POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Master of Science in
Ingegneria Informatica

Automatic Modeling System: a database based infrastructure to develop, validate and evaluate scientific models. An application to combustion kinetic models

Candidate

Alice Rigamonti
Student Id. number 820018

Thesis Supervisor

Prof. Barbara Pernici

Assistant Supervisor

Prof. Tiziano Faravelli
Ing. Matteo Pelucchi

Academic Year 2015/2016

Automatic Modeling System: a database based infrastructure to develop, validate and evaluate scientific models. An application to combustion kinetic models

Master thesis. Politecnico di Milano

This thesis has been typeset by LaTeX and the smcthesis class.

Author's email: alice1.rigamonti@mail.polimi.it

Abstract

Scientific data repository is a concept which has been developed in the past few years. All digital libraries used until now are tied to a specific context and cannot develop or share their resources with other scientific communities, making cultural growth difficult. Over the past few years, research has been aimed towards creating a universal digital library, not just for textual documents, but also scientific and experimental data sharing. These data must have certain authentication, documentation and formatting elements, to allow sharing and conversion. The Chemical Reaction Engineering and Chemical Kinetics Group of the Politecnico di Milano is experimenting sharing and formatting issues for the experimental data in its research field. There are several data formats and data repositories in the Reaction Engineering and Chemical Kinetics field, but none of these can be used by the community as a whole. This paper explores the idea of a universal structure for data archiving for all European universities. The main goal is to create a data repository to manage internal data and chemical simulations studied within the Politecnico di Milano. This software automates all study processes of simulations and unifies the use of current programs such as OpenSMOKE++, Gnuplot e CurveMatching. A database has been created to save and manage data and an interface has been developed to make the researcher's study process quicker and more efficient. This database is the starting point for sharing data and creating formats to extend sharing to other universities , allowing the exchange of structured and easily importable information in the various repositories.

Abstract in Italiano

Scientific data repository community é un concetto sviluppato negli ultimi anni. Le librerie digitali diffuse negli anni passati sono tutte relative ad un contesto specifico e non possono svilupparsi e condividere le loro risorse con altre comunitá scientifiche bloccando cosí la crescita culturale. In questi anni molte ricerche stanno cercando delle soluzioni per la creazione di un' unica biblioteca digitale non solo per documentazioni testuali, lo studio si estende sulla gestione e l'interscambio di dati scientifici e dati sperimentali. Questi dati devono avere diverse caratteristiche di formattazione, specifica delle risorse, autenticazione e documentazione per poter permettere la condivisione e la conservazione. Il Chemical Reaction Engineering and Chemical Kinetics Group del Politecnico di Milano ha verificato questo problema di condivisione e formattazione dati sperimentali nel proprio campo di ricerca. Esistono tanti data format e data repository nel campo "Reaction Engineering and Chemical Kinetics" ma nessuna delle proposte esistenti é utilizzabile dall' intera comunitá. Alcuni data format come ReSpecTh hanno una buona struttura ma molto generalizzata. Questa tesina é nata per proporre una struttura comune per tutte le universitá europee che lavorano in questo contesto. Lo scopo principale é quello di creare un scientific data repository interno al Politecnico di Milano per la gestione dei dati e delle simulazioni chimiche. Il software automatizza tutti i processi di studio delle differenti simulazioni e raggruppa l'utilizzo di tutti i programmi utilizzati come OpenSMOKE++, Gnuplot e CurveMatching. É stato creato un database per la memorizzazione e gestione dei dati ed é stata sviluppata un'interfaccia per rendere piú veloce ed efficace possibile tutto il processo di studi dei ricercatori. Il software sviluppato é stato esteso per supportare la collaborazione con differenti universitá per poter scambiare informazioni strutturate e facilmente importabili nei differenti repository utilizzati.

Contents

Chapter 1

Introduction

1.1 The content

The Chemical Reaction Engineering and Chemical Kinetics Group (CRECK) at Politecnico di Milano works on the development of new fuels and new combustion technologies. The group is mostly focused on the development of chemical kinetic models to describe pollutant formation and to explore the viability of new fuels in new and conventional combustion systems.

Standard procedures in the development of scientific models requires an iterative process, summarized in Figure 1.1

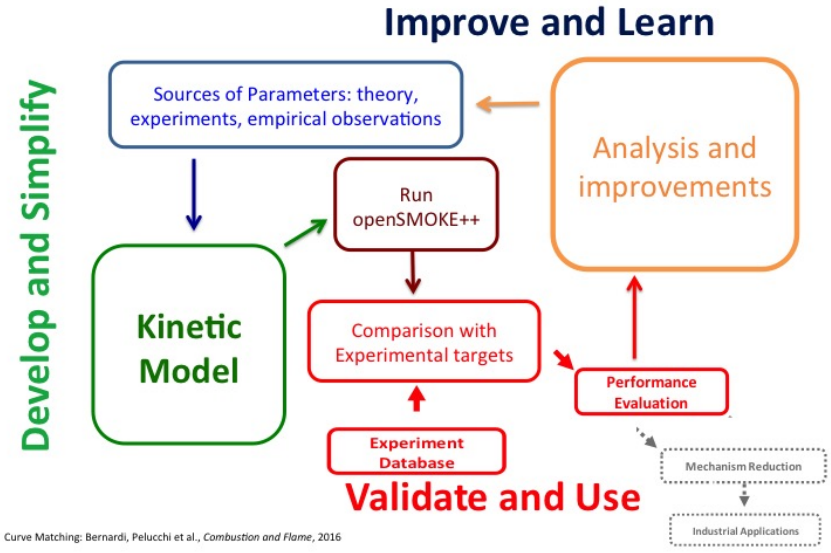


Figure 1.1. Iterative process flow

From a first state of the art analysis of the scientific literature, the second step requires the collection and organization of experimental data. Such experimental data constitute the targets for model validation, i.e. they are used to prove the accuracy of model predictions through extensive comparisons with results from

numerical simulations. Despite this approach applies to any scientific area, there exist disciplines where the daily users would particularly benefit from a systematic organization of both the large amount of experimental data available and of the evolution of models in time. A clear example of this is combustion science and engineering, and in particular the branch dealing with the development of kinetic mechanisms to describe the chemistry involved in the combustion of a given fuel. Correctly assessing the efficiency and the pollution potential of a fuel is mostly a chemical kinetics problem. A deep understanding of details in chemical kinetics allows the tailoring of a fuel or fuel blend for an existing technology (engine, gas turbine, industrial burner etc.) or the tuning of an engine for a given fuel [1]. The reliability of such models, and the advances achieved over the last 30/40 years due to the advent of high-end computing facilities and to the implementation of fundamental theories [2], have driven a dramatic increase in the number of models available and, more dramatically, in the number of experimental targets (ignition delay times, laminar flame speeds, pollutant formation etc.).

The Chemical Reaction Engineering and Chemical Kinetics (CRECK) group of the Department of Chemistry and Chemical Engineering at Politecnico di Milano has been playing a major role in the field of chemical kinetic modelling since the dawn of this science [3]. An enormous amount of data and different versions of the CRECK model have been stored over these decades, and their organization and implementation in a fully automated system strategically constitutes a key step for future developments, extensions and investigations. The complexity of such models also motivates the need of a smart system to manage and control its daily update and refinement. As an example, the CRECK model, known for its particularly "small" size contains about 500 chemical species and 20000 reactions. Overall, this roughly translates into 100,000 parameters, strongly interacting with one another.

In general, a user collects new experimental data from a new publication, saves the data in text (txt), comma separated values (csv) or excel (xls) format. The second step requires the identification of the parameters needed for the numerical simulations. Once the input is compiled, kinetic simulations are performed with the OpenSMOKE++ framework of Cuoci et al. [4]. The numerical simulations produce output files that have to be compared with experimental data, according to the standard graphical visualization. At this point an evaluation procedure is used to discriminate between a "good" model and a "bad" model. In the last 2 years two different methods have been proposed to automatically evaluate the performances of combustion kinetic models [5] [6], in order to avoid subjective graphical evaluations. This procedure not only applies to the cases where new experimental data have to be considered, but also to the more common situation where a new release of the CRECK model has to be produced. Theory driven updates and refinement of model parameters happen on a daily basis, to improve model performances at a given condition. Due to the strong interaction between the different parameters, it is important to verify that the modification meant to improve the model reliability at some specific condition, did not make it worse for some other target.

The aim of this thesis is to implement a fully automatic method translating all these steps in a well constrained routine, of use for kinetic modelers and of easy extension to scientific modelers in general.

It is possible to summarize this process into 4 steps:

1. Data Storage, Formatting and Sharing.
2. Compilation of input file for numerical simulation
3. Execution of numerical simulation with OpenSmoke++
4. Graphical comparison of model and experiments
5. Evaluation of model performances

The following section describes in detail point 1,2 and 4. Concerning point 3 and 5, a detailed description of openSMOKE++ numerical framework and of model performances evaluation [6] are out of the scope of this thesis However, the two steps are implemented in automatic tool described in the following.

1.2 Data Sharing

1. Experimental data file reading
   The User firstly needs to obtain data from scientific papers, or from the attached supporting information. Different issues are associated with this:
   (a) Time needed to recover the data, if not already well organized in the supporting information to the paper
   (b) Risk of replicating work already done
   (c) Need of improving the data altready stored by someone else
   (d) Formatting of Experimental data
2. Compilation of OpenSMOKE++ input files
   The step requires the correct interpretation of experimental data in order to assign the proper parameter for the numerical simulation(reactor configuration, fuel composition, temperature, pressure, time etc.)
3. Execution of Numerical simulations and output files analysis
   Depending on the reactor configuration and on the size of kinetic model the time for simulation can range from seconds to hours. Therefore it is important to store and share the results from numerical simulations, avoiding replication of work.
4. Evaluation of results
   Results obtained by one user, if not shared, might be repeated by another user causing avoidable time costs. If performances evaluation were shared, users could use previous results. If matching were shared, Users could access results of previous users. This way they could improve different results and compare their opinions or studies
5. Experimental data from other repositories
   Another issue is the exchange of experimental data with already available repositories or sources (other laboratories, other databases etc.). Such data might not be structured in the same way as different reserach groups might be

interested in storing different information or the same information in a different way. This constitutes an issue mainly because the information contained in an experimental data from another sources might fit the requirements of other solvers similar to OpenSMOKE++ (CHEMKIN [7], Canterra etc.) but not OpenSMOKE++ format. Properly storing the needed information is useful to avoid the risk of running simulations using wrong specifications in the future.

Format of file of experiment

Experimental data files and OpenSMOKE++ outputs might have different structures. Differnt series of points can be contained: $(X_1, Y_1)(X_2, Y_2)$ etc. series and $(X, Y_1)(X, Y_2)$ etc. series. Different separators between the values can also be used: comma(,) , space, tab etc.
Agreeing on a shared format and rules for formatting data facilitate data sharing.

Graph creation method

Different tools can be used to graphically compare experimental data and model results: Excel, Origin, Gnuplot, R.

1.3 Solution strategy

After having analysed the work flow and the issues of kinetic modelers we can elaborate a strategy to solve such criticism. The first step is to build a database essential to store data and input files and create a software managing all the data in the database. It is necessary for the user to be able to insert data to create the OpenSMOKE++ input file without mistakes and with the proper structure. It is necessary that all these operations, currently executed manually, become automated. Automation of the processes has been fundamental to save all the data in the database, and to structure the information needed by OpenSMOKE++ simulation and in the experimental data. Three important steps are: the insertion the pdf format file of the scientific paper with its experimental data and standard reference information (title, authors, year, journal, volume, issue, pages, figures of interest). The second step deals with managing the experimental data, the OpenSMOKE++ input parameters (mixture condition, reactor type, reactor configuration, temperature, pressure etc.). The input file is then linked to the experimental data and its reference paper. Clearly, both the input information and the experimental data can be modified at any time if needed. The third step is the execution of the simulation with OpenSMOKE++ and the comparison of the results with the experimental data previously stored. This step has the very important feature of allowing the user to search for particular fuels, conditions, targets of interest to be executed through a filter. Such a filter will limit or extend the validation targets according to the user specifications. Diagrams comparing experiments and models are automatically generated and stored together with the output files.

Respecth experimental file

Other research groups use different formatting formats: as Matlab, Excel and XML. In the combustion community, the Chemical Kinetics Group at ELTE Budapest University, developed an extremely useful XML format (ReSpecTh [8]), containing the data points together with the conditions needed to run the simulations. It was decided to adopt the same format, making it applicable to OpenSMOKE++ requirements.

The XML file contains:

- File paper name and references
- Type of experiment
- Initial Condition
- Data point of experiment
- Column name and unit measure of columns data point

The ReSpecTh database already contains 968 dataset [8] of experimental data of interest for combustion mechanism validation. The format was slightly modified to be able to directly include such data into the framework developed herein. To converge to a common format a collaboration with ELTE Budapest University was activated within the COST CM 1404 Action (SMARTCATs, www.smartcats.eu). The group from POLIMI needed to introduce specific fields within the ReSpecTh format to describe experimental data and to carry out simulation in OpenSMOKE++ . The joint effort was promoted through COST Short Term Scientific Missions and the outcomes were presented at the 2nd General Meeting and Workshop on Smart Energy Carriers in Industry of SMARTCATs Action held at Instituto Superior Tecnico, Lisbon, Portugal. [9]

1.4 Curve Matching

The recent years have observed an increase of the experimental and theoretical data of combustion processes. The new experimental techniques, together with the increase of the measurement accuracy, produced wider and wider sets of experimental data. The continuous and fast growth of the computer performances enhanced the development of modeling activity. As in the development of any scientific models, extensive and frequent comparisons with experimental data have to be performed. All these comparisons are usually performed using plots in which the experimental data and the calculated curves are plotted together. This is a very effective approach, which has been used since the dawn of science, whenever a model had to be compared with empirical observations. In some cases the qualitative result is not sufficient. Moreover, when a very large number of comparisons has to be done, the procedure could result in impractical times. An agreement evaluated on a quantitative basis can better highlight measurements or models limits, but it becomes strictly necessary if an automatic procedure of comparison is required. The first step in this complex process of assessing models validity is the definition of a standardized and quantitative

approach to estimate the agreement between models and experimental data, and especially to compare the performances of different models in reproducing the same set of measurements [6]. To pursue this goal, a novel framework named Curve Matching was introduced by the CRECK group at POLIMI. This innovative method was introduced in the automatic modeling system developed in this study.
Implementation of CurveMatching integration and automatization is described in this document in section 4. In this document you can find in section 1 a general introduction to the context. The document continues with the state of the art in section 2 where existing studies about different scientific data repository and chemical reactor and kinetics repository are described. In section 3 is a the analysis of the software describing initial proposal, costs analysis and the creation of the database.

Chapter 2

State Of The Art

2.1 Context

The Chemical Reaction Engineering and Chemical Kinetics group (CRECK) at POLIMI had the necessity to implement a structure to manage research data and to build a shared repository to store and manage models and experiments. Two types of information have to be managed: a) experimental data, b) inputs and outputs from OpenSMOKE++ simulations. To build the repository in the most efficient way, an analysis of existing Research Data Repository structures and Digital Libraries in the context of chemistry and chemical kinetics had to to be carried out.

2.2 State of the art

The concept of Digital library and Data Repository has been widely used for many years to collect data and to manage large amount of information about specific sectors [12]. A Digital library is an important structure for distribution of information and to promote culture diffusion. The same concept is applicable to scientific data repositories related to specific areas of research. In the last years many proposals have been made in terms of expansion and sharing of digital libraries, given the key role played by data in our society. There is a universal agreement on the benefits of "data sharing and reuse" as a mean to accelerate science performances [13].
It is possible to identify six main ways that explain how a Data repository can increase performances [12]:

- Support, dissemination and quick exchange of experimental research.
- Contribution to collect larger number of information
- Improve the productivity of scientists, engineers and students.
- Encourage new discoveries and innovations.
- Speed up technological transfer
- Provide access and quality to everyone

Lars Mayer presented an analysis about the changing of digital space and about how digital libraries and data repositories should reallocate resources to innovation, publishing and research [15]. Single digital libraries exist and required acquisition of good digital storing, in single digital libraries the storing is well managed and well formatted. It has been necessary extend storing of digital and metadata at community libraries. In this case preservation of the work is more complex. From these conditions Mayer creates a life cycle stage of resources represented in figure 2.1.[15]

Figure 2.1. Life cycle Digital Libraries [15]

There are five steps related to digital data, where the first one represents the creation of it. The second step is the Description where Specific of metadata and their preservation are stored and linked to digital data, next the two steps are storing in the repository. Data storing can be reused thanks to open access policy. Data reused can be modified or updated and inserted again in the data repository restarting cycle from first step.

Attempts to structure the community are under investigation, like for example OAIS (Open Archivial Information System) [16] and TRAC (Trusted Repositories Audit and Certification) [17]. OAIS defines information as ”any type of knowledge that can be exchanged, and this information is always expressed (i.e., represented) by some type of data” [16]. In the OAIS model, the represented information contains semantic and structural information. OAIS defines PDI (Presentation Description Information) as a story of information. PDI contains creation, modifications and association about an information stored. Four types of information compose a PDI:

- Provenance: describes the source of the Content Information. The provenance helps to have a clear understanding to where provenance data should be mantained in the digital environment.
- Context: describes how the content relates to other information outside the information package, including why it was produced. It facilitates understanding and interpretation of the Content Data.

- Reference: provides one or more identifiers, or systems of identifiers, by which the content information may be uniquely identified.
- Fixity: provides a wrapper or protective shield that protects the content information from undocumented alteration. With provides of file measurements such as byte count, record count, and record length a system can calculate and associate checksums for each data file has recently been deployed.

TRAC formalizes libraries and long-term management of digital assets. The main concepts are preservation and sustainable accesses. TRAC defines elements and actors of nine Digital Preservation Policies. The main elements are: Mandate, Categories of commitment, Level of preservation and Access and use criteria. Firstly mandates are defined as commitments and stakeholders that are involved in the digital content, in this way long-term goal is supported. Another main element is Categories of commitment that provides a scope for the types of materials that will be digitally preserved and made accessible. Levels of Preservation supports long-term adherence to an established digital preservation conceptual model. Finally Access and Use Criteria identify what elements affect proper implementation of the policy: infrastructure, feasibility of solutions, standards adoption, and evolution of standards and best practices. Community data repository present immature system and presence of single digital library. A new structure is needed for single libraries to take part in the community. Digital libraries have to shifting from "content-centric" system of simple organise and with personal structure that provide access to particular collections of data to "person-centric" system that provide facilities for communication, collaboration and any kind of interaction among scientists, researchers, and the general audience interested in topics of pertinence. [18] Concept of person-centering extends to the research data publishing contest. "Research data" is the very broad and heterogeneous range of materials produced during a research activity [14]. Science repository is called to manage research datasets produced in different contexts and by different teams. Digital Libraries are used by different sets of stakeholders that represent academia, industry, founding agencies and publishers. For certain types of important digital objects there are well-curated, deeply-integrated, special-purpose repositories such as 3TU.Datacentrum(http://datacentrum.3yu.nl, CSIRO Data Access Portal , Dryad, Figshare(https://figshare.com/) , Zenodo (http:// zenodo.org/). These repository accept a wide range of datatypes in a wide variety of formats.

Analysing these repositories Candela and Castelli proposed a flexible method to create more robust repositories that contain and manage a wider heterogeneity of data. The paper describes the main features [18]

- Formatting
  the structure of a dataset must be in a certain format to maintain usability. A dataset is a unit of information. There are two types of format associated to it, the file format and the content format. Having data properly formatted is a pre-requisite for any use of the dataset. To create a general repository it has been studied generalization of formatting. Generalist repositories cannot make any assumptions on the data and content formats. This has lead to the development of approaches that aim to be generic and as much formatted as possible. This makes the management of the system difficult.

Table 2.1. Foundation

Name	Type	Founded	Country	Context
3TU. Data-centrum	Institution	2008	NLD	Cooperation between 3 University, It provide to community a research data archive
CSIRO DAP	Institution	2011	AUS	Manage, discover and share data across different research fields.
Dryad	Organization	2008	USA	Initiative of a group of journals.They set up a community-governed infas-tructure for data archiving and pol-icy.
Figshare	Company	2011	GBR	Created by PhD student as a way to store manage and freely disseminate any kind of research output.
Zenodo	Organization	2013	CHE	Enable researches to preserve and share any kind of research output. It focuses on produced in the context of the long-tail (in the context of those scientific domains in which activity is performed in a large number of rel-atively small labs and by individual researchers who collectively produce the majority of scientific results) of science.

- Documenting
  a dataset has to be retrieved, understood and reused. The goal is to make datasets available for validation and reuse both within the scientific community that has produced it and other community. It is necessary to create auxiliary data to datasets to provide contextual information about them. This infor-mation can be what the dataset is about - descriptive metadata - and how it has been obtained - provenance. This documentation influences dataset discoverability, understandability, verification and pratical re-use. Finding a complete documentation to apply to datasets is very difficult because existing repositories have a large heterogeneity auxiliary data to datasets.

- Licensing
  Two licences are involved, the one agreed between the repository and the data owner, and the second one agreed between the repository and the data consumer. The main licence is relative to data owner, a re-use license concerns at least attribution and control on commercial exploitation. Moreover the data-owner must check whether the licence is compatible with one of those supported by the repository. The goal is to find a progressive orientation to establish common licences and not only proprietary ones. Usually access licences are publics but in some cases require data consumer registration.

- Publication cost
  the value of founding for guarantee an open access to dataset. There are two type of costs: the first one is a monetary cost, the second focuses on the use of services. All repositories have the tendency to charge data owners when publishing their dataset rather than data consumers. The costs depend on the level of curation or preservation for a given number of years or a given level of quality. Another cost relates to the use of service. The direction to take is to reduce the repository operation cost. There are many techniques to reduce the costs of curation by automatically extracting parts of necessary information through analysis and mining of related artefact, e.g. papers.

- Validation: cogency and soundness of published data to ensure quality of repository. Scientific data repositories help the dataset validation phase by offering practices and services for both dataset pre-publication and post-publication. In the contest of generalist data repositories, dataset validation is at an initial stage. It is characterized by the definition of ”data-quality”, through validation criteria and general revisions.

- Availability
  Availability guarantees that published datasets are at consumer's disposal. Most repositories store multiple copies of the datasets, either on their own premises, or in third party service providers. For preservation of repositories the use of migration practices is common, but becomes a challenging feature to preserve for open ended sets of data. Some repositories do not guarantee usability and completeness of deposited objects over time, in some case the migration to another format can not preserve the original information. If migrated files do not contain all the information available in the original file format, then can be excluded from migration. In order to guarantee current and future availability the repository has to address both technical and economic challenges.

- Discoverability and access
  Dataset discovery and access is the facility enabling consumers to find out about the existence of dataset, and to be able to get access to them, namely to the dataset content and the associated documentation. This facility is called metadata and includes user-driven functions and semi-automatic function. The main discovery facility is search on database by query on fields of entities, this is called database discovery. The main approach is based on keywords and fields, the set of supported fields is repository specific. Discovery is a service that can be provided by third party for example through API.

- Citation
  Is the practice to provide a reference to datasets intended as a description of data properties that enable credits and attribution, discover, interlinking and access. Communities often have their own way to cite their data. It would be very useful that repositories offer facilities enabling specific communities to customize the way their own data should be cited, preserving interoperability at the same time.

Comparing different repositories it has been underlined that a large variety of metadata exist, some metadata schemes have common features in terms of types of data across different disciplines [14]. In order to give a general information attribute of metadata are specify main attribute necessary:

- Availability: enabling access to the dataset and its content.
- Bibliometric data: report dataset statistic.
- Coverage: describing spatial, temporal and taxonomic coverage.
- Date: when dataset was created.
- Format: formatting perspective including the size.
- License: policies ruling.
- Minimal description.
- Paper reference: references to related pubblications.
- Project: initiative leading to production of the dataset.
- Subject: including keyworks, tags and subject headings.

One of the issues previously raised had to deal with dataset formatting, a prerequisite for any use of datasets. To create a community data library it is necessary to create a standard approach to save dataset and, at the same time, to create a generalist data repository. This condition restricts the facilities that repository can offer to data publication [14]. It is important that the user can upload the type of data used for research and experiments, and that the dataset is "intelligibile" to permit cross disciplinary reuse of datasets. The second main criticism is the validation and reuse of datasets by other organization and in other disciplines. Usually a dataset is associated with a data paper. Eleven classes have to be included in a paper for metadata : availability, bibliometric, coverage, data, format, license, minimal description, paper references, project, provenance and subject [14]. To connect and manage dataset and documentation, the concept of "data paper" has been defined. Each overlay documentation was expected to contain (a) metadata about the overlay document (b) information about and from the quality process for which the document was constructed, and (c) basic metadata from the referenced resource to aid discovery and identification [13]. Journals now contains different types of data papers. Two types of journals content have been identified, the "pure journal" is the journal that contains only data paper, the "mixed" journal contains any type of paper including data paper. The data paper is identify as a set of two elements that have to be materialized into concrete and identifiable information objects: the data set (the subject of the data paper) and the data paper itself (the artefact produced to describe the data set). Data Paper is important to build a coherent data repository because contains more information about documentation Data Paper can solve or reduce costs of curation and research, it can be added fields of research and with this type of file the link with citations is not a problem. [13]. By Analysing existing templates the following classes of data set have been identified before:

- Availability: provide data set access attributes, namely, a DOI or a URI.
- Provide explicit declaration of any factor that might influence the related dataset.
- Coverage: to provide data set ”extent” attributes.
- Format: to provide information oriented to promote the actual reuse.
- License to provide information oriented data set policy.
- Microattribution to provide appropriate credits to each author of the paper by capturing in detail the contribution of each author
- Project
- Provenance: to provide information describing methologies
- Quality: qualitative aspect of information
- Reuse to provide information promoting potential uses.

Different tools to create a correct data paper exist. For example GBIF network, specifically conceived to support metadata for datasets and to have an interface to add all classes of documentation.This tool is specifically conceived to support the production of metadata for data sets of primary biodiversity data to be published through the GBIF network [19]. However, the tool is also equipped with a facility for automatically generating a data paper manuscript from the data set metadata. The author is requested only to produce the Introduction section of the data paper.

The general data format is still one of the main criticisms, as metadata in science might have different structures. The dataset paper in science, which promote ”hybrid” model, metadata associated to dataPaper can have different formatting and different standard. In many repositories there is an extension to allow publication of metadata as a zip file that contains supplementary material.

2.3 PriMe

Chemical kinetic models rely on a large amount of experimental and theoretical information and methods, commonly used to develop and validate such models [21]. These data are spread over different data repositories and often they cannot be effectively used due to different formatting. Commonly these data have to be transformed and rearranged by subjects having an expert knowledge. Frenklach et al. at the University of California at Berkeley, firstly highlighted and investigated such issues in combustion kinetics and studied a process to store data, creating a repository to store combustion experiments and models, the PrIMe Kinetics framework [21]. This repository is open to any type of organization and institution that treat these models. First of all PriMe defines concept of Process Informatics asa data-centric approach to developing predictive models for chemical reaction systems. It will deals with all aspects of integration of pertinent data of complex systems (industrial processes and natural phenomena) whose complexity originates from

chemical reaction networks. The goal of Process Informatics is to build targeted knowledge from the entire community and to provide the wealth of information in its entirety to every user [21]. As specified in the PrIMe report in August 2002 [22] two types of accesses and actors exist. In fact, the main consumers are data providers and model users. The first one deposits her/his new observations or new computational results, the second one can peruse the entire database. About licenses, the report firstly specifies "Open Membership": access is open to any scientist who wants to be involved (data provider). The second license defines "Open Sources" access: all data and models submitted to the project or created by the project will become "public domain",mostly for model users. Finally PrIMe rely on "Democratic Governance", i.e. the project management follows a rational leadership guided by the combustion community consensus. Clearly PrIMe represented a pioneering approach to combustion kinetic modelling, based on the scientific collaboratory paradigm, through a data-centric system framework. The main elements of PrIMe include: a data Warehouse which is a repository of data provided by the community, a data Library which archives community evaluated data, and computer based tools to process data and to assemble data into predictive models [23]. To build the structure of the database and to store all data, PriMe starts by defining chemical reactor models. Different reasons justify this first categorization: the first is that the modelling of a combustion process requires a reaction model. Most disagreements are between models and experiments and most controversies begin with and trace to the selected reaction model. Secondly chemical kinetics has accumulated much needed data and the missing data can be evaluated using quantum and reaction rate theories [23].

2.3.1 Prime data models

The data Model includes entities allowing the creation of a simulation case and storing PriMe information about its results. Relevant entities are: bibliography for referencing data sources, chemical element and chemical species involved in the simulation, reactions and reaction rates (rate expression and parameters), the model associated to the reactor, experimental data to compare with the simulation output and data attribute used to specify features of the experimental data. Experimental data, data attribute and initial status (mixture composition, temperature, pressure etc.) are specified in the PrIMe XML file format. XML formatting allows exchanging experiments between researchers within the community.

2.3.2 Software Infrastructure

PrIMe is an online application. The central feature of PrIMe is the PrIMe Workflow Application (PWA) that links all components together and create an automation of dataflow. The infrastructure enables interaction with different users, having diverse objectives and varying roles. An overall diagram of the software infrastructure is shown in Fig. 2.2

The PrIMe Workflow Application (PWA) is a web based application that unifies the components of PrIMe into a single interface. Workflow Components are the features that allow every operation on the data. Such features also permit sharing of data-analysis and scientific simulation codes, providing to the community the

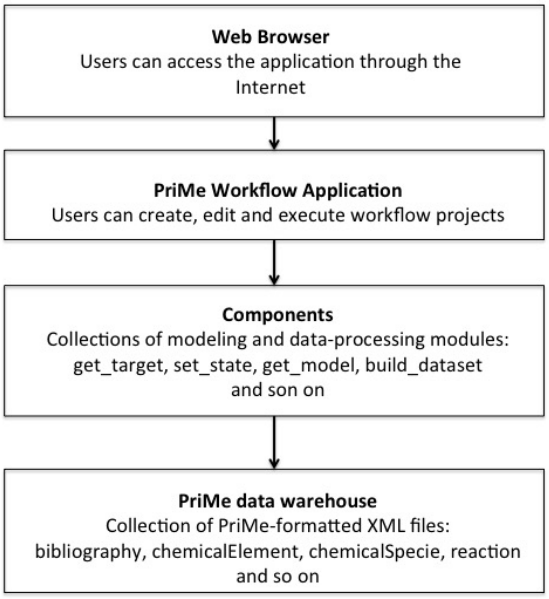


Figure 2.2. Software Infrastructure [24]

capability of applying different methods of analysis on the same data, thus allowing meaningful comparisons and interpretations.

2.3.3 PrIME criticisms

The combustion kinetics community has found some difficulties using PrIMe. First of all it was observed that the insertion of new experimental data presented some criticism: firstly for the usability of its features. The community has observed [25] that is not easy add new experimental data because all active parameters are not recalculated, the operator of the active database provides new releases from time to time. A bug about details of experimental data was also found: it does not provide any information on how the data should be interpreted, easily generating misunderstandings within the users. Other difficulties are related to the references. A large amount of literature references are contained in PrIMe, preventing the possibility of finding a specific single file or experimental dataset. Finally, PrIMe does not specify what is required to complete the simulation in case of failure, requiring a very expert user.

Following these considerations, the Chemical Kinetics Laboratory of the Eótvós University created a data structure improving PrIMe XML formatting. Eótvós University builded a data repository (http://respecth.hu/) containing XML experiments with additional information compared to PrIMe.

2.4 Respecth

ReSpecTh has several unique and noteworthy features. Some of the datasets contained in ReSpecTh are built upon data mining, using discrete mathematics

methods. ReSpecTh includes a searchable bibliographical chemical kinetics data collection, a database of high-temperature gas kinetics elementary reactions with information on the available experimental data, and capacity for a detailed assessment of derived quantities. ReSpecTh, in the portion concerning combustion chemistry, contains experimental data on specific reactions, experimental data used for model validation, and detailed reaction mechanism of exemplary combustion system and utility programs [23]. ReSpecTh set up a dedicated database http://respecth.hu, indirect experiments are searchable according to the type of experiment, experimental conditions and the type of fuel and batch gas. In this way ReSpecTh can easily be used to locate all experimental data at a given set of experimental conditions, which can be of great benefit for mechanism developers. The direct experiments and the theoretical determinations can be searched according to the stoichiometry of the elementary reaction and the range of temperature and pressure [8]. Table 1.2 shows experimental and theoretical data related to combustion available in ReSpecTh

All combustion data are stored according to the ReSpecTh Kinetics Data (RKD) Format Specification. This data format consists of a set of specifications, aiming to provide an unambiguous definition of the storage of combustion experimental data and rate coefficient determinations [8]. The RKD format specification defines an XML data format to provide flexible data representation and allows for an easy extension of the format specification [23].

The ReSpecTh kinetics data format is an XML based format. All data are stored in XML elements and attributes. The main element of the XML file (root element) contains all the data elements that characterize an experiment. It has been possible to divide XML data Element in three types:

1. Reference of experiment
   It contains a reference about the file paper and the type of experiment to simulate as ”fileAuthor”, ”fileVersion”, ”ReSpecTh version”, ”bibliography link”, ”Experiment type” and ”apparatus”. The main Data Element is the ”Experiment type” that is mandatory. ”Experiment type” specifies the model of reactor and the type of experiment. For instance, ignition delay times (i.e. autoignition propensity of a given fuel) are measured typically in shock tubes and rapid compression machines, for which the simulation requires the solution of a batch reactor. Other important DataElements are references about paper, fileAuthor, bibliography link specific references, DOI and name of the file paper. Data element apparatus is specifically about the Reactor (e.g. batch, plug flow, jet stirred rectors) but simulation of ReSpecTh already contains this information in ”Experiment Type”, making the apparatus not mandatory. The structure is:

```
<ReSpecThVersion>
    <major></major>
    <minor>0</minor>
</ReSpecThVersion>
<fileAuthor>Tamas Varga, ELTE - Alice - Matteo</fileAuthor>
<bibliographyLink preferredKey="Mueller 1999"/>
<experimentType>Concentration time profile measurement</experimentType>
<apparatus>
    <kind>Batch Reactor</kind>
```

```
        <mode>A</mode>
    </apparatus>
```

2. Initial Condition
   "Common properties" represents initial condition of experiment. It is fundamental for the execution of experiment simulation. "Common properties" contains Data Element "property" that specify species link or component associated to its value. For example species=H2 has measure='Mole fractions' and value=0.2. Example of the structure:

```
    <commonProperties>
        <property name="pressure" units="atm">
            <value>0.3</value>
        </property>
        <property name="temperature" units="K">
            <value>880</value>
        </property>
        <property name="initial composition">
            <component>
                <speciesLink preferredKey="H2"/>
                <amount units="mole fraction">0.0050</amount>
            </component>
            <component>
                <speciesLink preferredKey="O2"/>
                <amount units="mole fraction"> </amount>
            </component>
        </property>
    </commonProperties>
```

3. Data group Data group contains experimental data points. In the first part there are the definition of experimental data columns and in the second one rows about experimental data.

```
    <dataGroup>
        <property name="time" units="s" id="x1" label=""/>
        <property name="composition" units="mole fraction" id="x2" >
            <speciesLink preferredKey="H2"/>
        </property>
        <property name="composition" units="mole fraction" id="x3" >
            <speciesLink preferredKey="O2"/>
        </property>
        <property name="composition" units="mole fraction" id="x4" >
            <speciesLink preferredKey="H2O"/>
        </property>
        <dataPoint>
            <x1>3.680000e-003</x1>
            <x2>4.965400e-003</x2>
            <x3>9.588880e-006</x3>
            <x4>4.965400e-003</x4>
```

```
        </dataPoint>
        <dataPoint>
            <x1>6.610000e-003</x1>
            <x2>4.255800e-003</x2>
            <x3>6.730000e-004</x3>
            <x4>4.597100e-003</x4>
        </dataPoint>
    <|dataGroup>
```

Starting from the large amount of experimental information stored in the ReSpecTh database the CRECK group at POLIMI proposed a collaboration to the ELTE group with the goal of exchanging data, thus speeding up the recovery of experimental data. Modifications to the ReSpecTh format have been necessary to efficiently interface ReSpecTh and OpenSMOKE++ for simulations. Added information are:

- Type of Experiment: in "apparatus" data Element add child of "reactor" and "model" of experiment to simulate.
- Equivalence ratio/fuel/oxidizer
  In specification of Initial status the ReSpecTh XML format provides only one method for the insertion of components, openSMOKE++ input can specify the Initial Status component as an association between equivalence ratio, fuel component and oxidizer component. It has been proposed to add these name types to the property of initial status. Fuel and oxidizer contain in data element childs that represent components involved for each of them.
- Axis to data point
  ReSpecTh XML format does not specify axis on data point column specification. This is because only experimental data with the structure XY_1Y_2 are managed. It has been proposed to add an attribute axis on the specification of columns. In this way it is much more simple to identify the structure of the experimental data.

Table 2.2. experimentalData

	hydrogen combustion	syngas combustion	ethanol combustion
related publications	[14, 15]	[16, 17]	[18]
ignition delay times - measured in shock tubes	770 datapoints in 53 datasets	732 datapoints in 62 datasets	444 datapoints in 39 datasets
ignition delay times - measured in RCM	229 datapoints in 20 datasets	492 datapoints in 47 datasets	20 datapoints in 3 datasets
laminar burning velocities	443 datapoints in 73 datasets	2116 datapoints in 217 datasets	1011 datapoints in 124 datasets
concentration profiles in flow reactors	152 datapoints in 17 datasets	443 datapoints in 73 datasets	1674 datapoints in 22 datasets
concentration profiles in shock tubes	-	436 datapoints in 4 datasets	8871 datapoints in 14 datasets
concentration profiles in JSR	631 datapoints in 9 datasets	90 datapoints in 3 datasets	328 datapoints in 4 datasets
direct measurements	1749 datapoints in 56 datasets for 10 elementary reactions	589 datapoints in 29 datasets for 5 elementary reactions	725 datapoints in 58 datasets for 11 elementary reactions
theoretical determinations	-	-	41 datasets for 13 elementary reactions
total number of data points	3974 data points	4462 data points	13114 data points
total number of data sets	228 data sets	435 data sets	305 data sets

Chapter 3

Software Analysis

3.1 Feasibility Analysis

Starting point

Before starting the analysis and development of the project the Department of Chemistry CMIC discussed about different possibile solutions to reach the final goal. The goal is create a software that stores all data used to study kinetics mechanism and automates all the steps of work that the researcher do manually.

Database has been build a unified to retrieve and work on the Data in order to automate the mechanism. It has been needed build coherent database because of the structure of experimental data file and the connection between experimental data and openSmoke simulation.

The first step was observing how a student works on the Data. It has been necessary observe how the researcher analyses the contents of file and how he executes different cases of model.

The following main steps have been identified

1. Recovery of file Paper. The paper is read,the experimental data is individuated
2. If the file Paper has attached files of experiment data points in question it is necessary to take this Data. Otherwise it needed read associated graph and extract data points.
3. When the student has experimental data he creates the input of openSmoke. It is been necessary specify initial condition and species of output. These species are the same contained into the data experiment file. OpenSmoke is executing.
4. At the end of openSmoke execution the student takes the output and reads interesting species of output.
5. Finally he compares species of experimental data with output values and creates graphics for each species for compare the experiment and the model.

After this observation a list of possible solution is created for reach the goal. In the next section we list all proposals that we think up.

Proposed Solutions

Different solutions were proposed, and experiencing the different scenarios iteratively led to the final proposal.

- FIRST PROPOSAL: Managing files

  The first need is to limit the information stored, to avoid additional costs in terms of memory and working time. It is necessary to focus on essential features of the experiments. Therefore only the connection between the experimental data file, OpenSMOKE++ and the pdf format scientific paper was kept.
  In this way there is a reduction of memory costs, the space occupy is the files size. This solution solves problem about space, OpenSMOKE++ execution time and output comparison time.
  On the other side this solution presents some criticisms. The first one has to deal with searching for specific experimental data in the database. If only the files are saved it is impossible to search for them in an efficient way (by fuel, temperature or pressure conditions). The second issue is that the input file cannot be updated and a new input file needs to be uploaded every time.

- SECOND PROPOSAL: Create Database and Search
  Implementation of OpenSMOKE++ input insertion fields has been necessary. In this way the input file is created before its execution. Search data insertion is needed, so we add fields to search data. It has been necessary to insert these data into Database and create attributes about search. Search data makes selection in the database easier than a search on file names. The search can be complete and the selection can be limited.

- THIRD PROPOSAL: Create a Database for Experimental data, References OpenSMOKE++ and Data Search

  The third proposal was to add to the second one features about managing experimental data and references to the scientific paper. It has been necessary to save all the data to be compared with model results and to modify experimental data directly in the database without visualizing the file. The user can change the specification using only the software without modifying the original file. The pdf format paper is attached to keep track of the data.

After having defined a strategy, its costs and the schedule to its implementation were assessed. The first solution has lower time costs than second one. In the first proposal it is necessary to upload reference files in the database. For the second proposal it is necessary to read all the fields of OpenSMOKE++ simulation and create input file for each run of OpenSMOKE++. At the end of execution it is required to read output files and compare them with experimental data.

Costs of the second proposal are higher than the first one, due to the complex structure of the database. It is necessary to save, for each experiment, all fields required in the simulation input. Moreover, it is necessary to save general information (fuel, temperature, pressure) about the experiment used for the data search.

The third solution is similar to the second solution, as only the information about the pdf paper is added. Clearly not many differences are obtained in terms of

space. This proposal allows secure management of the data and gives a structure to data experimental data and File paper.

Function Point Analysis

To carry on analysis it has been needed evaluate the cost of third proposal by the Function Point analysis [20]. FPA is based on the number of functions that have to developed in the software. We start identifing Functional User Requirements for each proposal explained before. It has been necessary divide Transactional functions to Data functions, apply rule by table of FP and estimate results. For the transactional function we identify three types of data [20]

- External Inputs (EI) that is an elementary process that processes data or control information sent from outside the boundary;
- External Output (EO) an elementary process that sends data or control information outside the boundary and includes additional processing logic beyond that of an External Inquiry;
- External Inquire (EQ) a form of data that is leaving the system. However, since automated counting tools cannot distinguish between External Inquiries and External Outputs, all External Inquiries will be included in and counted as External Outputs.

For Input translation, File Types Referenced shall refer to the number of file referenced read or updated. Data function can divide in External Interface Files(EIF) a user recognizable group of logically related data or control information, which is referenced by the application being measured, but which is maintained within the boundary of another application. Internal logical file (ILF) user recognizable group of logically related data or control information maintained within the boundary of the application being measured. For the data function it is needed to identify Data element types and Record element types and compare them with Data Function Complexity for evaluating the Data Function Size. Before starting we define the object DET as data element type. It is a unique user recognizable, non-repeated attribute that is part of an ILF, EIF, EI or EO.

When we have identify FTR (data function read and/or maintained by a transactional function) we can compare it with following tables:

With this consideration we can start by identify DET component involved. We can identify 4 ILF that are data search runtime, experimental data, OpenSmoke Data and file paper. These are the four most important inputs that we want to save and manage in our database. For each of them we can define DET contained in. For research Data we have seven observable DET: Component of experiment, Keyword, Range of pressure, range of temperature, reactor, model and equivalence Ratio. For the experimental Data we manage in database Ignition Time Specie, Ignition time output, reference page about the experiment, type structure, file paper, experiment rows and experiment columns. We soffirm on OpenSmoke data, these data are manage in dynamic and extended way, so we identify major data that are Reactor, model, eleven dynamics element that depend on reactor selected as the subDictionary. We can have eight subDictionaries to manage openSmoke data

Table 3.1. External Input complexity

	1-4 DET	5-15 DET	>15 DET
<2 FTR	1	1	2
2 FTR	1	2	3
>2 FTR	2	3	3

Table 3.2. External Output complexity

	1-5 DET	6-19 DET	>19 DET
<2 FTR	1	1	2
2 FTR	1	2	3
>2 FTR	2	3	3

Table 3.3. External Input complexity

External Input Complexity	External Input weights
1	3
2	4
3	6

Table 3.4. External Output complexity

External Output complexity	External Output weights
1	4
2	5
3	7

with a total of 41 fields to manage. In the end we evaluate File Paper ILF that involve nine DET as Name of file paper, references, authors, file associated to it and experimental data associated with its references and image. In the analysis we have a EIF file that use after the execution of openSmoke are OpenSmoke output that involve one file with dataPoint of simulation. Below we show the table with Data Function and complexity:

Table 3.5. Data Function evaluation

	TYPE	**RETs**	**DETs**	**COMPLEXITY**
Data of Research	ILF	2	7	Low
Experimental Data	ILF	3	7	Low
OpenSmoke Data	ILF	11	41	High
File Paper Data	ILF	3	9	Avarage
Open Smoke Output	EIF	1	1	Low

We can calculate Function point size for each Type of Data Function by these operations

$$ILF_s = 7\sum ILF_{DFC=1} + 10\sum ILF_{DFC=2} + 15\sum ILF_{DFC=3}$$

$$EIF_s = 5\sum EIF_{DFC=1} + 7\sum EIF_{DFC=2} + 10\sum EIF_{DFC=3}$$

We make refereces to Table 3.5 and we calculate our functions point

$$ILF_s = 7 \cdot 1 + 10 \cdot 1 + 15 \cdot 1 = 46$$
$$EIF_s = 5 \cdot 2 = 10$$

For the Transactional Data we find the operations that the software can have that involved use of Data Function, in the table 3.6 we show the operations and their relative FTR and DET. In the last column we can see the complexity relative to operation.

Table 3.6. Transactional Function evaluation

	TYPE	**FTRs**	**DETs**	**COMPLEXITY**
Insert Experiment/Simulation	EI	3	51	High
Update Experiment/Simulation	EI	3	51	High
Read experiment for list	EO	2	11	Avarage
Insert file paper	EI	1	9	Low
Update file paper	EI	1	9	Low
Execute OpenSmoke	EO	2	37	High
Read OpenSmoke output	EI	1	1	Low
Compare OpenSmoke and experimental Data	EO	3	5	Low
Read for update experiment/simulation	EO	3	51	High

As the Data function we calculate the Transactional point size by these expression:

$$EI_s = 3 \sum EI_{EIC=1} + 4 \sum EI_{EIC=2} + 6 \sum EI_{EIC=3}$$
$$EO_s = 4 \sum EO_{EOC=1} + 5 \sum EO_{EOC=2} + 7 \sum EO_{EOC=3}$$

We make reference Table 2.2 and we execute calculation

$$EI_S = 4 \cdot 3 + 6 \cdot 2 = 24$$
$$EO_s = 5 \cdot 1 + 7 \cdot 2 = 19$$

Finally we can calculate the Automated Function Point size

$$AFPs = ILF_s + EIF_s + EI_s + Eo_s$$
$$AFPs = 46 + 10 + 24 + 19$$

3.2 Requirement Analysis and Specfication Document

3.2.1 Introduction

Section goal

The goal of this section makes a definition to functionality and definition about the software in question, this section specifies the condition and precondition to use the product. We start with the definition of dictionary and the analysis of

different scenarios to explain the goal and different study. In next sections we can find different example to applications of this software. Major beneficiaries of this section are customer, teams of developing and testing team.

General goal

The Department of Chemistry wants automate processes of execution of simulation and comparison with openSmoke simulation output and experimental data. Goal of this software is to give to students the possibility to manage all data with interface and the possibility to store these data in a common structure to share with all students. Students can insert, update and execute simulation of openSmoke and manage it in the specific form. Moreover students can associate and modify experimental data for each simulation insert. Students can execute more simulation in one time and they can check the results of all of them at the end of process. Other goal is propose to client a confortable and intuitive interface and speed up the process of simulation and study of old insert data.

Definitions

- OpenSmoke++ : is software that the Department use to create an output of reactor from initial condition and changes in the simulation. These conditions are signal in the input file that changes depending on reactor.

- Experiment: is the association between Experimental data, Simulation Data and research data.

- Experimental data: are data taken from the Paper. These represent data take from graphs of experimental experiment. Experimental data represent data point of species of experiment with a variable as time, temperature or pressure. Variable rapresents X-axis and the species are on Y-Axis.

- Simulation Data: are data used for creating input of openSmoke.

- File Paper: is the file where researchers take experimental data. In File paper there are all information about experiments in study as Initial Condition, changes in runtime and Reactor or model used. Students start from these file to take a data and information about these.

- Research Data: are data associated to experiment and simulation used for the research of it. These data are also use to give general information to the experiment.

- Reactor: vessels designet to contain chemical reactions. In our case is the core of the simulation of openSmoke. The interface and input file of openSmoke depends on reactor.

- Specie: species mass fractions used from Reactor to track variation in chemical composition.

- PreProcessor: is the specification of kinetics use. PreProcessor is executing with openSmoke but is independent from the experiment. It is specify and execute at execution time.

- OpenSmoke output: is file output generated by openSmoke. This is tabular file with column with output value from execution of openSmoke. OpenSmoke output contains all species or only species specify in input and variable as temperature, pressure and time depending on the case of reactor.

- Graphics: is graph that represents comparison between experimental data and OpenSmoke output.

3.2.2 Description of product

Present situation

The Department of Chemistry doesn't have a software to manage data. The students store experimental data, input of openSmoke and output of OpenSmoke in a local directory. The researchers create and store in the same place graph of comparison between output of openSmoke and experimental data. The Department has openSmoke software to generate the simulation and creates output file to analyse. This software has a documentation with rule of use and explanation of the structure of input file. These structure depend on the Reactor of simulation and for each reactor there are different filed and SubDirectory of input. Before doing an analysing of the software we study the rule of openSmoke and different input types of it. We must also keep to rules of input and we examine structure of output of it. Beyond openSmoke documentation we have also different type of experimental data and we must create a structure and standards to manage these data and insert it in the database in the correct way. Finally we have to consider an easy way to insert these data and search it. In this way the researcher don't have difficulty to manage his data, use correctly his software and lose his time.

Stakeholders

- The data entry researcher: interested to a system easy to use. This researcher wants an easy system to insert data of simulation, experimental data, research data and file paper. The researcher inserts huge quantities of data for create experiment association. This researcher wants an easy interface. The insert fields must be simple to find and organized in a linear mode. The researcher expects that the association between simulation of OpenSmoke++ and experimental data is easy to do. He expects the same from association between experimental data and File paper.

- The evaluating researcher: is the researcher that research simulations to execute and check comparison between OpenSmoke++ output and experimental data. This researcher must have the possibility to research interested experiments so he expects wide research and he wants personalize the research based on his expectation. The research has to be with general and fundamental field, the student must be able to do all logical operations between fields proposed.

The evaluating researcher can execute PreProcessor so he can select different file of kinetics, termodinamics etc to create PreProcessor's folder used for the simulation. Other important functionality that the evaluating researcher use is the execution of OpenSmoke++, after the selection of experiments the researcher starts the execution and at the end he expects to find in the selected folder graph with comparison between experimental data and OpenSmoke++ output.

Software interface and requirements Server

Other software products required to hardware for support application are

- DBMS(Database Management System):
  - Name: MySQL
  - Mnemonic: MySQL
  - Version number: 5.6.14
  - Source: http://dev.mysql.com/downloads/mysql/
- Java Virtual Machine:
  - Name: Java virtual Machine
  - Mnemonic: JVM
  - Version number: 8
  - Source: https://www.java.com/it/download/
- JDK:
  - Name: Java development kit
  - Mnemonic: JDK
  - Version number: 8
  - Source: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Operating system:
  It is not necessary a specific operating system but it must be able to support DBMS MySql and it must have java virtual machine install on it.

Software interface and requirements Client

For this version of the software Client must be a user of the server where is save the software. They must have a link of the software and install on this account following software:

- OpenSmoke:
  - Name: OpenSmoke++
  - Mnemonic: OS++

  - Source:

- Gnuplot
  - Name: gnuplot
  - Mnemonic: gnu
  - Version Number: 5
  - Source: http://www.gnuplot.info/download.html

3.2.3 Function of the product

Functional Requirements

In the system there is one actor that is the researcher. He is the user of the software and he executes all the function of the system. We describe all requirements below

- Mange of experiment
  - Insert/update simulation
  - Insert/Update experimental data
  - Insert/Update references paper for each experimental data
  - Insert/Update search data
- Manage of File Paper
  - Insert/Update name and authors of File Paper
  - Insert/Update pdf of file Paper
  - Insert/Update references and Image of experimental data of file Paper
- Execution of Simulation
  - Search and selection of experiments
  - Execution of Pre Processor

Non functional requirements

Design that instructs choices to interface is create a logical flow of the work. For doing this we create different form for each data type to insert or update. The main form is the form that include Simulation data and search data, we make this form dynamics respect the selection of the Reactor. In this way the researcher doesn' t have one form for each reactor to choose. The design allows the insert on experimental data linked to the simulation and it allows associating reference paper specification to every experimental data inserted. The design is composed by a form for data search used to simplify to the user the selection and for give a complete visualization of the operation to make. For the output of simulation and comparison of the experiment we organize in hierarchical way all the directory.

3.2.4 System Model

This Section describes all scenarios of functional Requirements. We identify the main scenario and describe it with different actors present in the Department.

Table 3.7. Scenario1

Name of scenario	Add Simulation Data
Goal	Insert data of one simulation in DataBase
Actors	Gohbad
Initial Condition	The user starts the program and selects the form of Insert Experiment
Events	Ghobad selects the reactor of the simulation. The software shows all the input fields about the reactor and all the Sub Dicrectory associated to it. Ghobad decides to insert a Shock Tube Reactor. He views the combo box model and he decides to select IncidentShock model. He inserts type, Incident Shock Velocity and Incident Shock Velocity Measure about the simulation. Then Ghobad wants to insert only 2 SubDicrectories of the simulation, so he clicks button "Before Shock Tube Status", the software opens the form related this Sub Directory, Ghobad inserts Temperature, Pressure in textBoxs, he inserts in the table species and mole associated as H_2 0.8 and O_2 0.2. He verifies that sum of moles is equals to 1, then he saves the subDirectory and he closes form. Thereafter Ghobad clicks on Output-Option sub Directory, the system opens form, he checks Verbose checkbox and he inserts into the table Output Species to study as H_2 O_2 . Ghobad saves the form and close it. Ghobas as a good researcher inserts, in top textbox of the insert experiment form, the keyword associate to the experiment and the range of temperature. Then Ghobad clicks on the save button and closes the form.

Table 3.8. Scenario2

Name of scenario	Add Experimental data from file and Directory
Goal	Insert of Experimental data in the database
Actors	Matteo
Initial Condition	The user has inserted all data about the simulation of OpenSMOKE++ and data about search
Events	Matteo wants associate two experimental data at the simulation. Matteo clicks on Button add Experiment File, the system opens form about experimental data. Matteo has 2 different type of experimental data, the first one is a txt file with the structure x-axis $y - axis_1 y - axis_2$ where y_1 and y_2 represent species H_2 and O_2 , second one is a folder with 2 file txt with the structure x-axis y-axis, two files represent species NC_7H_{16} and O_2 with Temperature as X-axis. Matteo decides to insert txt file, so he selects on the radio button that represents the structure of the file. He clicks on "Add Experiment By File". He selects the file that he wants to insert and the system shows it in the table below. Matteo wants to change the name of the column of this file, he finds the table with the list of the column and changes names of it. Thereafter Matteo inserts experiment by directory, he clicks on the button "Insert experiment by directory" and selects the directory interest. The system shows data in a new Table and Matteo changes names of column again.

Table 3.9. Scenario3

Name of scenario	Add Experimental data from Respecth File
Goal	Insert of Experimental Data from file xml
Actors	Cristina
Initial Condition	The researcher is in the experimental data form, she doesn't insert simulation data yet. She starts from the insert of experimental data to insert experiment
Events	Cristina wants to insert experiment information and experimental data from Respecth file. She clicks on Insert XML Experiment and selects the ReSpecTh file that she wants use. After selection of the file Cristina can show data points about the experimental data inserted in the table of data point experiment. She doesn't change names of the columns because are all correct and the references of file paper is already associated as the system reads it from the file selected. Cristina closes the experimental data form. She can see reactor and model that the system takes from the file. She opens the subDirectory associated to Initial condition of experimental data and she verifies the condition that the system reads from paper, finally she saves this subDirectory.

Table 3.10. Scenario4

Name of scenario	Export of Respecth File
Goal	Export Respecth file from experiment
Actors	Prof. Faravelli
Initial Condition	Professor Faravelli opens an experiment that was inserted before, the system opens the experiment form.
Events	Professor Faravelli wants to create ReSpecTh file of one experiment to send to Budapest Unversity. He clicks on Create XML File, the system proposes to Professor Dialog form and he selects the form where to save the file. At the end of execution the system opens the xml file that he creates. Now Faravelli can open the specified Directory, he sends the created file.

Table 3.11. Scenario5

Name of scenario	Search Experiments
Goal	Make a list of experiment interested by selection of search data
Actors	Matteo
Initial Condition	Matteo opens on Search and Execute panel
Events	Matteo wants a list of experiments of Batch Reactor with Temperature in the range of [200-1000] K and that contains in GasStatus directory species NC_7H_{16} and O_2. He is on the search Form, he select Batch Reactor in the combobox of Reactor, he selects the logical operation AND associated to the temperature and write range of temperature. Then Matteo selects the logical operation AND near Species of experiments and write into textArea the logical operation NC7H16+O2. After these passages Matteo clicks on the button of research. The system Shows the list of experiments with specifications that are needed.

Table 3.12. Scenario6

Name of scenario	Update experiment
Goal	Update initial condition of experiment, his range of search and columns name of experimental data
Actors	Ghobad
Initial Condition	Ghobad has executed search of experiment and he has the list of experiment selected
Events	Ghobad has a list of experiments and wants to update experiment number 3 that is a Batch Reactor. He clicks on the row where there is experiment number 3, the system shows a messageBox that ask to Ghobad if he wants update the experiment. He choses yes. The system opens the same form of insert experiment , all fields of the form are compiled with the value of experiment. Ghobad wants update initial condition so click on the button "Initial Status" and change the Temperature from 300 K to 200 K and the pressure from 1 atm to 1 bar. Then he clicks on update SubDirectory and closes this form. Ghobad wants change the range of search of temperature and the unit of measure of pressure, so he goes in Minimum Temperature and changes its value from 300 K to 200 K then he goes to Pressure fields and changes the unit of measure from atm to bar. Finally Ghobad wants change columns name of the experimental data associated to experiment. He opens the form of experimental data, selects the table of experimental data points and in the table of names column he changes the name of x-column from T[K] to P[atm] as the name of OpenSMOKE++ output column. He closes the form of experimental data and clicks on updating experiment.

Table 3.13. Scenario7

Name of scenario	Execute Pre Processor
Goal	Create directory of Pre processor of symulation
Actors	Cristina
Initial Condition	Cristina is in Search form
Events	Cristina wants to execute PreProcessor for create folder relative to it. She clicks on checkbox Execute PreProcessor and the system shows her panel of preprocessor. She inserts file about Kinetics, Thermodynnamics and Transport, then she chooses the output folder where she wants save preProcessor. Finally she clicks on Execute Kinetics and the system runs and creates preProcessor to use for simulation.

Table 3.14. Scenario8

Name of scenario	Execute OpenSMOKE++
Goal	Execute OpenSMOKE++ simulation for 3 experiment selected
Actors	Tom
Initial Condition	Tom is in the search panel, he has selected the pre processor directory that he has create before and give the search of experiments by condition.
Events	Tom wants execute some experiments from the list obtained by the search. He selects the check box that corrispond to experiments that he wants execute. After selected all experiments interested Tom clicks on execute experiment and chooses the folder where he wants see the results. At the end of execution the system shows a txt file with the list of experiments without resolution.

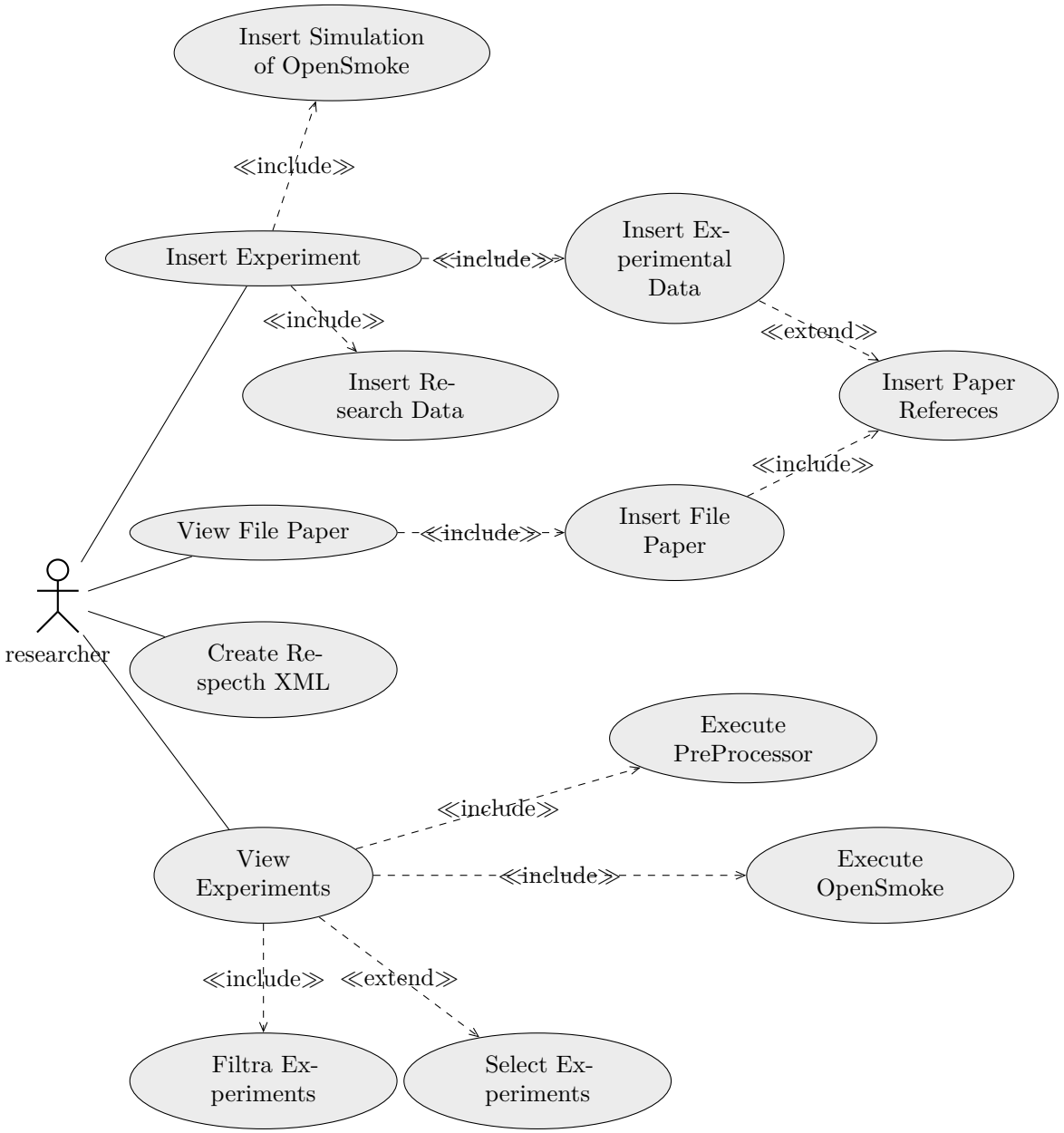
Table 3.15. Scenario9

Name of scenario	View comparison of expeirment
Goal	View comparison between expeirmental data and simulation of OpenSMOKE++
Actors	Matteo
Initial Condition	Matteo has executed experiments selected from search data list
Events	Matteo wants to view the results of simulation of OpenSMOKE++ and plots of comparation of different species. He opens the directory selected at the moment of execution. In the directory he views one directory for each experiments. In the directory of experiment there is a input file of OpenSMOKE++, the output of simulation and the folder of output. The graphics with comparison between species of simulation and species of experimental data is in output folder. Matteo opens the folder, he runs the gnuplot file created and he opens the file png created with gnuplot script. In the image there are graphics about comparison.

Table 3.16. Scenario10

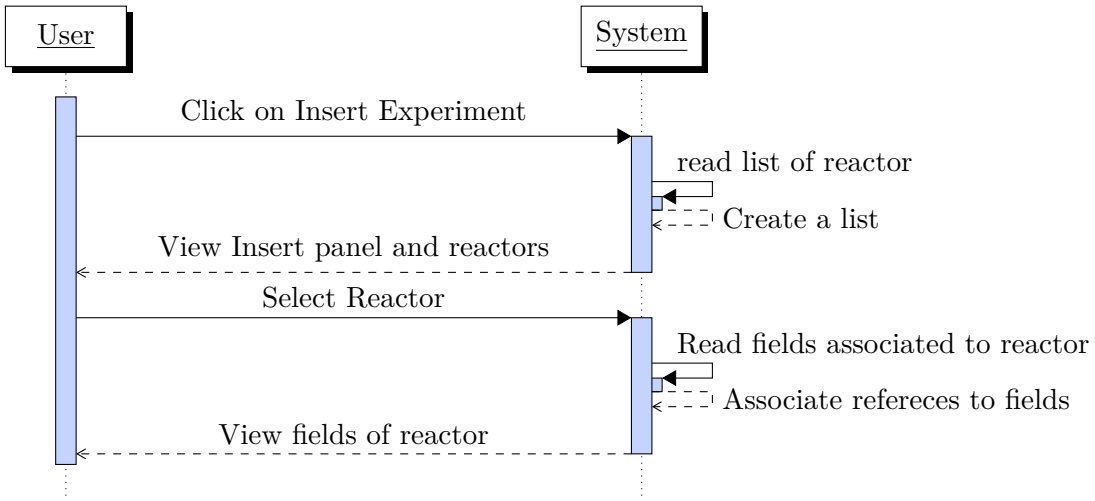
Name of scenario	Test of OpenSMOKE++ simulation
Goal	Testing of OpenSMOKE++ execution in one experiment
Actors	Gohbad
Initial Condition	The user opens the experiment that he wants testing
Events	Gohbad wants to testing a experiment for checking errors in OpenSMOKE++ input. In the form of experiment Gohbad find a panel with button to testing. He wants test the experiment with two different preProcessor created before.First he click on the button of PreProcessor folder and select the directory where it is in. Later he click on button of execution of PreProcessor and selects folder of output. At the end of the first execution Gohbad can check results of simulation. Now he wants test the same experiment with other PreProcessor. He selects directory belong to second PreProcessor and executes again OpenSMOKE++ test.

3.2.5 Use Cases


Insert Simulation of OpenSmoke
≪include≫
Insert Experiment
≪include≫
Insert Experimental Data
≪include≫
Insert Research Data
≪extend≫
Insert Paper Refereces
≪include≫
View File Paper
≪include≫
Insert File Paper
researcher
Create Respecth XML
Execute PreProcessor
≪include≫
View Experiments
≪include≫
Execute OpenSmoke
≪include≫
≪extend≫
Filtra Experiments
Select Experiments

1. Select reactor

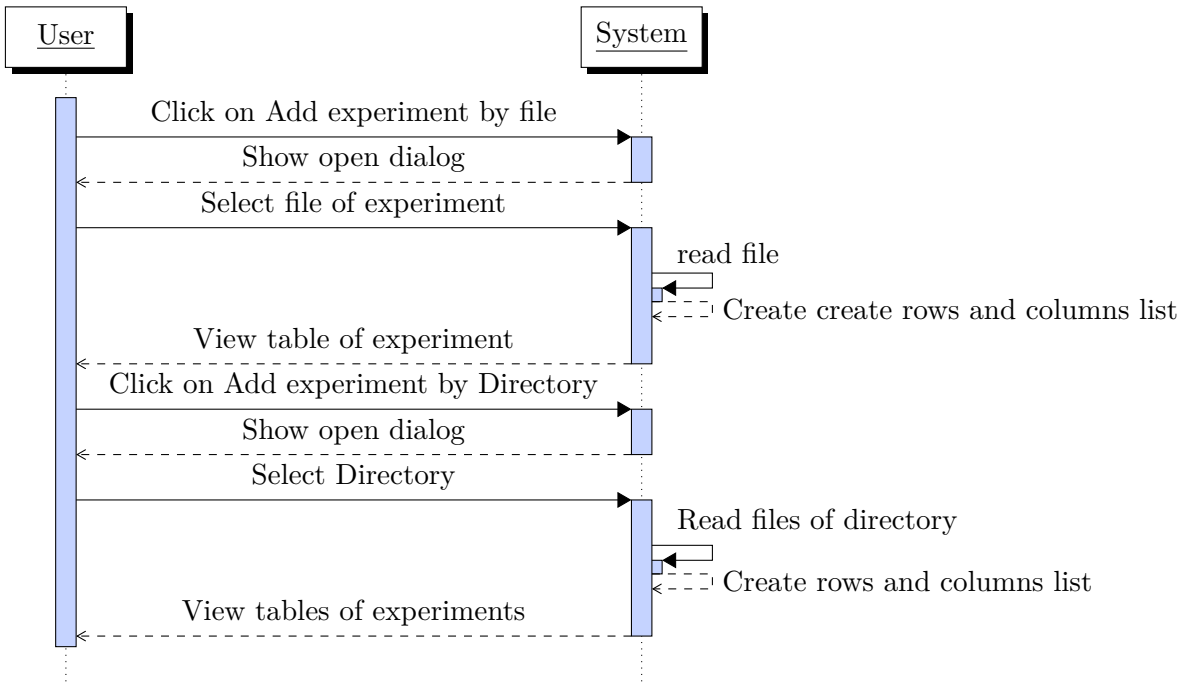
Description	The user opens insert form and select reactor
Goal	User opens insert form and views initial option
Actor	Data entry researcher
Entry condition	The user opens the software
Exit condition	The user is ready to insert data
Flusso d'eventi	- 1. The user opens Insert Experiment panel 2. The system reads list of reactor 3. The system views the Insert panel and combo box of reactor 4. The user selects reactor 5. The system reads element of reactor 6. The system assigns reference of table, attribute and dataType to object to view 7. The system shows to the user fields for insert expreriment 8. The user can inserts all fields of experiment



2. Experimental Data insert

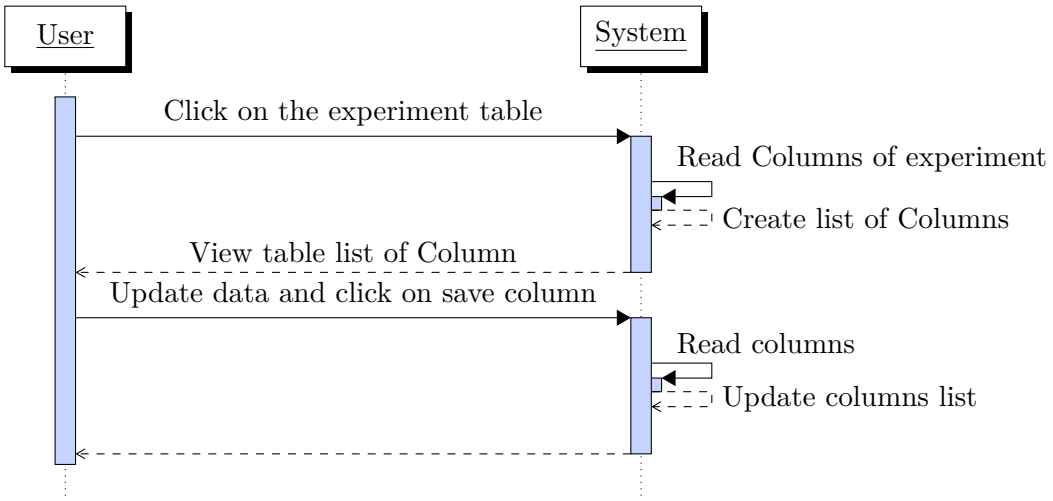
Description	Add experimental data
Goal	Add experimental data to experiment
Actor	Data entry researcher
Entry condition	Click the button of insert experimental data and View panel of experimental data. The researcher Insert two experiments with different type
Exit condition	Experimental data is associated to experiment user see two table with two experiment insert
Flusso d'eventi	

1. The user selects radio button of structure of file and click Add experiment
2. The system shows to user open dialog
3. The user selects the file
4. The system reads file
5. The system creates row and columns of the table
6. The system adds and view the table in tab panel
7. User select adds experiment by directory
8. The system shows open dialog
9. The user selects folder with files
10. The system reads files
11. The system creates rows and columns of the experiment
12. The system adds and view a table in tab panel
13. The user can see two table with two experiments



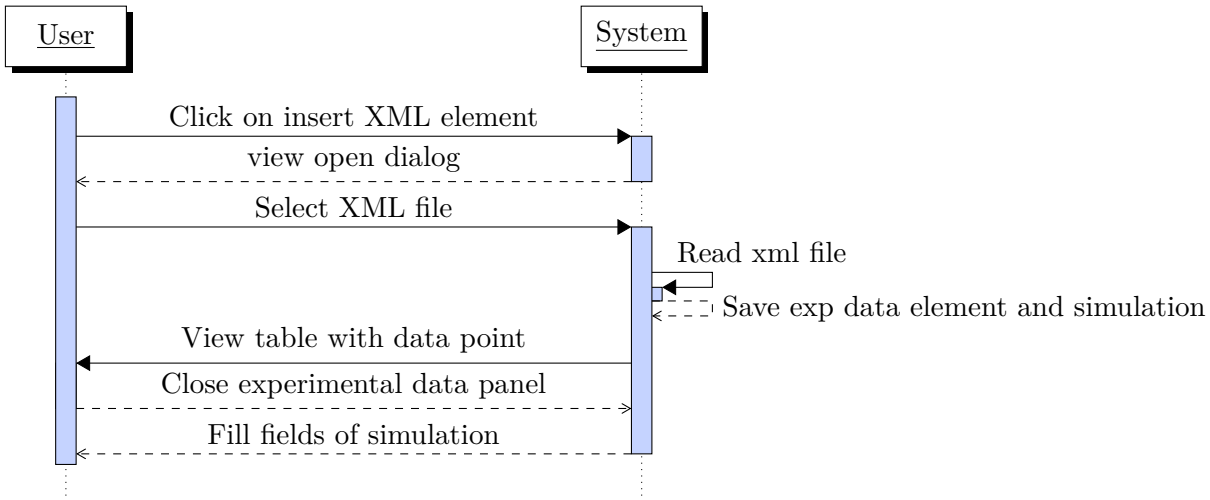
3. Manage experimental data column

Description	Change names of columns experimental data
Goal	For each column of experimental data change the name of columns and references paper
Actor	Data entry researcher
Entry condition	The user has inserted experimental data associated to experiment. The user view tables of experimental data.
Exit condition	The name and definition of axis of the columns experiment be change
Flusso d'eventi	1. The user clicks on the table 2. The system reads columns of experiment 3. The system views table with list of columns 4. The user updates data 5. The user clicks on save columns definition 6. The system updates list of columns



4. Insert Respecth XML

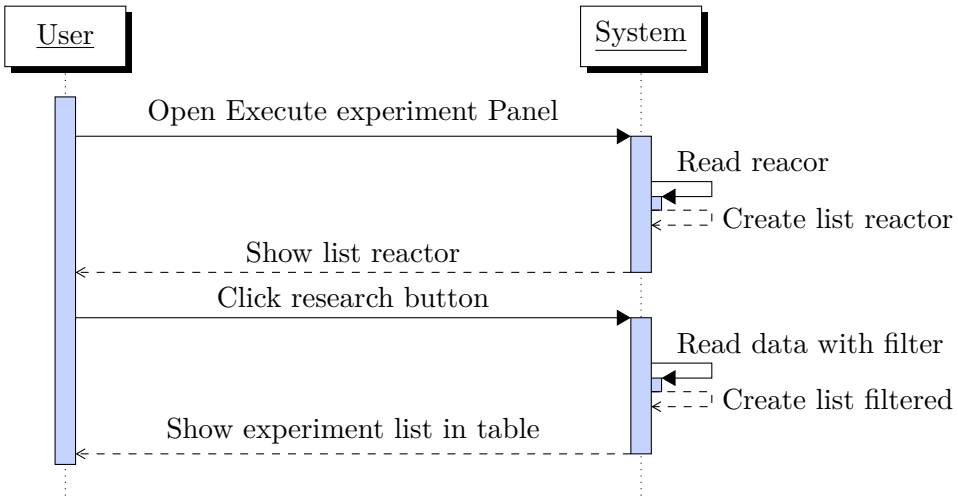
Description	Insert Respecth experiment
Goal	Insert Respecth experiment and view all data
Actor	Data Entry researcher
Entry condition	The researcher is in the experimental data insert panel
Exit condition	Fields of reactors and initial condition are selected in insert experiment panel
Flusso d'eventi	1. The user clicks on insert XML Experiment 2. The system views open dialog 3. The user selects xml file 4. The system reads xml file 5. The system saves reactor, model, initial condition and create lists of columns and rows 6. The system views table with data point of experiment 7. The User closes experimental panel 8. The system writes Reactor, Model and initial condition in the insert experiment panel 9. The user views fields of Reactor model and initial condition fill



5. Research experiments

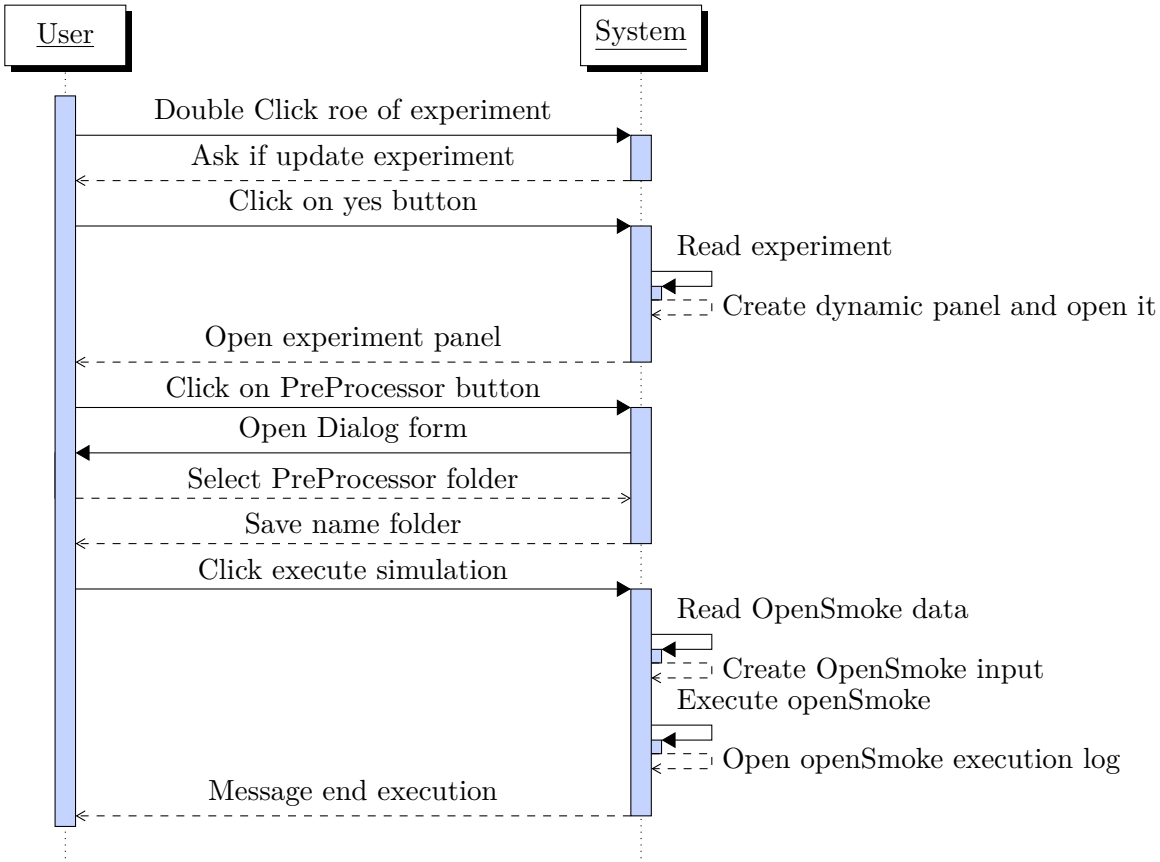
Description	Research Experiment
Goal	Research experiment by research form
Actor	The evaluating researcher
Entry condition	The user start software
Exit condition	The user view list of experiment look for
Flusso d'eventi	

1. The user opens Execute Experiement Panel
2. The system reads reactor and create a list
3. The system shows list
4. The Users fills in fields of research data
5. The system filters experiment
6. The system shows list of experiment in a table
7. The User views experiment list



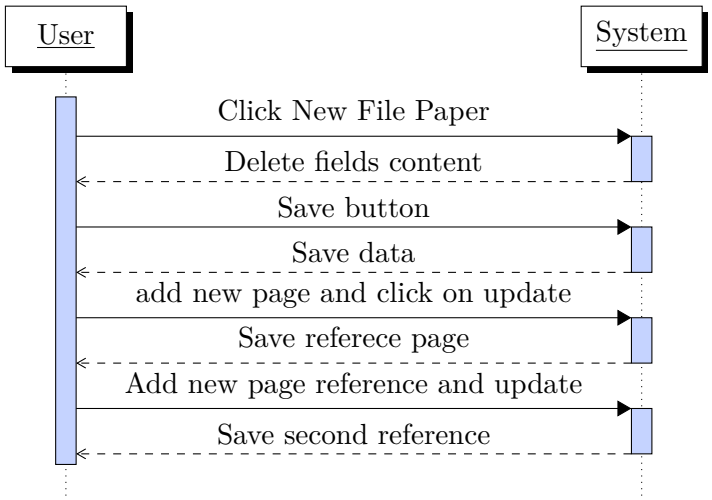
6. Test openSmoke simulation

Description	Testing OpenSmoke
Goal	Test a insert simulation
Actor	evaluating researcher
Entry condition	The users is in execute panel and he has the list of experiments
Exit condition	Execution of openSmoke for experiment selected
Flusso d'eventi	1. The users double clicks on experiment row 2. The system asks to user if he want update experiment 3. The users clicks on Yes button 4. The system reads experiment 5. The system creates dynamic panel 6. The system opens experiment panel 7. The users clicks on PreProcessor button 8. The system opens dialog form 9. The user selects folder of PreProcessor 10. The system saves name of folder 11. The user clicks on execute simulation 12. The system reads data of simulation and sub directory 13. The system creates input of openSmoke 14. The system executes openSmoke 15. The system views file log of execution 16. The system advises the end of execution


User
System
Double Click roe of experiment
Ask if update experiment
Click on yes button
Read experiment
Create dynamic panel and open it
Open experiment panel
Click on PreProcessor button
Open Dialog form
Select PreProcessor folder
Save name folder
Click execute simulation
Read OpenSmoke data
Create OpenSmoke input
Execute openSmoke
Open openSmoke execution log
Message end execution

7. Insert Paper refereces

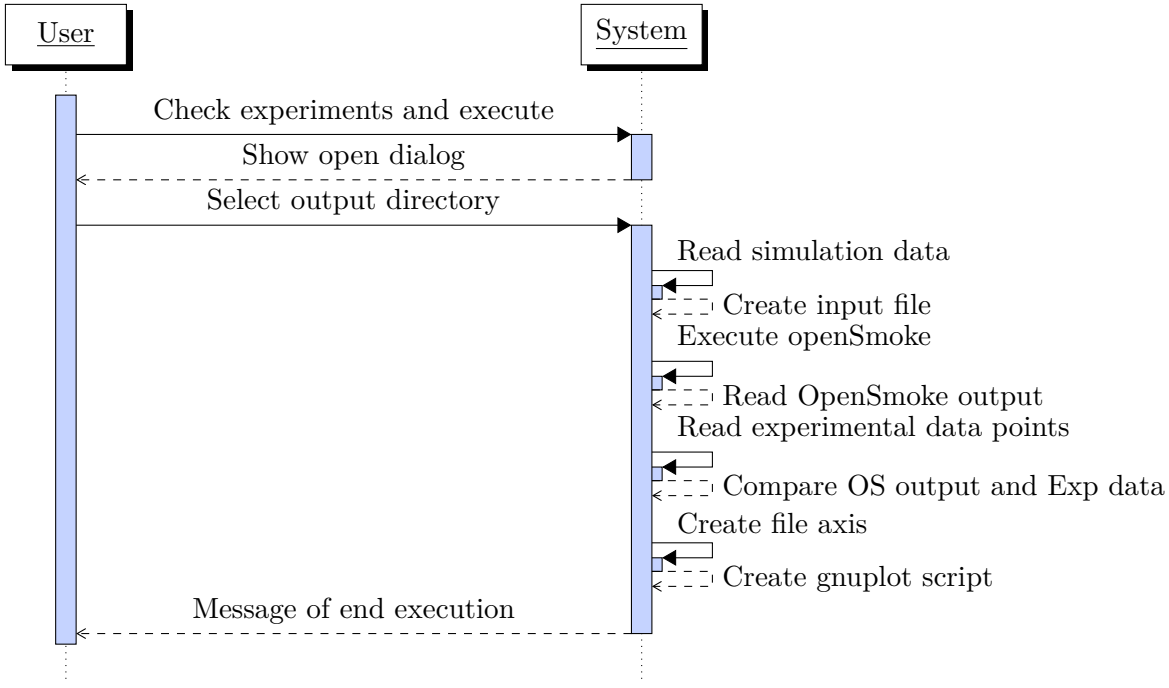
Description	Insert of file Paper
Goal	Insert new File Paper
Actor	Evaluating researcher
Entry condition	The user open manage file paper panel
Exit condition	The user have insert new File paper references
Flusso d'eventi	1. The user Clicks on New file paper 2. The system deletes content in all fields of interface 3. The user inserts references of file paper and click on save button 4. The system saves data 5. The user adds new page reference and click on update 6. The system saves reference page 7. The user adds new page reference and click on update 8. The system saves second reference page



8. Execute simulation

Description	Execute simulations
Goal	Execute simulation od openSmoke of selected experiments
Actor	Evaluating researcher
Entry condition	The user research experiments in Execute Experiment panel
Exit condition	The system execute all simulation of experiments selected
Flusso d'eventi	

1. The user checks experiments and click on execute
2. The system shows open dialog
3. The user selects Directory of output
4. The system reads data of simulation
5. The system creates input file
6. The system executes openSmoke
7. The system reads output of openSmoke
8. The system reads experimental data point
9. The system compares OpenSmoke output and experimental data point
10. The system creates file with XY for each specie
11. The system creates gnuplotScript for create graph
12. The system advises the end of execution



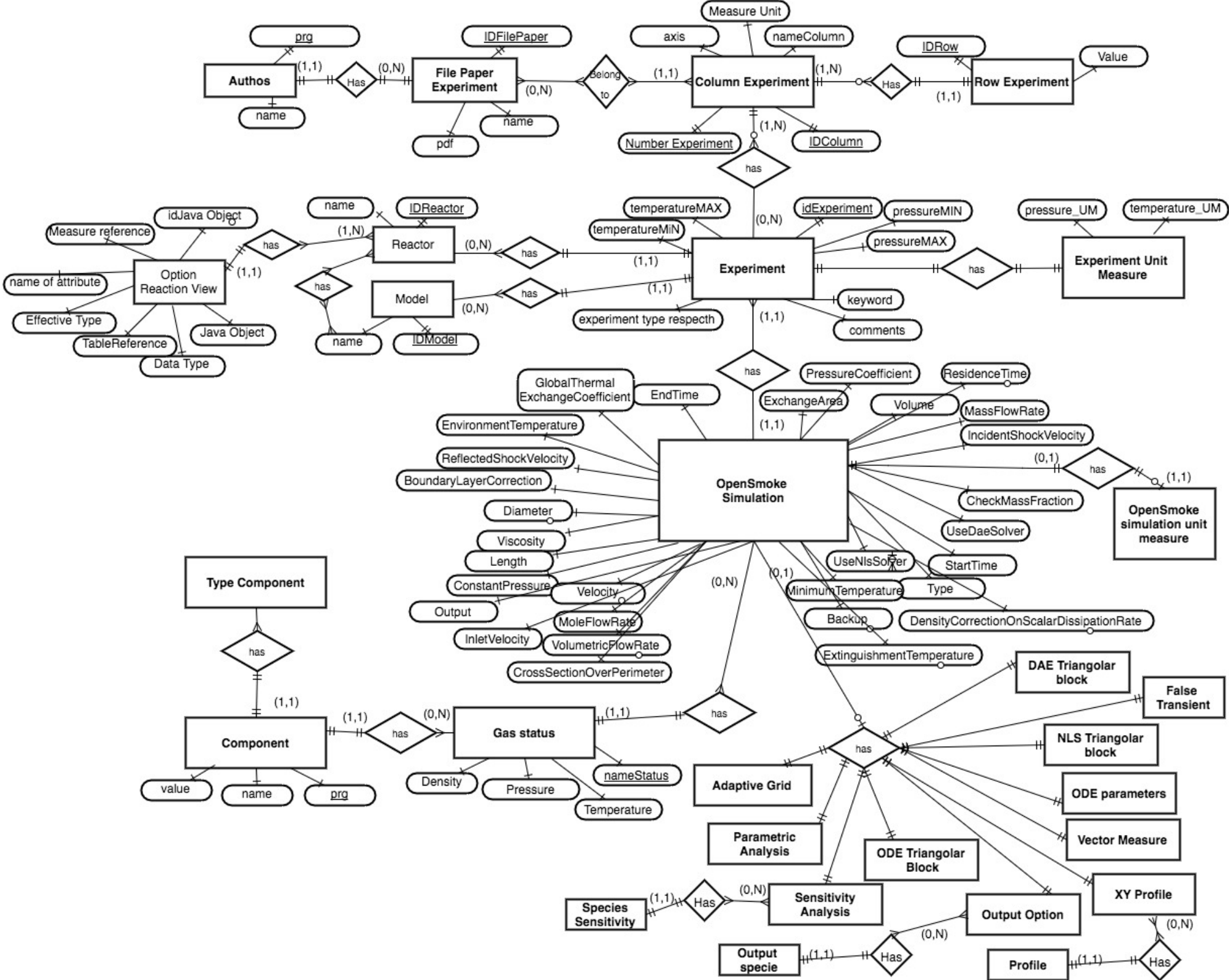
3.2.6 Design Document

The purpose of Design document is to translate the users requirements and users processes into a technical design that will be used to develop the application. In this section is explain design choice and make clear the structure of software.

Entity Relationship model

The main entity in the system is Experiment. This entity is the link between simulation of openSmoke data, experimental data and research data. Experiment contains all fields of research data, it is in relation with Experiment Unit of Measure that contains its unit of measure. OpenSmoke Data and OpenSmoke data Unit of Measure contain all fields of different reactors, they are in relation with id of Experiment. Experimental data are represented from the table DataColumnExperiment and DataRowExperiment, these entities represent experimental data points, each experimental data point belongs to experiments is identify by fields numberOfExperiment. Entities of experimental data are in relation with main entity Experiment by idExperiment primary key. The definition of Entities is split in five set, the first one is set of experiment with main entity Experiments, this set contains entity in relation with openSmoke and experimental data. Second important set contains SubDictionary definition. Since openSmoke contains 12 SubDictionaries, their use depending on reactor, an entity has been created for each subdirectory. These entities can be in relation with other entities that represent lists of data. Entities that represent initial status are GasStatus and ComponentExperiment, the first one contains Temperature, Density, pressure and equivalence ratio. GasStatus is in relation with primary key idExperiment of OpenSmokeData. An experiment can have more gasStatus in its definition. The second one contains rows of moles fraction species of reaction in relation with GasStatus by idExperiment and name of gas status. A mole fractions species can have more type of component for each nameStatus. Entities that represent Output Option and Output Species are OutputOption and Species, first one is connected with idExperiment of OpenSmokeData. The second one is connected with idExpeirment of OutputOption and contains list of output species. It has been necessary to create the same relation between SubDirectory of ParametricAnalysis, in relation to parametricsListValue, XY Profile in relation with profile that represent x-Axis and y-axis point. Another relation between SubDictionary and its list is between SensitivityAnalysis and SpeciesSensitivity. Other entities of subDictionary refer to AdapriveGrid, DAETriangolarBlock, FalseTransient, NLSTriangolarBlock, ODEParameters, ODETriangolarBlock and VectorMeasure. Third set of entities is creates for storing file paper data. It has been created three entities to represent it. The first one is FilePaper that contains list of name of file , second one is entity about authors of file paper is in relation with File paper by idPaper. Third entity is the relation between file paper and experimental data associated to experiment, this table is in relation with id of file paper and primary key of experimental data column. It has been needed create a entities that contains a list of column for each experimental data. These types of entities are content in fourth set. Major entities

included in this set are Reactor and Type Model, these entities are foreing key of correspondent fields in Experiment. Reactor is in relation with Type Model because for each reactor there are different type model associated. The other list entity in relation with major entity ComponentsExperiment is list of type components that contains different type of initial status species. Other entities are list of combo box value, It has been create them to read element. This entities are not in relation with others. Last but not least set contains OptionReactorView entity. This entity is use to create dynamic interface by selection of reactors. It is in relation with Reactor entity. For each reactor OptionReactorView contains list of interface object to create. It has been necessary to specify for each object of the table, name of fields, data type of openSmoke and data type of database. With this specification we can create dynamical interface and dynamical query of insert, update and delete.


prg
Authos
name
Has
(1,1)
(0,N)
IDFilePaper
File Paper Experiment
name
pdf
Belong to
(0,N)
(1,1)
Measure Unit
axis
nameColumn
Column Experiment
(1,N)
Has
IDRow
Row Experiment
Value
(1,1)
(1,N)
Number Experiment
IDColumn
has
(0,N)
idExperiment
pressureMIN
pressureMAX
temperatureMAX
temperatureMIN
Experiment
keyword
comments
experiment type respecth
(1,1)
has
pressure_UM
temperature_UM
Experiment Unit Measure
name
IDReactor
Reactor
(1,N)
(0,N)
has
Model
name
IDModel
idJava Object
Measure reference
Option Reaction View
name of attribute
Effective Type
TableReference
Data Type
Java Object
(1,1)
GlobalThermal ExchangeCoefficient
EndTime
PressureCoefficient
ResidenceTime
ExchangeArea
Volume
MassFlowRate
IncidentShockVelocity
EnvironmentTemperature
ReflectedShockVelocity
BoundaryLayerCorrection
Diameter
Viscosity
Length
ConstantPressure
Velocity
Output
MoleFlowRate
InletVelocity
VolumetricFlowRate
CrossSectionOverPerimeter
OpenSmoke Simulation
(0,1)
(1,1)
CheckMassFraction
UseDaeSolver
UseNlsSolver
StartTime
MinimumTemperature
Type
Backup
DensityCorrectionOnScalarDissipationRate
ExtinguishmentTemperature
OpenSmoke simulation unit measure
Type Component
Component
value
name
prg
(1,1)
(0,N)
Gas status
Density
Pressure
Temperature
nameStatus
DAE Triangolar block
False Transient
NLS Triangolar block
ODE parameters
Vector Measure
XY Profile
Adaptive Grid
Parametric Analysis
ODE Triangolar Block
Species Sensitivity
Sensitivity Analysis
Output Option
Output specie
Profile

Logical Model

It has been needed introduce relation between tables to switch from Entity Relationship model to logical model. Relation are between entities of the same set.

- The primary key of Experiment is the attribute idExperiment. It is the main primary key of the project. Experiment entity is in relation with
  - OpenSmokeData, there is a reference between id of Experiment and the primary key idExperiment of OpenSmokeData.
  - DataColumnExperiment there is a reference between id of Experiment and attribute idExperiment of DataColumnExperiment.

  Experiment contains temperature and pressure default that is value transformed in the measure unit default. This is use for research by temperature and pressure. Default measures are Kelvin for temperature and atmosphere for pressure.
- DataColumnExperiment has primary key composed by idExperiment, numberOfExperiment and idColumn.
- PaperExperiment has been created by relation between DataColumnExperiment and File Paper. Paper Experiment contains id of Paper Experiment and id Of Data column to represent the association.
- Reactors and Model are foreign key of Expeirment. The experiment contains idReactor and idModel referenced.
- Reactors and model are in relation by the table M_ReactorModel. This table contains associations between a reactor and models.
- All SubDictionary table have as primary key idExperiment referenced with OpenSmokeData except GasStatus.
- GasStatus has primary key compose by idExperiment and GasStatus name. An experiment can have more initial status with different name.
- Components Experiment is in relation with GasStatus. It has been created a primary key composed by primary key of GasStatus, component id and progressive of row.
- OptionReactorView has reactor id as foreign key.

We list tables with primary key
AuthorPaper($\underline{idFilePaper}, \underline{prg}$, name)

Data_ColumnExperiment($\underline{idExperiment}, \underline{idColumn}, \underline{numberExperiment}$, nameColumn, measureUnit,Axis)

Data_RowExperiment($\underline{idExperiment}, \underline{idColumn}, \underline{numberExperiment}, \underline{idRow}$, value)

E_Experiment(*idExperiment* , idReactor, pressureMIN, pressureMAX, temperatureMIN, temperatureMAX, keyword, temperatureMIN_Default, temperatureMAX_Default, pressureMIN_Default, pressureMAX_Default, comments, experimentTypeReSpecTh)

E_Experiment_UM(*idExperiment*, pressure_UM, temperature_UM)

E_AdaptiveGrid(*idExperiment*, type, length, length_UM, initialPoint, center, center_UM, width, width_UM, fixedPoint, fixedPoint_UM, maxPoint, maxAdaptivePoint, GradientCoefficient, CurvatureCoefficient, threshold, regridPoint)

E_DAETriangolarBlock(*idExperiment*, daeSolver, relativeTolerance, absoluteTolerance, maximumNumberOfSteps, maximumStep, minimumStep, initialStep, maximumOrder, linearAlgebra, meanResidual, verbosityLevel, preconditioner, Jacobian, sparseSolver)

E_ComponentExperiment(*idExperiment*, *idTypeComponent*, *nameGasStatus*, *prg_component*, name, value)

E_ParametricValues(*idExperiment*, *prg* , value)

E_Specie(*idExperiment*, *prg_specie*, name_specie)

E_SpecieSensitivity(*idExperiment*, *prg_specie*, name_specie)

Ex_adaptivegrid1d(*idExperiment* ,Type, Length, Length_UM, InitialPoints, Center, Center_UM, Width, Width_UM, FixedPoint, FixedPoint_UM, MaxPoints, MaxAdaptivePoints, GradientCoefficient, CurvatureCoefficient, Threshold, RegridPoints)

Ex_DaeTriangolarBlock(*idExperiment*, DaeSolver, RelativeTolerance, AbsoluteTolerance, MaximumNumberOfSteps, MaximumStep, MinimumStep, InitialStep, MaximumOrder, LinearAlgebra, MeanResidualThreshold, VebosityLevel, Preconditioner, Jacobian, SparseSolver)

Ex_FalseTransient(*idExperiment*,FalseTransientSolver, Tolerance, StepTolerance, RelativeTolerance, RelativeError, MaximumNumberOfIterations, MaximumSetupCalls, MaximumSubSetupCalls, Scaling)

3.2.7 Application Design

Develop of software it has been needed create an application multi platform. In this way optimal solution is create a desktop application with simple way to iterate with database. The application is develop in java. The software run on desktop connected with Polyecnic network. Data are stored in MySql database. Database Schema calls openSmoke. Server Contains java controller and MySql connector. The application has Client-Server structure and it executes external program as openSmoke and Guplot. In conclusion levels of application are Client, Web Server and Data Server

3.2.8 Navigation Model

It has been necessary divide the model description in tree parts to simplify the structure. Three set are Insert experiment, Research experiment and Manage of File Paper. It has been needed specify the symbol $ that indicate that the page are reachable from all other page.

Insert Experiment

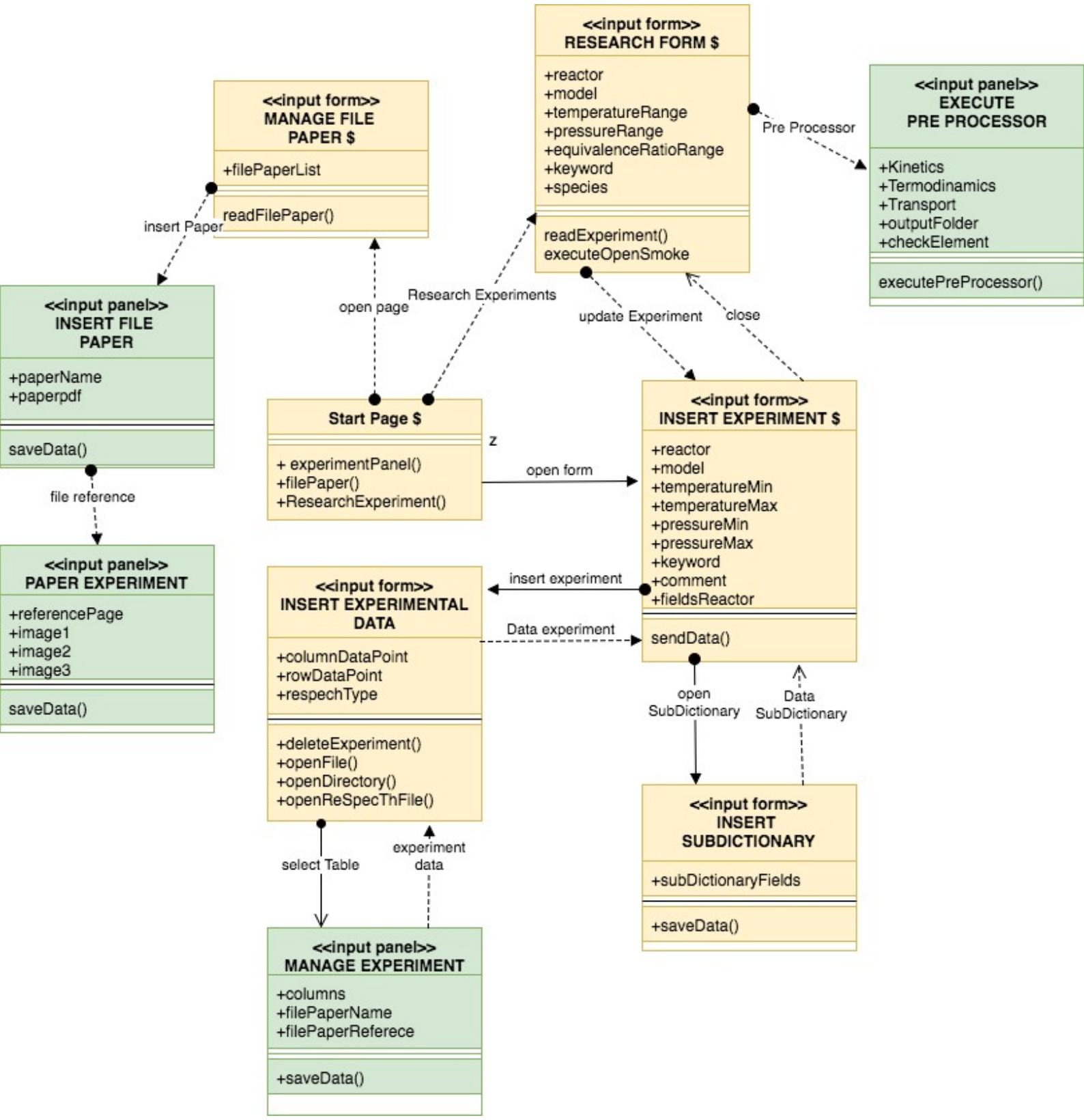
From starting page the user can insert experiment by click Experiment panel button. Experiment panel page is dynamic. When change reactor the system refresh the page. From Experiment panel we can open experimental data panel, a insert of experiment file the page show OpenDialog form to the user. From table of experiment we can go to panel of experiment column and file paper experiment to change information about it. In experiment panel by click the button the system open SubDictionary page depending on name of button. There is one page for each subdirectory, in this schema we group all pages in one generic class. Fields of simulation insert are dynamic, we group them in one definition as fieldsReactor.

Research Experiment

Research experiment panel is reachable from any page. Execute PreProcessor panel is reachable from this page. By click the experiment row research panel open Insert Experiment input form and pass it all data about experiment.

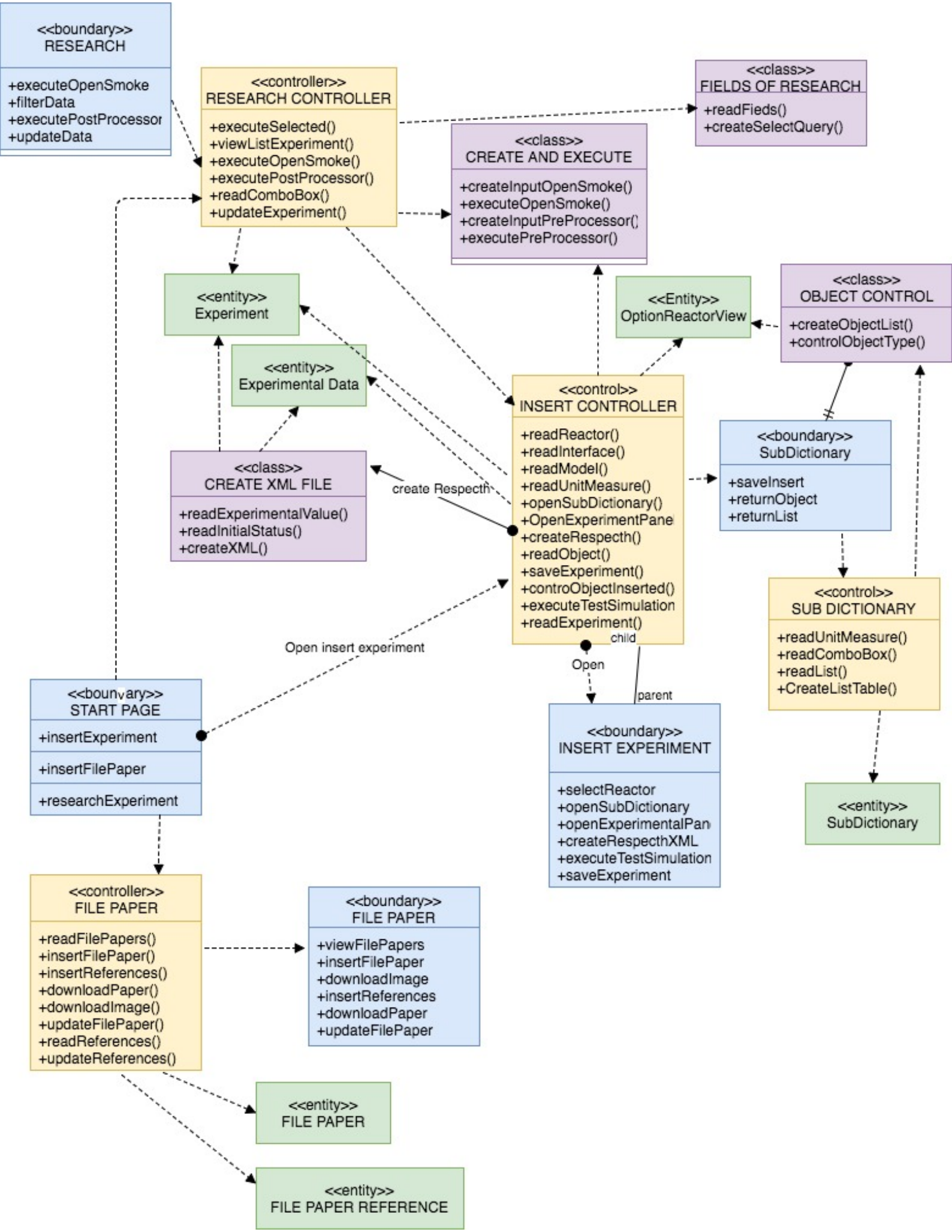
File Paper

From Start page we can go to Manage file paper form. In this form is group different panel to mange file paper and experimental data associated. File paper page is connected with insert File paper Panel. Paper experiment panel is reachable by click on experimental references in insert file paper page.


<<input form>>
RESEARCH FORM $
+reactor
+model
+temperatureRange
+pressureRange
+equivalenceRatioRange
+keyword
+species
readExperiment()
executeOpenSmoke
Pre Processor
<<input panel>>
EXECUTE
PRE PROCESSOR
+Kinetics
+Termodinamics
+Transport
+outputFolder
+checkElement
executePreProcessor()
<<input form>>
MANAGE FILE
PAPER $
+filePaperList
readFilePaper()
insert Paper
<<input panel>>
INSERT FILE
PAPER
+paperName
+paperpdf
saveData()
file reference
<<input panel>>
PAPER EXPERIMENT
+referencePage
+image1
+image2
+image3
saveData()
open page
Research Experiments
update Experiment
close
Start Page $
+ experimentPanel()
+filePaper()
+ResearchExperiment()
z
open form
<<input form>>
INSERT EXPERIMENT $
+reactor
+model
+temperatureMin
+temperatureMax
+pressureMin
+pressureMax
+keyword
+comment
+fieldsReactor
sendData()
insert experiment
Data experiment
<<input form>>
INSERT EXPERIMENTAL
DATA
+columnDataPoint
+rowDataPoint
+respechType
+deleteExperiment()
+openFile()
+openDirectory()
+openReSpecThFile()
open
SubDictionary
Data
SubDictionary
<<input form>>
INSERT
SUBDICTIONARY
+subDictionaryFields
+saveData()
select Table
experiment
data
<<input panel>>
MANAGE EXPERIMENT
+columns
+filePaperName
+filePaperReferece
+saveData()

BCE Component

Components are modeling using Entity-Control-Boundary model. It has been necessary to permit to develop software by Model View Controller pattern.


<<boundary>>
RESEARCH
+executeOpenSmoke
+filterData
+executePostProcessor
+updateData
<<controller>>
RESEARCH CONTROLLER
+executeSelected()
+viewListExperiment()
+executeOpenSmoke()
+executePostProcessor()
+readComboBox()
+updateExperiment()
<<class>>
FIELDS OF RESEARCH
+readFieds()
+createSelectQuery()
<<class>>
CREATE AND EXECUTE
+createInputOpenSmoke()
+executeOpenSmoke()
+createInputPreProcessor()
+executePreProcessor()
<<entity>>
Experiment
<<entity>>
Experimental Data
<<Entity>>
OptionReactorView
<<class>>
OBJECT CONTROL
+createObjectList()
+controlObjectType()
<<control>>
INSERT CONTROLLER
+readReactor()
+readInterface()
+readModel()
+readUnitMeasure()
+openSubDictionary()
+OpenExperimentPane
+createRespecth()
+readObject()
+saveExperiment()
+controObjectInserted()
+executeTestSimulation
+readExperiment()
<<class>>
CREATE XML FILE
+readExperimentalValue()
+readInitialStatus()
+createXML()
create Respecth
<<boundary>>
SubDictionary
+saveInsert
+returnObject
+returnList
<<control>>
SUB DICTIONARY
+readUnitMeasure()
+readComboBox()
+readList()
+CreateListTable()
<<entity>>
SubDictionary
Open insert experiment
child
Open
parent
<<boundary>>
START PAGE
+insertExperiment
+insertFilePaper
+researchExperiment
<<boundary>>
INSERT EXPERIMENT
+selectReactor
+openSubDictionary
+openExperimentalPan
+createRespecthXML
+executeTestSimulation
+saveExperiment
<<controller>>
FILE PAPER
+readFilePapers()
+insertFilePaper()
+insertReferences()
+downloadPaper()
+downloadImage()
+updateFilePaper()
+readReferences()
+updateReferences()
<<boundary>>
FILE PAPER
+viewFilePapers
+insertFilePaper
+downloadImage
+insertReferences
+downloadPaper
+updateFilePaper
<<entity>>
FILE PAPER
<<entity>>
FILE PAPER REFERENCE

Chapter 4

Software changes

During the development of the software the introduction of two features has been necessary. The first is introducing the calculation of Batch reaction Ignition Delay Time(IDT). Ignition delay time is the time interval between the start of injection and the start of combustion, some expeirmental data of batch reactor has to be compared with the IDT. The output of openSMOKE++ doesn't supply the IDT calculation, there is an external software "PostProcessor_IDT" that calculates it. The second is the introduction of CurveMatching calculation, for each simulation it has been necessary the the output of execution of openSmoke and compare the different model at the input by the User.

4.1 Ignition Delay Time

To calculate ignition delay times (i.e. the time needed for a fuelair mixture to auto-ignite without a spark or pilot flame) after the execution of OpenSMOKE++ a post processing step was needed.

The scenario to the use of Ignition Delay Time is show in the table 4.1:

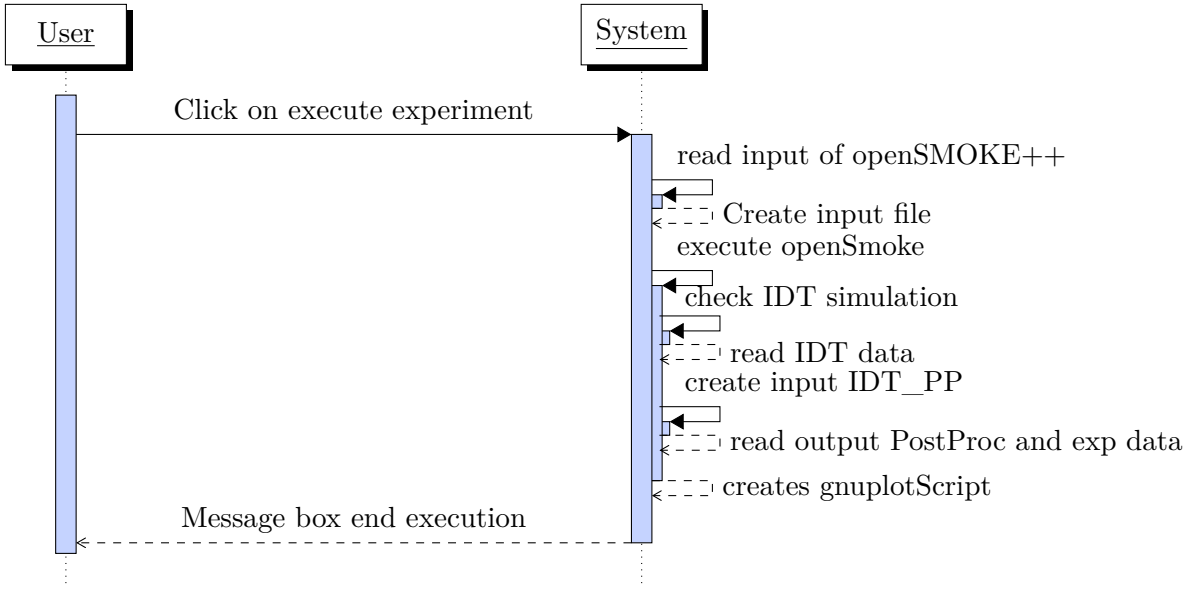
Table 4.1. IDT Scenario

Name of scenario	Insert Ignition Delay Experimental data
Goal	Signal Ignition Delay experimental data for the post processor
Actors	Gohbad
Initial Condition	The User is in the experimental data panel, the user adds experimental data that must be compared with post processor execution
Events	Gohbad wants signal that experimental data just insert must be compare with output of openSMOKE++ post processoring with IDT post Processor. He clicks on the check box IDT and he views the panel with fields of IDT post-Processor. Gohobad inserts into keyspec COH2, he selects Output unit Time equals ”hr” and Output unit Temperature equals ”100T”. He close the experimental data panel and save the experiement.

At the execution of simulation of openSmoke the software create the input file of post processor, later the system execute post processor IDT and take its output to compare experimental data. The user case of this step is:

Description	The user execute simulation that need IDT_PostProcessor execution
Goal	The user can can check the match between experimental data and output openSmoke execution
Actor	Insert data student
Entry condition	The user research Batch experiment with IDT experimental data
Exit condition	The user execute gnuplot script and view graph of comparison
Flusso d'eventi	

1. The user checks experiment to simulate and click on execute simulation
2. The system reads data of openSMOKE++ simulation
3. The system creates input file
4. The system executes openSMOKE++
5. The system checks if simulation need IDT post processing
6. The system reads IDT fields of input file
7. The system creates IDT input
8. The system executes Post Processor
9. The system reads output of post Processor
10. The system reads experimental data
11. The system creates gnuplot script



Fields of Ignition Delay Time and the execution of it are stored into the database. The Entity involved is E_Experiment, it has been added fields as IDT. The Experiment table changed with:

E_Experiment(*idExperiment* , idReactor, pressureMIN, pressureMAX, temperatureMIN, temperatureMAX, keyword, temperatureMIN_Default, temperatureMAX_Default, pressureMIN_Default, pressureMAX_Default, comments, experimentTypeReSpecTh, IDT, OutputUnitTime, OutputUnitTemperature, KeySpec)

4.2 Curve Matching

OpenSMOKE++ is executed with a specific model many times. Curve Matching is used to compare the performances of a model (or more models) in reproducing a set of experimental data. Curve Matching provides an error value concerning the variables of interest (ignition delay time, laminar flame speed, species concentration) for each model. The goal of this update is to implement every step in an automatic fashion: select the experiments, execute the simulations, run the Curve Matching, store the results

It has been necessary add new software feature:

1. Save the output of openSMOKE++ execution associated to the model
2. Create setting CurveMatching fields
3. Create Selection of model and execution of CurveMatching
4. Save results of CurveMatching execution
5. Manage history of results

Store openSMOKE++ output

At every execution of OpenSMOKE++ it is necessary to save the output associated to the model. Selection of the model is specified by the User. A new textbox has been added to the execution form. The user can save the execution as a definitive model or not: in the first case the model is evaluated through Curve Matching. The scenario of execution of openSMOKE turn into:

Table 4.2. openSMOKE++ execution

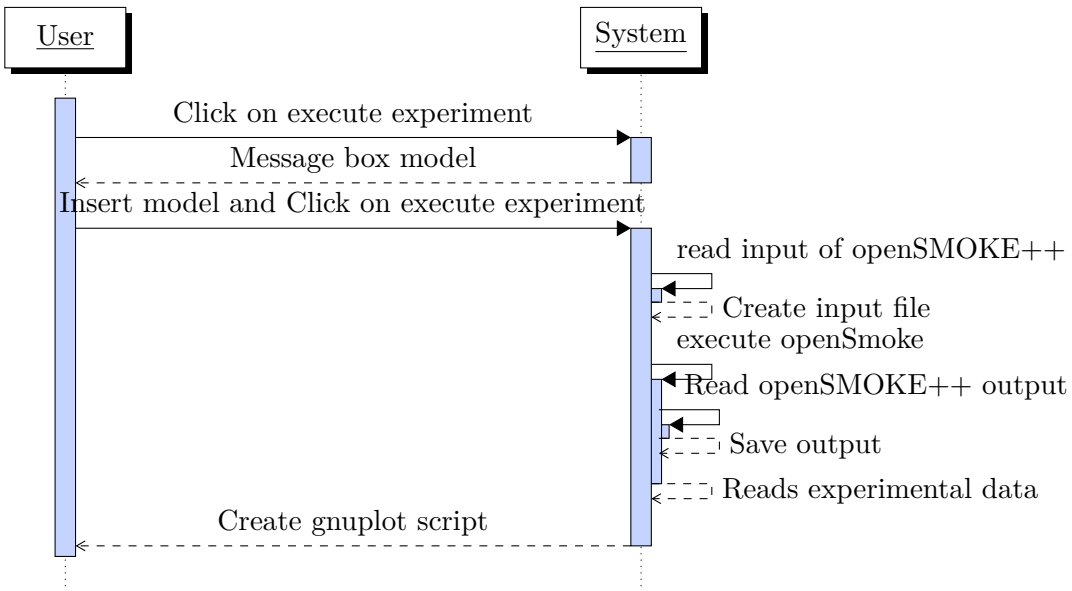
Name of scenario	Simulation execution of openSMOKE++
Goal	Execute openSMOKE++ and store output
Actors	Matteo
Initial Condition	The User research experiment that wants execute and selects the preProcessor directory
Events	Matteo wants to execute two simulation and he wants to associate it to a model ”M1”. Matteo selects three experiments content into the table of experiments. He inserts into the field of model "M1". He clicks on the button ”execute experiment”. Matteo reads the message box appear that asks him if he wants save the output as not definitive model. Matteo realizes that he wants to save the output as definitive so clicks no. Matteo checks the flag definitive model and execute OpenSMOKE++ simulation.

Also the software process change as:

Description	The user execute simulation of openSMOKE++
Goal	The user associated openSMOKE++ output with its model
Actor	Insert data student
Entry condition	The user research experiments that wants execute
Exit condition	The user execute simulation of openSMOKE++

Flusso d'eventi

1. The user checks experiment to simulate and click on execute simulation
2. The system shows message box to specify the Model of execution
3. The user checks definitive model and write model
4. The system reads input of openSMOKE++
5. The system create input file of openSMOKE++
6. The system executes openSMOKE++
7. The system reads openSmoke++ output
8. The system saves into database openSMOKE++ output
9. The system reads experimental data
10. The system creates gnuplot script



To store the output of openSMOKE++ new entity into database has been added. Entities are " openSMOKE columns" and "openSMOKE rows". These entities are in relation with Experiment entityies Columns contains the model that the user specifies and the date of execution The row entity content is in relation with columns entity and contains the values from OpenSMOKE++ outputs.

Setting fields

The execution of CurveMatching expects files containing the specific settings. Default values are set to avoid human error from non-expert users. It has been necessary to add a dedicated panel to change the default setting of CM if needed. In table 4.3 the scenario related to changes of the setting fields is reported.

Table 4.3. CurveMatching Settings Scenario

Name of scenario	Change CurveMatching setting
Goal	Change CurveMatching settings value
Actors	Andrea
Initial Condition	The User opens the software, he is in the start page
Events	Andrea wants to change settings default value of CurveMatching, he cick on section "Curve Matching Settings". Andrea reads default value related to fields of settings and change the value of defaultRelativeError from 0.1 to 0.15. Andrea saves the changes and he moves on research panel to execute curve matching for an experiment.

The values of changes is stored into the database at the moment of execution. The modified values are stored into the database at the time of execution. The change event creates objects into the system. Such objects are used then the user executes the CurveMatching.

CurveMatching Execution

Curve Matching execution is associated to an experiment. To execute CurveMatching it is necessary to select existing models to compare and start the execution. The system creates all the input files necessary to this process. The input must be saved in "input" directory of CurveMatching and files must have the following structure.

1. In the "Input" directory the system must create folder of the experiment
2. In the "Experiment folder" there are two file for each specie of the experiment
    - File _exp
      TThe experiment file represents all the experimental data points taken from the experimental data entities. The structure of the file contains x-axis, y-axis and error axis. The error axis is specified at the moment of

experimental data insertion and it is used by CM to compute the error values.

- File _mod
  The model files contains openSMOKE++ output data point of the model selected to the user. For each model x-axis and y-axis are specify.

3. Experiment formatting of file
   The starting name for model and experiment file must be the same.The name of the experimental data file is the same of Y-axis definition. The Model file contains all models and the starting name of X-axis and Y-axis must be the same of experimental data file specification. It has been necessary to show an example to explain the experimental formatting of species files. Suppose that one wants to execute CM for a Perfectly Stirred Reactor simulation. This experiment contains CO2 species concentration profiles versus reactor temperature. PSR model has 2 definitive model associated to it. The user wants to execute CM for both models. Files must have these structures:

   - _Exp File:
     - File name: CO2_x_exp
     - X-axis name: T[k]
     - Y-axis name: CO2_x
     - Error-axis name: ERROR
   - _Mod File:
     - File name: CO2_x_mod
     - For each model:
       * X-axis name: T[k]_ModelName
       * Y-axis name: CO2_x_ModelName

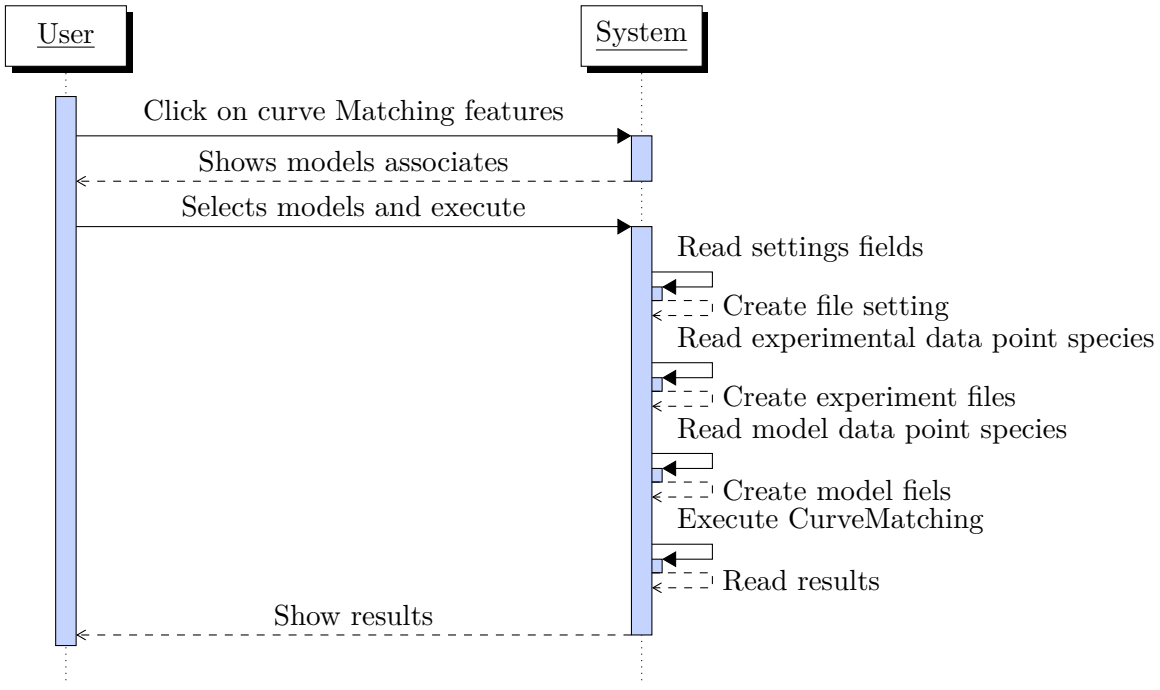
Table 4.4. CurveMatching Execution Scenario

Name of scenario	Execute CurveMatching
Goal	Execute Curve Matching
Actors	Andrea
Initial Condition	The user selects an existing experiment and opens experiment form.
Events	Andrea selects experiment of Perfectly Stirred reaction. He clicks on CurveMatching button and look for the existing model associated. Andrea reads three models associated to the experiment: Model1, Model2 and Model3. He wants study the data point comparison of Model1 and Model2 so he selects these experiments and start the execution of curve Matchin. At the end of execution Andrea reads the results of curve Matchin

The software at the execution time creates inputs and reads outputs of CurveMatching and shows them to the user.

Description	The user execute Curve Matching
Goal	Execution of CurveMatching
Actor	Evaluating student
Entry condition	The user research experiments and open its form of updating
Exit condition	The user execute CurveMatching
Flusso d'eventi	

1. The user checks experiment to simulate and click on CurveMatching execution
2. The system shows models associatds to experiment
3. The user checks models and click on execute
4. The system create file setting
5. The system reads experimental data species
6. The system reads Models data point species
7. The system creates experimental files
8. The system creates model files
9. The system executes CurveMatching
10. The system reads results
11. The system show results to the user



Save results

Results of CM are located in "Result" directory. The output of CM specifies two files that it has been necessary to store into the database. These file are ” _KInt” and ”_KMatrix”: he first one contains overall values of error associated to the model over the whole experimental conditions of interest. K_matrix has instead a more detailed overview of specific cases.

Table 4.5. KMatrix structure

	Specie1	Specie2	Specie3
Model1	error(1,1,)	error(1,2)	error(1,3)
Mode2	error(2,1)	error(2,2)	error(2,3)

The ”_KInt” file contains a vector with the average of error associate to each model, the structure is:

Table 4.6. KInt structure

Index	Model
error_M1	Model1
error_M2	Model2

At every execution of CM the results have to be saved, therefore two entities have been introduced CurveMatching Int and CurveMatching Matrix that represent these structures. Both must be associated to the entities "OpenSMOKE output column" that represent the model of OpenSMOKE++ for each species.

Manage history

It has been needed to manage the history of execution and results of Curve Matching. Entities introduced above represent the history stored in the database. The User can be looking for the history of an experiment in the panel of Curve-Matching. When the system shows list of models associated to experiment if the model has error of CurveMatching already associated, the error value is show. It has been necessary to introduce the possibility to show for each model the error specific to species specified before.
The system read all error history from entities ”K_Matrix” and ”K_Index” specify before.

Table 4.7. View Curve Matching data

Name of scenario	View history of CurveMatching
Goal	The user shows history of curve matching associated to an experiment
Actors	Matteo
Initial Condition	The User opens experiment form that wants study
Events	Matteo wants check the results of curve matching of selected experiment. He clicks on CurveMatching button and opens the CurveMatching panel. Matteo looks all model associated to experiment and he reads that Model1 is already executes and CurveMatching calculate error equals to 0.128. Matteo wants to read the error for each species associated to model M1 so he double-clicks on row of M1 model. Matteo reads that M1 has Curve Matching error equal to 0.109 for species CO2 and error equal to 0.132 for species H2.

4.2.1 Entity RelationShip changes

It has been necessary introduced new fields for Experiment entity that introduce IDT attribute. It has been needed add four entities "Output openSMOKE column", "Output openSMOKE row", "CurveMatching k_matrix", "CurveMatching k_int". The ER diagram is show in figure 4.1

Logical Model

The relation between Entities can be transformed in this way:

- The Entity "Output openSMOKE column" is in relation with Experiment by idExperiment. The primary key is composed by idExperiment, idOpenSmoke-Execution that identify the execution of openSMOKE.

- The Entity "Output openSMOKE row" is in relation with "Output openSMOKE column" by the primary key of "Output openSMOKE column". The primary key of openSMOKE row is composed by openSmoke column primary key and idRow that identifies the rows of output file.

- The Entity "CurveMatching K_Matrix" is in relation with "Output openSMOKE column" by its primary key. The database can store CurveMatching result of every openSMOKE output that is definitive.

- The Entity "CurveMatching K_Matrix" is in relation with "Output openSMOKE column" by the key idExperiment and idOpenSMokeExecution(that identify the model). idExperiment and idOpenSMOKE execution compose the primary key

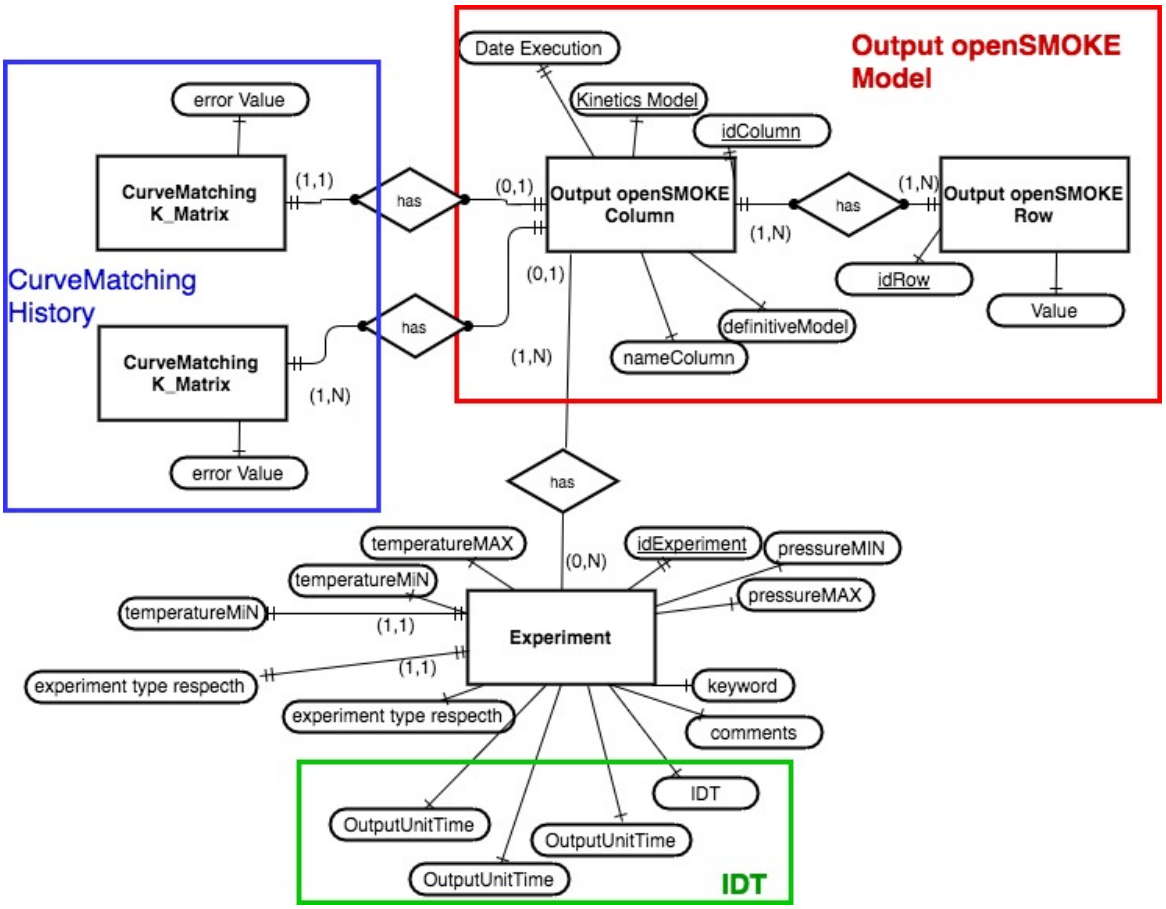


Figure 4.1. Flow

We lists the tables describe before:

Data_column_opensmoke(*idExperiment*, *idOpenSmokeExecution*, *idColumn*, DateExecution, KineticsModel, nameColumn, definitiveModel)

Data_row_opensmoke(*idExperiment*, *idOpenSmokeExecution*, *idColumn*, *idRow*, value)

CurveMatching_Matrix(*idExperiment*, *idOpenSmokeExecution*, *idColumn*,error)

CurveMatching_Int(*idExperiment*, *idOpenSmokeExecution*,value)

Bibliography

[1] Bergthorson, Jeffrey M., and Murray J. Thomson. *"A review of the combustion and emissions properties of advanced transportation biofuels and their impact on existing and future engines."* Renewable and sustainable energy reviews 42 (2015): 1393-1417.

[2] Klippenstein, Stephen J. *"From theoretical reaction dynamics to chemical modeling of combustion."* Proceedings of the Combustion Institute (2016).

[3] A. Cuoci, A. Frassoldati, T. Faravelli, E. Ranzi, *OpenSMOKE++: An object-oriented framework for the numerical modeling of reactive systems with detailed kinetic mechanisms"*

[4] Dente, M., E. Ranzi, and A. G. Goossens., *"Detailed prediction of olefin yields from hydrocarbon pyrolysis through a fundamental simulation model (SPYRO)."* Computers & Chemical Engineering 3.1-4 (1979): 61-75.

[5] Olm, Carsten, et al. *"Comparison of the performance of several recent hydrogen combustion mechanisms."* Combustion and Flame 161.9 (2014): 2219-2234.

[6] Bernardi, M. S., et al. *"Curve matching, a generalized framework for models/experiments comparison: An application to n-heptane combustion kinetic mechanisms."* Combustion and Flame 168 (2016): 186-203.

[7] Kee, R.J.; Miller, J.A.; Jefferson, T.H. *"CHEMKIN: a general-purpose, problem-independent, transportable, FORTRAN chemical kinetics code package"* Mar 1980; 203 p; Available from NTIS.

[8] T. Varga, T. Turanyi, E. Czinki , T. Furtenbacher, A. G. Csaszar *"ReSpecTh: a joint reaction kinetics, spectroscopy, and thermochemistry information system "*

[9] M. Pelucchi, A. Rigamonti, B. Pernici, A. Frassoldati, T. Faravelli *"tTowards a fully automated system to develop, validate and evaluate combustion kinetic mechanisms*

[10] Ranzi, Eliseo, et al. *"A wide range modeling study of methane oxidation."* Combustion science and technology 96.4-6 (1994): 279-325.

[11] Ranzi, E., et al. *"Lumping procedures in detailed kinetic modeling of gasification, pyrolysis, partial oxidation and combustion of hydrocarbon mixtures."* Progress in Energy and Combustion Science 27.1 (2001): 99-139.

[12] KAREN CALHOUN *"Il ruolo sociale della biblioteca digitale"* DOI: 10.3302/2421-3810-201602-004-1

[13] Leonardo Candela, Donatella Castelli, Paolo Manghi, Alice Tani *"Data Journals: A Survey"* DOI: 10.1002/asi.23358

[14] Massimiliano Assante, Leonardo Candela , Donatella Castelli, Alice Tani *"Are Scientific Data Repositories Coping with Research Data Publishing?"*

[15] Mary Vardigan, Cole Whiteman *"Describing Roles & Measuring Contemporary Preservation Activities in ARL Libraries"* May 2009

[16] Lars Meyer, ARL Visiting Program Officer *"ICPSR meets OAIS: applying the OAIS reference model to the social science archive context"* 23 August 2007

[17] Ana Krahmer,Mark Edward Phillips *"Communicating Organizational Commitment to Long-Term Sustainability through a Trusted Digital Repository Self-Audit"* 27.07.2016

[18] Leonardo Candela, Fuat Akal, Henri Avancini, Donatella Castelli, Luigi Fusco, Veronica Guidetti, Christoph Langguth, Andrea Manzi, Pasquale Pagano, Heiko Schuldt, Manuele Simi, Michael Springmann, Laura Voicu *"DILIGENT: integrating digital library and Grid technologies for a new Earth observation research infrastructure"* 15 May 2007

[19] Free and Open Access to Biodiversity Data *"The Global Biodiversity Information Facility" - http://www.gbif.org/*

[20] Object Management Group *"Automated Function Points (AFP)"* 2014

[21] - *"PROCESS INFORMATICS?A NEW PARADIGM FOR BUILDING COMPLEX CHEMICAL REACTION MODELS"* -

[22] Michael Frenklach, Mike Pilling, David Golden *"PrIMe Report"* August 2002

[23] Michael Frenklach *"PrIMe Next Frontier Large, Multi-dimensional Data Sets"* August 2002

[24] XIAOQING YOU, ANDREW PACKARD, MICHAEL FRENKLACH *"Process Informatics Tools for Predictive Modeling: Hydrogen Combustion"* 2011

[25] The Institute of Research on Combustion (CNR) organized the Workshop on "Data Collection and Mining toward the Virtual Chemistry of Smart Energy Carriers" *"ReSpecTh: a joint reaction kinetics, spectroscopy, and thermochemistry information system"* October 2016

[26] - *" IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications - http://standards.ieee.org/findstds/standard/830-1998.html "*

[27] - *" IEEE Systems and software engineering Life cycle processes Requirements engineering "*