieve* type goal node. This tree is refined in terms of AND/OR decomposition such that subobstacles leafs nodes are linked to countermeasure goals through resolution links. Quantitative (through weighted scores) and qualitative (through labels such as --) reasoning algorithms are used to select the alternative option with minimum risk for the system. This is done by eliminating obstacles nodes, reducing their likelihood, or attenuating their consequences. Aiming to provide a more complete risk management approach with KAOS models, Islam [79] extends KAOS layers in order to represent both technical and non-technical components with Goal-driven Software development Risk Management (GSRM) framework. As in [170], Islam's approach also adopts obstacles as negative contributes towards the goal satisfaction. However, the framework splits the model into four layers that, although related, are independent from each other. The *goal layer* represents KAOS goal model with AND/OR decomposition. The *risk-obstacle layer* identifies potential risk factors that contribute negatively to goals satisfaction. The *assessment layer* details the risk factors into risk events, which are composed by likelihood and severity properties. This layer also includes the risk metrics, which are used to obtain likelihood of risk events based on their linked risk factors. However, the author does not provide in depth information about this calculation. Finally, the *treatment layer* identifies available actions in order to mitigate

holding risks.

Based on *i** models, the approach proposed by [109] introduces *i*-risk* which extends *i** by integrating actors cultural characteristics with risk analysis techniques. The authors define risks as unsatisfied dependencies, goals, tasks or unavailable resources, which are divided into outer risks (e.g., unsatisfied dependency) and inner risks (e.g., unfinished task). With that, the risk analysis is carried on looking for denied dependencies, where broken dependencies are turned into risks for the *depender*. This makes hidden risks to appear, as the analyze does not consider only denied dependencies, but all existing *depender* links. For example, if the provider does not deliver the expected service, it represents a risk for the consumer. And if the consumer does not pay, it represents a risk for the provider. Having all risks identified, the authors use *i** actors cultural factors in order to define the risk impact degree. In order to demonstrate the approach, a cultural Chinese database was used as example where they learned that "people do not mind their personal time being taken up by work, so the risk of deviation [of project delivery] from schedule is weakened".

In Tropos methodology the risk analysis was extended towards more advanced techniques by several approaches, from which the approach presented by Asnar et al. [15] is the most relevant throughout this thesis. The authors introduce the Goal-Risk (GR) framework which, differently from Tropos, concepts are organized in three layers: *asset layer*, *event layer*, and *treatment layer*. Analogous with the layers presented by Islam [79], the *asset layer* represents strategic interests of the actors that are expected to be achieved by the system, in which goals are refined through AND/OR decomposition. The *event layer* represents uncertain circumstances that are not controlled by actors. These circumstances, i.e. events, impact positively or negatively over goals. Events are also refined through AND/OR decomposition and have two attributes: likelihood and severity. The likelihood is calculated based on events that support or prevent the event occurrence through qualitative values: (L)ikely, (O)ccasional, (R)are, or (U)nlikely such that L > O > R > U. On the other hand, severity represents the degree of impact of an event within the goal fulfillment. Such degree is qualitatively defined as strong positive (++), positive (+), negative (-), and strong negative (--). Finally, the *treatment layer* contains sequences of actions that are used to fulfill goals or to mitigate negative events. As in the other layers, actions can be refined through AND/OR decomposition.

Figure 2.5 depicts the proposed model layers where goals are shaped as ovals, events as pentagons and tasks as hexagons. Moreover, the model distinguishes four types of relations. *Decomposition* relations (depicted

as solid line with logical operation in between and without arrow) are used to represent subgoals, subevents and subtasks which follow refinement strategies proposed in *i**/Tropos. *Contribution* relations (depicted as solid line with filled arrow) represent the impact of one node (goal, event or task) over another within the same layer or between treatment and event layers. The relation is also labeled with the impact degree (ranging from ++ to --) and the type of propagation that can be satisfaction, denied, or both. Based on propagation rules and the contribution relation labels, positive and negative evidences are qualitatively represented as (F)ull, (P)artial or (N)one, where F > P > N. *Impact* relations (depicted as dash line-arrows) represent the impact of an event over one or more goals. An event can characterize a risk (negative impact) for one goal and, at the same time, be an opportunity (positive impact) for another goal. As impact relations indicate the impact of events on goals, they are the target for alleviation relations. *Alleviation* relations (depicted as solid line with hollow arrow) are responsible to mitigate negative risky impact relations by reducing their severity (e.g., -- $\mapsto$ -) through the enactment of tasks in the treatment layer.

The selection of the most suitable set of tasks and their expected benefits are evaluated by a qualitative risk reasoning process. In short, this process consists in: i) finding alternatives solutions using as input GR model labels (through *backward reasoning* algorithm) and acceptable risks/quality constraints; ii) evaluating each candidate solution against relevant risks and possible treatment tasks in the case of identified unacceptable risks/qualities; and iii) assessing and enacting suitable tasks in order to mitigate the risks through *forward reasoning* algorithm, which propagates the positive and negative evidences of the solution throughout the model.

2.4.2 Goal-driven SBA adaptation

SBA adaptation can be based on both reactive and proactive. The adaptations presented in the previous section are mainly reactive, i.e., the adaptation is triggered on service failure events. However, even if a failure is not identified the adaptation can be triggered in order to improve somehow the current scenario (such as higher performance or lesser energy consumption) due to new available services or new user's requirements. Moreover, adaptation actions can also be triggered in order to prevent foreseen failures through forecasts mechanisms. Gehlert and Heuer [69] approach focus on the first situation. The authors propose service replacement adaptation every time there is a new available service that better contributes towards the application goals fulfillment

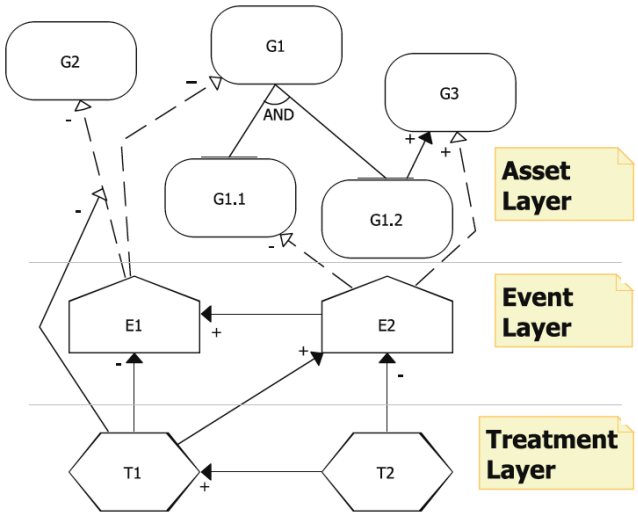


Figure 2.5: Goal-Risk model from Asnar et al. [15]

(self-optimization) and provides equivalents functionalities. In order to do so, the authors use a goal-driven approach that enables satisfaction analysis of single goals with respect to the entire model provided by Tropos. After the identification of functional equivalent new services, the approach distinguishes four different situations for adaptation: i) the new service provides equal goals satisfaction; ii) the new service provides different goals satisfactions ratios; iii) the new service adds new functionalities to the application which are expressed as new hard-goals in the model (goal extension); and iv) the new service has less functionalities, but can be combined with other services in order to fulfill the expected goals into a better way than before (goal reduction). Based on that, the Tropos quantitative reasoning algorithms are used in order to calculate goals satisfiability and deniability, which properly identify the gains of the adaptation.

Franch et al. [66] introduce MAESoS, a **M**onitoring and **A**daptation **E**nvironment for **S**ervice-**o**riented **S**ystems that keeps stakeholders goals aligned with the system runtime behavior. The approach tackles both system design-time and runtime aspects. At design-time, i^* models represent system actors and stakeholders goals, quality models assess defined goals, and variability models are used to identify and to select suitable system adaptations when they are needed. At runtime, a monitoring system collects all involved services behavior data in order to detect goals and quality requirements violations. If so, the variability models are used to perform semi-automated corrective adaptation ac-

tions. These variability models support two types of adaptations based on execution performance and stakeholders' variations. Considering a running example, the authors demonstrate how to trace the impact of runtime system behavior within high-level goals and vice versa. However, the approach does not consider the inter-dependencies among different adaptation actions, which may limit the variability models scope. Such need is partially accomplished by [121] through the usage of the Belief-Desire-Intention (BDI) agent models of Tropos. Tropos methodology is extended in order to support interrelationships between goals and the system environment where SBA failures and correspondent recovery actions are represented as design abstractions. Based on Tropos models, the authors introduce a fault modeling dimension, which captures errors that may lead to failures, their symptoms, and the linking between symptoms and possible recovery actions.

Due to high dynamic changeable scenarios in which service-based applications are into, Dalpiaz et al. [58] propose a model-based self-reconfiguration mechanisms in response to failure or context change. In the approach, failures are characterized by the identification of no progress or inconsistent behavior towards the goal fulfillment. Although the monitoring, diagnosis and reconfiguration mechanisms are framed into a timed analysis, the approach is not able to recognize the cause of the identified failure. Driving the system adaptation according to dynamic environmental changes is the aim of [83], in which goals evolve according to requirements changes. The concept of live goals proposed in [18] to support service composition adaptation at runtime changes the model itself according to the identification of non-satisfied goals. However, the approach does not discover new dependencies relations nor perform timed analysis. Chopra et al. [50] use the autonomous and heterogeneous agent' characteristics from Tropos in order to reason about SBA at business level through commitments. Commitments represent relationships between two agents trough conditional propositions, which capture the model elements context. For instance, in an auction application the "bidder" may has a "payment" commitment with the "seller" if he "won the bid". An agent commitment support analysis is performed in order to certify that all agent's commitment will be fulfilled or to avoid the creation of commitments that are impossible to be fulfilled by the agent. The identification of how the environmental context affects goal fulfillment is described in the next subsection.

2.4.3 Context-aware goal-driven approaches

A survey on context awareness conducted by [108] highlights that the effective use of context is a complex issue not fully addressed by several existing context-aware systems. In context-aware SBA, Ceri et al. [46] use conceptual modeling contexts to trigger application adaptation actions. In fact, context is one of the most common factors to guide the adaptation. Tackling service composition problem, the authors in [129] exploit the context multi-granularity characteristic in order to identify different context attributes and correlations. The vertical granularity is divided into three levels (role, attribute, and content) while the horizontal granularity allows the merging of related context roles into a more general one or their split. Although goal-driven modeling provides a useful way to specify the system desired behavior, the more complex the system, the higher the numbers of goals and their interdependencies. A model developed at design-time is often based on assumptions that can be verified only at runtime. Thus, the modeling process results in a complex activity that, occurring at the early stages of the software development process, is crucial for the quality of the system.

In order to make the model more flexible within high dynamic environments, model contextualization, which involves identification and utilization of context information are evaluated. Considering both design-time and runtime models, the approach proposed in [147] solves a service selection problem through a context annotated goals-based models at design-time and goal-refinement method at run-time. Ali et al. [4, 5] formalize the use of context information within Tropos models through six contextual variation points, which identifies the context data placement within the model. The authors define context as "a partial state of the world that is relevant to an actor's goals" [5]. In this way, context is used to ensure an existing relation between two model nodes by checking if the context holds or not in terms of conditional statements. Similarly to goals refinement, the identified context information is refined through AND/OR decomposition, where context statements are composed by one or more context facts. The difference between statements and facts are regarding to their verificability, in which facts can be verified by agents and statements cannot. Also, the authors propose a new context-aware reasoning technique for goal-based models, which is used to derive context information from requirements variants and user priorities and to derive the minimum-cost tasks that have to be enacted in order to meet quality requirements.

Najar [125] proposed a context-aware intentional framework for SBA adaptation called CI-SOA. Differently from *i**, the CI-SOA focuses on

the dynamic aspects of SBA context information that makes stakeholders goals also evolve. This is done through the interconnection of two models: context model and user's behavior schema, which is represented as a rule-based system. The context model follows a multilevel ontology representation which is divided into upper and domain-specific ontologies. The first captures generic knowledge regarding to the user (e.g., profile and role), environment (e.g., location and time) and computational entity (e.g., service and network). The second level is the upper level knowledge specialization in several sub-domains. For example, location can be specialized into town and street while network into latency and bandwidth. In general, the framework is composed by four components. The *context management* component is responsible for collecting and interpreting context data in order to extract relevant knowledge about users and environment. The *user behavior management* component traces users' historical behavior and properly stores it. The *request processor* component represents the users' needs that are analyzed and defined as users' intentions. Finally, the *proactive trigger* component is activated by service discovery requests and is responsible to deduce user's intentions based on the analysis of their traced behavior against their context model.

In order to support design decision based on system environmental impact, Zhang et. al [187] uses Goal-oriented Requirements modeling Language (GRL) [8, 107], which is a variant of *i**, to represent system agents social and intentional behavior. Through detailed quantitative analysis, the aim is to estimate the system environmental impact according to some environmental-friendliness non-functional requirements (softgoals). Such quantitative analysis is possible due to the insertion of small quantitative calculation models within tasks, which are particular ways of doing something towards the goal achievement, and softgoals relationships. Such calculation models capture context of the underlying environment in order to provide quantitative values of how much each task contribute (positively or negatively) to satisfy top levels softgoals. The calculation model is represented as a 4-tuplea attributes: `<inputs,outputs,restrictions,error>`. The input attribute represents detailed information with respect to the execution of the linked task. The model output provides the expected value of the linked softgoal. The restriction attribute states in what circumstances the model can be used. Finally, the error attribute provides the model calculation error. Considering a packing and delivery managing system model, one possible task is the "use of RFID device". Assuming that such task is linked to a "low energy consumption" softgoal, the calculation model between them is following described:

Input: use frequency (times/a)
Output: energy consumption (kWh/a)
Restrictions: RFID model = Motorola MC family
Error: +/-20%

The existence of calculation models within each task–softgoal relation provides a clear view of the impact of each design-time alternative option within the system non-functional requirements. However, these models are rather simplistic with respect to the restriction attributes, which can be runtime dependent in some cases. Moreover, a mix approach between quantitative and qualitative analyses shall speed up the process in very complex system models.

2.5 Summary

The identification of energy inefficiencies within data centers is a challenging issue mainly due to the high number of involved components. We have studied the most relevant characteristics of these components from a energy perspective and how they are related to each other in order to understand the consumed energy flow. The major limitation of most approaches is that they do not have a strong link between the application and the infrastructure, i.e., the application does not abstract the infrastructure capabilities and vice-versa. This chapter points out how current service-based metrics and adaptation mechanisms could be used to improve energy efficient based on a service oriented architecture. In order to provide such integration, we have studied goal-driven models, which are able to map all involved components through specific relations types that enable impact propagation analysis.

3 Service-based applications measurement and composition

Service-based models [6] are commonly adopted within modern data centers environments. As the utilization of these models are steadily growing for many applications domains, their impact on the data center infrastructure is becoming more and more significant. In such scenarios, the available computing resources and applications are accessed as-a-service, in which computational capacity is provided on demand to many customers who share a pool of IT resources. The Software-as-a-Service (SaaS) model can provide significant economies of scale where multiple service providers can offer functionally equivalent web services that may differ for their offered non-functional aspects.

Such non-functional aspects include performance and energy aspects, in which performance is calculated using quality attributes (e.g., response time) and energy is obtained through power models (e.g., VM power consumption [30]). Focusing on SaaS models and despite the research work towards application monitoring and measuring (discussed in the previous chapter), we believe they are not enough as they do not make a clear connection between quality and energy aspects within heterogeneous data center environments. We argue that, in order to improve the

design and execution of SBAs, the very first step is to assess the ongoing situation. This is commonly performed by data collection and data representation through metrics. A wide variety of energy metrics have been used, however it is difficult to compare different obtained results [179]. Even though metrics to measure data center power efficiency, proposed by Green Grid, Uptime Institute, Lawrence Berkeley National Laboratory, and Greenpeace, are quite similar the comparison of their results should be made with caution, since the values are calculated in different ways or measured following different methods and equipment.

Considering the approaches discussed in Section 2.2 of the previous chapter, this chapter focuses on data representation, in which raw monitored variables (including both runtime and design-time phases) are properly transformed and represented as metrics named indicators. According the Oxford Dictionaries Online [1], an indicator is defined as "a thing that indicates the state or level of something". In this chapter, indicators are used to represent meaningful information about the underlying environment at different granularity levels. To do so, we took the following approaches:

- Quantitatively represent the application BP characteristics using metrics such as size and complexity. This kind of information is considered as static since it is defined during the process design-time phase. Such metrics are used to compose the BP profile, in which single resources needs and workflow behavior are represented.

- Evaluate the SBA from both quality and energy aspects in order to enable trade-offs between the expected execution (represented by the constraints) and the available execution conditions, which sometimes does not attend all constraints. This is modeled as a Service Composition (SC) problem in which indicators attributes (in particular the soft thresholds characteristics) are used to define a Concrete Execution Plan (CEP), i.e. all abstract tasks of the BP are invoked by one or more concrete services, even though not all constraints are satisfied.

- Take into consideration the energy aspect mentioned in the previous item. To do so, a new indicator is introduced, namely SBA Energy-Efficiency Indicator (SBA-EEI), in order to measure up single and composite services using aggregation functions. The proposed indicator is detailed in Section 3.2, which benefits are demonstrated in Section 3.3.

[1] `http://oxforddictionaries.com`

- Aggregate indicators into a more general index that provides meaningful information about one or more aspects of the system and take into consideration cross architectural layers indicators. The goal is to enable analysis at different granularity levels in order to support suitable adaptation strategy selection when necessary. Section 3.4 describes the proposed aggregation system that is able to combine different indicators from different granularity levels.

3.1 Business process measurement

In order to capture the BP main characteristics some metrics can be used to extract this information [143]. In this thesis we concentrate on BP resource attributes, which represents the computational resources required and used during the BP execution. As illustrated below, the BP characteristics and the data from the monitoring system are used to calculate key/green performance indicators.

3.1.1 Key/Green Performance Indicators

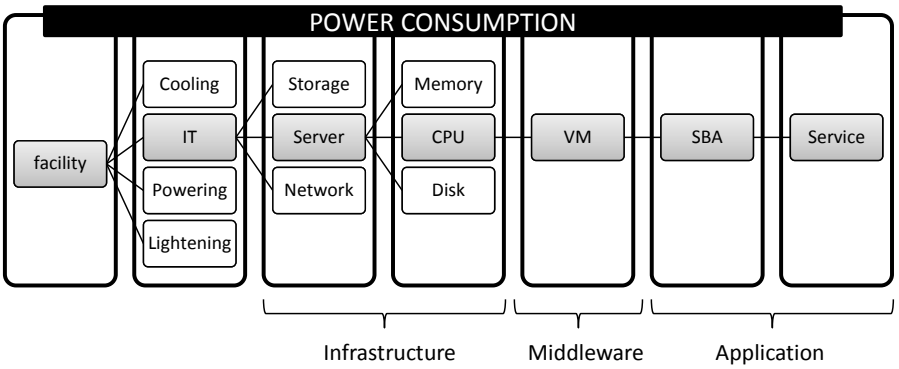
In [115] we have presented two different types of indicators: Key Performance Indicators (KPIs) and Green Performance Indicators (GPIs). KPIs measure performance indexes that are based on quality models, while GPIs measure the system greenness with respect energy consumption. All types of indicators span through the three architectural layers and some of them might be part of both groups.

In order to achieve business objectives, indicators are used to report the company's ongoing situation against its defined objectives. This is commonly done by a set of KPIs, which is defined as "a quantitative or qualitative indicator that reflects the state/progress of the company, unit or individual" [142]. The same definition is also adopted by the GPIs, but focusing on company's energy related objectives. However, the utilization of GPIs requires an extended and more general definition of indicators, such that both KPIs and GPIs characteristics are represented. Towards this direction, in [115] we discuss and propose a more detailed definition of indicators (suitable for KPIs and GPIs) and delineate how these indicators can be aggregated in order to provide meaningful information about the underlying system to the system manager.

Figure 3.1 depicts, as an example, how power consumption spans across different architectural layers, which goes from the entire facility (towards left side) through the resource used by a single task/web-service (towards right side). While we focus on the IT component, we argue that there is a gap between the server and the service layers. In order to deal

with the five layers shown in the figure, we have combined them into three layers: *infrastructure* (e.g., server and CPU), *middleware* (e.g., VM), and *application* (e.g., SBA and service). The infrastructure layer involves the server and its physical devices. In the power consumption example shown in the figure, CPU is highlighted as it usually has higher influence over power consumption with respect to memory and disk considering an average server. The Virtual Machine (VM) is placed within the middleware layer and the power consumption calculation follows a model-based approach [87]. Finally, the application layer involves the SBA and its elements, which are represented by concrete services (Web-Services - WS) that implement the business process abstract tasks.

Figure 3.1: Power consumption layered view



For instance, the indicator `energy consumption`, which represents the power consumption over a period of time of a physical device or logical component, can be applied across all the three considered layers such that: i) *Server/CPU* represents the entire server energy consumption, including operating system and all running virtual machines; ii) *VM* represents the energy consumption of one or more virtual machines with respect to its allocated capacity; and iii) *SBA/Service* represents the energy consumption of a specific application inside the VM and comprises all task/WS involved in it.

In order to cope with the issues described, an indicator (GPI or KPI) is defined according to the meta-model described in Figure 3.2. The abstract class *Indicator* defines both GPIs and KPIs which are identified by the attribute *type*. The attribute *importance* is based on the user's preferences and it dictates an indicator priority in case of multiple violations.

The indicator violation is defined by its thresholds sets, in which one set is composed by two lower-bounds and two higher-bounds. Each in-

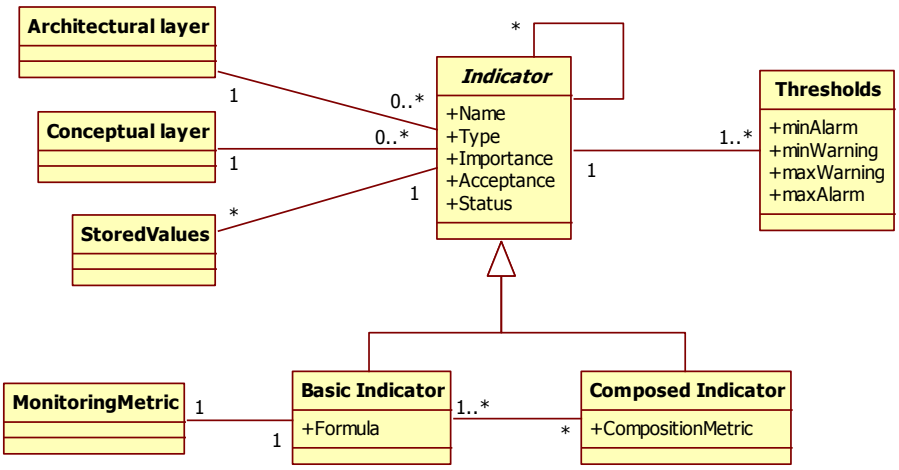
dicator has one set of thresholds at least. Based on that, an indicator is not violated when its current value is within both warning and alarming boundaries of all sets, named as **green**. Otherwise, there are two types of violation. In the first type, named as **yellow**, the indicator value violates min or max warning threshold and does not violate alarming ones. In this case, although the indicator is not within the desired situation, it is considered acceptable. In the second type, named as **red**, the indicator value violates both warning and alarming threshold. In this case, it indicates an unacceptable situation. Alarming indicators have priority to be solved with respect warning ones as they cause higher damage to the system. The current indicator situation, i.e. green, yellow or red, is represented by the attribute *status*.

The indicator last attribute, *acceptance*, defines the accepted time interval of which an indicator can stay in a violation state (warning or alarming violation) without requiring adaptation. This attribute is important in order to avoid adaptation overload, in which violation is temporary and might not require adaptation. For instance, let us suppose that the action VM migration may violate the indicator availability of an application. In this case, the acceptance attribute shall dictate the maximum amount of time the indicator availability can stay in a violation state without triggering a new adaptation action, i.e., the maximum amount of time the VM migration action can take without causing a new side-effect.

The indicator value is obtained through the indicator formula calculation, which uses the information provided by the monitoring system represented by *MonitoringMetric* in the figure. Indicators are divided in basic and composed indicators. An indicator is called as basic (*BasicIndicator* class) when its calculation formula is either a direct measure from the monitoring system variable, like application response time, or a combination of several monitoring variables within a formula, like number of transactions per second (TPS). On the other hand, composed indicators (*CompositionMetric* class) formulae use other indicators as input values, such as application performance that is the ratio of TPS and power. In this case, there is no value transformation and they represent compatible values. For instance, the server power consumption value in Figure 3.1 is obtained either through the monitoring system (physical installed sensor) or the sum of its components by the composition metric: $\sum \big(Power(memory) + Power(cpu) + Power(disk)\big)$. In [48] such dependency is named as Operational Dependency.

The indicator architectural layer represents where the indicator is placed within our three-layer architecture. As described before, our approach takes into consideration three layers that define the indicator

Figure 3.2: Indicators meta-model



granularity. In addition, indicators are classified in three conceptual layers in order to identify the level of significance of an indicator from an organizational perspective [93]. The three identified layers (operational, tactical and strategical) represent different degrees of importance into the following risk analysis.

3.2 SBA Energy Efficient Indicator (SBA-EEI)

Considering that energy efficiency is related to IT loads and most data center servers remain running at low utilization rates or in idle mode for a considerable amount of time, we introduce an Energy Efficiency Indicator at the application level called SBA-EEI [114]. The indicator goal is to explore the data center heterogeneity and to identify existing spaces for usage improvement based on the utilization of low power servers. Making better usage of low power servers, the impact shall be reflected into the data center cooling and power systems as well. Due to servers heterogeneity in most data centers, servers can be classified into different classes. In our approach, we classify them from the slowest to the fastest through different energy consumption levels. Zenker [186] says that, using a multi-dimensional coefficient, it is possible to compare results among different environments.

Based on that, the first important assumption adopted in [114] is that energy consumption is directly proportional to computational power performance [19]. Although this is not true in all cases - e.g. very old

Server Class	Idle (0–0.12)	Normal (0.13–0.67)	Burst (0.68–1)
Slow	178	222	266
Average	486	607	728
High	6,485	8,106	9,727

Table 3.1: Classes of servers and their correlation power and utilization rate

equipment with low computational power and high energy consumption rates - we assume that all the heterogeneous servers considered are new and homogeneous with respect to this aspect. According to this basic assumption, three classes of servers have been created that can be separated according to their quality and energy consumption rates. This partition of servers into classes was inspired from Koomey's report [96], in which, servers were divided into: *volume*, *mid-range*, and *high-end* according to their overall system costs. Table 3.1 presents these classes of servers (*slow*, *average*, *high*) and their weights with respect to energy consumption during three different utilization periods (*idle*, *normal*, and *burst*) that were measured in Watts. The latter numbers were derived from measurements performed by [133, 166, 173, 96].

By inspecting the data shown in Table 3.1, the energy efficiency (ee) metric of a single application service is computed based on its executed server class, taking into account possible server modes during a fixed time period. Despite the fact that Table 3.1 presents the *burst* column with the maximum energy consumption rates, we do not consider these values for the following reasons: i) the usage of an admission control scheme [176] is assumed, which is responsible to maintain the number of execution services within the *normal* utilization level by dropping the overload requests; ii) beyond the *normal* utilization limit, although the energy efficiency will increase, the boundary of accepted quality, which for instance involves execution time, will be exceeded. Energy efficiency can be computed by Eq. 3.1. According to that, $ee_j^{\Delta t}$ of the service s_j executing in server class class(j) is computed by dividing the amount of energy consumption of the real execution of the service (i.e., when the server is in the *normal* mode) with the total energy consumed by the server in our specific time unit of reference.

$$ee_j^{\Delta t} = \frac{ec_{\text{class}(j)}^{\text{normal}} \cdot t_j^{\text{normal}}}{ec_{\text{class}(j)}^{\text{idle}} \cdot t_j^{\text{idle}} + ec_{\text{class}(j)}^{\text{normal}} \cdot t_j^{\text{normal}}} \tag{3.1}$$

For example, suppose we want to calculate the energy efficiency of service s_1 that is executed in a *slow* server in one specific time unit, where 45% of the time is executed in *normal* mode and 55% in *idle* mode. Then, from Eq. 3.1 and Table 3.1 we will have that : $ee_1^{\Delta t} = \frac{222 \cdot 0.45}{178 \cdot 0.55 + 222 \cdot 0.45} = 0.505$

As can be easily seen from Eq. 3.1, there is a direct relationship between energy consumption, service execution time, and energy efficiency. The object of research is how to exactly compute this quantitative dependency based on Eq. 3.1. Considering the fact that the execution time of the service is measured according to our specific time unit of reference and that the service is executed in a certain server class, we can derive Eq. 3.2. The numerator of Eq. 3.2 calculates a service's total energy consumed in *normal* mode by multiplying the energy consumed in this mode according to our unit time reference ($ec^{\text{normal}}_{\text{class}(j)}$) with the total time spent by this service in this mode ($\frac{ec_j - ec^{\text{idle}}_{\text{class}(j)} \cdot et_j}{ec^{\text{normal}}_{\text{class}(j)} - ec^{\text{idle}}_{\text{class}(j)}}$) with respect to its total execution time et_j. The denominator of Eq. 3.2 is the total energy consumed ec_j by this service. In other words, Eq. 3.2 has the same physical meaning as Eq. 3.1, as it calculates the percentage of energy consumed by a service in *normal* mode with respect to the total energy consumed by this service. Moreover, this new formula expresses our inquired quantitative dependency as it dictates the way the energy efficiency of a single service can be computed by measuring its execution time and its total energy consumption. The latter two metrics can be computed for each service through a monitoring layer [106], which provides information on the fly about application workload, resource utilization and power consumption.

$$ee_j = \left\lceil \frac{ec^{\text{normal}}_{\text{class}(j)} \cdot \frac{ec_j - ec^{\text{idle}}_{\text{class}(j)} \cdot et_j}{ec^{\text{normal}}_{\text{class}(j)} - ec^{\text{idle}}_{\text{class}(j)}}}{ec_j} \cdot 100 \right\rceil \tag{3.2}$$

Besides the aforementioned dependency, other types of dependencies and constraints can be derived from a service's past execution (in all classes of servers) and from its specification in a service profile (obtained from the BP profile previously presented). Without considering other quality attributes like availability and reliability, we can have constraints 3.3 and 3.4 on execution time and energy consumption defining the range of admissible values of these two dimensions. Equality constraint 3.5 defines how the price of the service is produced from a cost model that takes into account the service's execution time, normalized by the server class, and energy consumption rates. We assume that the

cost of a service depends linearly on the time it needs to execute and on the amount of energy consumed, where the first partial cost depends also on the server class (see constant $\alpha_{\text{class}(j)}$). Of course, apart from linear, other types of functions could be used instead [54].

$$et^{min}_{\text{class}(j)} \leq et_j \leq et^{max}_{\text{class}(j)} \tag{3.3}$$

$$ec^{min}_{\text{class}(j)} \leq ec_j \leq ec^{max}_{\text{class}(j)} \tag{3.4}$$

$$pr_j = \alpha_{\text{class}(j)} \cdot et_j + \beta \cdot ec_j \tag{3.5}$$

Based on the above analysis, a service can operate in different quality and energy levels and constraints can be used to capture the service's quality and energy efficiency in all these levels. Then, according to the application domain in which this service is used, user preferences can be issued in the form of constraints and a service discovery process can be executed in order to select those services that satisfy the user needs.

3.3 Energy-aware design of SBA

While in the previous section the problem of energy and quality-aware selection of single services was analyzed, we consider now the case of composite services, for which a service is built from other services at runtime when the user's requirements are issued to a broker or service composition engine. The composite service construction is separated into two sequential phases: i) an abstract execution plan is built; ii) one service is selected for each abstract task of the execution plan. As described in Chapter 2, various techniques have been proposed for automatically or semi-automatically creating an abstract execution plan of a composite service based on the functional needs and the functional capabilities of available services. We do not deal with this phase and we assume that the execution plan is already in place as an outcome of the first phase or as a product of an expert that designs the process (e.g., in the form of Abstract BPEL).

In the second phase, based on the abstract execution plan, for each abstract task a set of functionally-equivalent services are selected as candidate services that implement the same functionality, but differ in their non-functional characteristics, i.e., quality and energy. The functional selection of these candidate services is a very well-known problem that has been efficiently solved with approaches like the one in [141]. The final goal of this phase is achieved by solving the Service Concretization (SC) or QoS-based Service Composition problem and is the focus of the

presented approach. According to this problem, the best service available at runtime has to be selected among all candidate ones for each abstract task taking into consideration global and local non-functional constraints.

In order to guarantee the fulfilment of global constraints, SC approaches use optimization techniques like MIP [184, 14] or Genetic Algorithms (GAs) [39]. However, most of these approaches usually consider the worst or most common case scenario for the composite service (that concerns the longest or hottest execution path, respectively) [184, 39] or they satisfy the global constraints only statistically (by reducing loops to a single task) [80]. Thus, they are either very conservative or not very accurate.

Most of these approaches present the following disadvantages, which are solved by the approach proposed below and published in [114] : i) they do not allow non-linear constraints like the ones we have outlined in the previous section; ii) they do not produce any solution when the requirements of the user are over-constrained, while in the following, we adopt two levels of constraints which allow the identification of warning and alarming violations separately; iii) they are very conservative concerning the fact that all execution paths have to satisfy the global constraints – even the longest ones that are not so likely have to satisfy all of the global constraints and, therefore, some good solutions are removed when these constraints are very tight – while we allow constraint violations (warning thresholds) for the unlikely execution paths; iv) they take into account only the worst or average value of a metric for each service and they also regard that all metrics are independent, while we allow ranges of possible values and metric dependencies; v) they do not take into account energy related metrics.

3.3.1 Main definitions and assumptions

The first main assumption is that a composite service is characterized by a single starting and ending task and that the composition of tasks follows a block structure. In this way, only structured loops can be defined, i.e., loops with only one entry and exit point. We name each abstract task of the service composition with the term *task* (t_i), while the set of services S_i to be executed for this task are called *candidate services* (s_j). We symbolize with I the total number of tasks of the composite service specification and with J the number of candidate services retrieved from the system's registry. The goal of the process that solves the SC problem is to find the optimum execution plan OEL^* of the composite service, i.e., the set of ordered couples $\{(t_i, s_j)\}$, indicating that task t_i is exe-

cuted by invoking service s_j for all tasks of the composite service. This is done until all the global constraints related to energy and quality are satisfied. The latter constraints are either explicitly specified by the user or can be implicit in the user profile. We assume that these constraints are expressed by the following upper or lower bounds depending on the monotonicity of the attribute.

Based on the past execution of the composite service stored in system logs or from the designer's experience, the following two types of information can be derived and evaluated [184, 39, 14]:

- *Probability of execution of conditional branches.* For every switch s, we symbolize with NB^s the number of disjoint branches out of s and with p_h^s the probability of execution of each disjoint conditional branch. For all these probabilities, the following constraint must hold: $\sum_{h=1}^{NB^s} p_h^s = 1$.

- *Loop constraints.* For every loop l, we define the expected maximum number of iterations IN^l as well as the probability p_h^l for every number of iterations h of the loop. For all these probabilities, the following constraint must hold: $\sum_{h=0}^{IN^l} p_h^l = 1$. A loop cannot have an infinite number of iterations, otherwise the composite service could not be optimized since infinite resources might be needed and consequently global constraints cannot be guaranteed [184].

These two types of information can be used to transform the abstract execution plan of a composite process to a Directed Acyclic Graph (DAG) through the use of loop peeling [16]. The latter method is a form of loop unrolling in which loop iterations are represented as a sequence of branches whose branch condition evaluates if loop l has to continue with the next iteration (with probability $\{p_h^l\}$) or it has to exit.

After loop peeling, from the transformed DAG, we can derive a set of K `execution paths` ep_k that identify all possible execution scenarios of the composite service. An `execution path` is a set of tasks $\{t_1, t_2, \ldots, t_l\}$ where t_1 and t_I are the initial and final tasks, respectively, and no tasks t_{i_1} and t_{i_2} belong to alternative branches. Every execution path ep_k has an associated probability of execution $freq_k$ that can be evaluated as the product of the probability of execution of the branch conditions included in the execution path. Moreover, we associate to each execution path ep_k a set A_k of the indices of the tasks included in it. In addition, each execution path ep_k has a set of *subpaths* that are indexed by m and denoted by sp_k^m. A subpath of an execution path contains those tasks of the execution path, from the initial to the end task, so that it does

not contain any parallel sequence. For every possible concrete execution plan CEP, i.e., all subpaths including the optimum one, we evaluate the quality dimensions under consideration under the hypothesis that the composite service is executed along the corresponding execution path using the aggregation patterns that will be analyzed below. In Figure 3.3, this is represented by the `assignment` class.

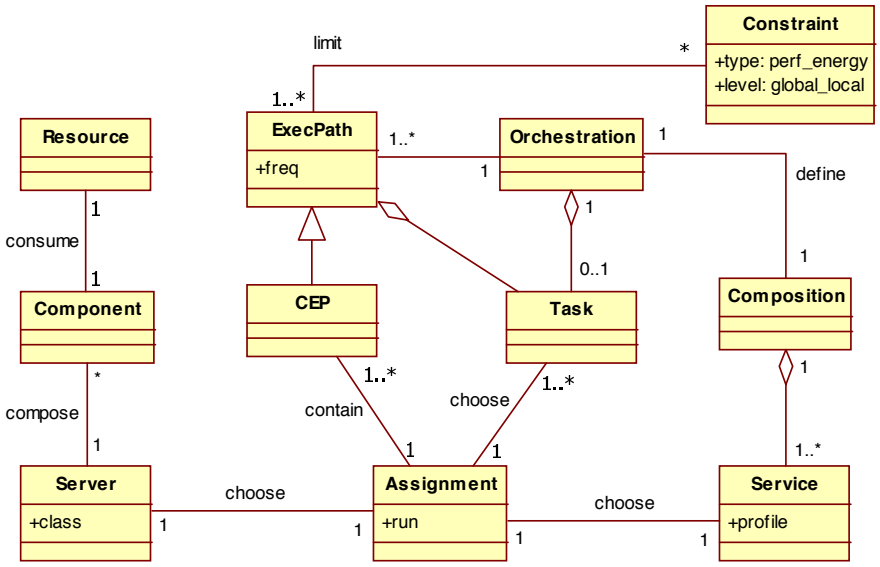
Figure 3.3 gives a more general idea of all involved elements in the presented SC approach, in which every service s_j is selected as a candidate service based on its advertised service profile sp_j that is stored in the system's registry. In this service profile, the functional, quality and energy capabilities of the service are defined based on information submitted by the service provider and the past execution of the service. Moreover, this service profile specifies the server class class(j) on which the service s_j executes. If the service runs also in a different server class class($j^{'}$), then it is considered as a different service $s_{j'}$ and its capabilities are stored in a different service profile $sp_{j'}$. It must be noted that the `assignment` class contains an attribute *run* indicating if the chosen service is currently running on the designated server class. In this way, the proposed SC approach will fetch only the available services that are deployed on their indicated server class at that time. We accommodate for the case where the resources are dynamically allocated in a hosting site, as if a service stops running on a service class and starts running on a different class through VM migration, by updating the `assignment` class and, in particular, the *run* attribute.

According to the approach described above, a service profile does not advertise only quality and energy level of a service by storing only one (average or minimum) value for every possible dimension. Instead, it contains all possible levels using the constraint set we have introduced, i.e., normal, warning, and alarming. Thus, the service profile sp_j of a service s_j contains a set of constraints that involve variables qd_j^n that are associated to the quality and energy dimensions $qd^n, n \in N$ where N is the total number of dimensions.

We have considered two of the most representative quality dimensions, namely *execution time* and *price*, and two energy dimensions, namely *energy efficiency* and *energy consumption*. For these four dimensions, we consider the following relevant information:

- Execution time (ET) is the expected duration in time that a service spends to fulfill a service request. For each service s_j it is represented as an integer variable et_j that takes values from the following domain of values: $[et^{min}, et^{max}]$. It is measured in a specific time unit like seconds.

Figure 3.3: SBA energy-aware design



- Price (PR) is the fee/cost that a service requester has to pay to the service provider for the service invocation. For each service s_j it is represented as an integer variable pr_j that takes values from the following domain of values: $[pr^{min}, pr^{max}]$. It is measured in a defined currency like euro. This dimension depends both on the execution time and energy consumption dimensions.

- Energy efficiency indicator (EE) is a measure of how efficiently a service uses energy (analyzed in Section 3.2). For each service s_j it is represented as an integer variable ee_j that takes values from the following domain of values: $[ee^{min}, ee^{max}]$. It depends on both execution time and energy consumption dimensions.

- Energy consumption (EC) is a measure of the total energy consumed by the service during its execution. For each service s_j it is represented as an integer variable ec_j that takes values from the following domain of values: $[ec^{min}, ec^{max}]$. It is usually measured in Watts or miliwatts per hour.

The aggregation pattern, within the SBA scope, for each of these four dimensions for every execution path is given in Table 3.2. Execution time of a composite service is computed by the maximum execution time calculated in all possible subpaths of the execution path. In case

Dimension	Aggregation Function
Execution Time	$et_k\,(CEP) = \max_{sp_m^k \in ep_k} \sum_{\substack{t_i \in sp_m^k \\ (t_i,s_j)\in CEP}} et_j$
Price	$pr_k\,(CEP) = \sum_{\substack{t_i \in ep_k \\ (t_i,s_j)\in CEP}} pr_j$
Energy Efficiency	$ee_k\,(CEP) = \frac{1}{\lvert A_k\rvert} \sum_{\substack{t_i \in ep_k \\ (t_i,s_j)\in CEP}} ee_j$
Energy Consumption	$ec_k\,(CEP) = \sum_{\substack{t_i \in ep_k \\ (t_i,s_j)\in CEP}} ec_j$

Table 3.2: Aggregation patterns for each considered dimension

of parallel executions, the longest subpath is taken. For each subpath, the execution time is calculated as the sum of all the execution times of the services that are contained in it. The price of a composite service is computed by the sum of prices of all component services contained in the execution path. The energy efficiency of a composite service is computed by the average of the energy efficiency value of each component service contained in the execution path. Finally, the energy consumption of a composite service is computed by adding the energy consumption of all component services contained in the execution path.

3.3.2 Proposed Approach

In [114] we formulate the SC problem as a Constraint Satisfaction Optimization Problem (CSOP) [153]. The main decision variables of our problem are the following:

$$z_{i,j} = \begin{cases} 1, & \text{if the task } t_i \text{ is executed by service } s_j : j \in S_i \\ 0, & \text{otherwise} \end{cases}$$

The goal of the SC problem is to maximize the aggregated quality and energy value by considering all possible execution paths ep_k of the abstract execution plan and their corresponding probability of execution $freq_k$. To this end, we have used the following optimization function for defining our problem:

$$\max \sum_{k=1}^{K} freq_k \cdot sc_k$$

In this way, we try to find the solution that is the best at least for the most frequent execution paths.

According to the above optimization function, a score sc_k is produced from the aggregated quality and energy value for each execution path. This score is obtained by applying the Simple Additive Weighting (SAW)

technique [78] to the list of considered dimensions. According to this technique, the raw aggregated values for each dimension are first normalized using a corresponding evaluation function that is specific for each dimension and then multiplied by the weight (i.e., the impact) of this dimension. This weight is either given explicitly by the user or is obtained from his profile. So by denoting the aggregated value of each dimension along a specific execution path ep_k with q_n^k and the user-provided weights of this dimension as w_n, the score of ep_k is obtained from the following equation:

$$sc_k = \sum_{n=1}^{N} w_n \cdot f_n\left(q_n^k\right)$$

Based on the above equation, different evaluation functions can be used to normalize the values of different dimensions. We have carefully chosen a specific type that allows the use of soft instead of hard global constraints for restricting the aggregated values for each dimension and for each execution path. Depending on the monotonicity of the dimension, we have used the following two (denoted by 3.6 and 3.7) evaluation functions for negative and positive dimensions, respectively:

$$f_n(x) = \begin{cases} a_n + \frac{q_n^{max} - x}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), & q_n^{min} \leq x \leq q_n^{max} \\ \max\left(a_n - \frac{q_n^{min} - x}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), 0\right), & x < q_n^{min} \\ \max\left(a_n - \frac{x - q_n^{max}}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), 0\right), & x > q_n^{max} \end{cases} \quad (3.6)$$

$$f_n(x) = \begin{cases} a_n + \frac{x - q_n^{min}}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), & q_n^{min} \leq x \leq q_n^{max} \\ \max\left(a_n - \frac{q_n^{min} - x}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), 0\right), & x < q_n^{min} \\ \max\left(a_n - \frac{x - q_n^{max}}{q_n^{max} - q_n^{min}} \cdot (1 - a_n), 0\right), & x > q_n^{max} \end{cases} \quad (3.7)$$

where q_n^{min} is either the minimum domain value for this dimension or a user-provided bound, q_n^{max} is either the maximum domain value for this dimension or a user-provided bound, x is the value to be normalized, and a_n is a number between 0.0 and 1.0 given by the user or the composite service designer in order to allow values outside the ranges specified within the constraints. Values within this range represent the indicator warning violation. In order to give a more specific example, the evaluation function of the execution time quality dimension is given

by the following formula:

$$f_{et}(x) = \begin{cases} a_{et} + \frac{ET-x}{ET-q_{et}^{min}} \cdot (1 - a_{et}), & q_{et}^{min} \leq x \leq ET \\ \max\left(a_{et} - \frac{q_{et}^{min}-x}{ET-q_{et}^{min}} \cdot (1 - a_{et}), 0\right), & x < q_{et}^{min} \\ \max\left(a_{et} - \frac{x-ET}{ET-q_{et}^{min}} \cdot (1 - a_{et}), 0\right), & x > ET \end{cases}$$

In this case, we can have that: $q_{et}^{min} = et^{min}$, or the user also provides another bound for the highest level (lowest value) of this dimension that is: $q_{et}^{min} = ET^{'}$.

As it can be observed from the latter equation, the aggregated dimension's value is allowed taking values outside the user requested bound or range of values but the produced normalized value decreases and gets to zero when the aggregated value's distance from the bound increases. Actually, the initial normalized value and how quickly this value decreases depends on both the design parameter a_n and the two bound values. Table 3.3 shows three examples of a_{et}, in which $ET - q_{et}^{min} = 5$ and $ET = 7$. In the first column we provide some input for the equation presented above, where in the first three rows the indicator is not violated. In the following columns we depict the obtained value for three different examples of a_{et}. Considering that indicators within its thresholds are considered **green**, values $\leq a_{et}$ state the the indicator ET as **green**. If the value is within a_{et} and zero, the indicator is considered as **yellow**. Finally, **red** indicators have values equal to zero. Have said that, the warning range of $a_{et} = 0.4$ is much shorter than a_{et}, where the indicator is considered as **red** with $x = 12$. In the example, if $a_{et} = 0.8$ the alarming threshold is $x = 27$; if $a_{et} = 0.6$ the alarming threshold is $x = 17$; and if $a_{et} = 0.4$ the alarming threshold is $x = 12$. So if we want to allow a very small amount of values outside the user requested bound, we have to use small values of the a_n parameter, depending also on the min and max bound values and their distance. It must be noted that the value of a_n can be predefined or produced from a predefined table that maps the distance between the min and max values of a dimension to the value of a_n. Similarly, the weights given to each dimension can be equal. In this way, the user quantifies only the bounds of the dimensions and no other parameter, so there is no cognitive overload for him.

The reason for using the above type of evaluation functions is that we do not want to rule out solutions that do not violate in a significant way the user's global constraints. In this way, if the SC problem is over-constrained, a solution can be found that violates in the smallest possible way the least number of global constraints. Of course, considering how the SC problem is formed, if this problem is not over-constrained, then

$f_{et}(x)$	$a_{et}=0.8$	$a_{et}=0.6$	$a_{et}=0.4$	condition
2	1.0	1.0	1.0	$q_{et}^{min} \leq x \leq ET$
5	0.9	0.8	0.7	
7	0.8	0.6	0.4	
10	0.7	0.4	0.2	$x > ET$
12	0.6	0.2	0.0	
17	0.4	0.0		
22	0.2			
27	0.0			

Table 3.3: Examples of relation between a_{et} and warning threshold

violating solutions will always get a lower score than the correct solutions. Moreover, we extend the approach proposed by [14], in which all execution paths, even the longest non-probable ones, have to satisfy the global constraints. In our approach, this assumption is relaxed by allowing solutions that violate some of the global constraints of the SC problem for the non-probable execution paths. Thus, solutions that satisfy the global constraints only for probable execution paths are not ruled out from the result set but they get a smaller score with respect to those solutions that satisfy the global constraints for all execution paths. We achieve this goal by using appropriate evaluation functions that return a zero normalized value for undesired aggregated dimension values and the following set of constraints: $[sc_k > 0,\ \forall k]$ that rule out those solutions that have a zero score for at least one execution path.

Based on the above analysis, the CSOP that has to be solved in order to determine the optimum execution plan OEP^* is the following:

$$\max \sum_{k=1}^{K} freq_k \cdot sc_k \tag{3.8}$$

$$\begin{aligned} sc_k = & w_{et} \cdot f_{et}(et_k) + w_{pr} \cdot f_{pr}(pr_k) + w_{ee} \cdot f_{ee}(ee_k) + \\ & + w_{ec} \cdot f_{ec}(ec_k) \end{aligned} \quad ,\forall k \tag{3.9}$$

$$sc_k > 0, \forall k \tag{3.10}$$

$$f_{et}(x) = \begin{cases} a_{et} + \frac{ET-x}{ET-q_{et}^{min}} \cdot (1-a_{et}), & q_{et}^{min} \leq x \leq ET \\ \max\left(a_{et} - \frac{q_{et}^{min}-x}{ET-q_{et}^{min}} \cdot (1-a_{et}), 0\right), & x < q_{et}^{min} \\ \max\left(a_{et} - \frac{x-ET}{ET-q_{et}^{min}} \cdot (1-a_{et}), 0\right), & x > ET \end{cases} \tag{3.11}$$

$$f_{pr}(x) = \begin{cases} a_{pr} + \frac{PR-x}{PR-q_{pr}^{min}} \cdot (1 - a_{pr}), & q_{pr}^{min} \leq x \leq PR \\ \max\left(a_{pr} - \frac{q_{pr}^{min}-x}{PR-q_{pr}^{min}} \cdot (1 - a_{pr}), 0\right), & x < q_{pr}^{min} \\ \max\left(a_{pr} - \frac{x-PR}{PR-q_{pr}^{min}} \cdot (1 - a_{pr}), 0\right), & x > PR \end{cases} \tag{3.12}$$

$$f_{ee}(x) = \begin{cases} a_{ee} + \frac{x-EE}{q_{ee}^{max}-EE} \cdot (1 - a_{ee}), & EE \leq x \leq q_{ee}^{max} \\ \max\left(a_{ee} - \frac{EE-x}{q_{ee}^{max}-EE} \cdot (1 - a_{ee}), 0\right), & x < EE \\ \max\left(a_{ee} - \frac{x-q_{ee}^{max}}{q_{ee}^{max}-EE} \cdot (1 - a_{ee}), 0\right), & x > q_{ee}^{max} \end{cases} \tag{3.13}$$

$$f_{ec}(x) = \begin{cases} a_{ec} + \frac{EC-x}{EC-q_{ec}^{min}} \cdot (1 - a_{ec}), & q_{ec}^{min} \leq x \leq EC \\ \max\left(a_{ec} - \frac{q_{ec}^{min}-x}{EC-q_{ec}^{min}} \cdot (1 - a_{ec}), 0\right), & x < q_{ec}^{min} \\ \max\left(a_{ec} - \frac{x-EC}{EC-q_{ec}^{min}} \cdot (1 - a_{ec}), 0\right), & x > EC \end{cases} \tag{3.14}$$

$$\sum_{j \in S_i} z_{ij} = 1 \quad , \qquad \forall i \tag{3.15}$$

$$et_i = \sum_{j \in S_i} z_{ij} \cdot et_j \quad , \qquad \forall i \tag{3.16}$$

$$x_{i_2} - (et_{i_1} + x_{i_1}) \geq 0, \qquad \forall t_{i_1} \quad \rightarrow \quad t_{i_2} \tag{3.17}$$

$$\sum_{i \in sp_m^k} et_i \leq et_k \quad , \qquad \forall k \tag{3.18}$$

$$pr_k = \sum_{i \in A_k} \sum_{j \in S_i} z_{ij} \cdot pr_j \quad , \qquad \forall k \tag{3.19}$$

$$ee_k = \frac{1}{|A_k|} \cdot \sum_{i \in A_k} \sum_{j \in S_i} z_{ij} \cdot ee_j \quad , \qquad \forall k \tag{3.20}$$

$$ec_k = \sum_{i \in A_k} \sum_{j \in S_i} z_{ij} \cdot ec_j \quad , \qquad \forall k \tag{3.21}$$

$$et_{\text{class}(j)}^{min} \leq et_j \leq et_{\text{class}(j)}^{max} \quad , \qquad \forall j \tag{3.22}$$

$$ec_{\text{class}(j)}^{min} \leq ec_j \leq ec_{\text{class}(j)}^{max} \quad , \qquad \forall j \tag{3.23}$$

$$ee_j = \frac{ec_{\text{class}(j)}^{\text{normal}} \cdot \frac{ec_j - ec_{\text{class}(j)}^{\text{idle}} \cdot et_j}{ec_{\text{class}(j)}^{\text{normal}} - ec_{\text{class}(j)}^{\text{idle}}}}{ec_j} \quad , \qquad \forall j \tag{3.24}$$

$$pr_j = \alpha_{\text{class}(j)} \cdot et_j + \beta \cdot ec_j \quad , \qquad \forall j \tag{3.25}$$

$$et^{min} \leq et_x \leq et^{max} \quad , \qquad \forall x = \{j, k, i\} \tag{3.26}$$
$$pr^{min} \leq pr_x \leq pr^{max} \quad , \qquad \forall x = \{j, k\} \tag{3.27}$$
$$ee^{min} \leq ee_x \leq ee^{max} \quad , \qquad \forall x = \{j, k\} \tag{3.28}$$
$$ec^{min} \leq ec_x \leq ec^{max} \quad , \qquad \forall x = \{j, k\} \tag{3.29}$$

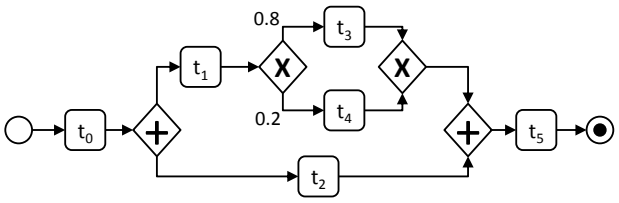
The optimization function (3.8) and the constraints (3.9 and 3.10) have already been explained. The equations from (3.11) to (3.14) define the evaluation functions of the four quality and energy dimension under consideration. Constraint set (3.15) enforces the fact that only one candidate service should be selected for each task. Constraint set (3.16) expresses that the execution time for each task is the execution time of its selected service. Constraint set (3.17) represents precedence constraints for subsequent tasks in the abstract execution plan. To explain, if $t_{i_1} \rightarrow t_{i_2}$, $i_1, i_2 \in I$, then the task t_{i_2} is a direct successor of task t_{i_1} so the execution of the former should start after the termination of the latter. The variable x_i denotes the starting time point of task t_i. Constraint set (3.18) expresses that the execution time of every execution path is obtained by calculating the maximum execution time of all corresponding execution subpaths of this path. Constraint sets (3.19–3.21) express the price, energy efficiency, and energy consumption of every execution path, respectively, based on the aggregation rules highlighted in Table 3.2. Constraints sets (3.22–3.25) express the constraints obtained from the service profile of every candidate service for the four considered dimensions. Finally, constraints sets (3.26–3.29) define those variables of the problem that are related to the considered dimensions and are specific for each service, task and execution path.

Local constraints can be easily added in the above definition of the SC problem as they predicate on properties of a single task. For instance, if the ee for a task t_y has to be greater than or equal to a specific given value v, then the following constraint should be added to the above definition:

$$\sum_{j \in S_y} z_{yj} \cdot ee_j \geq v$$

Based on the above analysis, we have shown that the SC problem for a composite process with a block structure can be mapped to a CSOP. The proposed approach advances the state-of-the-art in QoS-based service composition by: a) taking into account not only single (average or minimum) values of independent quality or energy metric but a range of values represented by indicator thresholds; b) producing a concrete execution plan even if the requirements set by the user are over-constrained through the use indicators warning thresholds; c) allowing for the use

Figure 3.4: Process example



of non-linear functions in the optimization (QoS-based Service Composition) problem to be solved; d) considering dependencies among quality and energy metrics into aggregated functions.

3.3.3 Proof-of-concept example

In this section, we provide a proof-of-concept example that highlights the significance of the proposed approach. Let us consider the process example depicted in Figure 3.4 that consists of six tasks, namely t_0, t_1, t_2, t_3, t_4 and t_5. According to this process, task t_0 runs first, then there is a split where tasks t_1 and t_2 run in parallel. After t_1 is executed, then we have a conditional branch, where t_3 is executed with probability 0.8 or t_4 is executed with probability 0.2. In the end, when either t_3 or t_4 and t_2 are finished, there is a join and the last task, t_5 is executed. Thus, this process has two execution paths, their execution probabilities and subpaths:

$$ep_1 = \{t_0, t_1, t_2, t_3, t_5\}, \begin{cases} sp_1^1 = \{t_0, t_1, t_3, t_5\} \\ sp_1^2 = \{t_0, t_2, t_5\} \end{cases}$$

$$ep_2 = \{t_0, t_1, t_2, t_4, t_5\}, \begin{cases} sp_2^1 = \{t_0, t_1, t_4, t_5\} \\ sp_2^2 = \{t_0, t_2, t_5\} \end{cases}$$

where the probability P of execution of sp_1^1 equal 0.8 and sp_2^1 equal 0.2. Moreover, $sp_1^2 = sp_2^2$ with probability equal 1.

Moreover, for the sake of simplicity, we assume that there are three services that can be used to execute any of the six tasks, where service s_1 runs in the slow server class, s_2 runs in the average server class, and s_3 runs in the fast server class. For each service we assume that we can derive the information in Table 3.4 from their profiles. In addition, we assume the following information: $\alpha_{\text{class}(1)} = 10, \alpha_{\text{class}(2)} = 50, \alpha_{\text{class}(3)} = 250, \beta = 0.5$, so the cost models of the services will be: $pr_1 = 10 * et_1 +$

$0.5 * ec_1$, $pr_2 = 50 * et_2 + 0.5 * ec_2$, and $pr_3 = 100 * et_1 + 0.5 * ec_3$, respectively.

	Exec.Time	Energy Consumption	Efficiency
s_1	$7 \leq et_1 \leq 10$	$1275.4 \leq ec_1 \leq 2088$	$ee_1 = \left[\frac{222 \cdot \frac{ec_1 - 178 \cdot et_1}{222-178}}{ec_1} \cdot 100\right]$
s_2	$4 \leq et_2 \leq 7$	$1992.4 \leq ec_2 \leq 3994.9$	$ee_2 = \left[\frac{607 \cdot \frac{ec_2 - 486 \cdot et_2}{607-486}}{ec_2} \cdot 100\right]$
s_3	$1 \leq et_3 \leq 4$	$6647.1 \leq ec_3 \leq 30478.8$	$ee_3 = \left[\frac{8106 \cdot \frac{ec_3 - 6485 \cdot et_3}{8106-6485}}{ec_3} \cdot 100\right]$

Table 3.4: Services obtained profiles

The last assumptions made in this example concern the value domain of the quality and energy variables, the normalization functions and their weights, and the user constraints. Concerning the variables, we assume that all execution time variables et_x have the domain $[1, 10]$, all price variables pr_x have the domain $[500, 20000]$, all energy efficiency variables ee_x have the domain $[0, 100]$, and all energy consumption variables ec_x have the domain $[1000, 31000]$. Moreover, we assume that all dimensions are equally important and should be evaluated in the same way, so we have that: $a_{et} = a_{pr} = a_{ee} = a_{ec} = 0.4, w_{et} = w_{pr} = w_{ee} = w_{ec} = 0.25$. Finally, the user provides the following constraints: $et^{max} = 27, pr^{max} = 2400, ee^{min} = 0.55, ec^{max} = 6000$.

Based on the user-supplied information, it is easy to see that the problem is over-constrained, so all of the current approaches would fail and not return any solution. However, our approach does not fail and produces the following solution $z_{0,1} = z_{1,1} = z_{2,1} = z_{3,1} = z_{4,1} = z_{5,1} = 1$, which has the highest score (0.2825) and violates in the least possible way the user constraints. In other words, all tasks of the process have been assigned to the first service, which is the cheapest and less energy consuming. Based on this solution, both execution paths will have the following values for their aggregated dimensions: $et_j = 28, pr_j = 3770, ee_j = 0.45, ec_j = 6840$.

3.4 Indicator aggregation through composed weighting system

Although the previous section presented aggregated quality and energy metrics such as response time and energy consumption, the aggregation is limited in two aspects: i) it comprehends only the application layer;

and ii) the aggregation function does not combine different metrics together. In this way, a more general aggregation approach has to be used in order to represent meaningful information derived from the combination of different indicators at different architecture layers. Even if related indicators, such as VM power and application energy consumption, are used in a composite way in order to simplify the calculation (the output of one is used as the input of the other), as a general rule, they usually cannot be directly compared nor aggregated. This is because they may represent different system layers, have heterogeneous scales and opposite monotonicity. The following approach proposes to make indicators comparable through normalization functions such that they can be compared and aggregated within meaningful clusters named as Green Index Functions (GIFs).

3.4.1 Proposed indicators aggregation system

The issue of indicators aggregation is not only related to Green IT. In many research fields, the information provided by a set of indicators have to be aggregated in order to provide a more general index or make different indicators comparable. In [115] we introduce an indicator aggregation system that was inspired in the ideas presented by [138], in which the authors measure sustainability impact over a land use scenario. The approach is composed of two phases. The first phase, called *indicator normalization*, normalizes the heterogeneous indicators values within [0,1] range based on the indicator thresholds and monotonicity. In this phase, indicators are independent from each other. The second phase, called *aggregation metric*, puts the indicators together as a single value that represents a specific group of indicators, i.e., GIF. As indicators may have different relevance within the GIF, weights are used to represent such differences. The next subsections explain these phases in more details.

Indicator normalization

In order to be able to apply aggregation metrics, the considered set of indicators values has to be normalized within a common numerical scale that abstract the diverse indicators scales (e.g., watt, percentage, and seconds). Most of normalization functions are based on max and min boundaries [138, 184], however we want to represent both warning and alarming thresholds. For that reason we have chosen a normalization function that allows the use of soft constraints [97, 114], where values between warning and alarming thresholds are also considered. Depend-

ing on the indicator monotonicity, equations 3.30 and 3.31 represent the decreasing and increasing dimensions respectively. In case of non-monotonic indicators, such as CPU payload, we split them in order to create two monotonic ones.

$$f_i(x) = \begin{cases} \frac{a_i^{min}}{w_i^{min}} + \frac{w_i^{max}-x}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{a_i^{min}}{w_i^{min}}\right), & w_i^{min} \leq x \leq w_i^{max} \\ \max\left(\frac{a_i^{min}}{w_i^{min}} - \frac{w_i^{min}-x}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{a_i^{min}}{w_i^{min}}\right), 0\right), & x < w_i^{min} \\ \max\left(\frac{a_i^{min}}{w_i^{min}} - \frac{x-w_i^{max}}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{a_i^{min}}{w_i^{min}}\right), 0\right), & x > w_i^{max} \\ 0, & x < a_i^{min} \end{cases} \tag{3.30}$$

$$f_i'(x) = \begin{cases} \frac{w_i^{max}}{a_i^{max}} + \frac{x-w_i^{min}}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{w_i^{max}}{a_i^{max}}\right), & w_i^{min} \leq x \leq w_i^{max} \\ \max\left(\frac{w_i^{max}}{a_i^{max}} - \frac{w_i^{min}-x}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{w_i^{max}}{a_i^{max}}\right), 0\right), & x < w_i^{min} \\ \max\left(\frac{w_i^{max}}{a_i^{max}} - \frac{x-w_i^{max}}{w_i^{max}-w_i^{min}} \cdot \left(1 - \frac{w_i^{max}}{a_i^{max}}\right), 0\right), & x > w_i^{max} \\ 0, & x > a_i^{max} \end{cases} \tag{3.31}$$

where x is the value to be normalized, w_i^{min} and w_i^{max} are min and max warning thresholds and a_i^{min} and a_i^{max} are min and max alarming thresholds of indicator i.

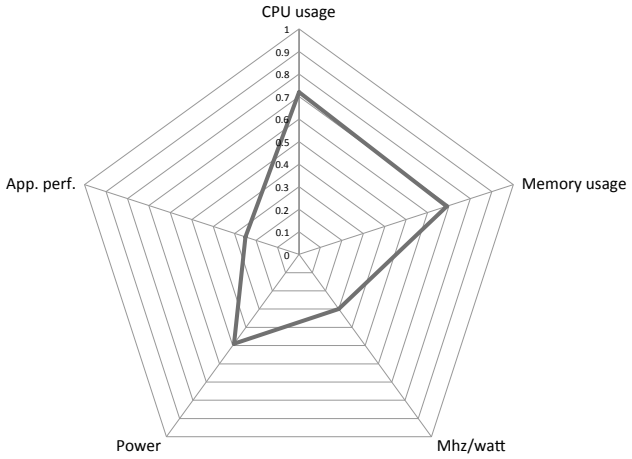
Let us consider, for instance, the indicator `App. Performance` AP, which represents the number of transactions per second divided by power ($\frac{TPS}{Power}$), with warning thresholds $w_{ap} = [1000; 1200]$ and alarming thresholds $a_{ap} = [600, 1000)$. If $x \in [600; 800)$ it gets a normalized value in $[0.1; 0.25)$; otherwise if $x \in [800; 1000)$ it gets a normalized value in $[0.25; 0.4)$; and finally, if $x \in [1000; 1200]$ it gets a normalized value in $[0.4; 0.55]$.

Values between the range $[0.4; 1.0)$ identifies indicator i as green, values between the range $(0.0; 0.4)$ identifies as yellow and, eventually, red if the normalized value is equal to zero.

Figure 3.5 depicts five normalized indicators measured in GAMES project that are heterogeneous in nature. All of them are regarding the infrastructure architectural layer and operational concept layer. The indicators represent the average value of 24 hours measurement with 1,000 of tuples of each indicator (except power consumption, which is respect to one month and 41,580 tuples). The indicators details about GAMES project testbed under consideration are presented in Chapter 7.

The indicator `CPU usage` provides the CPU load and it is measured by dividing the amount of CPU used by the amount of CPU allocated. In the same way, `Memory usage` is the ratio of the amount of used memory by the total memory. The `Mhz/watt` indicates how effectively power

Figure 3.5: Example of normalized indicators from GAMES project



is being consumed by the processor. For both usage indicators, warning and alarming thresholds are defined as 60 and 15 respectively. The thresholds of `Mhz/watt` is defined as $w^{min} = 65$ and $a^{min} = 25$. `Power consumption` indicator is the power consumed by the server cluster such that $w^{max} = 400$ and $a^{max} = 500$. Finally, `App. performance` has $w^{min} = 1000$ and $a^{min} = 600$.

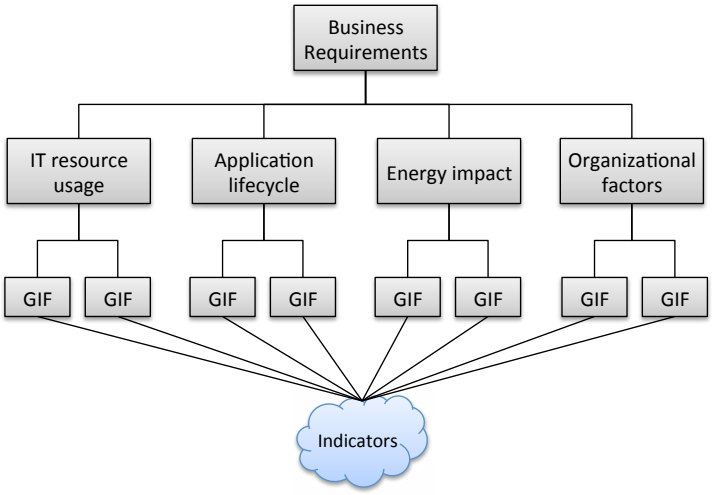
Aggregation metric

Using the indicators normalized values, we argue that the indicators have to be aggregated in order to provide meaningful information indexes. To do so, indicators are aggregated in GIFs. The GIF is defined as a set of inter-related indicators (by nature, architectural or conceptual layers) that, when aggregated, provide relevant information about one aspect of the system or component. We adopt weighted sum technique, in which a specific value (weight) is attributed to each normalized indicator within the aggregation function. The definition of such weight is usually considered out of scope or a user input issue. However, in this approach we identify the indicator weight based on risk management analysis as follows.

The first step is to identify possible GIFs and what indicators can be part of them. The aim of a GIF is to aggregate related indicators in order to provide a more general index that represents one or many aspects towards the system goals, including of course, system greenness. For instance, the GIF `server usage` aggregates the indicators `CPU usage` and `memory usage`. Figure 3.6 shows four main GIFs defined by [93] which

are used as umbrella for user defined ones, named specialized GIFs. The specialized GIF `server usage`, for instance, is placed under the main GIF `IT resource usage`. Specialized GIFs aggregate both GPIs and KPIs where some of them might be part of one or more GIFs depending on the indicator scope. The indicator scope identification is based on the attributed described in Figure 3.2.

Figure 3.6: Indicators hierarchy



Considering the presented indicators attributes, the goal is to aggregate them within the following four main GIFs:

- IT resource usage: It measures how the resources are being consumed in terms of utilization. This GIF is related to effectiveness once the proper utilization of the resource (e.g., CPU, memory, and disk) reduces overall wastes in terms of energetic and financial terms.

- Application lifecycle: It measures the needed effort to design, execute and maintain the application within the data center. It includes well-known quality metrics, such as availability and reliability, but also software engineering metrics that computes from the analysis requirements phase until the software decommissioning.

- Energy impact: It represents energy directly-related indicators such as power, energy, Mhz/watt, and application performance. It is closely related to IT resource usage GIF once several indicators from both GIFs are the same or have narrow relationships.

- Organizational factors: It represents higher level indicators such as governmental compliance, supply chain outsourcing and human resource involvement during the data center maintenance.

Knowing what are the indicators that can be aggregated within each main GIF, specialized GIFs can be created by the users. When a specialized GIF is created, the user selects what indicators will be part of it among the available indicators for the main GIF. For example, the indicator `CPU usage` is selected for the specialized GIF `server usage`, which is under the main GIF `IT resource usage`. The next step is to calculate the weight of each indicator within the aggregation metric. The aim of weighting is to compensate each indicator's relevance within the GIF, as some indicators may appear in several GIFs with different importance.

We propose a weighting system that is composed by three weights. The first weight is regarding the total number of indicators within one GIF, such that $w_1 = \frac{1}{N_k}$, where N_k is the total number of indicators within GIF k. The second weight w_2 is regarding to the relevance of the indicator within the GIF based on the running scenario. For example, in a high performance scenario, the indicator `CPU usage` has higher relevance over the indicator `memory usage` although both of them are part of the specialized GIF `server usage`. This relevance is defined by the user during the GIF indicators selection phase and varies within the range [0,1]. Finally, the third weight w_3 represents the negative and positive impacts of the indicator violation and fulfillment respectively towards the GIF. Such information is mapped as an impact index. Indicators with higher degree to damage the system get higher weights such that they have bigger influence over the GIF. For instance, considering the GIF `server energy consumption`, indicators related to the CPU power consumption may have higher importance than I/O once CPU related events represent higher risk for server energy consumption.

In order to proper represent such impact index according to the administrator preferences, the user has to select one of three desired levels: i) *avoidance* does not allow warning nor alarming indicators violation; ii) *reduction* does not allow alarming violation and do allow warning; and iii) *retention* allows both warning and alarming indicators violation. Such preferences reinforce or weaken the indicator impact index within a certain GIF. The final weight of an indicator within the GIF is composed by the mean value of each described weigh.

3.5 Summary

This chapter describes how BP can be measured with respect to their static information and how quality and energy dimensions can be used within service compositions. Based on some indicators attributes, in special the different thresholds, the proposed approach manages to deal with over constrained situations. In the first part of this chapter we propose some metrics in order to extract the BP main characteristics in a quantitative manner. This information is therefore used to compose the BP profile, which supports the calculation of indicators like energy consumption.

Taking into consideration both quality and energy aspects of the SBA is the main goal of the service composition problem formulated in the second part. The goal is to find out the best trade-off between them in order to solve a service composition problem even during over constrained scenarios. Hence, a new energy efficiency indicator for a single service is introduced, which maps directly the relationship between energy consumption and execution time. The energy dimension requires considering novel aspects for service quality evaluation. In particular, the proposed method considers soft constraints, nonlinear relationships among quality dimensions, and ranges for quality values.

A more general approach is described in the last part of the chapter with an indicator aggregation system that is based on composed weighting system. It supports indicators interrelationships when they are heterogeneous with respect to their architectural layer. The presented aggregation aims to make indicators comparable through normalization functions and to properly express the indicator relevance within aggregated values. Thus, the normalization deals with all four indicators thresholds and the aggregation calculates the indicator violation impact from the system perspective.

4 Energy-aware adaptation of co-designed business processes

As previously discussed, SOA has been proposed to realize systems that are flexible compositions of services created to facilitate reuse of components, service sharing and, therefore, to reduce costs. In highly dynamic environments, both from IT and business perspectives, requirements for such systems are rapidly changing and interactions among enterprises are reactive to variable contexts. Considering such scenario, adaptivity becomes a major asset that sets the basis for developing adaptive systems able to react to variable operation conditions [90].

The design of energy efficient ISs can therefore leverage on some characteristics of the service-oriented approach and its flexibility and adaptivity features. The approach can be applied at different levels and phases in IS development and maintenance, from a more strategic view to a technological support of adaptive approaches, introducing a new focus on Green IT/IS during the system design. According to [174], green IT focuses on equipment utilization and energy efficiency while green IS refers to the design and implementation of sustainable business processes. In our view, the monitoring, analysis, and control of green IT equipment have to be supported towards the development of a green IS control sys-

tem, which shall provide filtering and decision-making mechanisms to achieve sustainable business goals. In order to build energy-aware ISs, the following two characteristics are comprehended in this thesis:

1. *Co-design*: The energy-aware IS processes and their underlying services and the IT infrastructure have to be *co-designed* in order to trade-off user business, functional, and quality requirements against energy related constraints, such as consumption and efficiency. The central role of expressing and assessing this trade-off is to support decision-making mechanisms that are based on risk and impact propagation analysis.

2. *Runtime Adaptation*: Energy efficiency has to be managed at runtime by exploiting the adaptive behavior of the system, considering green IT/IS perspectives and interactions of these two aspects in an overall unifying vision.

Considering the indicators described in the previous chapter, this chapter deals with energy-aware adaptation of co-designed BPs in order to recover (or avoid) indicators (KPIs and GPIs) violation. More specifically, the goal of this chapter is to propose an approach for designing *Energy-aware Business Processes* (E-BP) extending the typical Business Process (BP) conceptual model, which aims to capture the energy consumption of the involved process abstract tasks. Energy consumption is constantly monitored by using specific indicators (GPIs), which have to be satisfied together with the more traditional functional and quality (KPI) requirements. By making use of *energy-aware adaptation*, the E-BP is able to enact specific *strategies* in order to modify its execution or structure in case energy consumption needs to be lowered or energy inefficiencies are identified.

4.1 A Layered approach for Energy-Aware Information Systems

Figure 4.1 presents the three main layers of an IS considered in this thesis: *infrastructure layer*, *middleware layer*, and *application layer*. The infrastructure layer includes all physical energy-hungry equipment. As we focus on servers, we consider the processor as the main energy consumer component, which is passive of runtime adaptation like frequency change. This frequency, measured in Hertz, specifies the internal core operation frequency where the higher it is, the better performance. As higher frequencies consume more power than low frequencies, modern processors

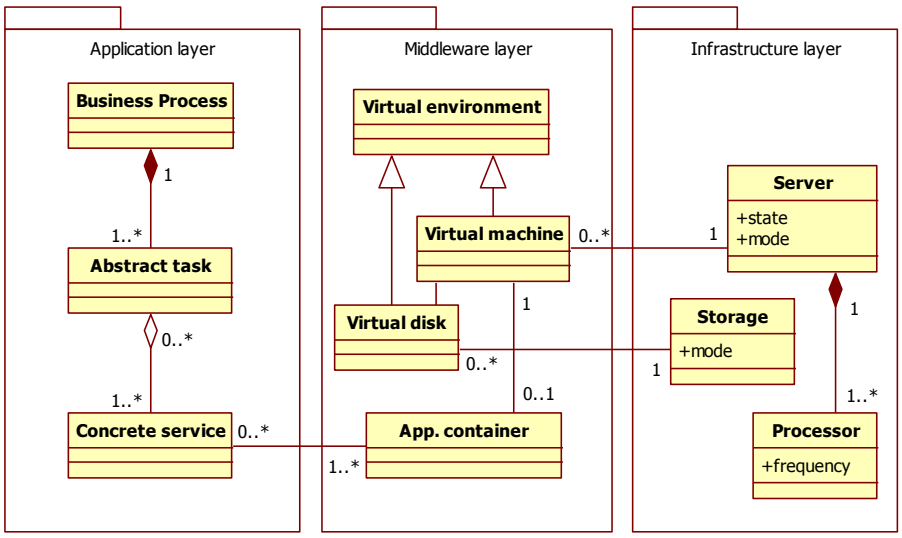


Figure 4.1: Three-layered IS model

are able to dynamically scale their operating frequencies through some defined P-states (performance states).

In addition, the server attributes `state` and `mode` represent server on/off and hibernation modes. We also consider the energy consumption and running modes (normal or acoustic) of the storage system. In this model we do not consider networking since the focus is on the relationships between applications and their used local resources. Thus, servers and storage systems represent the available IT resources for the middleware layer, in particular, one server can hold several virtual machines. The virtual machine is one type of virtual environment, which can also represent virtual disks for the storage system.

The middleware layer manages virtual resources reservation, workload distribution, and resource monitoring. As shown in the figure, the middleware layer comprises the virtual environment (i.e., VMs and virtual disks) and the application container, in which concrete services are deployed. Due to management issues, one VM holds at maximum one application container. The calculation of power consumption at the middleware layer is made through power models. For instance, [88, 30] demonstrate how VM power consumption values can be estimated based on their physical resources usage.

The application layer involves the concrete services, abstract tasks and business processes. Concrete services (usually represented by web-services in SOA) represent the implementation of small operations that,

when coordinated, perform the activity described by one or more abstract tasks. Abstract tasks represent the business process objectives described as actions. A business process "consists of a set of activities that are performed in coordination in an organizational and technical environment" [177], where activities are represented by abstract tasks, user tasks and routing tasks. The calculation of power consumption at this layer follows a model-based approach [63, 65].

As our aim is to focus on the application layer, we split our analysis into two different relations depicted in Figure 4.1. The first relation, a composition between business process and abstract task (BP–AT), deals with the characteristics of the BP, such as individual requirements and previous/next flow pattern. The second relation, an aggregation between abstract task and concrete service (AT–CS), deals with available implementations characteristics, in particular, their published non-functional dimensions such as response time and costs. More details about both relations are presented in the following:

BP–AT: This composition relation represents the many abstract tasks that compose a business process and how they are organized as a workflow. Each abstract task is defined by both functional and non-functional requirements. Focusing on non-functional ones, they are expressed as constraints on quality and energy dimensions that are expected to be satisfied during the BP execution, i.e., indicators thresholds defined in Chapter 3. For instance, the business process BP_n has maximum response time threshold ($^{max}RT_{BP_n}$), which is measured in seconds, while every single abstract task $at_{n,m}$ of BP_n also has maximum response time threshold ($^{max}rt_{at_{n,m}}$) for all existing execution path in the process, such as:

$$^{max}RT_{BP_n} \geq \sum_{m=1} {}^{max}rt_{at_{n,m}} \tag{4.1}$$

Dealing with energy aspects, in [41] we introduce the service-based Energy-aware BP (E-BP), in which the abstract tasks are defined not only in terms of their quality and energy characteristics, but also with specific information to guide application energy assessment (like minimal resources requirements) and task elasticity for adaptation (like optimal tasks flags). All this information is annotated as metadata properties, which is provided by GAMES methodology described in Chapter 7. The considered metadata are:

- Flow: provides information regarding the BP routing tasks and the composing tasks.

- Resource: contains the minimal and optimal resource information about needed resources to execute single or composed BP tasks.
- Data: provides information with respect to data (such as variable type) exchanged through the BP.
- Constraints: includes quality and energy constraints with respect to single or composed BP tasks, which are represented as indicators thresholds.

The result of this phase is an E-BP annotated with all the information about the execution and the deployment.

BP–CS: This aggregation relation represents the next step of the application co-design phase, in which information about available services (concrete services) is gathered in order to satisfy the requirements described in the previous relation. Considering the service concretization problem described in the previous chapter (Chapter 3 Section 3.3), we assume to have a list of candidate services that are functionally equivalent, but differ regarding non-functional characteristics. Each candidate service of the designed-to-be SBA runs in a specific application container that has specific quality and energy capabilities. Based on that, some KPIs and GPIs are pre-calculated, i.e., estimated based on the information published by each candidate service. This information composes the service profile, which guides the proposed CSOP, also described in the previous chapter, to select the best concrete service candidate.

In order to demonstrate how the 3-layers model fits with SOA applications, Figure 4.2 shows a BP example. The example represents a system for publishing online news, which involves actors like editor and photographer. The system is described as: as soon as an event is indicated to be covered, the system has to assign a correspondent and a photographer (represented by the abstract task at_1). After have covered the event, both of them shall upload the material with respect to the event, i.e., the article text and photos (at_2 and at_4). The text is automatically reviewed by the system (at_3). When all the material is available within the system, the editor approves or rejects it (at_5). In case of rejection, the editor has to indicate the required changes (at_6). It is assumed that the probability of an article be rejected is 30% and it can happen only once. The process ends when the approved article is published online through the company's website (at_7) and both correspondent and photographer get paid (at_8). The figure clearly separates the layers and their elements after solving the SC problem. For instance, the abstract

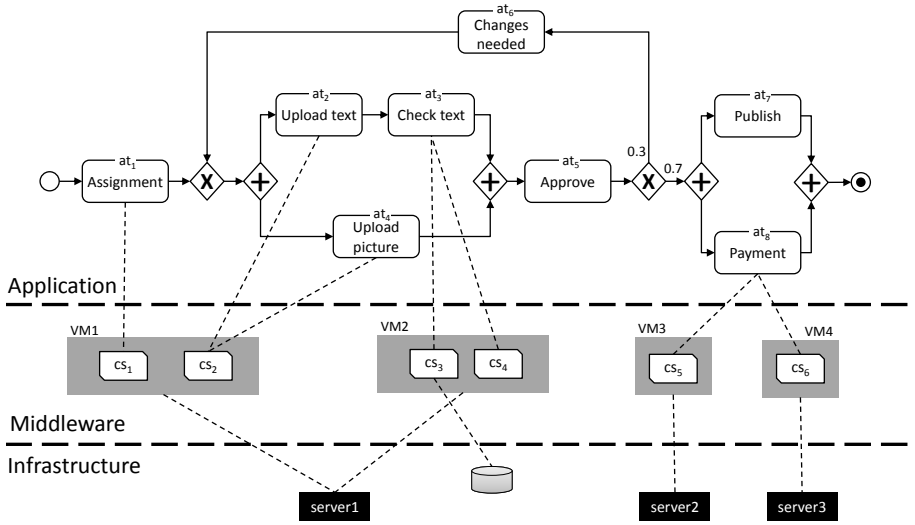


Figure 4.2: Online News Publishing System - example scenario

task at_3 is performed by the composition of two concrete services, cs_3 and cs_4. These services are deployed on VM2 (which also represents the application container) and running on server $server1$.

4.2 Business process energy estimation

The information gathered at the application layer, in particular the relation between BP and abstract task, represents both quality and energy aspects of the BP. In order to cope with the latest, in [41] we have presented an approach to estimate the BP energy consumption based on the 3-layers model previous described. Estimating the application power consumption is important to define some GPIs thresholds and their index function within the GIFs normalization and aggregation phases. It starts from the infrastructure layer with measured power consumption P of individual servers $sv_i \in SV$, where SV is the available servers. At the middleware layer, power models are used to extract the power consumption of the virtual machine $vm_{i,j}$, which is running on top of server sv_i with configuration parameters C, such that $C_{vm_{i,j}}$. The configuration parameters represented by C involves elements such as number of processors allocated to $vm_{i,j}$ and the processor frequency. As we are focusing on servers, the considered configuration parameters are the number of processor cores allocated to the VM and the VM manager (hypervisor) type: native or hosted. Thus, the power consumption of $\{vm_{i,j}\}$ can be defined as:

$$P_{vm_{i,j}}(t) = f\big(C_{vm_{i,j}}, P_{sv_i}(t)\big) \tag{4.2}$$

where the VM power consumption depends on the allocated resources $C_{vm_{i,j}}$ and the server power consumption $\{P_{sv_i}(t)\}$ at time t.

Having $P_{vm_{i,j}}(t)$, the next step is to estimate the power consumption of the deployed services, i.e., concrete services. With respect to Figure 4.1, we are not interested in the application container power consumption as each VM contains only one container. However, one application container can have many concrete services $cs_{i,j,k}$. Similarly to the calculation of $P_{vm_{i,j}}(t)$, the estimation of a concrete service power consumption is model-based. Barlatos et. al [21] argue that it is influenced by three factors: i) *nature of the I/O*, which considers the input and output data type of the service and its internal computational complexity; ii) *server configuration*, which in our approach is represented by the virtual machine allocated resources $C_{vm_{i,j}}$; and iii) *web-service design*, which is based on internal logic structures and technologies used during the implementation phase. As described in the previous section, this information is available within the service profile. Considering that, the calculation is simplified by the following model:

$$P_{cs_{i,j,k}}(t) = f(P_{vm_{i,j}}(t), \Phi_{cs_{i,j,k}})) \tag{4.3}$$

where $\Phi_{cs_{i,j,k}}$ represents the service profile. It is important to point that, for the sake of simplicity, we assume hereafter to have only one E-BP, with several instances, on the servers.

The relationship between the deployment configuration and the power consumption helps the designer to realize which is the maximum energy consumption of E-BP. To this aim, the designer needs to know for each task $at_{n,m} \in \text{E-BP}_n$ what is the set of concrete services $\{cs_{i,j,k}\}$ that are expected to perform it. Note that, due to dynamic binding mechanisms decision-making process (which are based on parameters like availability) a different set of concrete services may be selected to execute an abstract task at runtime. In case of deviation of the expected power consumption (defined at design-time) and obtained one at runtime, adaptation actions are enacted. These actions are guided by the application energy assessment and tasks elasticity, previously defined. Thus, the abstract task $at_{n,m}$ power consumption is calculated as following:

$$P_{at_{n,m}}(t) = \sum_{k=1}^{K} P_{cs_{i,j,k}}(t) \tag{4.4}$$

where K is the number of concrete services cs selected to implement the abstract task $at_{n,m}$. Moreover, we consider that a concrete service

represents the atomic implementation of the SBA and, therefore, it may implement more than one task (such as cs_2 in Figure 4.2). As they constitute different invocations of the same service, Eq. 4.4 holds.

Having the power consumption of all abstract tasks that compose the BP, the next step is to calculate the expected energy consumption of the entire BP, which is the energy consumption of all abstracts tasks. Considering that energy is defined as the area of the power graph over time, the energy consumption of task $at_{n,m}$ can be estimated as:

$$\overline{E}_{at_{n,m}} = \int_{t^0}^{t^0+rt_{at_{n,m}}} P_{at_{n,m}}(t)dt \tag{4.5}$$

where t^0 is the task starting time.

The estimation of the entire BP is done based on the aggregation function described in Chapter 3 Table 3.2, in which the entire BP energy consumed is estimated by summing up the abstract tasks energy consumption:

$$\overline{E}_{BP_n} = \sum \overline{E}_{at_{n,m}} \tag{4.6}$$

However, we are not only interested in the total energy consumption of the BP, but the partial consumption of the orchestration in order to enable the identification of energy leakage (i.e., inefficiencies). This is done using the tasks reduction rules proposed by Cardoso et al. [45] through steps, in which these rules are based on flow patterns: sequence, parallel, conditional, simple loop and dual loop. Considering the BP depicted in Figure 4.2, which is composed of eight abstract tasks $[at_1 \dots at_8]$, the approach starts from the most internal ones and ends when the BP is represented as one single aggregated task. Each step represents a partial orchestration aggregation and is depicted as grey square. This is demonstrated in Figure 4.3, where 6 steps are used to reduce the 8-tasks BP into one aggregated task. The quality aggregation functions are applied at each step based on the aggregated tasks flow pattern. Considering the task response time, we have:

$$step2 : at_{23} = \{at_2, at_3\} \quad \therefore rt_{at_{23}} = rt_{at_2} + rt_{at_3} \quad \text{(sequence)}$$
$$step3 : at_{34} = \{at_{23}, at_4\} \quad \therefore rt_{at_{34}} = max_{\{at_{23},at_4\}}\{rt_{at_{23}}, rt_{at_4}\} \quad \text{(parallel)}$$
$$step3 : at_{78} = \{at_7, at_8\} \quad \therefore rt_{at_{78}} = max_{\{at_7,at_8\}}\{rt_{at_7}, rt_{at_8}\} \quad \text{(parallel)}$$
$$step4 : at_{45} = \{at_{34}, at_5\} \quad \therefore rt_{at_{45}} = rt_{at_{34}} + rt_{at_5} \quad \text{(sequence)}$$
$$step5 : at_{56} = \{at_{45}, at_6\} \quad \therefore rt_{at_{56}} = \frac{rt_{at_{45}} + rt_{at_6} - (1 - p_{at_6}) \cdot rt_{at_6}}{1 - p_{at_6}} \quad \text{(dual loop)}$$
$$step6 : at_{18} = \{at_1, at_{56}, at_{78}\} \quad \therefore rt_{at_{18}} = rt_{at_1} + rt_{at_{56}} + rt_{at_{78}} \quad \text{(sequence)}$$

where p_{at_6} is the probability of execution of the branch at_6, i.e., 30%. In the last step, the execution time of the aggregated task at_{18} represents the execution time of the entire BP. However, it is possible to distinguish the response time of the partial orchestration $\{at_2, at_3, at_4\}$, which involves sequential and parallel patterns.

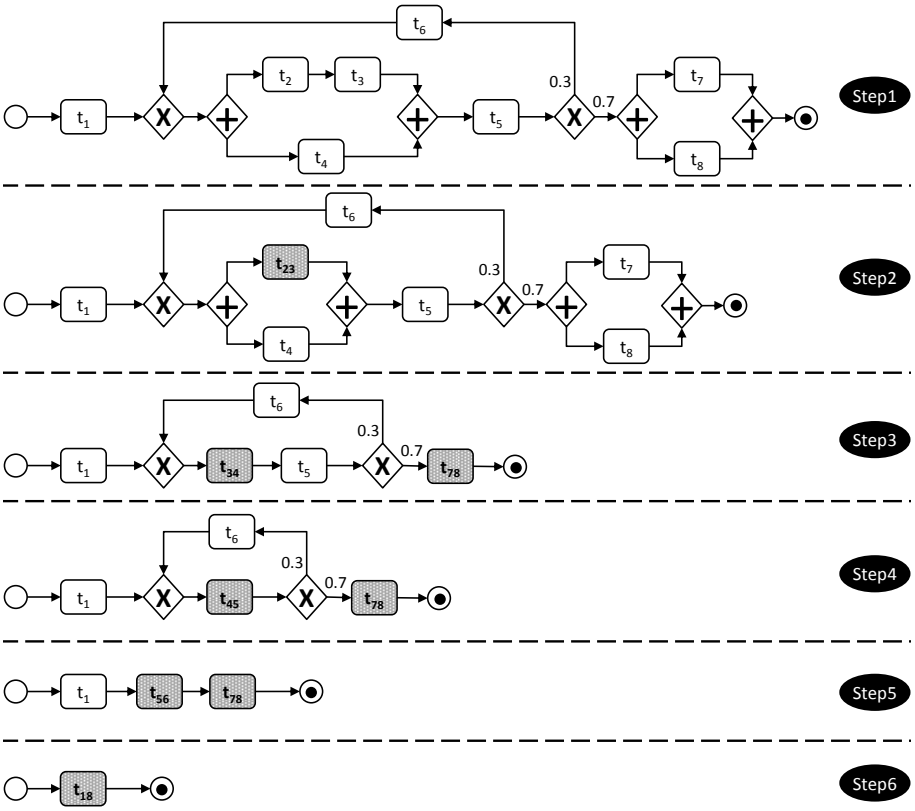


Figure 4.3: BP reduction steps

4.3 Identifying BP energy leakage

The real power consumption of an application can be obtained by mining the data gathered from sensors. According to the power consumed by each device and the resources assigned to the components at the different layers and their usage, we can obtain the real energy consumption $E_{BP}(T)$. The evaluation of the selected indicators is based on the application data saved in log files.

The estimated values of power consumption at runtime allow the designer to identify energy inefficiencies i.e., energy leakage. *Energy leakage*

represents the resources that are not being used in the best possible way and is defined as the difference between the actual energy consumption $E_{BP}(T)$ related to the process execution and the estimated one $\overline{E}_{BP}(T)$. For example, the case in which $E_{BP}(T) < \overline{E}_{BP}(T)$ it might (i) be a sign that a source is not working properly and thus wasting energy or (ii) reveal an error in the resource reservation process. In the former case, the source of this leakage has to be identified between resources allocated to the process. The latter case occurs when, for example, during the process execution all the instances consume less energy than expected and indicators are fulfilled. In this situation, it is possible to state that the amount of reserved resources is overestimated.

Figure 4.4 shows an example of energy leakage based on the BP depicted in Figure 4.2, in which the power consumption of some tasks is less than expected and, on the other hand, the BP takes longer to end. The figure represents the BP execution when at_6 is not performed, i.e., the article is approved by the editor at first time. The most difficult issue is the definition of (i) the task responsible of the leakage and/or (ii) the inefficient resource (e.g., processor or storage inefficiency).

The identification of the responsible abstract task and its concrete service(s) is not trivial since the duration of the tasks might be shorter or longer than expected. Moreover, the active task in a specific time instant could be different from the task expected in the execution plan. In fact, changes in the process execution can affect the estimated execution time of the different tasks. In the figure, the execution of the aggregated task at_{34}, which partially represents the BP orchestration $(at_2 \wedge at_3) \oplus at_4$, impacts significantly on the entire BP execution time. Following the service concretization suggested in Figure 4.2, they can be represented as $(cs_2 \wedge (cs_3 \wedge cs_4)) \oplus cs_2$, where cs_2 is invoked twice (by abstract tasks at_2 and at_4) while the services cs_3 and cs_4 are sequentially executed by task at_3. Let us consider that during the execution of at_3, more specifically service cs_3 that is responsible to interact with a large lexical database, the disk controller decides, for internal reason, to slow down the disk access (highlighted by a circle in the detailed graph on the bottom right of Figure 4.4). This action reduces the service power consumption P, but may increase the overall response time RT.

In fact, as the calculation of energy consumption is based on both power and execution time, switching the disk to quiet mode is not always associated with energy consumption reduction since it decreases the power, but may increase the execution time. Checking, at a given time, if the estimated power is lower than the actual one, at_5 should be blamed for the leakage instead of at_{34}, i.e., the set $\{at_2, at_3, at_4\}$. To solve this situation, both curves have to refer to the same time line by

tightening or relaxing one of them. Anyway, it is possible to say that the set $\{at_2, at_3, at_4\}$ is consuming more than expected and thus it could be the source of the problem. The service concretization of this set of tasks is detailed in the lower right corner of the figure where service cs_3 is pointed as the source of problem due to its substantial power reduction and time increase. In addition, it is noticed that at_1 is consuming less power than expected and thus revealing an overestimation problem. Notwithstanding this divergence with the expected energy consumption, the response time of the service is still satisfied. In both cases, energy inefficiency is detected and changes can be applied in order to improve it.

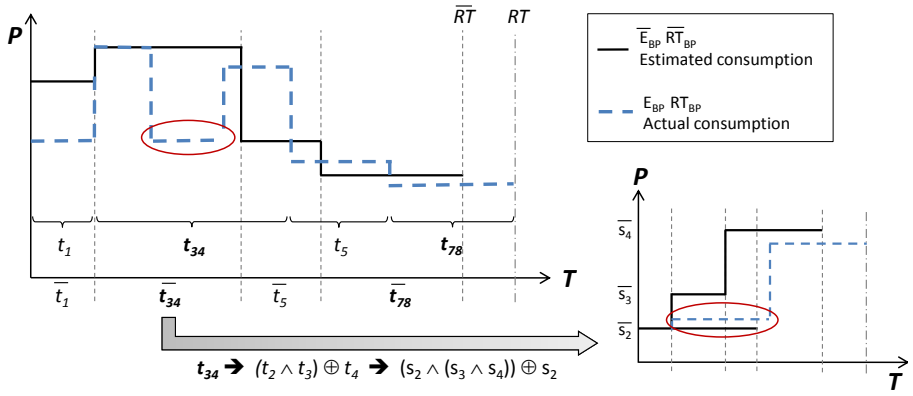


Figure 4.4: Power and response time estimation vs consumption (The area represents the energy)

4.4 Business process co-design and adaptation to reduce energy consumption

Producing optimized configurations at design-time provides static solutions of problems faced when running SBAs in an energy-aware IS. The highly dynamic environment under which these processes are running imposes new challenges to their engineering and provisioning. Thus, self-adaptation is an important feature in order to overtake undesired conditions at runtime, which differ from those assumed during their initial design. Adaptability can be achieved by equipping the IS with self-managing mechanisms that enable it to detect or even predict system context changes and to react on them with adaptation actions that compensate for deviations in functionality, quality or energy of the running business processes.

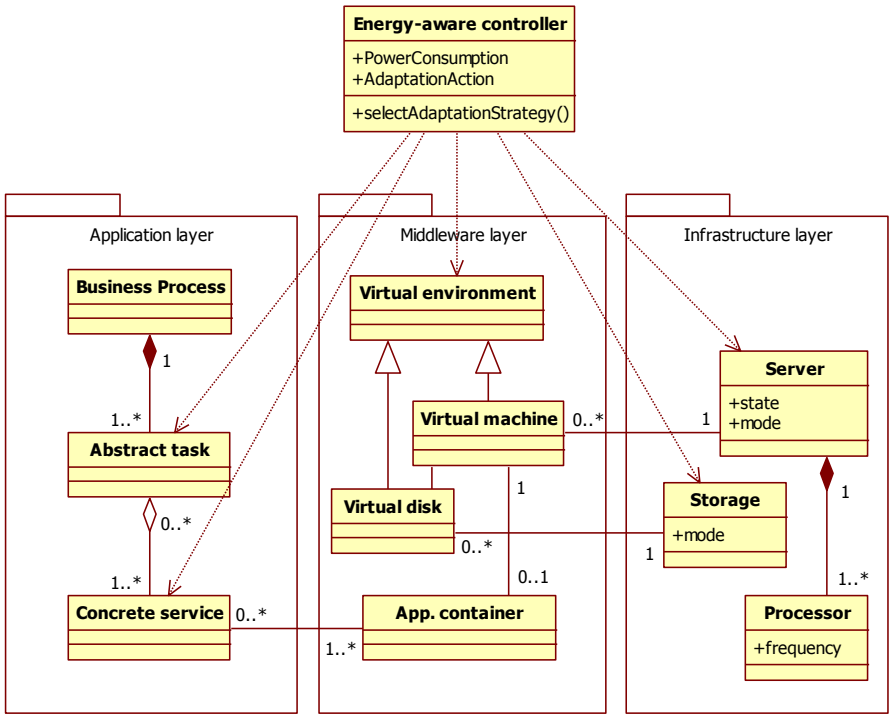


Figure 4.5: The energy-aware controller over the 3-layered IS model

As described in Chapter 2, adaptation controllers are runtime components and have concentrated on minimizing costs, including energy costs, to satisfy applications computing and communication requirements. However, we argue that adaptation actions enacted at runtime should be aligned with the BP design characteristics. In the proposed approach, this consideration is of foremost importance as decisions made at the application layer directly influence the decisions at middleware and infrastructure layers and vice-versa. In order to establish such level of dependency, Figure 4.5 introduces the `energy-aware controller`, which is a centralized approach to determine the most convenient configuration at different layers with respect to optimization of quality and energy parameters. The proposed controller is inspired by the strategy publishing pattern [188], where it gets inputs from local layers elements (through dependency relationship) and produces cross-layers outputs. In this way, at the application layer, concrete services will be deployed among virtualized resources by trading-off quality and energy requirements and, at the middleware layer, workload will be distributed among servers by trading-off service execution quality and energy constraints.

The figure shows that the energy-aware controller relates all the layers, in which both BP and IS characteristics are considered to create a co-designed E-BP. An E-BP is *adaptive with respect to energy consumption* when the amount of resources needed can be adapted so that the E-BP can run maximizing the use of the resources and minimizing the power consumption. Such adaptivity feature regards the flexibility of resources and BPs management. Focusing on the BP flexibility, it aims to enable (or disable) different sets of configuration parameters values that can be used at runtime. To do so, the BP designer makes use of BP requirements and IS characteristics to define:

1. Maximum and minimal resources configuration in order to satisfy energy and quality constraints, respectively.
2. Abstract tasks and execution paths degree of importance throughout the BP such as critical ones.
3. Warning and alarming thresholds based on IS characteristics and not only BP requirements.

4.4.1 Business process co-design

The concept of co-design suggests that, although actors involved in the design process have different objectives or perspectives, all should be taken into consideration somehow. In the BP design, these actors' objectives can be represented by desired quality and energy requirements, which are represented as indicators thresholds. In order to properly set BP design parameters, we argue that the underlying environment should be taken into consideration, i.e., co-designed. The different configuration of these parameters composes the BP flexibility, which is created by the BP co-design. This flexibility is divided into three main components: i) indicators warning max and min thresholds, ii) enablement (or disablement) of adaptation actions available at runtime, and iii) definition, at design-time, of the environment condition (context) in which adaptation actions can be enacted.

As described in the previous chapter, BP requirements are represented as indicators thresholds. These thresholds are divided into four dimensions, i.e., max and min warning and alarming. Focusing on warning thresholds, they represent situations that are not optimal, but also do not cause significant damage to the system. When trading-off quality and energy aspects, these warning thresholds become key features in order to avoid the violation of alarming thresholds. During the BP co-design, the definition of warning thresholds values (i.e., parameter a_n) is based

on both alarming min/max and the IS gathered data. For instance, let us consider the indicator `CPU payload` at server level with minimum alarming threshold equal to 30%. Depending on the characteristics of the application (such as CPU intensive) and the available computational resources provided by the server (or VM), the range between alarming and warning threshold can vary. Hence, the value 40% can or cannot represent an indicator warning threshold violation.

In order to provide more alternative adaptation actions at runtime, the BP co-design provides possible alternative actions that can enabled (or disabled) according to the process characteristics. An adaptation action is defined as a single change at runtime over application, middleware or infrastructure layer, so that undesired situations are recovered or avoided. They can be triggered by the violation of an indicator alarming threshold (*reactive* adaptation) or by the menace caused by the violation of an indicator warning threshold (*proactive* adaptation). Moreover, the actions are categorized based on their main impact, such as: *less quality*, which represents reduction of performance or energy requirements; *less functionality*, which represents reduction of functional requirements; and *resource reallocation*, which represents resources changes and may cause quality reduction or increase.

Although the enactment of an adaptation action is performed at runtime according to the runtime controller, the actions that involve the BP have to be designed and enabled at design-time. For instance, although the adaptation action `service migration` is managed at runtime within the VM migration scope (e.g., servers and network availability evaluation) at the middleware layer, the service becomes unavailable until the action is not fully completed. The harmful consequences of such unavailability can be higher to the BP goals than the violation of payload indicator, for example. Thus, the enablement (or disablement) of this action is defined at design-time, which is based on the BP characteristics and availability conditions (e.g., the VM in which a set of services is performing a certain abstract task cannot migrate when there is a transaction going on).

Table 4.1 sums up some of the energy-aware adaptation strategies that we consider in our approach. The definition of all these actions is made at design-time while the action request is made at runtime. However, some of them are managed by the runtime controller, like `service migration` as VM migration, while some are managed by the designer in a semi-automated manner, like `process re-design`. This is specified by the column "managed by", which can be design or runtime.

In addition to the enablement or disablement of the adaptation actions described in Table 4.1, the BP co-design also specifies the actions

Table 4.1: Energy-aware adaptation actions at application level defined at design-time

	Action	**Description**	**Enacted by**	**Type**
1.	BP redesign	Redefinition of the process functionalities	designer	less functional
2.	Structure change	Changes with respect to routing tasks	designer	less functional and less quality
3.	Optional flow	One or more execution paths can be skipped	runtime	less functional
4.	Optional task	The execution of one or more abstract tasks can be skipped	runtime	less functional
5.	Non-critical task	Task quality and energy constraints can be relaxed	designer	less quality
6.	Eliminate redundancy	Tasks that are redundant are decommissioned	runtime	less quality
7.	Service replacement	Service can be replaced by functional equivalent ones	runtime	less quality
8.	Service migration	Services can migrate together with their associated VM	runtime	resource reallocation
9.	SLA renegotiation	Renegotiate to reduce functional and non-functional minimum requirements	designer	less quality and less functional

availability conditions. These conditions are interpreted by the runtime controller in order to verify the availability of one action under certain BP conditions. They do not trigger the adaptation action request (which is done by the runtime controller), but they make an action ready (available) to be requested. Thus, these conditions are represented as rules applied to either abstract task ($at_{n,m}$) or concrete service ($cs_{i,j,k}$). Table 4.2 represents some conditions, which can be described as:

1. A business process BP_n can be redesigned if there is no instance running.

2. A business process BP_n can change its flow if there is at least one routing task, i.e., BP_n has more than one execution path (ep).

3. An execution path ep_k can be avoided if there is no critical task in it.

4. An abstract task $at_{n,m}$ can be skipped if it is not critical.

5. An abstract task $at_{n,m}$ is defined as critical according to the designer preferences.

Table 4.2: Energy-aware adaptation action availability conditions

	Action	**Availability conditions**
1.	BP redesign	$(BP_n.redesign) \leftrightarrow \nexists(BP_n.instance)$
2.	Structure change	$(BP_n.flowChange) \leftrightarrow \exists(at_{n,m}.routingTask)$
3.	Optional flow	$(ep_k.optional) \leftrightarrow \nexists(at_{n,m}.critical \in ep_k)$
4.	Optional task	$(at_{n,m}.optional) \leftrightarrow \neg(at_{n,m}.critical)$
5.	Non-critical task	$(at_{n,m}.critical) \leftrightarrow (designer.def)$
6.	Eliminate redundancy	$(at_{n,m}.decommissioned) \leftrightarrow \exists((at_{n,m} \equiv at_{n,h}) \wedge (at_{n,h} \in ep_k))$
7.	Service replacement	$(cs_{i,j,k}.replace) \leftrightarrow \neg(cs_{i,j,k} \in at_{n,m}.critical)$
8.	Service migration	$(at_{n,m}.migrate) \leftrightarrow \nexists Trans(at_{n,m}.onGoing)$
9.	SLA renegotiation	$(cs_{i,j,k}.reneg) \leftrightarrow \neg(cs_{i,j,k} \in at_{n,m}.critical)$

6. A redundant abstract task $at_{n,m}$ can be decommissioned if there is an equivalent one in the BP execution path ep_k.

7. A concrete service $cs_{i,j,k}$ can be replaced if it does not implement a critical abstract task $at_{n,m}$.

8. A concrete service $cs_{i,j,k}$ can migrate (within the scope of VM migration) if it is not running a business transaction.

9. The SLA of a concrete service $cs_{i,j,k}$ can be renegotiated if it does not implement a critical abstract task $at_{n,m}$.

4.4.2 Energy-aware adaptation

Having defined the BP adaptation actions through a co-designed BP and their enactment rules, the runtime controller is responsible to evaluate when and how these actions will be required. This is made through the development of two control loops: local and global. A set of local control loops is associated with each server component, in which the virtual machine is running. This loop deals with individual concrete service deployment. On the other hand, one global control loop takes care of the entire infrastructure layer (composed by several servers) as well as the whole BP (composed by several concrete services). In this way, indicators (KPIs and GPIs) are evaluated from two different granularity levels and so are the adaptation actions.

Based on the BP adaptation actions defined at design-time and available at runtime and the set of adaptation actions that are exclusively runtime (depicted in Table 4.3), the runtime controller selects which are the most appropriate ones so that undesired situations are recovered or avoided. When adaptation is required, the selection is based

Table 4.3: Energy-aware adaptation actions at middleware and infrastructure levels runtime exclusive

	Action	**Description**	**Group**
1.	VM migration	The VM container execution from one server to another	consolidation
2.	VM deploy	Create a new VM	consolidation
3.	VM undeploy	Decommission a VM	consolidation
4.	VM reconfiguration	Reallocate more or less resources for the VM	consolidation
5.	Change CPU P-state	Increase or decrease CPU P-state level	power management
6.	Change disk mode	Change the disk to acoustic or normal mode	power management
7.	Change server mode	Hibernate/wake-up servers that are expected to stay idle for short period of time	power management
8.	Shutdown server	Turn-off/on servers that are not expected to be used for long period of time	power management

on the action characteristics and historical information gathered from log files. In particular, the decision-making process uses the GAMES knowledge database, which contains the results of the past adaptation actions together with the description of the critical situation that they were expected to solve.

As mentioned before, the runtime controller is in charge of requesting for adaptation actions. This request is triggered by indicators threshold violation, which can be warning or alarming. Warning violations request *proactive* actions since they do not represent high damage to the system but indicate that the probability of an alarming violation is high. On the other hand, alarming violations request *reactive* actions and they have priority over warning ones. However, it might happen that more than one adaptation actions are needed to solve an indicator violation. Thus, we say that an adaptation strategy is composed by a coordinated set of adaptation actions, which may involve heterogeneous actions.

Adaptation actions can be totally independent from the design phase (e.g., `change CPU P-state`), be defined at design-time and requested and enacted at runtime (e.g., `service migration`) or be defined at design-time, requested at runtime and enacted at design-time (e.g., `structure change`). In our proposed approach [41], the selection of the more suitable strategy is based on the definition of a set of adaptation rules $\mathcal{R}$ that link the violation of an indicator (warning or alarming) with the set of associated adaptation strategies. More formally, an indicator $I_{h,obj}$

is associated with a system object $obj \in \{\text{E-BP}_n \cup \{at_{n,m}\} \cup \{cs_{i,j,k}\} \cup \{vm_{i,j}\} \cup \{sv_i\}\}$ such that the adaptation rule $R_{h,obj}$ is defined as:

$$R_{h,obj} = \langle I_{h,obj}, \{\langle V_{hw,obj}, \langle As_{hwm}, Conf_{hwm}, Imp_{hwm}\rangle\rangle\}\rangle \tag{4.7}$$

where $V_{hw,obj}$ is the type of violation, i.e., warning or alarming, of indicator $I_{h,obj}$ for the specific system object obj; As_{hwm} is the set of adaptation strategies to be enacted in case of violation $V_{hw,obj}$; $Conf_{hwm}$ is the confidence associated with the effective execution of the action associated to the violation; and Imp_{hwm} is the degree of the importance of the action that depends on the impact that it has on the energy consumption.

Due to existence of dependencies between indicators, such as basic and composed indicators presented in the previous chapter, the impact of an adaptation action can be propagated through the indicator network. This dependency is identified by the function $Dep : I \times \mathcal{P}(I) \rightarrow \mathcal{P}(I)$, such that each indicator $I_{h,obj}$ identifies the set of correlated indicators $I'_{l,obj} \subset I$. Such relationships can be computed using data mining techniques that are described in Chapter 6.

Algorithm 1 details the adaptation strategy selection and enactment perfomed by the `Energy-aware controller` in Figure 4.5. This algorithm represents actions that are required and enacted at runtime. The monitoring module gathers all the data necessary to compute the values of all the indicators $Val_{h,obj}$ and to evaluate the associated requirements. The evaluation of requirements can be seen as a function $Eval : R \times Val \rightarrow \{AS, \emptyset\}$ which, given a value for $I_{h,obj}$, checks both warning and alarming possible indicators violations $V_{hw,obj}$ (line 29) and, in case of violation, returns the strategy As_{hwm} associated with the highest importance and confidence values (line 35). If more than one indicator is violated, the system starts considering root indicators in our GIFs hierarchy until leaf ones (line 2). In order to avoid failures or performance reduction at the application level, relationships among indicators are exploited (line 12). In fact, the function $Eval$ will be applied to all correlated leaf level indicators $I'_{l,obj}$: $Eval(R_{l,obj}, val_{l,obj})$ (line 15). Once all the adaptation strategies have been applied, the value of $I_{h,obj}$ should satisfy all constraints, which means that a second evaluation of $Eval$ should not enact any other action for the considered indicator (line 17). If instead $Eval$ tries to enact other actions, this means that enacted adaptation actions did not succeed. In this case, the function $Eval$ will be applied directly to the examined indicator (line 23). Despite the activation of all the adaptation actions, if the value of $I_{h,obj}$ still violates an alarming indicator, a human intervention is required and the application should be redesigned or restructured (line 10). As some adaptation

strategies could negatively affect the system performance, they can be executed only for a predefined time interval or until the analyzed indicator does not represent any violation anymore.

Algorithm 1 Adaptation Strategy Enactment Algorithm

Require: $rule_{*,obj}$ (all the rules related to object obj)
Ensure: adaptation strategies enactment to eliminate indicators' violation
function PROCEDURE_ENACTMENT($rule_{*,obj}$) {
{the procedure is continuously executed during the assessment phase}
sort $rule_{*,obj}$ from root to leaf level based on indicators
for all $rule_{h,obj} \in rule_{*,obj}$ **do**
  $ind_h \leftarrow$ related indicator instance from $rule_{h,obj}$
  $value_{h,obj} \leftarrow Val(ind_h, obj)$ //get indicator value from assessment tool
  $eval_h \leftarrow$ EVAL($rule_{h,obj}$, $value_{h,obj}$)
  if $eval_h = \emptyset$ **then**
    do nothing //there is no violation in $rule_{h,obj}$
  else if $eval_h =$ human intervention required **then**
    message(human intervention required for $rule_{h,obj}$)
  else
    for all $indLow_l \in$ Dep(ind_h) **do**
      $rule_{l,obj} \leftarrow$ associated rule with $indLow_l$
      $value_{l,obj} \leftarrow Val(indLow_{l,obj})$
      enact adaptation strategy from EVAL($rule_{l,obj}$, $value_{l,obj}$)
      update $value_{h,obj}$
      if EVAL($rule_{h,obj}, value_{h,obj}$) = $\emptyset$ **then**
        exit FOR cycle
      end if
    end for
    if EVAL($rule_{h,obj}, value_{h,obj}$) $\doteq$ adaptation strategy **then**
      //apply adaptation directly to the examined indicator
      enact adaptation strategy from $eval_h$
    end if
  end if
end for
}
function EVAL(rule, value): STRATEGY {
  check that the indicator has warning or alarming thresholds violations
  if there is one or more violated constraints
    select the best adaptation strategy not used yet
    if there is no remaining strategy
      return human intervention required
    else
      return adaptation strategy
    end if
  end if
  else return $\emptyset$
}

4.5 Summary

In this chapter we have demonstrated how the BP co-design can help the identification of non-optimal situations, like the existence of energy leakage, and support the system adaptation manager in selecting and executing actions at runtime aligned with design-time preferences. The BP energy consumption estimation starts from the analysis of the characteristics of the activities composing the process and the resources it requires. On the basis of the actual use of the resources during execution, a way to improve energy efficiency of the process is proposed. The approach enables the identification of energy leakages and/or of indicators (KPIs or GPIs) threshold violations.

Depending on the violation type (warning or alarming) suitable adaptation actions are selected to be enacted. We differentiate these actions according to their enactment environment, design-time or runtime. In this way, the runtime controller is not completely independent from the BP design but, instead, they support each other in order to define and to enact actions that are suitable for both BP goals and runtime controller.

The approach drawback relies on the identification of the impact of an adaptation action throughout the system. Although indicators dependencies are represented trough the function $Dep()$, the relationships between indicators and adaptation actions are not analyzed. This analysis shall have a deep impact in the action selection opening up new strategies causing minimum side-effects from a global perspective. This issue is solved in the next chapter, in which a contextualized goal-based approach identifies the relationships between indicators, adaptation action and their trigger events.

5 Using a goal-based model to identify and analyze system threats

The adaptive behavior proposed in the previous chapter aims to support the capability of reacting in a (semi)automatic way in case of unexpected situations. Such situations are expressed by indicators states, in particular, indicators violation (indicators are detailed in Chapter 3) that represent system threats. Considering that indicators represent system objectives, we argue that the adaptation strategy selection phase should take a higher level of abstraction, in which objectives are clearly defined such that the effects and trade-offs of different adaptation options can be evaluated. A suitable way to define these goals is using goal-based models presented in Requirements Engineering. These models allow the designers to specify goals of the system and their relationships. Some of the most well-known approaches are described in Chapter 2, which includes Tropos [33], i* [181], and KAOS [59]. In order to support the selection of suitable adaptation actions such that side-effects are minimal, we adopted the goal-based risk model proposed by [15]. The model links system goals (represented by indicators thresholds) to events that

might either positively or negatively affect them. The approach also includes the set of actions that can be enacted to intensify or diminish the effects of these events.

As the model and its elements (goals, events and actions) are proposed at design-time, it is often based on assumptions that can be verified at runtime. Thus, we believe the model has to support three main features: i) raised events identification that represent a threat towards the system goals fulfillment, ii) feedback aware adaptation action selection that analyzes previous executed actions in order to create adaptation strategies, and iii) relationships evolution according to a systematic comparison between the expected and the obtained outcome after the enactment of adaptation actions. In this chapter, we deal with event identification and feedback aware adaptation action selection and how these features can be achieved through the introduction of contextualized nodes and timed event analysis. The approach aims to facilitate quality and energy trade-offs such that relationships and their impact on pools are easily identified and hidden ones are discovered. The initial version of the model was created based on GAMES knowledge [25], in which relevant GPIs and KPIs are identified by the project methodology. Moreover, the project monitoring system was used to provide the required input data for the proposed approach.

Figure 5.1 shows the main elements that compose the proposed framework. First, the `monitoring` system provides raw data about the underlying environment. This information is used to calculate the defined indicators by the `indicator calculation` module. The `event identification` is responsible to verify both monitored variables and indicators values in order to recognize possible situations that represent a threat to the system. This is done according to the system goal-based model defined by GAMES assessment phase. If a threat is identified, the module creates several event occurrences that are analyzed by the `event analysis` module in order to separate the different types of events and, in particular, the ones that require the enactment of adaptation actions (meaningful events). The `adaptation selection` module is responsible to select the adaptation actions that eliminate the system threat without creating new ones. Thus, single actions are aggregated as adaptation actions and send to the `adaptation parameterize` module. In this module the system manager, who is the user responsible for the system environment, shall approve or disapprove the adaptation actions and, if approved, he should provide some necessary actions parameters values. In parallel with the `adaptation selection`, the `history-based analysis` module tries to identify possible misleading relationships within the goal-based model that do not provide effective adaptation actions. The output

of this module implies a new adaptation attempt, a model modification or both. The model modification comprehends: i) relationships impact labels update, and ii) revelation of unexpected and not yet modeled relationships. The identified model changes are verified by the system manager in the `evolution verification` module in order to ensure the goal-based model soundness. In this chapter we cover the `event identification` and the `event analysis` modules, while `adaptation selection` and `history-based analysis` modules are tackled in the next chapter.

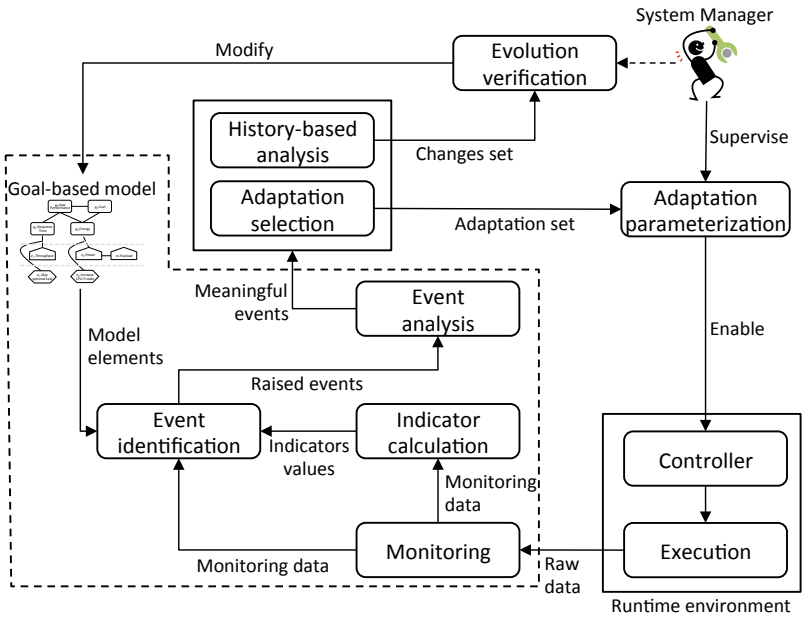


Figure 5.1: Framework overview

5.1 Model specification

In order to represent our goal-based model we have used the Goal-risk model proposed by [15] with some modifications in order to accommodate our objectives. These modifications are regarding to the incorporation of context dependent relations, in which IF-THEN clauses are used to define the relation impact/severity level. Although we adopted the same set of relations types, we paid more attention to the relations responsible to enact adaptation actions. For this reason, the model is defined by a set of three elements $\langle \mathcal{N}, \mathcal{R}, \mathcal{A} \rangle$, in which $\mathcal{N}$ represents the set of nodes, $\mathcal{R}$ is the set of relations among nodes, and $\mathcal{A}$ is the set of special relations,

called alleviation, between one node and one node or relation. A node $N \in \mathcal{N}$ is composed of a set of attributes that are:

- *Type:* Nodes are divided into three different types, that are goals G (strategic interests to be achieved), events E (circumstances that impact over the achievement of goals), and adaptation actions A (actions used to support the achievement of goals).
- *SAT and DEN evidences:* It represents the evidence that the node will be satisfied (SAT) and/or denied (DEN), which are qualitatively represented as *(F)ull*, *(P)artial* or *(N)one* such that $F > P > N$.

The set of relationships is divided into different types and they are: AND/OR-decomposition, contribution and impact, where a source node N_s holds a relation with a target node N_t such that

$$\mathcal{R} : \{(N_1, ..., N_s, ..., N_n) \xrightarrow[r]{p} (N_1, ..., N_t, ..., N_n)\}$$

r is the relation type and p is the propagation of the contribution/impact level. AND/OR-decomposition relations refine nodes into more fine-grained ones. Contribution relations propagate SAT and DEN from the source node to the target node. In [15] the authors propagate four levels of contributions, that are: $--$ (strong negative), $-$ (negative), $+$ (positive), and $++$ (strong positive). Table 5.1 describes the satisfaction and denial evidence propagation rules from the source node N_s to the target node N_t considering their qualitative representation. The relation propagates both SAT and DEN evidence when the propagation type is not specified like, for instance, $N_s \xrightarrow{++} N_t$. Finally, impact relations state an effect (positive and/or negative) of an event node $e_i \in E$ towards the satisfaction of a goal node $g_p \in G$ and, thus, it is defined as $E \xrightarrow[impact]{p} G$. This impact is qualitatively measured in the same way as contribution relation, i.e., $p = \{--, -, +, ++\}$. In fact, negative impact relations are the target of alleviation relations.

Alleviation relations aims to link adaptation actions nodes A with negative impact relations in order to eliminate or diminish an event risk. This is done in two different ways:

- reduce the event *likelihood*: this is done using a direct relation between an action $a_v \in A$ and an event $e_i \in E$ such that $A \xrightarrow[alleviation]{\{-S,--S\}} E$. The event probability is calculated based on its SAT and DEN evidences, in which increasing the DEN evidence reduces the event

Table 5.1: Evidence propagation rules according to [15]

Relation	SAT(N_t)	DEN(N_t)
$N_s \xrightarrow{+S} N_t$	Min(SAT(N_s),Partial)	None
$N_s \xrightarrow{++S} N_t$	SAT(N_s)	None
$N_s \xrightarrow{+D} N_t$	None	Min(DEN(N_s),Partial)
$N_s \xrightarrow{++D} N_t$	None	DEN(N_s)
$N_s \xrightarrow{-S} N_t$	None	Min(SAT(N_s),Partial)
$N_s \xrightarrow{--S} N_t$	None	SAT(N_s)
$N_s \xrightarrow{-D} N_t$	Min(DEN(N_s),Partial)	None
$N_s \xrightarrow{--D} N_t$	DEN(N_s)	None

occurrence. This is the reason we use only $\{-S, --S\}$ as they propagate the SAT of action a_v towards the DEN of event e_i according to Table 5.1.

- reduce the event *severity*: this is done linking an action $a_v \in A$ and an impact relation between, such that $A \xrightarrow[alleviation]{\{-,--\}} [E \xrightarrow[impact]{\{-,--\}} G]$. As mentioned, the severity of event relationships is dynamic defined depending on the identified context scenario through IF-THEN clauses.

Considering the main components of the goal-based model above described and the highly dynamic environment that we are dealing with, new techniques to identify an ongoing problem and properly react against it are needed. To do so, we added new analysis tools into the model in order to support real-time events identification based on streams monitored data, context identification for each model node, and feedback analysis in order to validate the outcome of taken decisions. The identification of ongoing problems is performed by the `event identification and analysis` modules depicted in Figure 5.1. Indicators violation might, or might not, require the enactment of adaptation actions. Thus, the first step is to identify an indicator violation and evaluate its importance throughout the model based on the indicator calculated value. This is done by the `event identification` module, which creates event occurrences. An event occurrence dynamically represents the event impact relation, in which relation severity and event likelihood are defined based on event context models (ECM). The next step is to evaluate the source of the problem in order to distinguish between new threats, threats

Table 5.2: Goals evidence calculation according indicator status

Indicator i_h.status	SAT(g_p)	DEN(g_p)	Fulfillment
Green	F/P	N	H
Yellow	F	P	Mh
	F	F	M
	P	P	M
	P	F	Ml
Red	N	F/P/N	L

already under treatment, and recurrent ones. The `event analysis` module is responsible for that such that it supports both `adaptation selection` and `history-based analysis` modules.

5.1.1 Defining goals, events and adaptation actions

In this subsection we formally describe the three types of nodes and how they are composed.

Goals. Goals G are expressed as indicators (GPIs/KPIs) fulfillment, in which one goal $g_p \in G$ is related to one indicator $i_h \in I$ and I can be a single indicator or an aggregated through a GIF. The goal fulfillment depends on the indicator value with respect to its thresholds. We defined five different levels of a goal fulfillment: (H)igh, (M)edium-(h)igh, (M)edium, (M)edium-(l)ow, and (L)ow, where $H > Mh > M > Ml > L$. According to the indicators definition in Chapter 3, an indicator thresholds are divided into sets of four dimensions values: maximum alarming a^{max}, maximum warning w^{max}, minimum warning w^{min} and minimum alarming a^{min}, such that $a^{max} \geq w^{max} \geq w^{min} \geq a^{min}$. The indicator fulfillment can assume three different states, that are: i) **green**, when warning and alarming thresholds are not violated; ii) **yellow**, when only warning threshold is violated; and iii) **red**, when both warning and alarming thresholds are violated. Based on that, the evidences of SAT and DEN and the fulfillment of a goal $g_p / g_p \triangleq i_h$ are calculated according the rules based on Table 5.2.

Events. Events E represent unexpected situations that impact positively or negatively over goals. Although they are defined by the users, events are defined in terms of indicators formula variables. They are expressed by two attributes: *status* and *likelihood*. The event *likelihood* (λ) is calculated according event evidences that support (SAT) or prevents

(DEN) an event occurrence, such that $\lambda(e_i) \leftarrow SAT(e_i) \wedge DEN(e_i)/e_i \in E$, where high SAT and low DEN values imply in high event likelihood [15]. However, in order to properly manage the event occurrences we added the attribute *status*. This attribute is responsible to identify the current event with respect to its occurrences and can assume the following values: `ready`, `new`, `wip` (work in progress), `done`, and `renew`. The parameter `ready` states that incoming event occurrences have no relation with past ones. The parameter `new` identifies a new event occurrence that was not yet managed by an adaptation action. The parameters `wip` means that one or more adaptation actions were selected as candidate actions to reduce the event *likelihood* but not yet executed or are currently under execution. If the executed set of actions reduced the event likelihood, the parameter `done` is assumed. This parameter has an expiration time, which represents the actions approval period. The approval period aims to validate the action effectiveness and identify possible solutions that are only temporary. Thus, if a new event occurrence shows up during this period, the event status is defined as `renew` and a different set of actions should be executed. Otherwise, the event status is returned to `ready`. The rules used to set the event status attribute and their transition states are described in Section 5.3 of this chapter.

Adaptation actions. Adaptation actions A are required when a goal fulfillment is *Low*, i.e., there is an indicator alarming violation. An adaptation action represents a single mechanism able to change one or more monitoring variables such that it supports goals fulfillment by avoiding indicators thresholds violation. This change can be at the infrastructure level (e.g., change the CPU frequency), middleware (e.g., allocate more memory for a VM) or application (e.g., skip the execution an abstract task within the BP). As described in the previous chapter, the proposed approach focuses on design-time adaptation actions, which are divided into two groups: design-time actions and mixed (design-time and runtime) actions. The design-time actions include actions that are selected and executed at design-time, such as SLA negotiation. On the other hand, mixed actions are enabled at design-time and executed at runtime, such as skip an abstract task of the BP.

An adaptation action $a_v \in A$ aims to eliminate or diminish a negative event occurrence through an alleviation relation. As described above, this type of relation is used to reduce negative event *likelihood* ($A \xrightarrow[alleviation]{\{-S,--S\}} E$) or *severity* ($A \xrightarrow[alleviation]{\{-,--\}} [E \xrightarrow[impact]{\{-,--\}} G]$), which is based on the event occurrence context rules.

Moreover, an adaptation action is composed by attributes like: *type*,

duration, *approvalPeriod* and *cost*. The attribute *type* specifies whether the action is design-time or mixed (design-time and runtime). The *duration* specifies the time interval the action is expected to use during its execution. On the other hand, *approvalPeriod* defines the time interval used to validate the action effectiveness. Finally, *cost* represents the action execution involved costs.

A metamodel of the described model is depicted in Figure 5.2, in which the abstract class `Node` generalizes goals, events, and adaptation actions. Thus, all nodes contain SAT and DEN evidences that are associated with a context model. Goals are expressed in terms of indicators thresholds, events in terms of event occurrences and actions are aggregated into adaptation strategies. Moreover, relation subtypes are indicated by enumerations. Note that alleviation relations are associated with event occurrences once adaptation actions aim to avoid raised events, which are represented as occurrences.

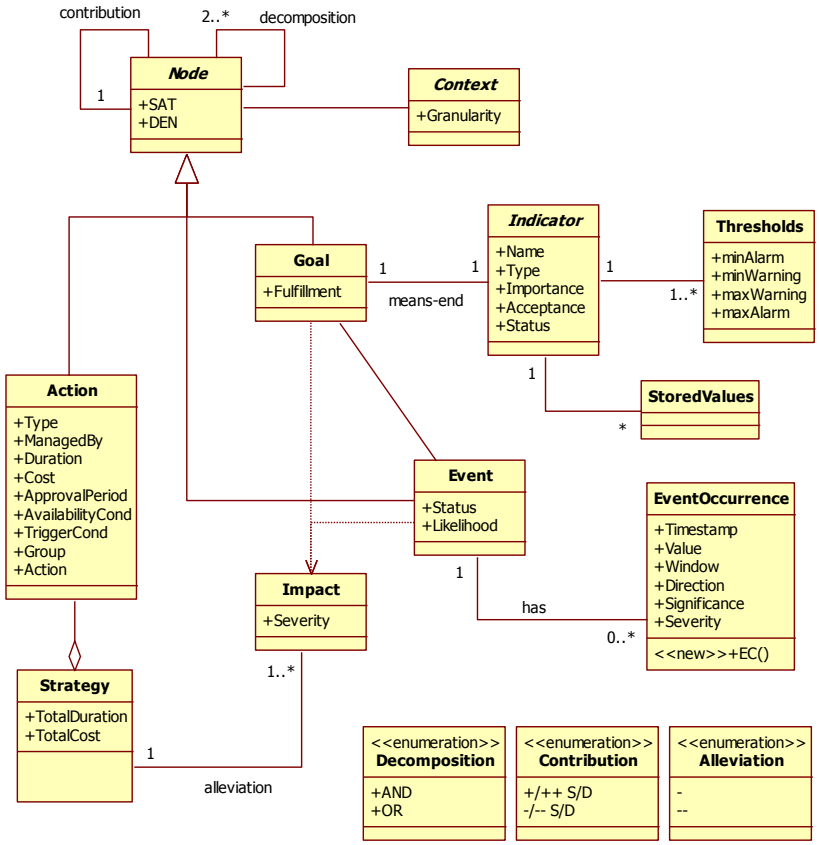


Figure 5.2: Metamodel of the considered model

Figure 5.3 shows a small example of the described goal-based model. At the goals layer (asset layer), `Energy` (g_4) represents the server power consumption over time, `Response Time` (g_3) and `Cost` (g_2) are, respectively, the execution time and cost associated with each application task. Finally, `App. Performance` (g_1) represents the number application transactions per second (TPS) divided by the consumed power. At the event layer we have three events that might cause negative impact over the defined goals. `Throughput` (e_1) and `Power` (e_2) directly affect g_3 and g_4 respectively. Note that the relation $e_2 \xrightarrow[impact]{?} g_4$ requires the event e_2 context rules since the impact severity can vary depending on the application response time. On the other hand, the contribution relation between e_2 and e_3 (`Payload`) always holds a negative label as they are directly related: the higher the payload value, the higher the power consumption value. At the adaptation actions layer (treatment layer), actions are mixed, i.e., they are foreseen at design-time and enacted at runtime. The definition of actions parameters such as how much the CPU frequency has to be reduced (i.e., `CPU P-state`) is supported by the system manager and e_2 likelihood reduction also depends on context information.

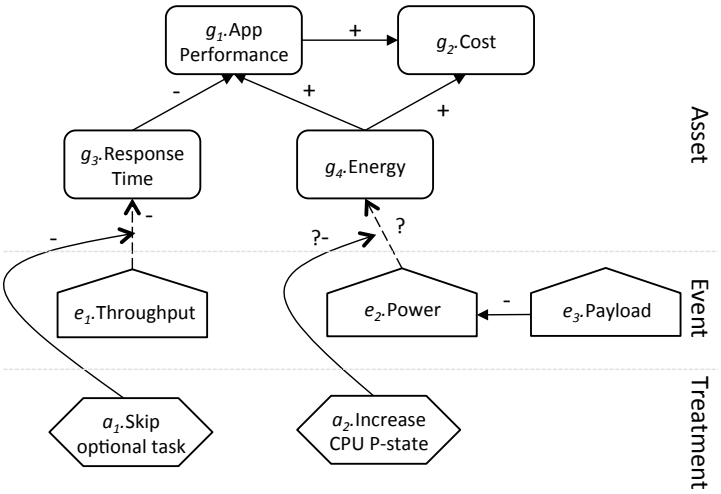


Figure 5.3: Partial model example

5.2 Event identification

The goal-based model described in the previous section supports the identification of different adaptation actions that could be taken in order to diminish the risk of an indicator violation. However, it does not specify what are the underlying circumstances that may restrict the enactment of an action (or a set of actions). These circumstances are represented by sensible changes in the monitored variables values, which might raise events at the model event layer. We argue that, depending on the gathered context information about the environment, the impact of these raised events towards the asset layer can be negative, positive or null. The identification of possible threats towards goals fulfillment is performed by the `event identification` module from Figure 5.1, which is shown in more details in Figure 5.4.

Based on the monitored data and the indicator calculated values, the proposed approach recognizes three different levels of threat, which might or might not result in the creation of an event occurrence. An event occurrence represents that an event is currently harming one or more goals fulfillment and, therefore, adaptation actions shall be enacted in order to restore desired indicators fulfillment levels. The creation of an event occurrence is based the underlying environment, which is represented by the instantiation of the defined set of context rules.

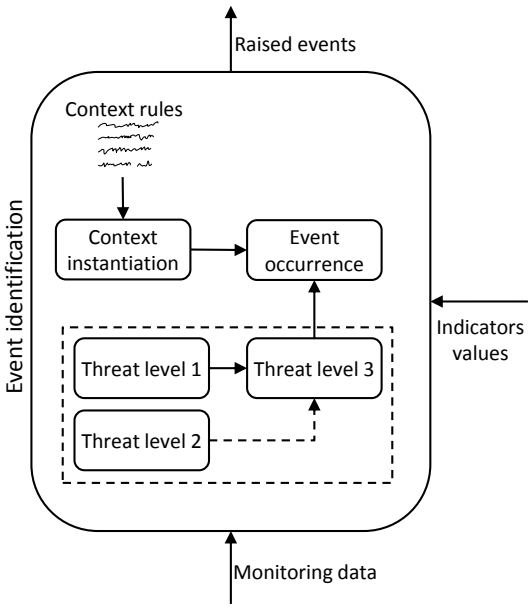


Figure 5.4: Event identification model detailed

5.2.1 Event context instantiation

In environments such as a data center, the monitoring system variables are likely to vary according to (i) the customer functional and non-functional new requirements; (ii) the outcome of past actions enactment; and (iii) the middleware and hardware environment changes. For this reason, the model should be able to support the identification of different situations, represented by the aggregation of monitored variable values. For instance, the event server `power` consumption can have a positive or negative impact over the goal `energy`. Considering that energy represents power over time, the increased power consumption may represent an application execution time reduction, which makes the server free to be powered-off. Otherwise, if the execution time reduction is not significant, i.e. the gains in performance did not represent gains in execution time, higher power consumption directly implies in higher energy consumption. Thus, it represents a negative impact. This simple example makes clear the event impact polarity (positive or negative) dependency with other monitored variables, which compose our context rules shown in Table 5.3.

Due to the spread usage of the word 'context' and to avoid misunderstandings, we adopt the context definition provided by [5]: "context is a partial state of the world that is relevant to an actor's goals", where *world* is the underling environment captured by the monitoring system and *actor* is the system itself. In fact, all defined goals are towards the system soundness and effectiveness. However, instead of using statements and facts to represent the context, we define several context rules that are represented as IF-THEN statements where the IF clause is composed by a set of monitored variables aggregated as AND/OR-decomposition. Instead, the THEN clause describes the significance and severity parameters that the verification of the IF clause represents towards the asset layer. At the event layer, these IF-THEN statements represent positive and negative effects of an event-goal impact relation. The ECM metamodel is depicted in Figure 5.5, in which the attribute `CondRule` represents the positive or negative effect of the IF clause and `SigVal` and `SevVal` represents significance and severity of the THEN clause, respectively.

Table 5.3 describes examples of `context rules` at the event layer with respect to the goal-based model shown in Figure 5.3. Event e_2 (`Power`) high values may have a positive impact on energy consumption if the server is expected to shift to hibernate mode ($s_1.mode = hib$) and stays for a minimum time interval ($\Delta s_1.mode(t) \geq s_1.mode.min$) due to a reduction of the application response time ($app_1.rt^{current} < app_1.rt^{before}$).

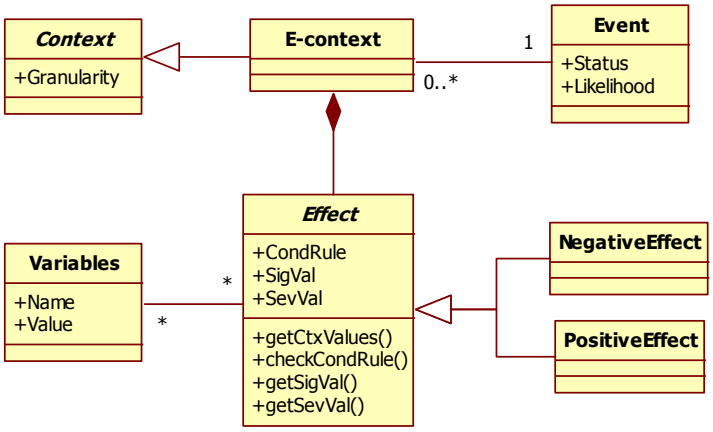


Figure 5.5: Event context metamodel

However, it may have a negative contribution if the current application execution time is greater or equal than before ($app_1.rt^{current} \geq app_1.rt^{before}$) or there is no server mode shift ($s_1.mode^{current} = s_1.mode^{before}$).

5.2.2 Identifying system threats

Monitored data, from application, middleware and infrastructure layers, is continuously produced by the monitoring system. All this information has to be analyzed so that threats are identified and properly treated. The three threat levels modules in Figure 5.4 are responsible to recognize situations that might threaten the indicators fulfillment. In order to deal with such a huge bunch of data rapidly, we adopt Stream Reasoning techniques proposed by [17] where streams generated by the monitoring

Table 5.3: Example of event context rules with respect to Figure 5.3

	Granularity	Server	
e_2: **Power** [high]	Positive Effect	CondRule	$s_1.mode = hib \wedge \Delta s_1.mode(t) \geq s_1.mode.min \wedge app_1.rt^{current} < app_1.rt^{before}$
		SigVal	0.6
		SevVal	+
	Negative Effect	CondRule	$app_1.rt^{current} \geq app_1.rt^{before} \vee s_1.mode^{current} = s_1.mode^{before}$
		SigVal	0.4
		SevVal	−

system are represented as materialized views of RDF [1] triples. These views are based on deductive rules and each triple is associated with an expiration time. Data streams are defined as unbounded sequences of time-varying data elements [2] and the proposed solution manages to inspect "continuous" data streams in real-time as illustrated below.

Differently from other types of data, streams are consumed by queries registered into a stream processor, that continuously produce answers. A common and important simplification applied by stream engines is that they process information within windows, defined as periods of time slots in which the flowing information should be considered. Such windows are continuously evolving due to the arrival of new data in the stream, whereas data falling outside of the windows is lost, in other words, it expires. The window size defines the stream *expiration time* which represents the triple arrival timestamp plus the window size, assuming the window size to be constant (time-invariant). We also assume time as a discrete and linear ordered variable parameter. Therefore, if the window is defined as 3 time slots long and a time slot is 5 seconds long, a triple entering at time τ will expire at time $\tau + 15s$. The expiration time of derived triples depends on the minimum expiration time of the triples it was derived from. In our approach, derived triples represent different levels of threats that may raise an event.

Considering the current materialized predicates window, represented by $\mathcal{W}$, the `event identification` module derives different levels of threat in order to raise an event based on the a set of rules. A threat is identified if one or more indicators are violated with respect to their warning and alarming thresholds. An indicator violation represents the first level of a threat, which might or might not raise an event based on the indicator alarming violation duration. This means that, a single indicator violation might not represent an ongoing system problem that requires an adaptation action. For example, during the VM migration action, the application availability indicator may be violated. However, if the migration action is executed normally, i.e., the VM is successfully migrated without errors, the application availability violation does not require an adaptation action and, therefore, an event occurrence should not be created.

The identification of an indicator violation is represented by the creation of the following entailment:

[1] Resource Description Framework (RDF) is a W3C recommendation for resource description [111].

$$
\begin{aligned}
i_h.\text{status} = \text{`yellow'} \longleftrightarrow i_h.\text{value} \in Y_h / \\
Y_h = \bigcup_{t=1}^{T}([a_h^{min}, a_h^{max}](t) \setminus [w_h^{min}, w_h^{max}](t)) \\
i_h.\text{status} = \text{`red'} \longleftrightarrow i_h.\text{value} \in R_h / \\
R_h = U_h \setminus \bigcup_{t=1}^{T}[a_h^{min}, a_h^{max}](t)
\end{aligned}
\tag{5.1}
$$

where v_h is the calculated value of indicator i_h and U_h is the set of all possible values the indicator can assume and, therefore, $v_h \in U_h$. The set of values that do violate the indicator's warning thresholds (max or min) is defined as Y_h. In the same manner, R_h represents the set of values that violates alarming (max or min) thresholds, in which $(Y_h \cup R_h) \subseteq U_h$. Finally, T represents thresholds sets of i_h where t defines the thresholds dimensions $\{a_h^{min}, w_h^{min}, v_h, w_h^{max}, a_h^{max}\}$ such that $a_h^{min} \leq w_h^{min} \leq w_h^{max} \leq a_h^{max}$.

The second level of a threat tries to identify warned violated indicators (*yellow*) that are likely to become alarmed violated indicators (*red*). We represent these indicators as *orange* ones. The identification of an *orange* indicator is similar to the identification of *yellow* ones, but narrowing alarming thresholds. Considering that W_h represents the set of triples of i_h such that warning threshold (min or max) is violated. The alarming thresholds are narrowed based on W_h standard deviation σ (R'_h). The calculation of $\sigma(W_h)$ is $\sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2}$ where N is the number of triples in W_h. As we aim only the triples that violates the narrowed alarming thresholds but do not violate the normal alarming thresholds, we have $R'_h \cap Y_h$. Thus, an indicator status is defined as *orange* according to the following:

$$
\begin{aligned}
i_h.\text{status} = \text{`orange'} \longleftrightarrow i_h.\text{value} \in R'_h \cap Y_h / \\
R'_h = U_h \setminus \bigcup_{t=1}^{T}[a_h^{min} + \sigma(W_h), a_h^{max} - \sigma(W_h)](t)
\end{aligned}
\tag{5.2}
$$

Finally, the third level of threat actually raises an event after we have identified indicators with *red* or *orange*. However, differently from the first level of threat, the violation lasts more than it is accepted by the

system. As described in Chapter 3, the indicator alarming violation is linked with a acceptance value acp_h, measured in number of monitored time slots, in which the indicator can stay violated without the enactment of adaptation action. For instance, if the acceptance of an indicator is 3 time slots, an alarming violation is considered as first level threat until its appearance in the fourth consecutive slot. As each triple is associated with a timestamp, the calculation of the violation duration is obtained by subtracting the timestamp of the last triple by the first one. Considering that V_h is the set of indicators with either *red* or *orange* status, we calculated the violation duration by $MaxTime(V_h) - MinTime(V_h)$, where $MaxTime()$ returns the last violated triple window and $MinTime()$ returns the first violated triple window. Thus, an event occurrence is created when the following expression hold:

$$\begin{aligned} MaxTime(V_h) - MinTime(V_h) > acp_h / \\ \forall v \in V_h : v.\text{status} = \text{`red'} \vee v.\text{status} = \text{`orange'} \end{aligned} \tag{5.3}$$

5.2.3 Creation of event occurrences

As presented in the previous section, an event is described by attributes and a set of occurrences that represents raised events over time. Hence, the creation of a new event occurrence $ec_{i,j} \in EC_i$ means that event $e_i \in E$ is raised, where EC_i is the set of occurrences of event e_i. Event occurrences are created by the `event occurrence` module in Figure 5.4. They are defined by the following attributes: $\langle$ *tim, val, win, dir, sig, sev* $\rangle$, where:

- ***tim****estamp*: It is the timestamp of the RDF triple that triggers $ec_{i,j}$ creation.
- ***val****ue*: It is the monitored variable value of the RDF triple that triggers $ec_{i,j}$ creation.
- ***win****dow*: It represents the window in which the RDF triple that triggers $ec_{i,j}$ is placed.
- ***dir****ection*: It dictates if the obtained value is increasing or decreasing based on threshold violation, i.e., max or min. This information is important in order to support the adaptation action selection phase.
- ***sig****nificance*: Considering the event context model previously described and current monitored variable values, the event occurrence

significance is within the range [0..1]. The calculation multiplies two variables: the significance of the considered monitored variable with respect to other variables that may trigger an event occurrence (defined by the user within the range [0..1]) and the normalized variable value (within the range [0..1]). The normalization is calculated as $1 - func(p_1, p_2, p_3)$, in which **funcNorm()** is the scaling functions of [184] and the set of parameters $\{p_1, p_2, p_3\}$ are the variable value, max and min limits. These limits are the violated indicator threshold (alarming) and the variable maximum (or minimum) existing values the variable can assume. The significance is important (together with impact) to define which event occurrences have priority to be eliminated by the adaptation actions.

- ***sev**erity*: This attribute defines, in a qualitative manner, the severity level of event occurrence $ec_{i,j}$ over one or more goals. In this way, it considers not just the relation between the violated indicator (which is represented in the model by a goal $g_p \in G$) and event e_i, but all goals impact relationships that event e_i has within the model. The relation between the violated indicator and the event occurrence is negative ($-$ or $--$), however the event can have positive ($+$ or $++$) impact over other goals. Thus we consider the set of goals that hold impact a impact relation with event e_i, named as $G' \subseteq G$:

$$e_i \xrightarrow[impact]{[++,+,-,--]} g_p / g_p \in G'$$

The creation of an event occurrence depends on both the set of RDF triples within the current materialized window ($\mathcal{W}$) and the event-context models (ECM) defined in the previous section. Algorithm 2 details the process of creating a new event occurrence $ec_{i,j}$. The algorithm input parameter is the RDF triple t_k that satisfies Eq.5.3, i.e., an indicator alarming violation with longer duration than it is accepted.

The first step is to find out the set of triples that represent a *red* or *orange* violation, according to Eq. 5.3. This is done by function `getVtriples()` (line 2), which retrieves the set of triples $\mathcal{W}' = \langle i_h$ ind:value $v_h \rangle$. As described in the previous section, goals (G) are defined as indicators thresholds and events (E) in terms of monitored variables. If the calculation of an indicator value depends on more than one variables, it means that one goal is impacted by more than one event. Thus, we need to identify the event $e_i \in E'$, where $E' \subseteq E$ represents the set of candidates events to be raised due to i_h violation. In order to create

E' we need to identify the goal g_p that represent the violated indicator threshold i_h through the function `getGoal()` (line 3). Based on g_p we select all events that hold impact relationship like $e_i \rightarrow g_p$ (line 4). The identification the right event that is triggering the indicator violation is based on the event context-model and, therefore, we need the select the event context-models with respect the event candidates E' (lines 5-6). An event context-model emc_i may have several positive and negative effects context (line 7). Therefore, an event is raised when negative conditional context rules hold (line 9-10), which are based on current context variable values (line 8). Note that an indicator violation can raise more than one event and, each event can create one or more event occurrences, depending on the defined event context conditional rules.

Algorithm 2 Creating new event occurrences

Require: t_k
1: $i_h \leftarrow \text{getInd}(t_k)$
2: $\mathcal{W}' \leftarrow \text{getVtriples}(t_k, i_h)$
3: $g_p \leftarrow \text{getGoal}(i_h)$
4: $E' \leftarrow \forall e_i \in E : e_i \xrightarrow{impact} g_p$
5: **for all** $e_i \in E'$ **do**
6: $\quad emc_i \leftarrow \sigma_{emc_i.event=e_i}(ECM)$
7: $\quad$ **for all** $neg\_effect_j \in emc_i.negEffect$ **do**
8: $\quad\quad$ ctx $\leftarrow$ getContextValues(neg_effect_j)
9: $\quad\quad$ **if** checkCondRule(ctx,neg_effect_j) **then**
10: $\quad\quad\quad ec_{i,j} \leftarrow$ new EC(e_i, i_h, neg_effect_j)
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: **end for**

Whenever an event has to be raised, a new event occurrence is created. Algorithm 3 describes the event occurrence constructor, in which all event occurrence attributes are properly filled. The function `getTim-_Val_Win()` sets *timestamp*, *value* and *window* respectively (line 1). The direction depends on which alarming threshold is violated, i.e., a^{max} or a^{min} (lines 2-10). The indicator attributes $MaxVal$ and $MinVal$ represents the maximum and minimum values the indicator can assume. The variables $limMax$ and $limMin$ are the maximum and minimum limits used by the normalization function $funcNorm()$ to scale the indicators value. This function is used as weight in order to define the event occurrence *significance* (line 11). The *severity* attribute is obtained by event context conditional rule (line 12). This value is defined by the user

during the context creation. Finally, the function `updateEventStatus` updates the status of the event according to the new event occurrence.

Algorithm 3 Creating new event occurrences - *EC* constructor

Require: e_i, i_h, neg_effect_j
1: $ec_{i,j} \leftarrow$ getTim_Val_Win(e_i, i_h)
2: **if** $i_h.a^{max}$ = 'violated' **then**
3: $ec_{i,j}$.dir $\leftarrow$ 'high'
4: $limMin \leftarrow i_h.a^{max}$
5: $limMax \leftarrow i_h.MaxVal$
6: **else**
7: $ec_{i,j}$.dir $\leftarrow$ 'low'
8: $limMin \leftarrow i_h.MinVal$
9: $limMax \leftarrow i_h.a^{min}$
10: **end if**
11: $ec_{i,j}$.sig $\leftarrow getSigVal(neg\_effect_j) *$ funcNorm(
12: $i_h.Val, limMax, limMin)$
13: $ec_{i,j}$.sev $\leftarrow getSevVal(neg\_effect_j)$
14: updateEventStatus(e_i)

Algorithm 4 describes the normalization function used to weight the event occurrence significance. Depending on the event occurrence monotonic function (identified by the attribute *direction*), the function scales the indicator value within the range [0..1]. In this way, the bigger the value the higher the significance weight should be, where 1 is the highest. This weight is multiplied by the indicator significance defined by the user and, therefore, event occurrences with higher significance have priority to be solved as they cause bigger damage to the system.

5.3 Event analysis

The existence of raised events through event occurrences indicates that adaptation actions have to be enacted. However, we argue the approach should be able to identify the different raised events in order to support the adaptation action instrumentation. To do so, the `event analysis` module identifies the current event *status* attribute with respect the event occurrences. The key idea is to map the existing relation between two events such that the system is able to recognize when one event is caused by another through the enactment of its related adaptation action ($E \rightarrow A \rightarrow E$).

In order to represent the diverse event status transitions, Figure 5.6

Algorithm 4 Creating new event occurrences - Significance normalization function

```
1: funcNorm (i_h.Val, limMax, limMin){
2: if limMax − limMin ≠ 0 then
3:    if ec_{i,j}.dir = 'high' then
4:       return 1 − (limMax − i_h.Val) / (limMax − limMin)
5:    else
6:       return 1 − (i_h.Val − limMin) / (limMax − limMin)
7:    end if
8: else
9:    return 1
10: end if
11: }
```

depicts the different states that the event attribute status can assume. The transition from one state to another is defined through several rules, which are defined from 1 to 5. As described in the previous section, the state `ready` indicates that $ec_{i,j}$ has no relation with past ones. The state `new` indicates an event occurrence that was not yet managed by an adaptation action. The state `wip` (work in progress) indicates that actions are under execution and we shall wait for their results. This waiting time is defined by the action attribute *duration*. When all actions are set as 'finished', the $ec_{i,j}$ assumes the state `done`. At this point, the event occurrence remains in this state until the action *approvalPeriod* expires, which guarantees the action effectiveness. Thus, if a new occurrence appears regarding to the same event during the approval period, it assumes the state `renew` and more actions should be executed. Otherwise, the state `ready` is assumed and new occurrences are not related to past executed adaptation actions.

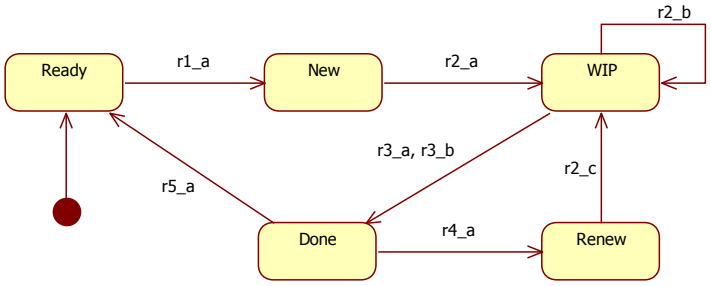


Figure 5.6: Event status transition states

The detailed description of the rules used in the states transitions depicted in Figure 5.6 are expressed in datalog as follows:

"*AdaptationRequired*"

$$r1_a : \mathcal{W}^{+}(e_i \text{ ev:status 'new'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'ready'}), \mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j})$$

"*OngoingAdaptation*"

$$r2_a : \mathcal{W}^{+}(e_i \text{ ev:status 'wip'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'new'}), \mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j}), \neg\mathcal{W}(a_v \text{ aa:enact 'finished'})$$

$$r2_b : \mathcal{W}^{+}(e_i \text{ ev:status 'wip'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'wip'}), \mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j}), \neg\mathcal{W}(a_v \text{ aa:enact 'finished'})$$

$$r2_c : \mathcal{W}^{+}(e_i \text{ ev:status 'wip'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'renew'}), \mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j}), \neg\mathcal{W}(a_v \text{ aa:enact 'finished'})$$

"*AdaptationCompleted*"

$$r3_a : \mathcal{W}^{+}(e_i \text{ ev:status 'done'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'wip'}), \neg\mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j}), \mathcal{W}(a_v \text{ aa:enact 'finished'})$$

$$r3_b : \mathcal{W}^{+}(e_i \text{ ev:doneAt } \tau) \text{ :- } \mathcal{W}^{+}\big((e_i \text{ ev:status 'done'}), ?\tau\big)$$

"*AdaptationStillRequired*"

$$r4_a : \mathcal{W}^{+}(e_i \text{ ev:status 'renew'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'done'}), \mathcal{W}^{ins}(e_i \text{ ev:occur } ec_{i,j}), \mathcal{W}(a_v \text{ aa:enact 'finished'})$$

"*AdaptationEffective*"

$$r5_a : \mathcal{W}^{+}(e_i \text{ ev:status 'ready'}) \text{ :- } \mathcal{W}^{before}(e_i \text{ ev:status 'done'}), \mathcal{W}(e_i \text{ ev:doneAt } ?doneTime), (e_i \text{ ev:approval } ?approvalPeriod), now() - ?doneTime > ?approvalPeriod$$

where ?*time* represents the current timestamp. In addition, the rules are based on the following materialized windows: $\mathcal{W}$ is current materialized window, $\mathcal{W}^{+}$ contains the derived triples to be added in $\mathcal{W}$, $\mathcal{W}^{before}$ represents the previous materialized window, and $\mathcal{W}^{in}$ represents the new triples that are coming from the `event identification` module, i.e., event occurrences or monitored variables values. Adaptation actions are required only for raised events that hold either 'new' or 'renew' status.

Figure 5.7 depicts four envisioned scenarios in which an event occurrence can be associated with the execution of adaptation actions. An event occurrence $ec_{1,1}$ is created at time window 6. In order to diminish its impact adaptation action a_1 is enacted (a_1^{start}). In the meanwhile the event occurrence continues to appear since the a_1 is not yet concluded (a_1^{end}). Up to window 8, the event occurrences $e_{1,2}, e_{1,3}, e_{1,4}, e_{1,5}$ are expected since a_1 is not concluded. However, occurrences of event e_1 should not be appear from window 9. This situation is represented in *scenario IV* and it does not mean that an event occurrence $ec_{1,j}$ will never appear again. Three other scenarios can also happen.

In *scenario I*, a_1 was not effective and a new occurrence of e_1 is created, i.e., $e_{1,6}$ and a_1 did not solve the problem. Depending on the next event occurrence window, it is possible to determine if the action a_1 was partially or fully failed. *Scenario II* represents the non-effectiveness of a_1, in which the action temporally solved the problem and $e_{1,6}$ appeared during the action approval period. This means that a_1 was not suitable to eliminate the threat of event e_1 and a different action should be enacted this time. In *scenario III*, the action a_1 fully solves e_1, but generates a side-effect (expected or not) represented by $e_{2,1}$, i.e., event e_2 was raised. The identification of a raised event as a side-effect of an action enacted to solve another event ($e_x - a_z - e_y$) is determined by two factors: i) the second event e_y appears within the action a_z approval period and ii) both events e_x and e_y share at least one monitored variable with action a_z.

In order to make clear the difference between the last three scenarios, let us consider a flat tire example. After the identification of the problem, one possible action to solve the problem is filling the tire with air. If the reason of the problem is a hole, it means that this action is not effective, since the tire will be flat again in few minutes or hours. This situation is represented in *scenario II*. Instead, if another action is chosen, such as replace the tire, different events can be raised that are related to the enactment of the action. In the example, the events car unstable (due to inadequate spare tire) or drive not safe (due to the lack of available spare tire within the car) can be raised. This situation is represented in *scenario III*. Even if the tire replacement effectively solves the problem, it

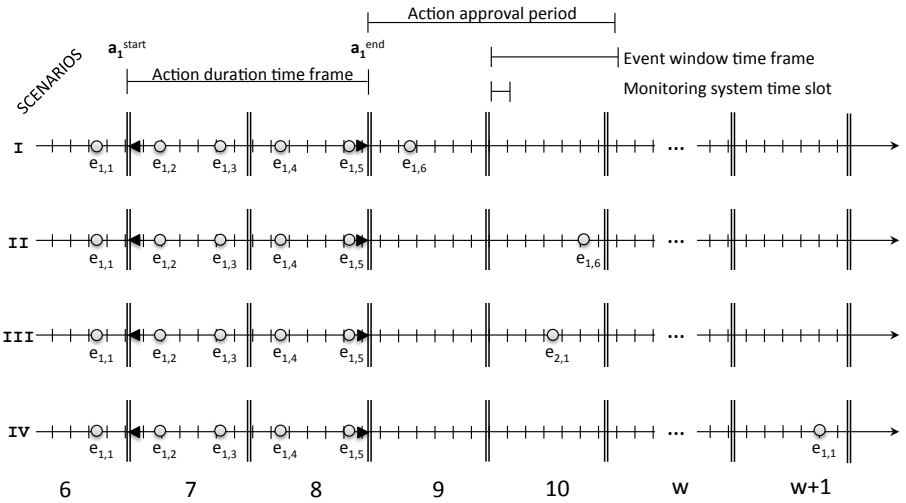


Figure 5.7: Timed scenarios

does not prevent that a new hole appears in the future (*scenario IV*). A more detailed analysis with respect to an adaptation action feedback and the implications of failed actions within both `adaptation selection` and `history-based analysis` modules from Figure 5.1 is described in the next chapter.

5.4 Summary

This chapter introduces a framework that is based on a goal-driven approach in order to identify system threats and provide suitable solutions. The novelty relies on the external supporting components, which help the identification of a system threat based on pattern recognition and context evaluation. Also, the event identification and analysis components provide mechanisms that extend the goal-risk model proposed by [15], where events are rather static elements. In our model, events are presented as event occurrences within a timed frame, in which analysis regarding significance, for example, can be made. Events are also well defined in terms of indicators thresholds violations and a less subjective analysis is described. The main limitation of the proposed event analysis is that both the context models and events are based on monitoring variables and, having both at the same level of granularity, is not always feasible.

6 System threats elimination: adaptation selection and model evolution

In this chapter we continue following the framework proposed in the previous chapter by describing in details the mechanisms that involve adaptation selection and history-based model evolution. After events have been identified as event occurrences and the status properly set, the next step is to use our goal-based model to support the creation of adaptation strategies such that system threats are diminished or eliminated. In order to do so, the adaptation selection mechanism has to consider all inter-relationships of the many involved elements such that enacted actions do not harm the system with undesired side-effects. Considering that, we believe that continuous monitoring should be done in order to verify expected obtained results and, whenever possible, recognize new scenarios based on past experiences through data mining techniques in order to be able to evolve the model.

The four modules described in this chapter are highlighted in the upper part of Figure 6.1, which shows the framework to perform the desired adaptation selection and model evolution. They are: `adaptation`

`selection`, `adaptation parameterization`, `history-based analysis`, and `evolution verification`. The first two modules are responsible to create an adaptation strategy based on the identified meaningful events, which are represented as system threats. The following two modules are responsible to identify frequent negative results obtained by the execution of a set of adaptation actions. The aim is to modify (evolve) either the current instance or the structure source of our goal-based model according to: i) the effective results obtained by the execution of a set of adaptation actions against an event occurrence and ii) the arriving of new event occurrences due to the previous execution of one or more adaptation actions. The chapter is divided into two sections, adaptation selection and history-based analysis.

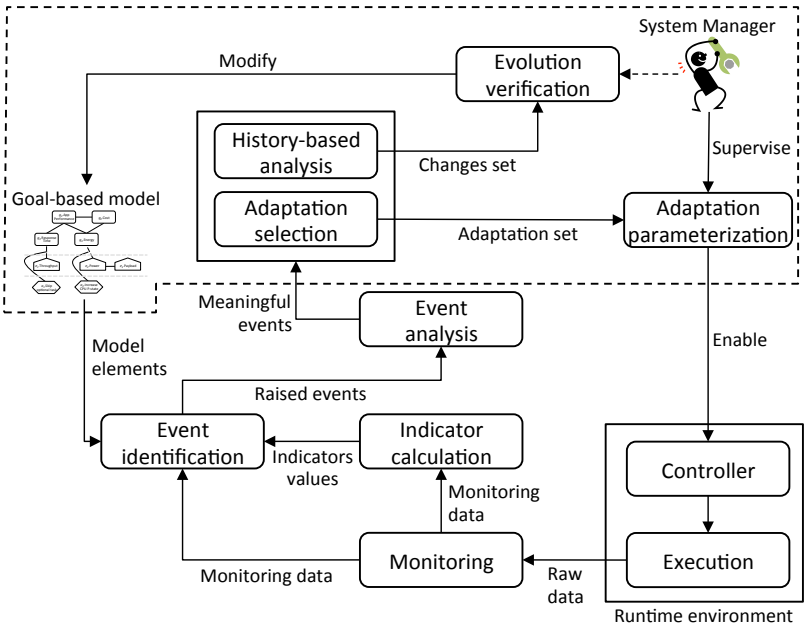


Figure 6.1: Framework overview - adaptation selection and model evolution

6.1 Adaptation selection

When the `event analysis` module generates an event occurrence $ec_{i,j}$ that is related to an event e_i identified as "new" or "renew" by the attribute *status*, the `adaptation selection` module process is triggered. The module selects the most suitable adaptation actions looking to eliminate the event occurrence $ec_{i,j}$. To do so, we use the set of adaptation

actions presented in Table 4.1 from Chapter 4, which are combined in order to create an adaptation strategy. An adaptation strategy is defined by a set of coordinated adaptation actions, i.e., actions that are executed in sequence and/or parallel (coordination). It can happen that this set is composed by only one action, so it does not require any coordination.

As depicted in Figure 6.2, an adaptation action is described by the following attributes: `Type` identifies if the action consequence is regarding functionality/quality reduction or resource reallocation; `ManagedBy` identifies if the action is managed by the runtime controller or by the designer at design-time; `Duration` is a time interval attribute that specifies the expected time interval, in terms of maximum and minimum, that the action takes to complete its execution; `Cost` is an interval attribute that represents the expected cost of the action execution and might depend on the action parameters defined by the system manager in the `adaptation parameterization` module; `ApprovalPeriod` is the expected time interval defined to validate the action effectiveness; `AvailabilityCond` represents the set of conditional rules [1] that should be satisfied in order to enable the action execution; `TriggerCond` also represents a set of conditional rules, but it describes triggering conditions that can be either reactive or proactive; `Group` identifies the action group from an energy perspective (e.g., consolidation and power management in Table 4.3); and finally `Action` is the adaptation action implementation, i.e., what the action should do.

Considering these attributes, the `adaptation selection` module is responsible for selecting the most suitable set of adaptation actions, which are represented into an adaptation strategy and aims to eliminate event occurrences that are causing one or more indicators violation. Figure 6.3 depicts the five steps used to do so. The first step (**Step 1**) identifies the incoming event occurrences that did not trigger adaptation yet. This identification is based on the event status attribute. The next step (**Step 2**) aims to cut off the list of existing adaptation actions in order to keep only the available and suitable ones. In the first case, available actions, we use the action's availability conditional rules. These rules are connected to single BPs that were previously designed to execute the action. For instance, the adaptation action `skip task` can be enacted if and only if the task is defined as `optional task`, i.e., the task is not critical (the action conditional rules are defined in Table 4.2). In the second case, suitable actions, we use the actions triggering conditions attribute in order to rule out actions from the list that were not designed to stop the incoming event occurrence. Each adaptation action is associated with events

[1] These rules are described in Table 4.2 in Chapter 4.

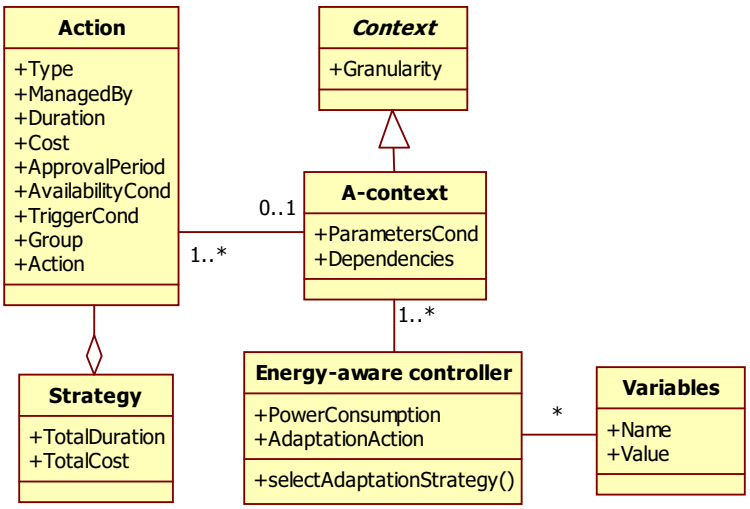


Figure 6.2: Adaptation action selection supported by action contexts

(event-trigger) or indicators violation (indicator-trigger) and, depending on the event occurrence, we keep only the actions that are directly or indirectly related to each other through an indicator violation, which is represented by contribution relations within the goal-based model. For instance, the action a_2 `Increase CPU P-state` depicted in Figure 5.3 is triggered by event e_2 `Power`, which has a direct impact over the goal g_4 `Energy`. However, it also impact indirectly over g_2 `Cost` and g_1 `App. Performance`.

Having the subset of actions provided by Step 2, the next step (**Step 3**) determines the optimal set of actions that are supposed to stop the arrival of new event occurrence with minimal side-effects. This step can be divided into four sub-steps, which are:

i An indicator violation may represent a violation of other indicators that compose the first one. Thus, the aim of this sub-step is to find the indicators-base that have been violated. This is done through `and/or decomposition` relationships among indicators within the goal-based model. This decomposition follows our GIF hierarchic described in Chapter 3.

ii Search for previous situations in which the current threat was identified and, most important, what adaptation strategies were selected with their obtained results. Looking at the goal-based model

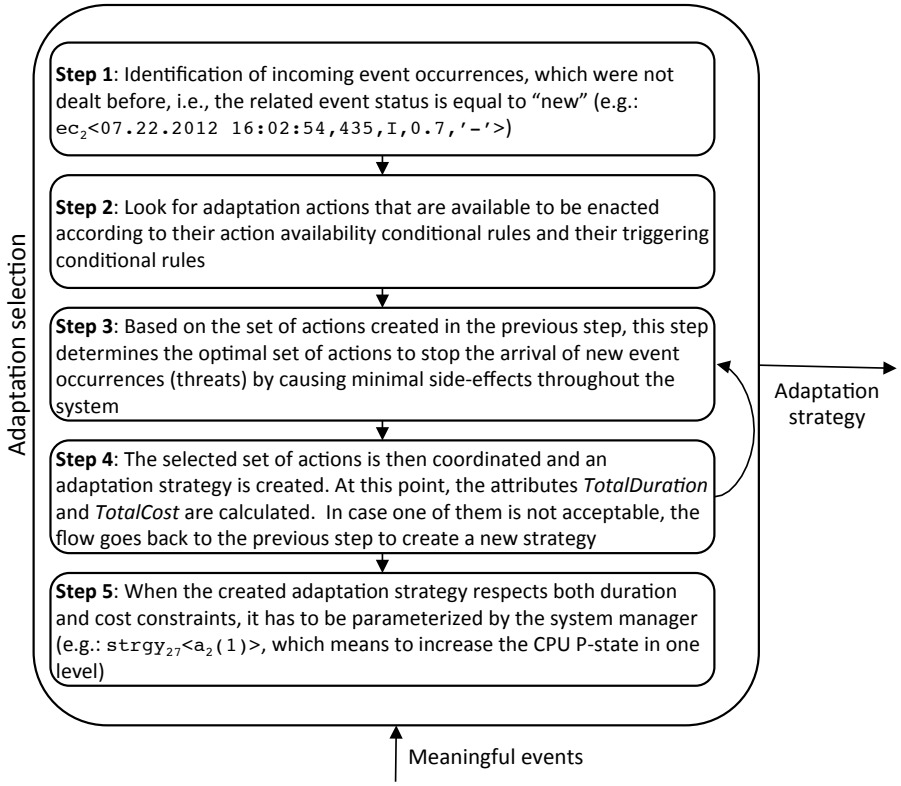


Figure 6.3: Framework details - adaptation selection module

we can have one or more adaptation actions able to mitigate a negative impact relation between one event and one goal. Searching historical log tables we are able to recover the tuple `EventID, ActionID, EventLikelihood_1, EventLikelihood_2, Duration`, in which we can observe the event likelihood reduction after the execution of an adaptation action, i.e., the effective result of an alleviation relation over an impact relation. A list of adaptation actions that did not reduce the event likelihood is created.

iii At this point a verification process estimates each action effects, both positive and negative. This is done through the Backward and Forward reasoning algorithms proposed by [15], which the first generates the set of input evidence in order to satisfy high level goals (top-down) and the second propagates the nodes input evidence throughout the model (bottom-up). The actions that propagate more negative effects than positive or do not satisfy minimal dura-

tion time or cost are ruled out of the candidate set. After that we ensure that all quality and energy constraints are satisfied (staying at green or warning levels) using the Constraint Satisfaction Optimization Problem approach described in Chapter 3.

iv Finally, the selection of the adaptation actions and their coordination are performed by Algorithm 1 proposed in Chapter 4 and represented by the `Energy-aware controller` depicted in Figure 6.2. The controller gathers the necessary context information from all the 3-layers (application, middleware, and infrastructure) in order to establish the actions parameters range according to the underlying hardware and software specification. This is represented by the `A-context` class, in which the attribute `ParametersCond` defines this range and the attribute `Dependencies` states possible additional hardware or software limitations for the adaptation strategy enactment. For the execution of action a_2 (increase CPU P-state), for example, it is necessary to know all the states supported by the processor under consideration (parameters range) and to ensure that it is working and not idling (dependency).

Once the set of adaptation actions is created, these actions need to be properly coordinated (when there is more than one action involved) in order to compose an adaptation strategy. This is done in **Step 4**, in which input and output parameters of each action are checked in order to identify immediate sequence and parallel patterns. Based on that, the attributes `TotalDuration` and `TotalCost` can be calculated. The total duration time is particularly sensible to the adopted flow pattern, sequence or parallel, for the actions execution. If the total duration time or cost exceed their constraints, the process returns to the previous step. Otherwise, the next step (**Step 5**) sends the created adaptation action to the `adaptation parameterization` module, in which the system manager shall validate the initially suggested actions parameters.

6.2 History-based analysis

In parallel with the execution of the module `adaptation selection`, the `history-based analysis` module analyzes past executed actions (which are arranged as strategies) and their related events from time to time in order to recognize patterns. These patterns are used to validate and to modify the current version of the goal-based model by creating new impact and contribution relationships (positive or negative) or adjusting the existing relation propagation probabilities. The analysis also benefits

the `adaptation selection` and the `event identification` modules by adjusting (refining) the expected duration time attribute of the adaptation actions.

When an adaptation strategy is created to eliminate an event occurrence, the impact relations of each action, defined in the goal-based model, state their expected effects. Sometimes it may happen that unexpected effects are observed and, therefore, we need to know when these situations require model modification. As mentioned, we deal with two types of modifications in our goal-based model: Add or remove impact (event – goal) or contribution (action – event, event – event) relations; Modify satisfaction (SAT) and denial (DEN) probabilities of contribution relations.

6.2.1 Identification of action feedback

The first important issue to be solved is the identification of an action feedback, which can be: *partially/fully failed* or *successful*. An action is fully failed when it has no impact over the event while partially failed actions are characterized by a temporary action success or creation of many side-effects. Finally, an action is identified as successful when it avoids the arrival of new negative event occurrences, which represent system threats. All these scenarios were discussed in the previous chapter through the timed analysis over an incoming event occurrence. However, we also want to be able to justify all the raised events caused by the action enactment. Therefore, we propose a post-enactment rule.

The key point is to express the existing relation between two events such that the system is able to recognize when one event is caused by another through the enactment of its related adaptation action (event – action – event). Through the identification of shared variables is possible to note that an action can have different effects on events, which depend on the monotonicity of the common variable within the events. For example, the events `Server power consumption` and `CPU server utilization` share a dependency relation with the CPU clock speed frequency variable. Higher CPU frequency gives better computational performance in terms of higher throughput that reduces the CPU utilization rate if we consider the same workload. However, it makes the server to consume more power. In these cases, the mitigation of one event may cause the intensification of the other and vice-versa. This property is expressed by the following rule:

$$(e_y, \text{isCausedBy}, (a_z, \text{isEnactedBy}, e_x)) \leftarrow$$
$$((a_z, \text{reducesLikelihoodOf}, e_x) \wedge (a_z, \text{increasesLikelihoodOf}, e_y)) \wedge$$
$$(Timestamp(e_y.status = \text{'new'}) < Timestamp(e_x.status = \text{'done'}) + a_z.approvalPeriod) \tag{6.1}$$

The diminish predicate notifies a reduction in the likelihood of the event occurrences, while the intensify predicate has opposite behavior. However, the rule is applied only if the first occurrence of e_x is within a certain time slot after a_z finalization. This is depicted in Figure 6.4 and expressed by the equation $Timestamp(e_y) < Timestamp(e_x) + a_z.approvalPeriod$, which ensures that an incoming e_y appeared during a_z approval period, i.e., e_x status was set as 'done' instead of 'ready' (a_z side-effect boundary). If the post-enactment analysis is not able to explain some side-effects, the model can be improved using mining mechanisms that are described in the next subsection.

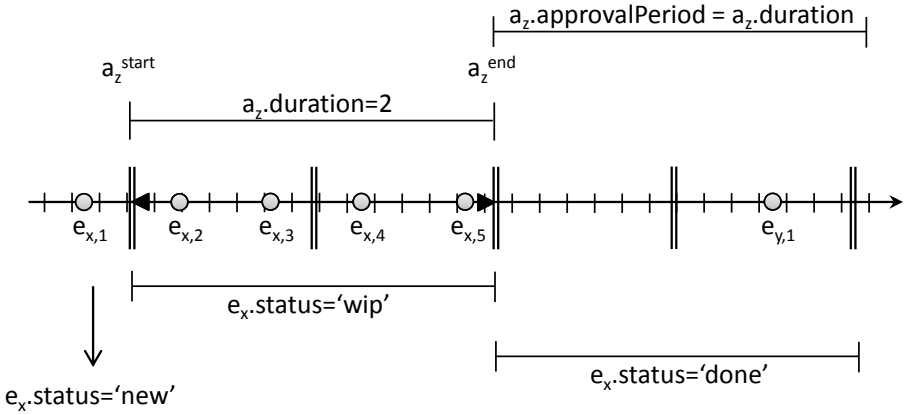


Figure 6.4: Adaptation action side-effect boundary

6.2.2 FIM to find frequent patterns

Impact relations concern negative effects of one event over one or more goals. A new impact relation event – goal ($e_x - g_p$) can be added when the appearance of an incoming event occurrence coincides with a goal fulfillment change, like from yellow to red. In some cases, it can be identified in a straightforward way looking at the event and goal shared variables. In this case the rule proposed in Eq. 6.1 is able to identify that. However, in other cases, it is not so explicit. For instance, let us consider that the event `server payload` has a negative impact relation towards

the goal `server cost`. In this example, it is reasonable to suppose that higher payload requests more power and power consumption represent cost. On the other hand, let us consider that the runtime controller allows high payload only during certain periods of the day when power costs are quite low. Thus, the negative impact may be not true while this scenario holds and it would be better to remove or change the impact relation.

Regarding contribution relations, we focus on action – event and event – event as relations that are subject to change. The other contribution relations, i.e., action – action and goal – goal, are considered static as they do not involve an event in the relation. On the other hand, action – event and event – event relations can be added, for instance, when we recognize the appearance of a new raised event e_y just after the execution of a certain adaptation strategy AS_k, which is enacted to eliminate a previous event e_x. In this way we have a complex relation like $e_x - AS_k - e_y$, which means that e_y appears due to the execution of AS_k in order to eliminate e_x. The most problematic issues in this situation are: to correlate the new event occurrence with the enactment of an adaptation strategy, to determine the single action(s) within the adaptation strategy set and to determine which relations shall be added/removed or modified, like $a_z - e_y / a_z \in AS_k$, $e_x - e_y$, or both. As mentioned, the idea of searching for indicators shared variables in order to identify relationships between two indicators can be used here as well. However, it is not enough to identify unforeseen contribution relations between model nodes when there is no shared variable, like the action `increase CPU P-states` and the event `payload`.

In order to provide a proper evolution technique for our goal-based model we search for patterns within historical data. The objective is to evolve the model with respect to the relations discussed above. To do so we use Frequent Item set Mining (FIM), which aims to find groups of items that co-occur frequently in a dataset. Such patterns are normally expressed as association rules in the format: IF [*situation 1*] THEN [*situation 2*].

A FIM algorithm requires as input the following parameters: i) the item base B set, which is the set of all items under consideration during the mining process; ii) the set of transactions T that represents some monitored changes of the item in B over a given period of time; and iii) the minimal support $\sigma_{min} \in \mathbb{R}, 0 < \sigma_{min} \leq 1$, which represents the minimal frequency pattern desired, i.e., how many times an item should appear in the transaction set to be considered frequent. Thus, the expected output of the algorithm is the set of frequent itemsets I

Table 6.1: The rules used to create our transaction T sets

	Relation	**Transaction T rule**
T_1	event – goal $(e_x - g_k)$	$changedFulfillment(g_p) \wedge (e_x.status =$ 'new' $\vee$ $e_x.status =$ 'wip')
T_2	action – event $(a_z - e_y)$	$e_y.status =$ 'new' $\wedge e_x.status \neq$ 'ready' $\wedge$ $a_z.enact =$ 'ended'
	event – event $(e_x - e_y)$	$e_y.status =$ 'new' $\wedge e_x.status \neq$ 'ready' $\wedge$ $a_z.enact \neq$ 'ended'

that can be represented by the following equation:

$$\Phi_T(\sigma_{min}) = \{I \subseteq B / \sigma_T(I) \geq \sigma_{min}\}$$

The item base B we consider involves our goal-based nodes, i.e., goals, events and actions whereas the creation of the transaction T set is driven by the rules described in Table 6.1. As we want to find patterns between raised events and goals that had their fulfillment attribute changed (event – goal, first row), and raised events, enacted actions and another raised events (event – action – event, rows two and three) we define two transaction sets. Let us consider that T_1 represents the impact relations in a given period of time (represented as time slots) and T_2 the set of event – action – event, where $T = (t_1, \ldots, t_n)$ with $\forall k, 1 \leq k \leq n : t_k \subseteq B$. Finally, σ_{min} defines the thresholds that makes this module asks for a model evolution and it is pre-defined by the miner analyst.

There are many different algorithms proposed to mine frequent item sets, such as: Apriori [3], which uses a candidate generation function that exploits the minimal support property; FP-Growth [71], which adopts a divide-and-conquer strategy and a frequent-pattern tree that eliminates the necessity of candidate generation; Tree-Projection [1], which projects the transactions onto the frequent-pattern tree in order to count the ones that have frequent itemsets; H-MINE [139], which dynamically adjusts the links within the mining process.

In our approach we adopt the algorithm proposed by Borgelt [31] called SaM (Split and Merge). The SaM algorithm is divided into two major steps, named Slip and Merge. The Split step moves all transactions that start with the same item into a new array and removes the common item. This step is recursively until it finds all itemsets the contain a split item. Then, the Merge step join the created sub-arrays using the well-known *mergesort* algorithm.

In order to analyze the model impact relations we consider only goals that have changed their status, in particular the unfulfilled ones, and

raised event that are classified as 'new' or 'wip' (working in progress) by the status attribute. Although SaM is not among the fastest approaches, due to the merge step, it uses quite simple data structures and processing schemes. This advantage is important due to memory limitations as we shall have many different mining processes, i.e., with different transaction sets, running in parallel.

6.2.3 Example

Let us consider the model depicted in Figure 5.3. As it is unlikely that the model input version represents all existing relations, new relations can be created based on the action execution outcome. Considering the incoming event occurrences described by Listing 6.1, e_2 `Power Consumption` increase has a negative impact over the goal g_4 `Energy Consumption`, represented by the relation $e_2 \xrightarrow{-D} g_4$ where the negative is due to the comparison of e_2 value e g_4 thresholds. In order to stop the arrival of new e_2 event occurrences, action a_2 `Increase CPU P-state` is enacted through the alleviation relation $a_2 \xrightarrow{-} (e_2 \xrightarrow{-D} g_4)$. However, during the action approval time, new event occurrences regarding e_3 `Server Payload` start to arrive.

Listing 6.1: Sample of incoming event occurrences of 30 time slots

```
1  e[2].status='ready'
2  e[3].status='ready'
3  ⋮
4  e[2,1]=<07.22.2012 16:02:54,435.00,0006,'I',0.4,'-'>
5  e[2].status='new'
6  ⋮
7  a[2].action.start()
8  ⋮
9  e[2].status='wip'
10 e[2,2]=<07.22.2012 16:04:24,454.00,0007,'I',0.4,'-'>
11 e[2,3]=<07.22.2012 16:06:24,427.00,0007,'I',0.4,'-'>
12 e[2,4]=<07.22.2012 16:07:54,485.00,0008,'I',0.5,'-'>
13 e[2,5]=<07.22.2012 16:09:54,455.00,0008,'I',0.4,'-'>
14 ⋮
15 a[2].action.end()
16 e[2].status='done'
17 ⋮
18 e[3,1]=<07.22.2012 16:15:24,99.00,0010,'I',0.6,'-'>
19 e[3].status='new'
```

```
20  ⋮
21  e[2].status='ready'
```

Considering that the situation above have occurred many times, the mining process applies the rules described in Table 6.1 in order to find new matching patterns. If the value of 'many' is greater than our minimal frequent pattern defined by the variable $\sigma_{min} = 0.5$, rule T_2 identifies a new relation between the enactment of action a_2 and the new event occurrences of e_3. At this point the approach is able to ensure that a_2 reduced the likelihood of e_2 (probability of event occurrences arrival), but also a_2 increased the likelihood of e_3.

In order to recognize this relation, which is not presented in the model, Eq. 6.1 is applied to ensure that event occurrences of e_3 were caused by the execution of action a_2. The Eq. 6.2 shows the new contribution relation action – event, which propagates $\{+\}$ evidence, i.e., the completion of a_2 will increase the likelihood of the arrival of e_3 event occurrences. The new version of the model is shown in Figure 6.5, in which the new identified relation is highlighted.

$$\begin{aligned}(e_3, \text{isCausedBy}, (a_2, \text{isEnactedBy}, e_2)) \leftarrow \\ ((a_2, \text{reducesLikelihoodOf}, e_2) \wedge (a_2, \text{increasesLikelihoodOf}, e_3)) \wedge \\ (Timestamp(e_3.status = \text{'new'}) < Timestamp(e_2.status = \text{'done'}) + a_2.approvalPeriod)\end{aligned} \tag{6.2}$$

6.2.4 Evolution verification

When modifications in the current version of the goal-based model are identified, either through shared variables or mining techniques, the information is sent to the `evolution verification` module. The aim of this module is to show these modifications to the system manager in a GUI way. In this stage of the model evolution an external expert validation is fundamental as special running situations of the system may lead to misleading changes that, instead of improving the current version of the model, can make it worst.

Another important point that we paid attention is regarding to two types of evolution. The system manager can apply a certain modification onto one specific model instance or the model structure source. When selecting the first option, model instance, the validated modifications do not impact on other model instances, i.e., it is kept exclusive to the current instance. Instead, if the second option is selected, the changes

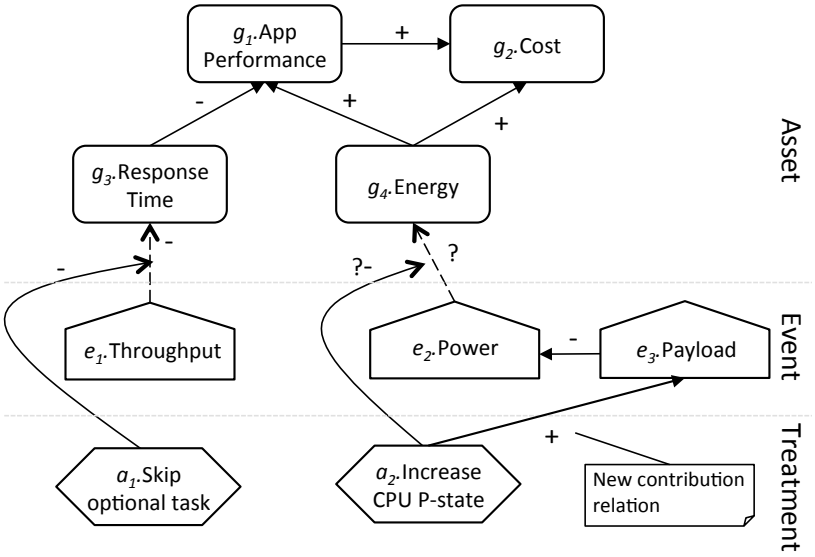


Figure 6.5: Partial model example - new relation identified

are applied to all current instances of the model. Of course a mixed approach can be used in which some modifications are permanent within the model and others are made for specific instances. In this way, every single modification has to be validated individually by the manager.

6.3 Summary

This chapter provides the proper information to complete the proposed framework description started in the previous chapter. It focuses on two main modules: `adaptation selection` and `historical-based analysis` and their extending modules, `adaptation parameterization` and `evolution verification` respectively. The adaptation selection uses both the constraint optimization approach, described in Chapter 3, as well as the adaptation selection algorithm presented in Chapter 4. Also, the model evolution techniques (through shared variables and data mining) are strongly based on the event analysis proposed in Chapter 5. Thus, this chapter concludes the description of the proposed framework that is able to recognize and react accordingly against system threats.

7 Implementation and validation

As mentioned in the beginning of this thesis, the goal is to provide mechanisms that support the identification and enactment of adaptation actions within data centers in order to maximize the trade-off between energy consumption and performance. In order to do so we need many other supporting mechanisms that shall be provided by service centers, which are considered as the future generation of current data centers. This is the reason why this thesis has a strong link with the EU project Green Active Management of Energy in IT Service centers [1] (GAMES), which took place from January 2010 to July 2012. GAMES architecture fulfills all required mechanisms that were assumed to exist throughout this thesis, like the monitoring and runtime environment modules.

In this sense, the approach proposed in this thesis makes use of many of GAMES architecture components while, at the same time, proposes new improvement techniques to optimize the following main aspects of GAMES:

- The identification of system threats that are not only based on indicators violation, but consider the different levels of violation. This provides a deeper analysis with respect to the cause of the rec-

[1] `http://www.green-datacenters.eu/`

ognized threat and, therefore, assists the selection of more effective solutions through the creation of adaptation strategies.

- The verification of high level goals is shaped from different heterogeneous indicators through normalization and aggregation formulas. These aggregation formulas allow both top-down and bottom-up dependencies analysis so that adaptation actions can be enacted towards the source of an identified threat. In addition, a graphical prototype tool is provided in order to improve the system manager experience both towards the goal-based model evolution and adaptation parametization.

- An evolutionary model that is continuously changing according to recurrent scenarios. In this way, the mining techniques described in the previous chapter are implemented over GAMES knowledge database.

Considering that, this chapter describes in more details the different components of GAMES architecture and how our approach fits among them. The first important module of GAMES used in our framework is the assessment module that supports the creation of the initial version of our goal-based model. Then, without the detailed monitoring system proposed in GAMES, the event analysis proposed in Chapter 5 would not have been possible. As we focus on the application design issues that enable the enactment of adaptation actions, the runtime controller proposed in GAMES is essential to analyze the runtime conditions and to validate which actions can be executed based on running context conditions, e.g., one server might not be available due to maintenance actions.

After positioning our presented framework within GAMES architecture, we use the obtained results from the project testbeds to fill in a simulated environment in which our approach is implemented and initial results are obtained. This is done using the Publishing News scenario proposed in Chapter 4, which is alike the eBusiness scenario proposed by GAMES.

7.1 GAMES architecture

The proposed architecture of GAMES provides the necessary mechanisms to deal with data centers energy issues at different levels simultaneously [25], which involves monitoring, analyzing, and improving energy efficiency of service centers. The sensing and monitoring infrastructure

is in charge of gathering the data produced by the sensor network installed on the data center. These sensors get information not only about the energy consumption of IT devices, but also about the performance and the usage of them. All this data enables the identification of event occurrences and the calculation of defined indicators. Querying GAMES knowledge base repository we can obtain information such as the number of servers that are running or the configurations of the virtual machines installed on a given server. This repository also keeps historical data, which are used by the proposed approach to validate the outcome of adaptation actions in specific contexts. GAMES architecture is composed by three main modules:

- DTE (Design-Time Environment): In this module, assessment and mining techniques are used to identify critical situations. It supports SBA and IT infrastructure co-design when more pervasive and long-term adaptations are necessary. This module also defines the indicators (KPIs and GPIs) that shall drive data center adaptations at both design-time and runtime. The output of this module is a set of optimized parameters and configuration files to be followed by the runtime environment.

- RTE (Run-Time Environment): This module is able to detect the occurrence of critical situations and, if required, automatically adapt the execution of the applications running on the data center. Such adaptations are short-term, such as CPU P-state adjustment. To do so, it combines the DTE output with the current context information provided by the sensing and monitoring system.

- ESMI (Energy Sensing and Monitoring Infrastructure): This module is responsible for collecting, parsing and storing the provided the sensors placed within the data center. Figure 7.1 depicts the existing sensors used to collect data about the environment at runtime and their hierarchy as a UML class diagram. Information about energy, such as CPU power consumption, and quality, such as number of web service transactions per second, are collected by physical and logical sensors that sense the infrastructure and middleware layers respectively. The collected data is parsed such that relevant information can be extract. This is used to calculate the defined indicators (GPIs and KPIs) and to provide needed context information for decision-making analysis, which is done through data mining tools. A fundamental element of the ESMI is the EPKB (Energy Practice Knowledge Base) where the data center

current and historical information about indicators and context, including configuration parameters, is stored. Querying the EPKB, for instance, it is possible to obtain the current number of servers that are running or the configurations of the virtual machines installed on a given server within a specific time interval or historical indicators values in order to perform mining pattern recognition techniques.

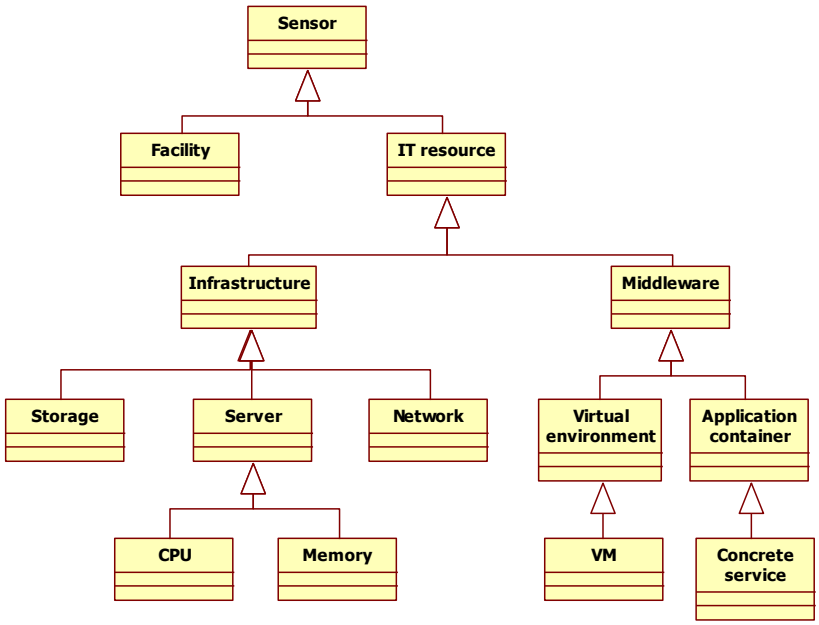


Figure 7.1: GAMES sensors within ESMI module

7.1.1 Methodology

Considering the described architecture, a six-step methodology is proposed in GAMES in order to assess, analyze and improve the greenness of the data center. Based on energy-aware adaptation, this methodology grants an energy efficiency SBA execution. It is composed of the following phases:

1. **Preliminary assessment**: Analysis of the current situation of the data center with respect to the facilities, the hardware and software installed, and the energy consumption. After this step, the data center manager is able to identify which is the level of maturity of the data center.

2. **General policies definition**: Based on the preliminary assessment, in this step the data center administrator shall define the goals to be achieved in order to increase the level of maturity. This step is usually performed at the strategic level, so the indicators adopted to identify the desired goals are coarse-grained.

3. **GPI/KPI definition**: Starting from the strategic goals previously defined, in this step constraints on measurable indicators are defined. Relevant indicators for the assessment of the energy consumed within a data center are related to the different components of the data center (e.g., storage, servers, applications) and can be defined at different granularities (e.g, single machine vs. the whole set of servers).

4. **Annotation**: This step focuses on the applications and enhances the usual design steps by requiring that each application is enriched with annotations useful for evaluating the power consumption, such as process relevant application data, temporal constraints and resources requirements. In details, annotations concern: (i) information regarding the business process control flow and thus the execution of certain activities in a process, (ii) energy and performance constraints within process flows, (iii) information regarding the used resources when executing a certain activity and (iv) information regarding the data used throughout a process.

5. **Execution**: In the data center the applications executions are monitored. The monitoring infrastructure collects information on the energy consumption and the resource usages.

6. **Validation**: The monitored data are used to calculate the current values for the relevant GPIs and KPIs. The obtained values are compared to the goals defined at the beginning in order to state if the goals are fulfilled. If not so, suitable adaptation strategies have to be selected and can be enacted at both design and run time.

The described methodology and its supporting architecture are used by our proposed framework in order to: (i) Identify relevant indicators and context environment definition; (ii) Monitor the runtime environment through ESMI sensors; (iii) Calculate indicators based on defined policies; (iv) Execute and track the selected adaptation strategies; (v) Notify enacted adaptation strategies for future analysis. The first item is performed within the DTE module, in which the preliminary assessment supports the definition of general policies and indicators. The second

and third items are performed inside ESMI module, which sets the monitored variable values. Finally, the last two items are performed within RTE module. Although GAMES methodology provides its own adaptation selection mechanisms, this thesis extends its functionalities by the following elements: (i) Spanned indicators relationships, normalization and aggregation; (ii) Timed-based event analysis, which provides different levels of indicators risks; (iii) An evolutionary mechanism that fits the proposed approach within different scenarios and identifies unforeseen issues. Figure 7.2 shows how GAMES modules interact with our proposed ones (highlighted in bold) in an interactive manner.

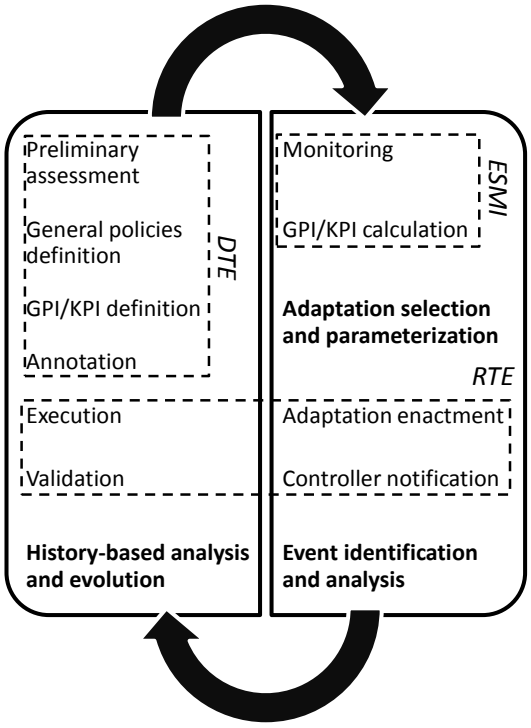


Figure 7.2: GAMES modules - methodology integrated with energy-aware controller

7.1.2 Testbed

The methodology proposed in GAMES is validated through experiments considering two scenarios: High Performance Computing (HPC) and Cloud Computing that are performed into two different testbeds physi-

cally separated. The testbed at HLRS [2] (Stuttgart, Germany) provides experimental values for HPC applications and a cloud-based eBusiness scenario.

The installed hardware at HLRS facility consists in one RECS [3] Cluster System that is composed by 18 server nodes Intel P8400 CPU (2x 2.26 GHz, 1066 MHz FSB) and 4 GB DDR3 dual channel RAM each. The cluster brings its own internal sensors for power and temperature monitoring, in which it is possible to gather information of each single node independently. Table 7.1 lists all sensors installed in this testbed, which are properly integrated by NAGIOS [4], which is a well-know monitoring tool designed for IT service centers and able to gather data from infrastructure and middleware layers. Further details about the testbed deployed hardware can be found in GAMES deliverables [105, 164].

Table 7.1: Data gathered from the testbed sensors

Sensor layer	**Sensor obtained data**
Infrastructure	Temperature of the Mainboards
	Clock Speed of the CPUs
	Temperature of the CPUs
	Payload of the CPUs
	Amount of used Memory
	Temperature of installed Disks within the Storage
	Energy consumption of the Storage Device
	Total amount of free space of each tier
	Fan speed of several system- or CPU-fans
Middleware	Power consumption for each virtual machine
	Power consumption for each virtual disc
	Amount of virtual machine used
	Amount of virtual disk used
	Processes deployed
	Processes running
	Processes failed

Note that the data is regarding only one cluster and, therefore, we cannot calculate indicators that are regarding the entire facility, such as PUE (power usage effectiveness) and DCiE (data center infrastructure efficiency). Also, we do not consider all monitored infrastructure provided by GAMES, instead we focus on a sub-set of indicators consid-

[2] `http://www.hlrs.de`
[3] `http://www.christmann-technologies.com/products/show/44`
[4] `http://www.nagios.org/`

ered relevant for our purposes. This set includes: power consumption, CPU usage, Mhz/watt and memory usage at the infrastructure level and number of VMs, process running and failed at the middleware level. At the application level we take into consideration application performance (i.e., transactions/Wh), energy consumption, price and execution time quality dimensions as indicators.

7.2 Implementation

This section provides details about the implementation of our BP example mentioned in Chapter 4 and the main modules of our proposed framework. These implemented components were experimented using as input the monitored data obtained from GAMES testbed at HLRS.

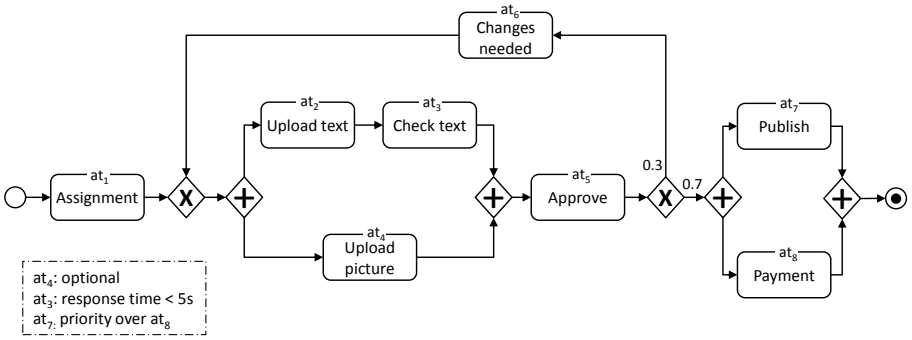
7.2.1 Business process specification

The proposed running BP example is based on the process used in Chapter 4, News Publishing, to demonstrate how the proposed 3-layers model fits with SOA applications. Figure 7.3 depicts the example in standard BPMN. The first abstract task at_1 is used by the editor to assign one correspondent and one photographer, when required, to cover a specific event. The correspondent shall write about it and upload the text (at_2), which will be revised by an automatic system review (at_3). In parallel the photographer shall upload the set of pictures with respect to the covered event (at_4). When both text and pictures (when required) are available the editor approve or reject the article (at_5). We considered 30% of probability of rejection, in which the editor asks for changes in the article (at_6). When the article is approve it shall be published in company's website (at_7) and both correspondent and photographer (when required) shall get paid (at_8).

In addition, the described system follows five main business rules, that are: (1) an article might not have pictures if the subject does not require it (e.g., political decisions made through written documents) or there was not a good picture to be published; (2) the automatic text verification (at_3) cannot take more than 5 second as the correspondent user is waiting to send it to the editor; (3) the editor can ask for changes only once [5]; (4) the action publishing task (at_7) has higher importance over the correspondent and photographer payment (at_8) as the news has a strictly tight expiration time; and (5) the complete payment is managed by an external sub-system.

[5] We do not consider the option of withdrawing the article after its review.

Figure 7.3: News Publishing example - BPMN



The proposed BP has two possible execution paths ep, which their respective probabilities of execution. The calculation of the considered quality dimensions (execution time, price and energy consumption) for each ep follows the approach described in Table 3.2 from Chapter 3. We define ep_1 as the most probable one (70%), which represents the article approval. The second execution path, ep_2, the article is rejected and new improved version is requested. Both execution paths are described below, in which $\langle\ \rangle$ indicates sequential execution, $\{\ \}$ parallel, and $\widehat{\ }$ optional task. In order to provide a more clear representation, ep_{sub} represents the execution of the abstract tasks at_2, at_3, at_4, at_5.

$$ep_{sub} = \{\langle at_2, at_3\rangle, \widehat{at_4}\}, at_5$$

$$ep_1 = \langle at_1, ep_{sub}, \{at_7, at_8\}\rangle$$

$$ep_2 = \langle at_1, ep_{sub}, at_6, ep_{sub}, \{at_7, at_8\}\rangle$$

We consider that for each abstract task at_x there is at least one concrete service cs_y that can be used to implement the expected functional requirements. The selection of concrete services that attend the minimum functionalities is the selection criteria. After receiving the customer request, all available service providers that are able to functionally satisfy, at least partially, the request requirements are listed. Several approaches deal with this phase [141, 72] and it is not our focus. The second selection criteria, called *service selection phase*, relies on the non-functional requirements, which may vary significantly. It consists in ranking and choosing the best concrete service for each abstract task according to global and local constraints, which are represented by our defined indicators thresholds. Such properties, which in this example

Table 7.2: Candidate concrete services and their qualities dimensions

Abstract task	Concrete service	Execution time	Price	Energy cons.
at_1	cs_{11}	1.30	1.2	65
at_2, at_4	cs_{21}	2.00	5.9	62
	cs_{22}	1.70	4.8	50
at_3	$\langle cs_{31}, cs_{41} \rangle$	2.20	5.2	65
	$\langle cs_{32}, cs_{42} \rangle$	2.50	2.8	55
	$\langle cs_{33}, cs_{43} \rangle$	1.60	3.4	51
	$\langle cs_{34}, cs_{44} \rangle$	1.50	3.3	72
at_5	cs_{51}	2.70	1.1	64
at_6	cs_{61}	0.80	0.5	48
at_7	cs_{71}	1.50	4.0	49
	cs_{72}	1.00	4.4	55
	cs_{73}	1.70	4.3	59
at_8	cs_{81}	7.50	2.0	64
	cs_{82}	5.50	3.0	70

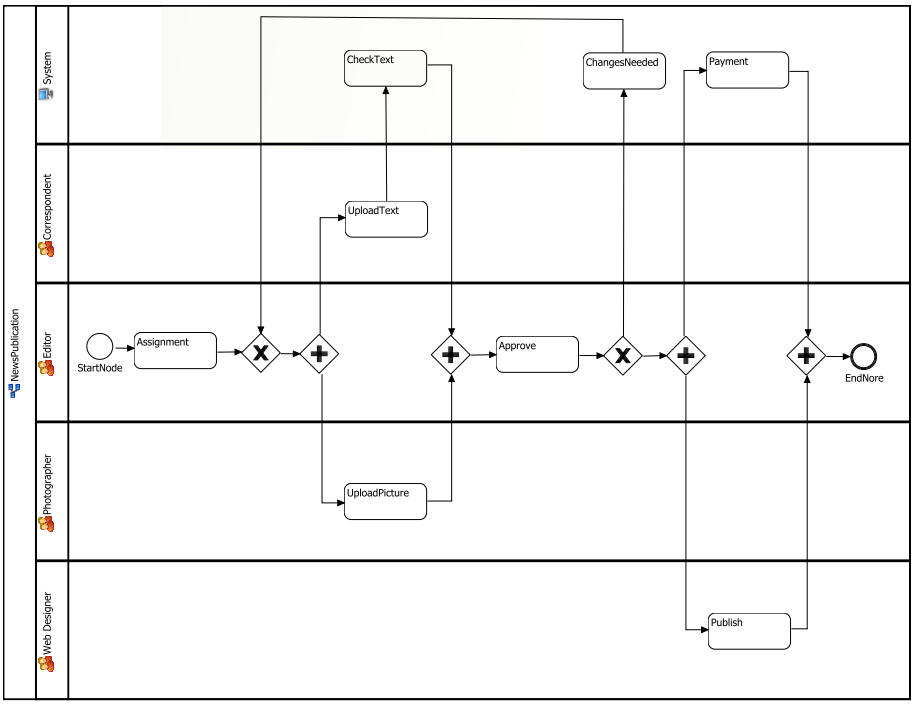
include response time, price and energy consumption, are agreed within the Service Level Agreement (SLA). As described in Chapter 3 this phase is formulated as Constraint Satisfaction Optimization Problem (CSOP).

Table 7.2 presents all concrete services candidates for our example during the service selection according to non-functional constraints (we assume that all listed services are functional equivalent and fulfill minimum functional requirements). It is worth pointing out that the abstract tasks at_2 and at_4 are functionally equivalent (file upload) and at_3 is composed by two concrete services (spell check and grammar check).

In order to implement and run our New Publishing BP we have used Together Workflow Editor [6] (TWE), which is an Open Source graphical editor to create and manage WfMC XPDL [7] process definition files. Figure 7.4 shows the TWE graphical interface used to implement our running example that was deployed and executed within Together Workflow Server [8] (TWS) workflow engine. Monitored data is used as input in our simulated environment described in the following.

[6] `http://www.together.at/prod/workflow/twe`
[7] `http://www.wfmc.org/xpdl.html`
[8] `http://www.together.at/prod/workflow/tws`

Figure 7.4: News Publishing example - TWE



7.2.2 Event analysis implementation

As we described in Chapter 5, the `monitoring` module keeps sending monitored data both to the `indicator calculation` and `event identification` modules. Monitored variable values as well as indicators values are provided by GAMES components, which keep sending during specific time intervals. Thus, our framework has to extract meaningful events from this bunch of data that represent system threats. To do so, the data is interpreted as continuous streams that are used to keep materialized view of RDF triples. As already discussed, an important implementation issue we had to deal with is the timed-stamped characteristic that incoming streams hold. This issue is solved by using Stream Reasoning techniques proposed by [17] with a special type of RDF triple that represents the different time slots.

The implementation of our timed stream reasoner was made using Jena framework [9], which is a well-known Java API to deal with semantic web applications and, in particular, with RDF and OWL. The reasoner is

[9] `http://jena.sourceforge.net/`

divided into two different knowledge types: static and dynamic. The static knowledge is represented by the goal-based module and specific technical information about the components within our 3-layers model. At the infrastructure layer, this information is regarding to the maximum frequency the processor can operate and the capacity of the storage arrays available, as shown in Listing 7.2. At the middleware layer, it is concerned to the VMs configuration parameters that cannot be changed at runtime. Finally, at the application layer it represents the annotated information associated with the BP.

Listing 7.2: Static knowledge sample

```
<rdf:RDF
   xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
   xmlns:dt='http://localhost/DC/'>

   <rdf:Description rdf:about='http://localhost/DC/clusterRECS'>
      <dt:hasCPU rdf:resource="http://localhost/DC/node01"/>
      <dt:hasCPU rdf:resource="http://localhost/DC/node02"/>
      ⋮
      <dt:hasCPU rdf:resource="http://localhost/DC/node18"/>
   </rdf:Description>
   <rdf:Description rdf:about='http://localhost/DC/node01'>
      <dt:type>Intel</dt:type>
      <dt:id>P8400</dt:id>
      <dt:nCores>2</dt:nCores>
      <dt:maxClock>2260</dt:maxClock>
      <dt:minClock>800</dt:minClock>
   </rdf:Description>
   ⋮
</rdf:RDF>
```

The dynamic knowledge refers to the materialized RDF views created initially from the static knowledge and then updated based on the incoming streams. All this information is stored in memory and represents the goal-model instance with all triples set as infinite expiration time. We represent it by the predicate T. As the window slides over the stream, the incremental maintainer:

1. Puts all incoming triples entering the window in a new predicate called T^{in};
2. Loads the current materialization and T^{in}, which represent the incoming triples;

3. Computes the production rules in order to identify the three different levels of threats;
4. Searches for expired triples;
5. Defines the set of triples to be added T^+ and removed T^- from the materialization;
6. Updates the RDF materialization according to T^+ and T^-;
7. Updates the time-stamped triples according to T^+ and T^-.

Jena2 inference subsystem

One of the biggest advantages of representing pieces of knowledge as RDF triples is the possibility to apply predefined inferences rules over it. Such rules will lead to new derived pieces of knowledge which combines two or more existing triples in an automated manner. To do so, many free tools can be easily found over the web. Example of these tools are MaRVIN [9], Sesame [34] and Jena [150].

Listing 7.3: Jena example - creating the RDF materialization and inference models

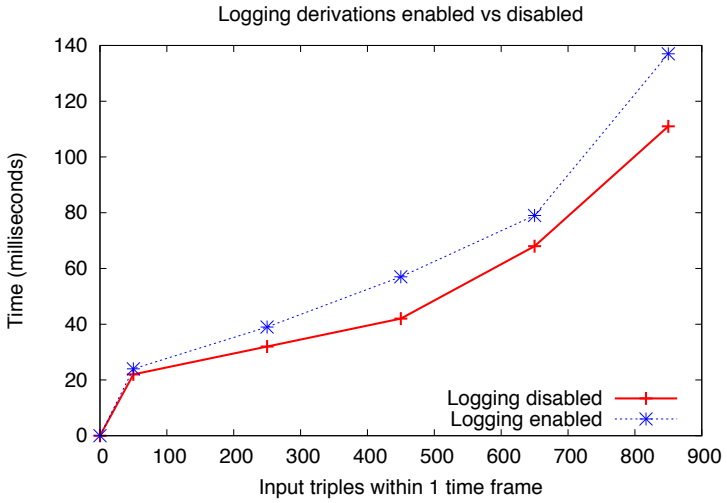
```
1  // Create a model representing the family
2  Model myRDFmodel = ModelFactory.createDefaultModel();
3
4  // Create a new generic rule reasoner to support user defined rules
5  Reasoner reasoner = new GenericRuleReasoner(Rule.parseRules(ruleSrc));
6  reasoner.setDerivationLogging(true);
7
8  // Create a new inference model over myRDFmodel using the reasoner above
9  InfModel inf = ModelFactory.createInfModel(reasoner, myRDFmodel);
10
11 // Create the RDF stream generator based on GAMES testbeds
12 RDFStreamsGenerator generator = new RDFStreamsGenerator(inf);
13 generator.start();
```

MaRVIN states for Massive RDF Versatile Inference Network which performs RDFs inference through a parallel and distributed platform. Its main advantage is regarding to the arbitrary scalability which computes the materialized closure gradually. Such approach releases the users to wait for the full closure to be computed. In Sesame the inference engine is within the *Storage And Inference Layer* (SAIL) which is completely independent and makes possible to implement Sesame on

top of a number of repositories without changing any other component. At the semantic level, the *RQL query engine* used in SAIL layer infer new statements using queries that distinguish between schema and data information. However, the most suitable for our purposes is Jena toolkit inference engine, which provides better flexibility and allows several reasoners to be plugged in it, including a generic rule engine that allows many customization of RDF processing and transformation. Listing 7.3 depicts a piece of Java code in which an empty RDF model and inference model are created together with a new generic rule reasoner. Note that the variable `ruleSRC` contains all predefined inference rules to identify the different threat levels.

Another important feature of Jena inference subsystem is its traceability function. Using a simple method – `InfModel.getDerivation(Statement)` – is possible to trace how an inferred statement was generated, which is one of the key concepts behind the proposed solution. To enable this functionality we just need to start logging all derivations, as it can be seen at line 6 in Listing 7.3. This functionality slightly increases the derivation process execution time as depicted in Figure 7.5, but does not compromise the proposed system.

Figure 7.5: The increased execution time of logging derivation function



Threat levels deduction

In order to create an event occurrence, the `event identification` module identifies three different levels of threat, in which only the third level shall actually trigger the creation of an event occurrence. As explained

in Chapter 5, the first level identifies indicators violation, warning or alarming, based on their calculated value and defined thresholds. These indicators are classified as 'yellow' or 'red', respectively. As not all violations lead to an event occurrence, the second level identifies the indicators that are current violating warning thresholds, but are also inclined to violate alarming thresholds. Indicators in this situation are classified as 'orange'. Finally, the third level represents indicators that are either classified as 'red' or 'orange' for a longer period than the value defined in the indicator acceptance attribute. In this case, they represent threats to the system and event occurrences shall be triggered.

The identification of all three levels of threats is implemented through deduction rules, in which the stream reasoner analyze the incoming stream and creates new RDF triples in the materialized view. Based on a set of rules, the mechanism determines the different levels of threat which each monitored indicator value represents to the system. Four inference rules are used for that as depicted in Listing 7.4. The first two rules, rule_1a and rule_1b, define if there is a warning or alarming indicator violation, respectively (threat level 1). Note that the variables $?Y$ and $?R$ represent the possible set of warning and alarming values of indicator $?a$, as defined in Eq. 5.1 from Chapter 5. The third rule, rule_2, recognizes indicators that shall be classified as 'orange' (threat level 2). The last rule, rule_3, infers the ones that shall trigger an event occurrence, thus representing a system threat to be eliminated by adaptation actions (threat level 3).

Listing 7.4: Jena inference rules for threat levels identification

```
String ruleSrc = ‘‘
[rule_1a: (?a http://localhost/EI/hasValue ?b), (?b http://localhost/EI/isIn ?c), (?c rdfs:range ?Y),
          uriConcat(?a, ‘$hasValue$’, ?b, ?d), (?d http://localhost/EI/hasTime ?time),
          uriConcat(?a, ‘$hasStatus$Yellow’, ?e)
          -> (?a http://localhost/EI/hasStatus ‘Yellow’), (?e http://localhost/EI/hasTime ?time)]

[rule_1b: (?a http://localhost/EI/hasValue ?b), (?b http://localhost/EI/isIn ?c), (?c rdfs:range ?R),
          uriConcat(?a, ‘$hasValue$’, ?b, ?d), (?d http://localhost/EI/hasTime ?time),
          uriConcat(?a, ‘$hasStatus$Red’, ?e)
          -> (?a http://localhost/EI/hasStatus ‘Red’), (?e http://localhost/EI/hasTime ?time)]

[rule_2:  (?a http://localhost/EI/hasValue ?b), (?b http://localhost/EI/isIn ?c), (?c rdfs:range ?Y),
          (?b http://localhost/EI/isIn ?d), (?d rdfs:range ?R’),
          uriConcat(?a, ‘$hasValue$’, ?b, ?e), (?e http://localhost/EI/hasTime ?time),
          uriConcat(?a, ‘$hasStatus$Orange’, ?f)
          -> (?a http://localhost/EI/hasStatus ‘Orange’), (?f http://localhost/EI/hasTime ?time)]

[rule_3:  (?a http://localhost/EI/isIn ?b), (?b rdfs:range ?V),
          (?c rdf:first sortByTimeAsc(?V)), (?d rdf:first sortByTimeInv(?V)),
          (?a http://localhost/EI/hasAcceptance ?e), diff(?c ?d ?g), greaterThan(?g ?e)
          uriConcat(?a, ‘$isIn$’, ?b, ?g), (?g http://localhost/EI/hasTime ?time),
          uriConcat(?a, ‘$triggerEventOccurr$’, ?h)
          -> (?a http://localhost/EI/triggerEventOccurr TRUE), (?h http://localhost/EI/hasTime ?time)]’’;
```

7.2.3 A prototype dashboard tool

Figure 7.6 depicts the indicators normalization and aggregation implemented within a dashboard prototype where the information is based on measured indicators from GAMES testbeds presented above. We focus on two specialized GIFs [10]: server utilization and server power, which are under `IT resource usage` and `energy impact` main GIFs hierarchy. In order to better represent these indicators, they are subdivided into high and low levels, which contain respective application/middleware and infrastructure level indicators. The aim of the proposed tool is to perform the indicators normalization functions and aggregation metrics in order to come up with high-level GIFs modeled as high-level goals in our goal-based model.

The upper-left corner of the figure depicts the aggregation regarding to all 18 nodes of the testbed described in Section 7.1. Each server node represents an aggregated value of all servers usage related indicators. The red area size means how far from the optimal situation the aggregated value is, which is obtained by calculating $1-\text{GIF}$. The bottom-left corner shows each selected indicator for the chosen GIF, that are `CPU usage` and `memory usage`. Note that 65 and 31 represent the indicator measured values, while 0.71 is the aggregated value of both indicators with respect to server node #11. In this example, the second weight of the indicator `CPU usage`, in the composed weighting system, is higher than `memory usage` indicator as it has higher importance in this testbed scenario. Finally, the right-side of the figure depicts four indicators values and their thresholds at server level regarding both usage (memory and CPU usage) and power (application performance and power consumption) that were introduced in Chapter 3.

The power consumption indicator (427 Watts) is the average value of one month measurement of the entire cluster and 30,538 tuples of data, of which 51 represent alarming violation and 26,615 warning violation (i.e., measured values higher than 400 Watts). Thus, if we want to calculate the aggregated value of the entire period interval, the system divides it into two steps. The first calculates the aggregation value of violated indicators separated from normal values and the second aggregates all of them within a single value. In this way, the weights used during the normalization phase are different for normal and violated indicators, representing different impacts in the threat identification module.

[10] Green Index Function, defined in Chapter 3

Figure 7.6: Dashboard tool - GIFs and indicators screen-shots

7.3 The impact on events and actions

Based on some GAMES modules, such as the monitoring system, and the testbed data obtained by Nagios we configure and run our framework modules within a controlled environment. As said before, we do not intend to replace any component of GAMES, but instead, provide additional mechanisms that can be used to improve the existing ones.

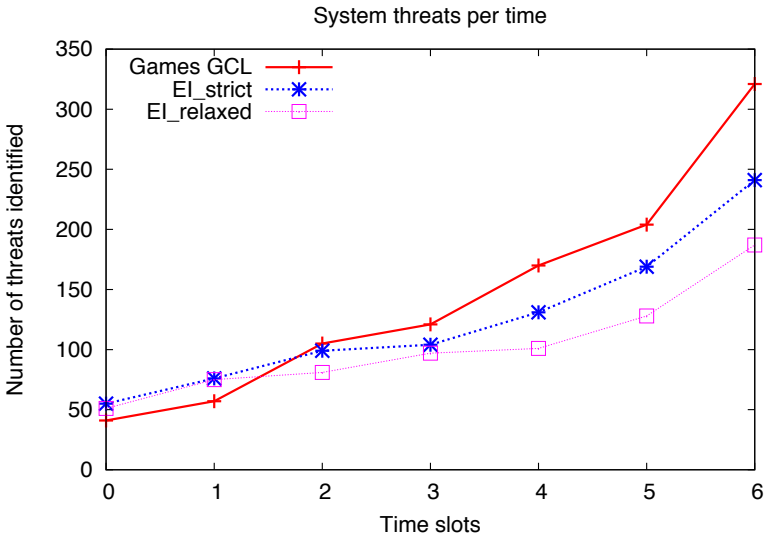
Looking at GAMES data we observed that the proposed controllers (Local Control Loop LCL for decisions at the server level and Global Control Loop GCL for decisions at the entire facility) do not have sophisticated mechanisms to identify system threats that require the enactment of adaptation actions. The aim of our proposed event identification and analysis modules is to reduce the number of adaptation actions execution by creating three levels of system threats, in which not all of them trigger adaptation. In this way, we aim to narrow the set of violated indicators that require adaptation without harming the overall system.

Analyzing the many database tables produced by Nagios we were able to reproduce situations that were responsible for triggering adaptation actions, represented by indicators violations. This was possible only because all experiments were performed three times: without games methodology intervention and two types of controllers approach: fuzzy and bio-inspired [25].

The graph shown in Figure 7.7 depicts the accumulated number of raised events that were treated as system threats (i.e., adaptation actions were triggered) within a period of 6 time slots of monitored data. It is possible to notice that GAMES GLC [11] always identified more

[11] Gobal Loop Controller.

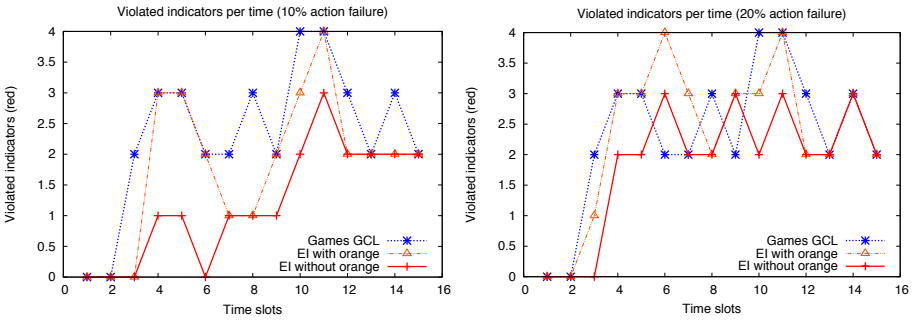
Figure 7.7: Graph identifying the number of threats before action enactment



threats than our Event Identification (EI) module from time slot 2 to 6. This is done using our proposed indicator flexibility, indicated by warning thresholds. The `EI_relaxed` represents a wider range for warning thresholds, i.e., varying the parameter `a` indicated in the equations presented in Chapter 3. Moreover, the indicator acceptance attribute and the three levels of threats avoid the triggering of unnecessary adaptation. For example, design-time actions like SLA renegotiation [12] may cause temporary quality degradation like availability violation. Therefore, if the SLA renegotiation action lasts as expected, the availability violation should not trigger any other adaptation. It is worth to mention that at time slot 6 the three scenarios (`GAMES GLC`, `EI_strict` and `EI_relaxed`) represent similar levels of goals satisfaction, although the first two have higher expectations with respect to the third one.

The proposed approach is also reflected in the number of violated indicators. The graphs depicted in Figure 7.8 compare the number of indicators alarming violation, i.e., 'red' status, in three scenarios: i) `GAMES GLC` that was executed following GAMES controllers in a real testbed at HLRS facility; ii) `EI with orange` that was executed in a simulated environment but with GAMES sensors data as initial input and takes into account the indicators classified as 'orange' status together with the 'red'

[12]The complete list of design-time adaptation actions is presented in Table 4.1 of Chapter 4.



(a) Violated indicators with 10% of action failure

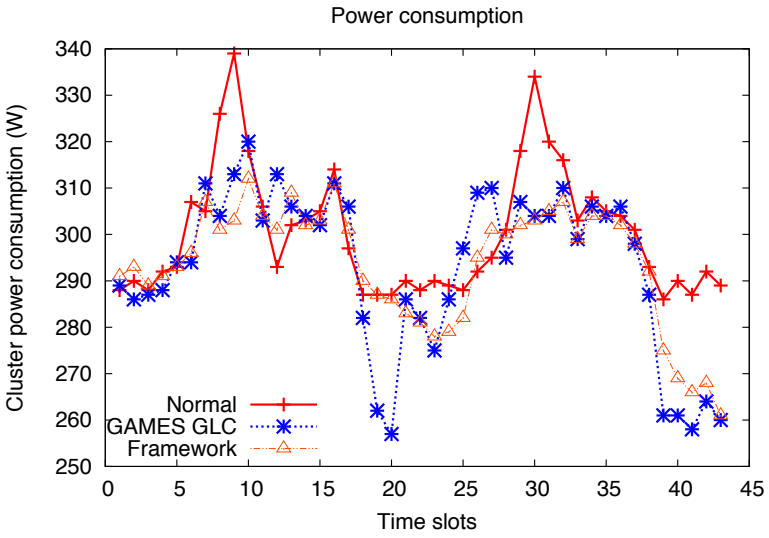
(b) Violated indicators with 20% of action failure

Figure 7.8: Number of violated indicators

ones; and iii) `EI without orange` that was also executed in a simulated environment, but did not consider 'orange' indicators as eminent system threats. The simulation was conducted using an expected adaptation action failure rate, in order to fairly compare our simulated experiments against GAMES testbed experiments. Figure 7.8a represent 10% of actions failure, while Figure 7.8b 20%. Our proposed approach reaches the desired levels of goals satisfaction considering a fewer number of violated indicators. Differently from the previous figure, in these figures we do not consider the number of system threats, but the number of violated indicators that can or cannot represent a system threat in our approach. Instead, in GAMES all indicators violation are considered as threats and require adaptation actions. In almost all time slots (in both graphs) our scenarios have less alarming violated indicators. It does not mean that our enacted actions got better results since we considered the same set of actions, but that warning violation are not seen as threats. Although the second scenario, `EI with orange`, has to deal with more violations, we argue that 'orange' indicators have to be considered as 'red' in this phase as they are very likely to become 'red'.

The graph depicted in Figure 7.9 shows the average power consumption of the entire cluster (i.e., the 18 server nodes) during a period of 43 time slots, which represent part of the execution of the BP. The first important improvement that both GAMES and our proposed framework provide is the elimination of two peaks of power consumption (time slots 9 and 30). Then, we can see that from slots 7 to 13 and from 24 to 32 our framework slightly overtakes GAMES controller by consuming less power, making clear the advantage of dealing with less raised events and, therefore, less adaptation actions. As adaptation actions are intrinsic re-

Figure 7.9: Cluster average power consumption during 43 time slots

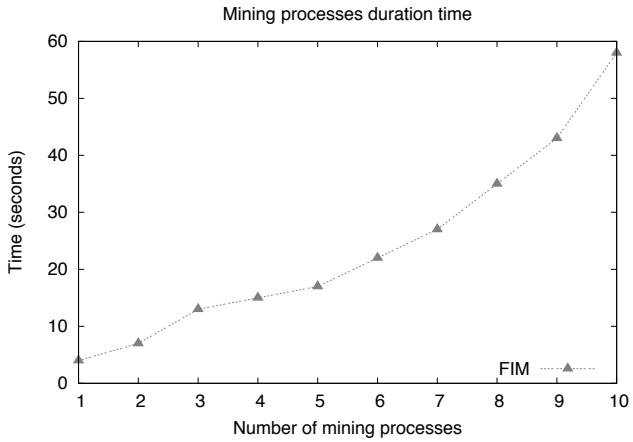


lated to other actions and events, it is unlikely that the enactment of an adaptation action does not produce any side-effect and, by doing so, the system can enter into an infinite cycle. The presented approach aims to reduce the number of raised events and enacted actions in order to increase the probability of overall success.

Regarding the goal-based model evolution mechanism, Figure 7.10 aims to demonstrate that the used mining technique can be performed within reasonable time – less than 1 minute. The experiment was set using one month of raw data obtained from GAMES testbed, which was used as input together with simulated data. An initial goal-based model was set, in which some relations were intentionally missing in order to ensure that the mining process (FIM algorithm) was able to identify all of them. There were 4 missing relations and after the execution of 10 mining processes, they were recognized [13] by the system and suggested to be added in the model. It is worth to mention that all modifications are presented to the system manager who shall approve or not them. As the mining processes execution time is an important issue at this phase, the presented graph shows that it increases significantly as the historical log table stores more tuples. Thus, in order to avoid unacceptable duration time, tuples have an expiration time defined by the system manager. In this way we limit the mining scope according desired preferences.

[13] An example of relation identification by the mining process is described in Chapter 6.

Figure 7.10: Mining process (FIM) execution time



7.4 Summary

In this section we explained in details how our proposed framework fits within GAMES architecture, in particular with the monitoring system. Moreover, we highlight how our framework can contribute to improve system threats identification, adaptation selection based on feedback loop and impact propagation, and the creation of high level system goals based on indicators normalization and aggregation. Thus, a brief overview about GAMES architecture was necessary to make these points clarified.

Using the project testbed monitored data we were able to apply our event identification and analysis modules based on an initial goal-based model version, which was taken from GAMES. Results were compared against GAMES LCL with respect the number of identified system threats (i.e. events that triggered adaptation mechanisms) and the number of indicators within alarming thresholds. Our proposed approach was able to keep less violated indicators based on a fewer number of raised events. This was possible due to our refined mechanisms that recognize different types of system threats and, therefore, the adaptation selection can focus on the most significant ones.

The FIM mining technique used to support our goal-based model evolution also played an important rule to refine the event identification and adaptation selection modules. Due the identification of unforeseen impact relations, the adaptation selection module could reduce the appearance of new indicators violations as a side-effect of the adaption action execution.

8 Concluding remarks

Though the research on green computing can follow several directions, this thesis aimed to cover service oriented aspects considering both the design and the execution of service-based applications. In this chapter we summarize the research work and provide future research directions.

8.1 Summary

In order to consider the energy aspects of a service-based application, we first propose some metrics in order to extract the application main characteristics in a quantitative manner, which are used to support the calculation of green performance indicators. For this matter, detailed information about these indicators and their dependencies are presented. Due the indicators heterogeneity, we propose the creation of Green Index Functions, which aggregate related indicators within one single value. The approach is divided into two phases: normalization and aggregation.

Approaching the problem to find the best trade-off between performance and energy consumption, we present a novel energy-aware and quality-based technique in order to solve the service composition problem, which takes into account non-functional characteristics through a global approach. Hence, a new energy efficiency metric for a single ser-

vice is introduced, which maps directly the relationship between energy consumption and execution time. The energy dimension requires considering novel aspects for service quality evaluation. In particular, the proposed method considers soft constraints, nonlinear relationships among quality dimensions, and ranges for quality values.

On the basis of the actual use of the resources during the application execution, a way to improve energy efficiency of the process is proposed. Our approach enables the identification of energy leakages and/or of indicators violations and the selection of suitable adaptation actions that can be applied to the process and its virtual execution environment in order to maximize the use of the resources and minimize the power consumption. We have demonstrated how the service-based business process co-design can help the identification of non-optimal situations and support the system adaptation in selecting and executing actions at runtime aligned with design-time preferences.

In order to support the adaptation action selection, we propose a framework that is based on a goal-based model to analyze the actions impact propagation throughout the system model. The novelty relies on the model external components, which supports the identification of system threats based on pattern recognition and context evaluation. Data-stream reasoning mechanisms are used to evaluate diverse scenarios and to identify relationships among system threats and enacted actions at runtime. In parallel, mining techniques are used to ensure that the considered goal-based model instance is adequate with the underlying system environment by evolving the model elements (like unforeseen relationships) according to monitored data. Thus, it is possible to have different instances of the same model that fit within different environmental configurations.

This framework is integrated within GAMES methodology, which provides the surrounding elements, such as monitoring system, to enable execution of the proposed approach. The GAMES project provides real monitored data from its cloud computing testbed scenario. This information is used as input data into our simulated experiments in order to make them as close as possible to real service centers.

8.2 Future directions

The research work presented by this thesis has provided a set of solutions to the problem under investigation. But still unsolved issues delineate many directions towards the extension of this thesis. This section aims to unveil some of these directions.

1. **Green performance indicators:** In this thesis we formulate a green performance indicator called SBA-EEI, however there is a big gap of indicators at the application layer. In particular, set of indicators that correlate the application development phase (coding) and its execution environment. Software engineering metrics that focus on the application life-cycle could take into consideration the application energy aspects. This research line would improve the application energy models and provide more accurate resource allocation algorithms.

2. **Application energy elasticity:** In Chapter 4 we describe an initial set of energy-aware adaptation actions which can be extended with new technologies both at middleware and physical infrastructure layers. These new technologies enable non-existent ways to reduce power consumption through dynamic and autonomous adaptation, in which hardware and software are able to clearly communicate to each other in order to express their current needs and capabilities.

3. **Cloud federation:** The solution proposed in this thesis is towards one single service center. Extending it to federated data centers new issues arise and the complexity of managing desired goals levels become an even more critical aspect. We believe that our proposed framework could abstract the underlying environment context, however, the goal-based model and the adaptation selection mechanisms have to be redesigned as the importance of communication turns up, for instance. Existing research towards this directions are presented in ECO_2Clouds [1] and FIT4Green [2] EU projects.

4. **Other research directions:** New challenges are identified when new application scenarios are taken. One type of application we believe that energy consumption reduction has huge potential is Bag-of-Tasks applications, which are composed of sequential and independent tasks, such as in massive searches, image manipulation, and data mining algorithms.

[1] `http://www.eco2clouds.polimi.it`
[2] `http://www.fit4green.eu`

Bibliography

[1] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, March 2001.

[2] C. C. Aggarwal. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th International Conference on Very Large Data Bases*, VLDB'94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[4] R. Ali, F. Dalpiaz, and P. Giorgini. A goal modeling framework for self-contextualizable software. In *Proc. of the 10th International Workshop on Enterprise, Business-Process and Information Systems Modeling*, volume 29 of *BMMDS/EMMSAD'09*, pages 326–338. Springer, 2009.

[5] R. Ali, F. Dalpiaz, and P. Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458, November 2010.

[6] G. Alonso, F. Casati, H. A. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

[7] F. G. Alvares de Oliveira, Jr. and T. Ledoux. Self-optimisation of the energy footprint in service-oriented architectures. In *Proc. of the 1st Workshop on Green Computing*, GCM'10, pages 4–9, New York, NY, USA, 2010. ACM.

[8] D. Amyot and G. Mussbacher. URN: towards a new standard for the visual description of requirements. In *Proc. of the 3rd international conference on Telecommunications and beyond: the broader applicability of SDL and MSC*, SAM'02, pages 21–37, Berlin, Heidelberg, 2003. Springer-Verlag.

[9] G. Anadiotis, S. Kotoulas, E. Oren, R. Siebes, F. van Harmelen, N. Drost, R. Kemp, J. Maassen, F. Seinstra, and H. Bal. Marvin: a distributed platform for massive rdf inference. `http://www.larkc.eu/marvin/btc2008.pdf`.

[10] A. I. Anton. Goal-based requirements analysis. In *Proc. of the 2nd International Conference on Requirements Engineering*, ICRE'96, pages 136–144. IEEE Computer Society, 1996.

[11] D. Ardagna, L. Baresi, S. Comai, M. Comuzzi, and B. Pernici. A service-based framework for flexible business processes. *IEEE Software Magazine*, 28(2):61–67, March 2011.

[12] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani. PAWS: A framework for executing adaptive web-service processes. *IEEE Software Magazine*, 24(6):39–46, November 2007.

[13] D. Ardagna and B. Pernici. Global and local QoS guarantee in web service selection. In C. Bussler and A. Haller, editors, *Business Process Management Workshops*, volume 3812 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin / Heidelberg, 2006.

[14] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Transaction on Software Engineering*, 33(6):369–384, June 2007.

[15] Y. Asnar, P. Giorgini, and J. Mylopoulos. Goal-driven risk assessment in requirements engineering. *Requirements Engineering Journal*, 16(2):101–116, June 2011.

[16] D. F. Bacon, S. L. Graham, and O. J. Sharp. Compiler transformations for high-performance computing. *ACM Computing Surveys*, 26(4):345–420, 1994.

[17] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. Incremental reasoning on streams and rich background knowledge. In *Proc. of the 7th International Conference on The Semantic Web: research and Applications - Volume Part I*, ESWC'10, pages 1–15, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] L. Baresi and L. Pasquale. Live goals for adaptive service compositions. In *Proc. of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 114–123, NY, USA, 2010. ACM.

[19] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[20] L. A. Barroso and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2009.

[21] P. Bartalos, M. Blake, and S. Remy. Green web services: Models for energy-aware web services and applications. In *Proc. of the International Conference on Service-Oriented Computing and Applications*, SOCA'11, pages 1–8. IEEE, December 2011.

[22] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, May 2012.

[23] V. M. Benjamim Werneck, A. d. Pádua Albuquerque Oliveira, and J. C. Sampaio do Prado Leite. Comparing GORE Frameworks: i-star and KAOS. In *Proc. of the Workshop in Requirement Engineering*, WER'09, July 2009.

[24] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, September 2010.

[25] M. Bertoncini, B. Pernici, I. Salomie, and S. Wesner. GAMES: Green Active Management of Energy in IT Service Centres. In *Information Systems Evolution*, volume 72 of *Lecture Notes in Business Information Processing*, pages 238–252. Springer Berlin Heidelberg, 2011. 10.1007/978-3-642-17722-4_17.

[26] N. Bhatia and N. Kapoor. Fuzzy cognitive map based approach for software quality risk analysis. *SIGSOFT Software Engineering Notes*, 36(6):1–9, November 2011.

[27] W. Binder and N. Suri. Green computing: Energy consumption optimized service hosting. In *Proc. of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM'09, pages 117–128, Berlin, Heidelberg, 2009. Springer-Verlag.

[28] B. W. Boehm. Software risk management. In C. Ghezzi and J. A. McDermid, editors, *Proc. of the 2nd European Software Engineering Conference*, volume 387 of *ESEC'89*, pages 1–19. Springer, September 1989.

[29] B. W. Boehm. Software risk management: Principles and practices. *IEEE Software Magazine*, 8(1):32–41, January 1991.

[30] A. Bohra and V. Chaudhary. VMeter: Power modelling for virtualized clouds. In *Proc. of the 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, IPDPSW'10, pages 1–8. IEEE Computer Society, April 2010.

[31] C. Borgelt. Simple algorithms for frequent item set mining. In J. Koronacki, Z. Ras, S. Wierzchon, and J. Kacprzyk, editors, *Advances in Machine Learning II*, volume 263 of *Studies in Computational Intelligence*, pages 351–369. Springer Berlin / Heidelberg, 2010.

[32] D. Bouley and T. Brey. Fundamentals of data center power and cooling efficiency zones. White paper, The Green Grid, 2009. `http://www.thegreengrid.org`.

[33] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

[34] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *Proc. Intl. Semantic Web Conf. (ISWC)*, pages 54–68, 2002.

[35] D. J. Brown and C. Reams. Toward energy-efficient computing. *Communications of the ACM*, 53(3):50–58, March 2010.

[36] A. Bucchiarone, C. Cappiello, E. Di Nitto, R. Kazhamiakin, V. Mazza, and M. Pistore. Design for adaptation of service-based applications: Main issues and requirements. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, volume 6275 of *Lecture Notes in Computer Science*, pages 467–476. Springer Berlin / Heidelberg, 2010.

[37] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, and V. Mazza. A context-driven adaptation process for service-based applications. In *Proc. of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems*, PESOS'10, pages 50–56, New York, NY, USA, 2010. ACM.

[38] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. QoS-aware replanning of composite web services. In *Proc. of the 2005 IEEE International Conference on Web Services*, ICWS'05, pages 121–129, Washington, DC, USA, 2005. IEEE Computer Society.

[39] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. QoS-aware replanning of composite web services. In *Proc. of the 2005 IEEE International Conference on Web Services*, ICWS'05, pages 121–129. IEEE Computer Society, 2005.

[40] J. Cao, Y. Liu, J. Luo, and B. Mao. Efficient multi-QoS attributes negotiation for service composition in dynamically changeable environments. In *Proc. of the 2010 IEEE International Conference on Systems, Man and Cybernetics*, SMC'10, pages 3118–3124, October 2010.

[41] C. Cappiello, M. Fugini, A. Mello Ferreira, P. Plebani, and M. Vitali. Business process co-design for energy-aware adaptation. In *Proc. of 4th International Conference on Intelligent Computer Communication and Processing*, ICCP'11, pages 463–470. IEEE, August 2011.

[42] E. Capra and F. Merlo. Green IT: Everything starts from the software. In *Proc of the 17th European Conference on Information Systems*, ECIS'09, pages 62–73, 2009.

[43] J. Cardoso. Evaluating the process control-flow complexity measure. In *Proc. of the 2005 IEEE International Conference on Web Services*, ICWS'05, pages 803–804, Washington, DC, USA, 2005. IEEE Computer Society.

[44] J. Cardoso. Complexity analysis of BPEL web processes. *Software Process: Improvement and Practice*, 12(1):35–49, 2007.

[45] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Web Semantics*, 1(3):281–308, 2004.

[46] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven development of context-aware web applications. *ACM Transactions on Internet Technology (TOIT)*, 7(1), February 2007.

[47] A. Chatzigeorgiou and G. Stephanides. Energy metric for software systems. *Software Quality Control*, 10(4):355–371, December 2002.

[48] D. Chen, E. Henis, R. I. Kat, D. Sotnikov, C. Cappiello, A. Mello Ferreira, B. Pernici, M. Vitali, T. Jiang, J. Liu, and A. Kipp. Usage centric green performance indicators. *ACM SIGMETRICS - Performance Evaluation Review*, 39(3):92–96, December 2011.

[49] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, pages 1–26. Springer-Verlag, Berlin, Heidelberg, 2009.

[50] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. In *Proc. of the 22nd International Conference on Advanced Information Systems Engineering*, CAiSE'10, pages 113–128, Berlin, Heidelberg, 2010. Springer-Verlag.

[51] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*, volume 5600. Springer, March 2000.

[52] L. Chung and J. C. Prado Leite. On non-functional requirements in software engineering. In *Conceptual Modeling: Foundations and Applications*, pages 363–379. Springer-Verlag, Berlin, Heidelberg, 2009.

[53] P. Clarke and R. V. O'Connor. The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and Software Technology*, 54(5):433–447, May 2012.

[54] M. Comuzzi and B. Pernici. A framework for QoS-based web service contracting. *ACM Transaction on the Web*, 3(3):10:1–10:52, July 2009.

[55] G. Cook. How clean is your cloud? Report, Greenpeace International, April 2012. `http://www.greenpeace.org/international/en/publications/Campaign-reports/Climate-Reports/How-Clean-is-Your-Cloud/`.

[56] G. D. Costa and H. Hlavacs. Methodology of measurement for energy consumption of applications. In *Proc. of the 11th IEEE/ACM International Conference on Grid Computing*, pages 290–297. IEEE, October 2010.

[57] M. Curley. Introducing an IT capability maturity framework. In *Enterprise Information Systems*, volume 12 of *Lecture Notes in Business Information Processing*, pages 63–78. Springer Berlin Heidelberg, 2009.

[58] F. Dalpiaz, P. Giorgini, and J. Mylopoulos. An architecture for requirements-driven self-reconfiguration. In *Proc. of the 21st International Conference on Advanced Information Systems Engineering*, CAiSE'09, pages 246–260, Berlin, Heidelberg, 2009. Springer-Verlag.

[59] A. Dardenne, A. Van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[60] A. Del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Defining process performance indicators: an ontological approach. In *Proc. of the 2010 International Conference on On the Move to Meaningful Internet Systems - Volume Part I*, OTM'10, pages 555–572, Berlin, Heidelberg, 2010. Springer-Verlag.

[61] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl. A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, 15(3-4):313–341, December 2008.

[62] T. Do, S. Rawshdeh, and W. Shi. pTop: A process-level power profiling tool. In *Proc. of the 2009 Workshop on Power Aware Computing and Systems*, HotPower'09, New York, NY, USA, October 2009. ACM.

[63] T. Do, S. Rawshdeh, and W. Shi. pTop: A process-level power profiling tool. In *Proc. of the Workshop on Power Aware Computing and Systems*, HotPower'09, New York, NY, USA, October 2009. ACM.

[64] B. Donnellan, C. Sheridan, and E. Curry. A capability maturity framework for sustainable information and communication technology. *IT Professional*, 13(1):33–40, January 2011.

[65] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proc. of the 34th annual international symposium on Computer architecture*, ISCA'07, pages 13–23, New York, NY, USA, 2007. ACM.

[66] X. Franch, P. Grunbacher, M. Oriol, B. Burgstaller, D. Dhungana, L. Lopez, J. Marco, and J. Pimentel. Goal-driven adaptation of service-based systems from runtime monitoring data. In *Proc. of the 35th IEEE Annual Computer Software and Applications Conference Workshops*, COMPSACW'11, pages 458–463, July 2011.

[67] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proc. of the Sixteenth International Conference on Machine Learning*, ICML'99, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[68] A. Fuxman, J. Mylopoulos, M. Pistore, and P. Traverso. Model checking early requirements specifications in Tropos. In *Proc. of the 5th IEEE International Symposium on Requirements Engineering*, RE'01, pages 174–181, Washington, DC, USA, 2001. IEEE Computer Society.

[69] A. Gehlert and A. Heuer. Towards goal-driven self optimisation of service based applications. In *Proc. of the 1st European Conference on Towards a Service-Based Internet*, ServiceWave'08, pages 13–24, Berlin, Heidelberg, 2008. Springer-Verlag.

[70] A. Ghose, K. Hoesch-Klohe, L. Hinsche, and L.-S. Lê. Green business process management: a research agenda. *Australasian Journal of Information Systems*, 16(2), 2009.

[71] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the 2000 ACM International Conference on Management of Data*, SIGMOD'00, pages 1–12, New York, NY, USA, 2000. ACM.

[72] Y. Hao and Y. Zhang. Web services discovery based on schema matching. In *Proc. of the Australasian conference on Computer science*, ACSC'07, pages 107–113. Australian Computer Society, Inc., 2007.

[73] R. Harmon and H. Demirkan. The next wave of sustainable IT. *IT Professional*, 13(1):19–25, January 2011.

[74] Q. He, J. Yan, H. Jin, and Y. Yang. Adaptation of web service composition based on workflow patterns. In *Proc. of the 6th International Conference on Service-Oriented Computing*, ICSOC'08, pages 22–37, Berlin, Heidelberg, 2008. Springer-Verlag.

[75] J. Hedman and S. Henningsson. Three strategies for green IT. *IT Professional*, 13(1):54–57, January 2011.

[76] J. Hielscher, R. Kazhamiakin, A. Metzger, and M. Pistore. A framework for proactive self-adaptation of service-based applications based on online testing. In *Proc. of the 1st European Conference on Towards a Service-Based Internet*, ServiceWave'08, pages 122–133, Berlin, Heidelberg, 2008. Springer-Verlag.

[77] K. Hoesch-Klohe, A. Ghose, and L.-S. Lê. Towards green business process management. In *Proc. of the 2010 IEEE International Conference on Services Computing*, SCC'10, pages 386–393. IEEE Computer Society, July 2010.

[78] C.-L. Hwang and K. Yoon. *Multiple Attribute Decision Making: Methods and Applications. A State-of-the-art Survey.* Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin, Heidelberg, 1981.

[79] S. Islam. Software development risk management model: a goal driven approach. In *Proc. of the doctoral symposium for ESEC/FSE on Doctoral symposium*, ESEC/FSE Doctoral Symposium'09, pages 5–8, New York, NY, USA, 2009. ACM.

[80] M. C. Jaeger, G. Mühl, and S. Golze. QoS-aware composition of web services: A look at selection algorithms. In *Proc. of the 2005 IEEE International Conference on Web Services*, ICWS'05, pages 807–808. IEEE Computer Society, July 2005.

[81] M. C. Jaeger, G. Mühl, and S. Golze. QoS-aware composition of web services: an evaluation of selection algorithms. In *Proc. of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part I*, OTM'05, pages 646–661, Berlin, Heidelberg, 2005. Springer-Verlag.

[82] T. A. Jenkin, J. Webster, and L. McShane. An agenda for 'green' information technology and systems research. *Information and Organization*, 21(1):17–40, 2011.

[83] Y. Jian, L. Liu, J. Wang, and E. Yu. Goal-driven adaptation of internetware. In *Proc. of the 2nd Internetware*, Internetware'10, pages 10:1–10:9, New York, NY, USA, 2010. ACM.

[84] C. Kaner and W. P. Bond. Software engineering metrics: What do they measure and how do we know? In *Proc. of the 10th International Software Metrics Symposium*, METRICS'04, 2004.

[85] J. G. Kang and K. H. Han. A business activity monitoring system supporting real-time business performance management. In *Proc. of the 3rd International Conference on Convergence and Hybrid Information Technology - Volume 01*, pages 473–478, Washington, DC, USA, 2008. IEEE Computer Society.

[86] A. Kansal and F. Zhao. Fine-grained energy profiling for power-aware application design. *SIGMETRICS Performance Evaluation Review*, 36(2):26–31, 2008.

[87] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proc. of the 1st ACM Symposium on Cloud Computing*, SoCC'10, pages 39–50, NY, USA, 2010. ACM.

[88] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proc. of ACM Symposium on Cloud Computing*, SoCC'10, pages 39–50, New York, NY, USA, 2010. ACM.

[89] K. Kant. Challenges in distributed energy adaptive computing. *SIGMETRICS Performance Evaluation Review*, 37(3):3–7, January 2010.

[90] R. Kazhamiakin, S. Benbernou, L. Baresi, P. Plebani, M. Uhlig, and O. Barais. Adaptation of service-based systems. In M. Papazoglou, K. Pohl, M. Parkin, and A. Metzger, editors, *Service Research Challenges and Solutions for the Future Internet*, volume 6500 of *Lecture Notes in Computer Science*, pages 117–156. Springer Berlin / Heidelberg, 2010.

[91] R. Kazhamiakin, M. Pistore, and A. Zengin. Cross-layer adaptation and monitoring of service-based applications. In *Proc. of the 2009 international conference on Service-oriented computing*, ICSOC/ServiceWave'09, pages 325–334, Berlin, Heidelberg, 2009. Springer-Verlag.

[92] K. H. Kim, A. Beloglazov, and R. Buyya. Power-aware provisioning of virtual machines for real-time cloud services. *Concurrency and Computation: Practice & Experience*, 23(13):1491–1505, September 2011.

[93] A. Kipp, T. Jiang, M. Fugini, and I. Salomie. Layered green performance indicators. *Future Generation Computer Systems*, 28(2):478–489, February 2012.

[94] R. K. L. Ko, S. S. G. Lee, and E. W. Lee. Business process management (BPM) standards: a survey. *Business Process Management Journal*, 15(5):744–791, 2009.

[95] Z. Kobti and W. Zhiyang. An adaptive approach for QoS-aware web service composition using cultural algorithms. In *Proc. of the 20th Australian Joint Conference on Advances in Artificial Intelligence*, AI'07, pages 140–149, Berlin, Heidelberg, 2007. Springer-Verlag.

[96] J. Koomey. Estimating total power consumption by servers in the U.S. and the world. Technical report, Analytics Press, February 2007. Available at `http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf`.

[97] K. Kritikos. QoS-Based Web Service Description and Discovery. PhD Thesis, Computer Science Department, University of Crete, Heraklion, Greece, December 2008.

[98] P. Lago and T. Jansen. Creating environmental awareness in service oriented software engineering. In *Proc. of the 2010 international conference on Service-oriented computing*, ICSOC'10, pages 181–186, Berlin, Heidelberg, 2011. Springer-Verlag.

[99] A. Lamsweerde. Conceptual modeling: Foundations and applications. chapter Reasoning About Alternative Requirements Options, pages 380–397. Springer-Verlag, Berlin, Heidelberg, 2009.

[100] E. Letier and A. Van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. *SIGSOFT Software Engineering Notes*, 29(6):53–62, October 2004.

[101] F. Li, B. Zhang, and L. Ge. Composite service adaptation supported environmental monitoring system. In *Proc. of the 5th International Conference on Genetic and Evolutionary Computing*, ICGEC'11, pages 315–318, Washington, DC, USA, 2011. IEEE Computer Society.

[102] Y. Li, X. Zhang, Y. Yin, and J. Wu. QoS-driven dynamic reconfiguration of the SOA based software. In *Proc. of the 2010 International Conference on Service Sciences*, ICSS'10, pages 99–104, Washington, DC, USA, 2010. IEEE Computer Society.

[103] M. Y. Lim and V. W. Freeh. Determining the minimum energy consumption using dynamic voltage and frequency scaling. In *Proc. of the 21th International Parallel and Distributed Processing Symposium*, IPDPS, pages 1–8, March 2007.

[104] D. Liu, Q. Wang, and J. Xiao. The role of software process simulation modeling in software risk management: A systematic review. In *Proc. of the 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM'09, pages 302–311, Washington, DC, USA, 2009. IEEE Computer Society.

[105] J. Liu, A. Kipp, D. Arnone, M. Berge, W. Christmann, R. Kat, E. Henis, and A. Ciuca. Validation scenarios. GAMES Deliverable 7.2, GAMES, December 2010. `http://www.green-datacenters.eu/`.

[106] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen. GreenCloud: a new architecture for green data center. In *Proc. of the 6th international conference industry session on Autonomic computing and communications industry session*, ICAC-INDST'09, pages 29–38, New York, NY, USA, 2009. ACM.

[107] L. Liu and E. Yu. Designing information systems in social context: a goal and scenario modelling approach. *Information System*, 29(2):187–203, April 2004.

[108] W. Liu, X. Li, and D. Huang. A survey on context awareness. In *Proc. of the International Conference on Computer Science and Service System*, pages 144–147, June 2011.

[109] W. Ma, L. Liu, W. Feng, Y. Shan, and F. Peng. Analyzing project risks within a cultural and organizational setting. In *Proc. of the 2009 ICSE Workshop on Leadership and Management in Software Architecture*, LMSA '09, pages 6–14, Washington, DC, USA, 2009. IEEE Computer Society.

[110] C. G. Malone, W. Vinson, and C. E. Bash. Data center tco benefits of reduced system airflow. pages 1199–1202. IEEE, 2008.

[111] F. Manola and E. Miller. Rdf primer. `http://www.w3.org/TR/rdf-primer/`, February 2004.

[112] M. Marwah, P. Maciel, A. Shah, R. Sharma, T. Christian, V. Almeida, C. Araújo, E. Souza, G. Callou, B. Silva, S. Galdino, and J. Pires. Quantifying the sustainability impact of data center availability. *SIGMETRICS Performance Evaluation Review*, 37(4):64–68, March 2010.

[113] A. Mehta, M. Menaria, S. Dangi, and S. Rao. Energy conservation in cloud infrastructures. In *Proc. of the 5th Annual IEEE International Systems Conference*, SysCon'11, pages 456–460, April 2011.

[114] A. Mello Ferreira, K. Kritikos, and B. Pernici. Energy-aware design of service-based applications. In *Proc. of the 7th International Joint Conference on Service-Oriented Computing*, ICSOC-ServiceWave'09, pages 99–114, Berlin, Heidelberg, 2009. Springer-Verlag.

[115] A. Mello Ferreira, B. Pernici, and P. Plebani. Green performance indicators aggregation through composed weighting system. In *Proc. ICT as Key Technology against Global Warming*, volume 7453 of *Lecture Notes in Computer Science*, pages 79–93. Springer Berlin / Heidelberg, September 2012.

[116] D. A. Menascé. QoS issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[117] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar. End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. *IEEE Transactions on Services Computing*, 3(3):193–205, July-Sept. 2010.

[118] R. Mirandola and P. Potena. A QoS-based framework for the adaptation of service-based systems. *Scalable Computing: Practice and Experience*, 12(1):63–78, 2011.

[119] M. Mohabey, Y. Narahari, S. Mallick, P. Suresh, and S. Subrahmanya. A combinatorial procurement auction for QoS-aware web services composition. In *Proc. of the 2007 IEEE International Conference on Automation Science and Engineering*, CASE'07, pages 716–721, September 2007.

[120] C. Momm, R. Malec, and S. Abeck. Towards a model-driven development of monitored processes. In *Proc. of the 8th Internationale Tagung Wirtschaftsinformatik*, WI'07, 2007.

[121] M. Morandini, L. Penserini, and A. Perini. Towards goal-oriented development of self-adaptive systems. In *Proc. of the 2008 international workshop on Software engineering for adaptive and self-managing systems*, SEAMS'08, pages 9–16, New York, NY, USA, 2008. ACM.

[122] S. Murugesan. Making IT green. *IT Professional*, 12(2):4–5, March 2010.

[123] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, June 1992.

[124] J. Na, Y. Gao, B. Zhang, L.-p. Huang, and Z.-l. Zhu. Improved adaptation of web service composition based on change impact probability. In *Proc. of the 2010 Third International Conference on Dependability*, DEPEND'10, pages 146–153, Washington, DC, USA, 2010. IEEE Computer Society.

[125] S. Najar. Context-aware intentional service framework for service adaptation. In *Proc. of the 4th International Conference on Research Challenges in Information Science*, RCIS'10, pages 663–672, May 2010.

[126] N. C. Narendra, K. Ponnalagu, J. Krishnamurthy, and R. Ramkumar. Run-time adaptation of non-functional properties of composite web services using aspect-oriented programming. In *Proc. of the 5th international conference on Service-Oriented Computing*, ICSOC'07, pages 546–557, Berlin, Heidelberg, 2007. Springer-Verlag.

[127] G. Nebuloni and T. Meyer. The role of x86 processors in the virtualized, energy-constrained datacenter. Technical report, An IDC Multiclient Study, November 2008.

[128] T. Neubauer and C. Stummer. Extending business process management to determine efficient IT investments. In *Proc. of the 2007 ACM symposium on Applied computing*, SAC '07, pages 1250–1256, New York, NY, USA, 2007. ACM.

[129] W. Niu, G. Li, Z. Zhao, H. Tang, and Z. Shi. Multi-granularity context model for dynamic web service composition. *Journal of Network and Computer Applications*, 34(1):312–326, January 2011.

[130] A. Nowak, F. Leymann, and R. Mietzner. Towards green business process reengineering. In *Proc. of the 2010 International Confer-*

ence on Service-Oriented Computing, ICSOC'10, pages 187–192, Berlin, Heidelberg, 2011. Springer-Verlag.

[131] A. Nowak, F. Leymann, and D. Schumm. The differences and commonalities between green and conventional business process management. In *Proc. of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC'11, pages 569–576, Washington, DC, USA, 2011. IEEE Computer Society.

[132] E. E. Odzaly and P. S. Des Greer. Software risk management barriers: An empirical study. In *Proc. of the 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM'09, pages 418–421, Washington, DC, USA, 2009. IEEE Computer Society.

[133] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In *Proc. of the 14th IEEE International Conference on Parallel and Distributed Systems*, ICPADS'08, pages 171–178. IEEE Computer Society, 2008.

[134] G. Ortiz and A. G. d. Prado. Web service adaptation: A unified approach versus multiple methodologies for different scenarios. In *Proc. of the 5th International Conference on Internet and Web Applications and Services*, ICIW'10, pages 569–572, Washington, DC, USA, 2010. IEEE Computer Society.

[135] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer Journal*, 40(11):38–45, November 2007.

[136] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer Journal*, 40(11):38–45, November 2007.

[137] M. P. Papazoglou and W.-J. Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, July 2007.

[138] M. L. Paracchini, C. Pacini, M. L. M. Jones, and M. Perez-Soba. An aggregation framework to link indicators associated with multifunctional land use to the stakeholder evaluation of policy options. *Ecological Indicators*, 11(1):71–80, 2011.

[139] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: Hyper-structure mining of frequent patterns in large databases. In *Proc. of the 2001 IEEE International Conference on Data Mining*, ICDM'01, pages 441–448, Washington, DC, USA, 2001. IEEE Computer Society.

[140] B. Pernici, editor. *Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[141] P. Plebani and B. Pernici. URBE: Web service retrieval based on similarity evaluation. *Knowledge and Data Engineering, IEEE Transactions on*, 21(11):1629–1642, November 2009.

[142] V. Popova and A. Sharpanskykh. Modeling organizational performance indicators. *Information Systems*, 35(4):505–527, June 2010.

[143] G. Pozzi, S. Morasca, A. Alessio, and A. Mello Ferreira. Software measures for business processes. In *Proc. II of the 15th East-European Conference on Advances in Databases and Information Systems*, ADBIS, pages 11–22, September 2011.

[144] H. Psaier, L. Juszczyk, F. Skopik, D. Schall, and S. Dustdar. Runtime behavior monitoring and self-adaptation in service-oriented systems. In *Proc. of the 2010 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, SASO '10, pages 164–173, Washington, DC, USA, 2010. IEEE Computer Society.

[145] H. Psaier, F. Skopik, D. Schall, and S. Dustdar. Behavior monitoring in self-healing service-oriented systems. In *Proc. of the 34th IEEE Annual Computer Software and Applications Conference*, COMPSACW'10, pages 357 –366, July 2010.

[146] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[147] N. A. Qureshi and A. Perini. Requirements engineering for adaptive service based applications. In *Proc. of the 18th IEEE International Conference on Requirements Engineering*, RE'10, pages 108–111, Washington, DC, USA, 2010. IEEE Computer Society.

[148] A. S. Rao and M. P. Georgeff. Modeling rational agents with a BDI-architecture. In *Readings in agents*, pages 317–328. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[149] H. Reijers and I. Vanderfeesten. Cohesion and coupling metrics for workflow process design. In J. Desel, B. Pernici, and M. Weske, editors, *Business Process Management*, volume 3080 of *Lecture Notes in Computer Science*, pages 290–305. Springer Berlin / Heidelberg, 2004.

[150] D. Reynolds. Jena 2 Inference Support, August 2009. `http://jena.sourceforge.net/inference/`.

[151] R. R. Rodriguez, J. J. A. Saiz, and A. O. Bas. Quantitative relationships between key performance indicators for supporting decision-making processes. *Computers & Industrial Engineering*, 60(2):104–113, February 2009.

[152] E. Rolón, J. Cardoso, F. García, F. Ruiz, and M. Piattini. Analysis and validation of control-flow complexity measures with BPMN process models. In T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, R. Ukor, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Enterprise, Business-Process and Information Systems Modeling*, volume 29 of *Lecture Notes in Business Information Processing*, pages 58–70. Springer Berlin Heidelberg, 2009.

[153] F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier Science Inc., New York, NY, USA, 2006.

[154] G. G. Roy and T. L. Woodings. A framework for risk analysis in software engineering. In *Proc. of the 7th Asia-Pacific Software Engineering Conference*, APSEC'00, pages 441–, Washington, DC, USA, 2000. IEEE Computer Society.

[155] L. Ruijie and H. Hong. Business process metrics modeling based on model-driven. In *Proc. of the 2010 IEEE International Conference on Software Engineering and Service Sciences*, ICSESS'10, pages 327–330, July 2010.

[156] S-Cube Partners. State of the art report on software engineering design knowledge and survey of HCI and contextual knowledge. Deliverable JO-JRA-1.1.1, S-Cube Network of Excellence, July 2008.

[157] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2):14:1–14:42, May 2009.

[158] L. Sánchez-González, F. García, J. Mendling, and F. Ruiz. Quality assessment of business process models based on thresholds. In *Proc. of the 2010 International Conference on On the Move to Meaningful Internet Systems - Volume Part I*, OTM'10, pages 78–95, Berlin, Heidelberg, 2010. Springer-Verlag.

[159] E. Saxe. Power-efficient software. *Communications of the ACM*, 53(2):44–48, February 2010.

[160] N.-H. Schmidt, K. Erek, L. M. Kolbe, and R. Zarnekow. Towards a procedural model for sustainable information systems management. In *Proc. of the 42nd Hawaii International Conference on System Sciences*, HICSS'09, pages 1–10. IEEE Computer Society, 2009.

[161] V. Sharma, A. Thomas, T. Abdelzaher, and K. Skadron. Power-aware QoS management in web servers. In *Proc. of the 24 th IEEE Real-Time systems Symposium*, RTSS'03, pages 63–72, 2003.

[162] H. Singh, D. Azevedo, D. Ibarra, R. Newmark, S. O'Donnell, Z. Ortiz, J. Pflueger, N. Simpson, and V. Smith. Data center maturity model. White paper, The Green Grid, 2011. `http://www.thegreengrid.org`.

[163] A. Tahamtan and A. M. Tjoa. Toward deadline aware low carbon supply chain. In *Proc. of the 1st International Conference on Logistics, Informatics and Service Science*, LISS 2011, pages 20–23, June 2011.

[164] J. Tao, A. Kipp, S. Ioan, T. Cioara, I. Anghel, M. Berge, R. Kat, D. Arnone, B. Pernici, P. Plebani, and A. Ciuca. Validation method and metrics. GAMES Deliverable 7.3, GAMES, December 2010. `http://www.green-datacenters.eu/`.

[165] M. Turner, D. Budgen, and P. Brereton. Turning software into a service. *Computer*, 36(10):38–44, October 2003.

[166] U.S. Environmental Protection Agency (EPA). Report to congress on server and data center energy efficiency – public law 109-431. Technical report, August 2007.

[167] U.S. Environmental Protection Agency (EPA). Report to congress on server and data center energy efficiency – Public Law 109-431. Technical report, August 2007.

[168] W. M. P. Van Der Aalst, A. H. M. T. Hofstede, and M. Weske. Business process management: a survey. In *Proc. of the 2003 international conference on Business process management*, BPM'03, pages 1–12, Berlin, Heidelberg, 2003. Springer-Verlag.

[169] A. Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proc. of the 5th IEEE International Symposium on Requirements Engineering*, RE'01, pages 249–, Washington, DC, USA, 2001. IEEE Computer Society.

[170] A. Van Lamsweerde. Requirements engineering: from craft to discipline. In *Proc. of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT'08/FSE-16, pages 238–249, New York, NY, USA, 2008. ACM.

[171] I. Vanderfeesten, J. Cardoso, H. A. Reijers, and W. Van Der Aalst. Quality metrics for business process models, May 2007.

[172] T. Velte, A. Velte, and R. Elsenpeter. *Green IT: Reduce Your Information System's Environmental Impact While Adding to the Bottom Line*. McGraw-Hill, September 2008.

[173] D. Wang. Meeting green computing challenges. In *Proc. of the International Symposium on High Density packaging and Microsystem Integration*, HDP'07, pages 1–4. IEEE Computer Society, 2007.

[174] R. T. Watson, M.-C. Boudreau, A. Chen, and M. Huber. Green IS: Building sustainable business practices. In R. W. (Ed.), editor, *Information Systems*, pages 1–17. Global Text Project, Athens, GA, USA, 2009.

[175] R. T. Watson, M.-C. Boudreau, and A. J. Chen. Information systems and environmentally sustainable development: energy informatics and new directions for the is community. *MIS Quarterly*, 34(1):23–38, March 2010.

[176] J. Wen Jun Xue, A. P. Chester, L. G. He, and S. A. Jarvis. Model-driven server allocation in distributed enterprise systems. In *Proc. of the 3rd International Conference on Adaptive Business Information Systems*, ABIS'09, March 2009.

[177] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[178] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann. Monitoring and analyzing influential factors of business process performance. In *Proc. of the 2009 IEEE International Enterprise Distributed Object Computing Conference*, EDOC'09, pages 141–150, Washington, DC, USA, 2009. IEEE Computer Society.

[179] J. Williams and L. Curtis. Green: The new computing coat of arms? *IT Professional*, 10(1):12–16, 2008.

[180] E. S. Yu. Conceptual modeling: Foundations and applications. chapter Social Modeling and i*, pages 99–121. Springer-Verlag, Berlin, Heidelberg, 2009.

[181] E. S.-K. Yu. *Modelling strategic relationships for process reengineering.* PhD thesis, Toronto, Canada, 1996. UMI Order No. GAXNN-02887 (Canadian dissertation).

[182] E. S. K. Yu and J. Mylopoulos. An actor dependency model of organizational work: with application to business process reengineering. In *Proc. of the conference on Organizational computing systems*, COCS'93, pages 258–268, New York, NY, USA, 1993. ACM.

[183] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, May 2004.

[184] L. Zeng, B. Benatallah, A. H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Transaction on Software Engineering*, 30(5):311–327, May 2004.

[185] L. Zeng, H. Lei, M. Dikun, H. Chang, K. Bhaskaran, and J. Frank. Model-driven business performance management. In *Proc. of the 2005 IEEE International Conference on e-Business Engineering*, ICEBE'05, pages 295–304, October 2005.

[186] N. Zenker and J. Rajub. Resource measurement for services in a heterogeneous environment. In *Proc. of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, ICTTA'08, pages 1–15. IEEE Communications Society, 2008.

[187] H. Zhang, L. Liu, and T. Li. Designing IT systems according to environmental settings: A strategic analysis framework. *The Journal of Strategic Information Systems*, 20(1):80–95, March 2011.

[188] M. Zouari, M.-T. Segarra, and F. Andre. A framework for distributed management of dynamic self-adaptation in heterogeneous environments. In *Proc. of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT'10, pages 265–272, Washington, DC, USA, 2010. IEEE Computer Society.