UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

PH.D. THESIS
IN
INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

ASSESSING AND IMPROVING INDUSTRIAL SOFTWARE PROCESSES

VINCENZO DE SIMONE

TUTOR: PROF. ANNA RITA FASOLINO

COORDINATOR: PROF. DANIELE RICCIO

XXX CICLO

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

Assessing and Improving Industrial Software Processes

Vincenzo De Simone

University of Naples Federico II

Contents

List of Acronyms

The following acronyms are used throughout this text.

ALM	Application Lifecycle Management
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
ASPICE	Automotive SPICE
ATL	ATLAS Transformation Language
BCM	Body Computer Module
CLoC	Commented Lines of Code
CMMI	Capability Maturity Model Integration
DSML	Domain Specific Modeling Language
ECU	Electronic Control Unit
EMEA	Europe, Middle East, and Africa
EXACT	EXcel Application Comprehension Tool

FCA	Fiat Chrysler Automobiles
FP	Feature Profile
GADGET	Gap Analysis Design and GEneration Tool
GUI	Graphical User Interface
HIL	Hardware-In-the-Loop
IPC	Instrument Panel Cluster
ITL	Incorrect Traceability Links
ITLRP	Incorrect Traceability Links Reduction Percentage
KPI	Key Performance Indicators
LoC	Lines of Code
MB	Model Based
MBD	Model Based Design
MDE	Model Driven Engineering
MTL	Missing Traceability Links
MTLRP	Missing Traceability Links Reduction Percentage
MIL	Model-In-the-Loop
MOF	Meta Object Facilities
MTL	Model-to-Text Language
OMG	Object Management Group
PA	Product Architecture

PLA	Product Line Architecture
QBGA	Questionnaire Based Gap Analysis
QVT	Query View Transformation
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SEI	Software Engineering Institute
SIL	Software-In-the-Loop
SPEM	Software & Systems Process Engineering Metamodel
SPICE	Software Process Improvement and Capability Determination
SPML	Software Process Modeling Language
SPL	Software Product Lines
SWC	SoftWare Component
SWF	SoftWare Factory
SWFTC	SoftWare Factory Test Cases
UML	Unified Modeling Language
VBA	Visual Basic for Applications
VP	Variation Point
V&V	Verification & Validation
XSLT	eXtensible Stylesheet Language Transformations

This page intentionally left blank.

List of Tables

List of Figures

Introduction

Software is undoubtedly becoming one of the key factors of our everyday life. People rely, often unwittingly, on software systems to carry out their usual daily tasks. A simple example of this enormous diffusion can be supported by the growing number of mobile applications that are downloaded and used worldwide[1]. As a further point, this shift toward software not only has impacts on the customers' market, but it is also affecting several industrial domains. In an increasing number of industrial domains, features that were realized in the past by means of mechanical, electrical or electronic systems are now governed by software. Even critical domains, such as avionics, automotive, railway, aerospace, health-care, have been interested by this spreading shift toward software. Just to give a practical example, most of the functions provided by a modern car are now realized by means of software. Nowadays, about 100 million Lines of Code (LoC) are deployed on 70 to 100 microprocessor-based Electronic Control Unit (ECU)s networked throughout the body of a typical modern car[2]. And these figures are bound to rise in the future. Also in the avionic domain, there was an increase from the 135 thousand LoC for the software

[1] *https://goo.gl/9Me7YU*

[2] *http://www.informationisbeautiful.net/visualizations/million-lines-of-code/*

deployed on the F16A in 1974 to the 24 Million LoC of the F35 Lighting II in 2012[3].

To be competitive on their respective markets and to design and produce quality products, guaranteeing better user experience, safety, security, maintainability, the companies in which software systems are developed need to correctly manage their software development processes, i.e. Software Processes. A software process represents the set of activities that leads to the production of a software product. These activities may involve the development of software from scratch or extending and modifying existing systems and by configuring and integrating off-the-shelf software or system components [1].

Research and industrial communities are devoting great efforts in order to propose approaches and tools for better supporting software processes, assessing them in real industrial settings and for improving their quality [2, 3, 4]. Even though, there are still several open issues and challenges related to software processes and their assessment and improvement in real industrial contexts.

One of the main issues that affects the management of complex phenomenon is the ability to understand and assess its state and behavior in order to propose effective and efficient solutions for mastering it. This statement can be easily applied to software processes that are distributed and sparse both geographically and from managerial viewpoint, involve a lot of different actors having different roles, require the execution of different time-consuming activities and the generation of a multitude of artifacts. Since that there is the need of new paradigms and enabling technologies for gathering all the key information regarding the process and to support its evaluation when it is needed. Moreover, companies involved in the development of software systems are continually asked to improve

[3] *https://insights.sei.cmu.edu/sei_blog/2015/09/managing-software-complexity-in-models.html*

the adopted development practices in order to maintain and increase the competitiveness in their markets [5]. These companies are required to adhere to well-known quality frameworks, such as the Capability Maturity Model Integration (CMMI) [6] or the Software Process Improvement and Capability Determination (SPICE) [7, 8] that specify how to carry out the development tasks. Moreover, when the software process is performed in safety-critical system domains, such as automotive, railway, or aerospace, the software companies are even obliged to demonstrate that they do not pose undue risk to people, property, or the environment, showing their compliance with a Standard Development Approach [9]. To cite just a few examples, Standard approaches for developing safety critical systems exist for automotive (ISO 26262 - Road Vehicles Functional Safety [10]), Medical (IEC 62304 Medical device software - Software life cycle processes [11]), and Nuclear (IEC 61513 Nuclear power plants - Instrumentation and control important to safety [12]) industries. The standard approach is to carefully code, inspect, document, test, verify and analyze the systems being developed. In these contexts, companies require to adopt appropriate approaches and technologies for supporting the preliminary evaluation and assessment of their capability to comply with the requirements of these Standards and Evaluation framework. Once identified the gaps which separate them from these standard, they can implement the needed improvement actions in order to demonstrate the quality of their software process and, consequently, the quality of the software products they develop.

As another relevant issue, software process execution involves a multitude of different artifacts that need to be managed throughout the entire software lifecycle. An artifact is defined as any deliverable that is created, consumed, or modified by an activity. Artifacts have a type, define a structure, and may be linked toward each other [13]. The complexity of the artifacts and of the dense, and often hidden, interconnection among them should be correctly managed during the software process. However, despite the relevance of traceability in software processes is well-known, the activities of traceability creation and management are not always adequately supported in real software projects. The lack of integration between the tools adopted in the development processes is one of the main causes of such an ineffective management, where traceability relationships are still manually generated and maintained.

As another factor that may influence the correct traceability management and also the effectiveness of the entire software process, there is the need to handle the increasingly complex artifacts involved in its execution. This demand is even stronger in domains in which software process are carried out as a phase of the entire product development process, such as the automotive, railway and others, that involve people, namely end-users, having little or no software development skills. In these contexts, solutions for supporting the comprehension of the artifacts involved in the software development process and for easing their management and traceability are required.

Another important issue is related to the great variability that companies need to manage when they carry out software process. Companies need to realize products that are tailored according to customers' requirements and preferences besides the different laws and the cultural preferences of the market in which they are sold. This tailoring has a great impact on the functionalities that the software products have to provide. Since that, companies are obliged to manage different versions of the same software product, tailored according the different requirements. In this scenario, they need to adopt cost-effective software development processes in order to manage the variability of the produced software. A case-by-case basis approach, where the software variability is managed at the end of the development process, can no longer be considered suitable to resolve these issues. More systematic solutions need to be introduced [14]. *Software Product Lines* (SPL) approach has proven to be a successful solution for handling the complexity and variability of the software developed in different domains as shown by several success stories reported in [15]. According to Clements and Northrop [16], *"a software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way"*. Although the SPL bring actual benefits to the software variability management also enabling software reuse, they strongly impact on the overall software life cycle and requires the definition of appropriate strategies supporting its effective adoption.

In order to address these issues and to guarantee software process improvement, I defined different approaches and tools exploiting software engineering principles and appropriate enabling technologies. I also evaluated the proposed solutions through Case Studies carried out in real industrial settings.

Thesis Contributions

This *Thesis* work contributes to the literature in software process assessment and improvement by proposing:

- an approach for supporting the design and execution of Gap Analysis processes with respect to Standards and Evaluation Frameworks that foresees the adoption of Application Lifecycle Management (ALM) systems and Model Driven Engineering (MDE) technologies;

- an approach for the automatic management of traceability links among the artifacts involved in the development process that relies on the integration of the tools supporting the enactment of the software process;

- a reverse engineering process and a tool for supporting the comprehension of spreadsheet based artifacts involved in software processes;
- an approach and a software architecture for supporting the introduction of Software Product Lines in real industrial processes. The approach has the aim of managing the variability and enabling reuse in software processes.

The feasibility of the proposed approaches and tools has been validated through real industrial case studies, conducted according to the guidelines defined by Runeson *et al.* [17]. Some of the reported case studies were conducted in the context of the APPS4Safety Research Project (PON03PE_00159_3)[4], that is partially funded by the Italian Ministry of Education and Research. Due to the complexity of software processes, that involve several actors, artifacts, activities and tools, I decided to employ the case study research methodology to assess the feasibility of the proposed approaches. In this way, I wanted to evaluate the proposed approaches in real software process settings. Moreover, in these studies I exploited qualitative research methods, since they increase the amount of information contained in the data collected. It also increases the diversity of the data and thus increases confidence in the results through triangulation, multiple analyses, and greater interpretive ability [18, 19]. The raw data of the conducted case studies was not reported due to ethical and logistical reasons. However, further material was made available in an external repository[5], such as the interview guides used for conducting the case studies or the produced artifacts.

This *Thesis* work includes material from the following research papers already published in peer-reviewed journals or conferences:

- *Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino,*

[4] *http://www.ponrec.it/open-data/progetti/scheda-progetto?ProgettoID=7111*
[5] *https://github.com/vindes2/Thesis-Material/*

and Stefano Scala. 2017. ***Improving traceability management through tool integration: an experience in the automotive domain.*** *In the Proceedings of the 10th International Conference on Software and System Process (ICSSP 2017). ACM. [20]*

- *Domenico Amalfitano, Anna Rita Fasolino, Porfirio Tramontana, Vincenzo De Simone, Giancarlo Di Mare, and Stefano Scala. 2015.* ***A Reverse Engineering Process for Inferring Data Models from Spreadsheet-based Information Systems: An Automotive Industrial Experience.*** *Part of the Communications in Computer and Information Science book series (CCIS, volume 178). Springer. [21]*

- *Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino, Porfirio Tramontana. 2016.* ***EXACT: A tool for comprehending VBA-based Excel spreadsheet applications.*** *Journal of Software: Evolution and Process 28(6). Wiley. [22]*

- *Domenico Amalfitano, Vincenzo De Simone, Anna Rita Fasolino, Mario Lubrano, Stefano Scala. 2016.* ***Introducing Software Product Lines in Model-Based Design Processes: An Industrial Experience.*** *In Proceedings of 13th Working IEEE/IFIP Conference on Software Architecture (WICSA 2016). IEEE. [23]*

Thesis Outline

This dissertation is organized as follows:

- Chapter 1 reports the background of this work, focusing on Software Processes and ALM. Moreover, it highlights the main elements of the research methodologies I exploited to validate the approaches proposed in this work;

- Chapter 2 describes the approach I designed for supporting companies in the definition and execution of Gap Analysis processes with Standards and Evaluation frameworks. Moreover, it reports the results of a case study I conducted in collaboration with my research group with the aim of assessing its feasibility;

- Chapter 3 shows the tool integration approach I proposed for supporting the traceability management of the artifacts involved in real software process and the results of an experiment I carried for assessing its validity;

- Chapter 4 describes the approaches and tools I defined for supporting the comprehension of the artifacts involved in software process that are based on spreadsheet systems and the case studies I conducted for validating them.

- Chapter 5 reports the approach based on SPL and the software architecture I defined for supporting the management of variability in software processes and the experience I gained by applying it in real automotive software processes;

- Chapter 6 reports conclusive remarks and possible future work.

Chapter 1

Background

This Chapter illustrates the main topics covered in this dissertation. More in detail, it reports an overview on Software process, that is the main topic of this Thesis work, and on ALM, chosen as enabling technology for solving the identified issues of software processes. Moreover, it illustrates a brief introduction of the research methodologies exploited in the Software Engineering domain, focusing on the one, i.e the Case Study Research methodology, that was adopted for validating the approaches proposed in this work.

1.1 Software Process

Software process describes the activities that are carried out to produce a software system. More precisely, a software process can be defined as the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product [24]. Often, software processes are also called "software development processes". Even though, this term is not exhaustive since many parts of software processes are not directly related to development (such as maintenance processes), they are also relevant in areas where not only software is produced (e.g., when building software-based systems).

Software processes are enacted, i.e. executed, in the real world and are characterized by different aspects. In order to produce the final software product, during its enactment different input artifacts are transformed into output artifacts by consuming other products such as guidelines, best practices. These transformations are carried out by human actors having specific roles that defines the need skills, permissions for carrying out the specific activities. Moreover, they are supported by tools that execute all or parts of the required activities. The software process can be further structured in subprocesses, activities, steps that define the different lifecycle of the software product.

1.1.1 Software Process Models

A software process model is a standardized format for planning, organizing, and executing software development processes. It describes the sequence of activities carried out in a software development process, and the relative order of these activities [1]. A multitude of different software process models have been proposed throughout these years, but they can be considered small variations of a set of basic software process models. A first effort for defining a comprehensive software process can be traced back in the 70s with the definition of the Waterfall software process by Royce [25], even though its definition was based on several previous work [26, 27]. This process defines the development of software through the execution of sequential activities. It is widely used since it reinforces good habits, such as define-before- design, design-before-code, identifies deliverables and milestones and can be used for mature products even with weak development teams. Although, the waterfall model presents different disadvantages: it is costly, it does not reflect iterative nature of exploratory development, it requires accurate requirements early in a project, it leads to late software deliveries, delaying the discovery of serious errors, it does not manage risks and it does not take into account possible decision changes or evolution. Since that, a lot of different process models have been proposed such as Exploratory Development, Throw-away Prototyping, Spiral Model [28]. All these process models have their advantages and cons and should be applied considering the context in which software is developed. Moreover, Agile software processes [29], such as Scrum [30], XP, etc., have been proposed in last decades proposing processes that can be adapted to changes and that do emphasis on people rather than activities and documents. Moreover, there is in the last decades the adoption in software process of approaches that focuses on models rather than on code, such as Model Driven Architecture[1] and Model Based Design[2]. These approaches exploit modern technologies for supporting code automatic generation from design models.

The most widely adopted software process model in Industrial setting is the V-model. It is considered as an extension of the typical waterfall model where relationships between each phase of the development life cycle and its associated phase of testing are introduced. It is a mature software process model and it allows to discover issues in the early stages of development and it can manage risks and changes. It is the reference model in different Standards for software development [31], even for safety critical domains [10].

1.1.2 Software Process Modeling

One of the challenges for software development organizations is to find the means of rationally describing and managing activities, resources, and constraints of their software development processes while taking into ac-

[1] *http://www.omg.org/mda/*
[2] *https://www.mathworks.com/solutions/model-based-design.html*

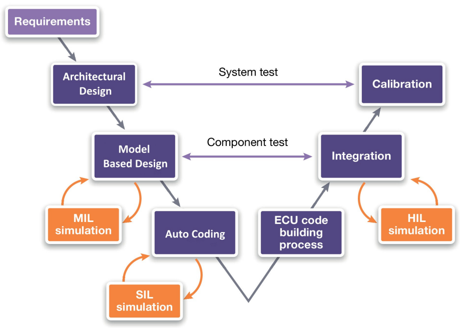


Figure 1.1. V-model

count all these characteristics. Software process should be appositely documented exploiting software process models. Process models can be used to reason about processes, and to test and improve them to meet increasing quality and cost expectations. Moreover, process models can also be exploited to automate repetitive and non-interactive tasks. The software community tried to answer the need for explicit process models with a wide range of Software Process Modeling Languages (SPMLs).

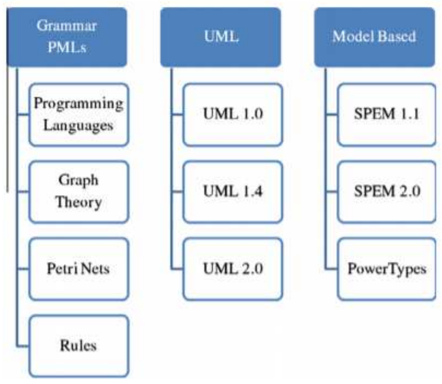


Figure 1.2. SPMLs Categories

As reported in [32], the proposed SPMLs can be groped in three different categories, as can be seen in Figure 1.2. The first group of grammar-based languages includes all studies that focus on formal languages, mathematical and programming, by means of rules or restrictions. A second group contains several versions of Unified Modeling Language (UML)-based SPMLs, and finally, the last group includes metamodel-based SPMLs or DSLs. Figure 1.3 outlines the correspondence among each SPML and the aforementioned groups. Following the proposed taxonomy where grammar-based SPMLs are shown with no background color or frame, UML-based SPMLs are shown with background color and model-based SPMLs are shown with a frame. Some of them were rules-based (e.g., MARVEL), others Petri net-based (e.g., SPADE) or programming languages-based (e.g., SPELL, APPL/A). While these first-generation languages were executable and put a strong emphasis on formality, they did not gain much attention from industry [33]. Their complexity, their use of low-level formalisms, and their inflexibility were among the causes of their limited adoption [34]. Moreover, from the comparison carried out in [35], resulted that the *Software & Systems Process Engineering Meta-model (SPEM) 2.0*[3] is the most widely adopted Modeling language used to describe Software Processes.

1.1.3 Software and Systems Process Engineering Meta-model

SPEM 2.0 was introduced as a language to define software and systems development processes and their components. The scope of SPEM is purposely limited to the minimal elements necessary to define any software and systems development process, without adding specific features for particular development domains or disciplines. The goal is to accommodate a large range of development methods and processes of different styles, cultural backgrounds, levels of formalism, lifecycle models, and communities. SPEM is a process engineering meta-model as well as conceptual

[3] *http://www.omg.org/spec/SPEM/*

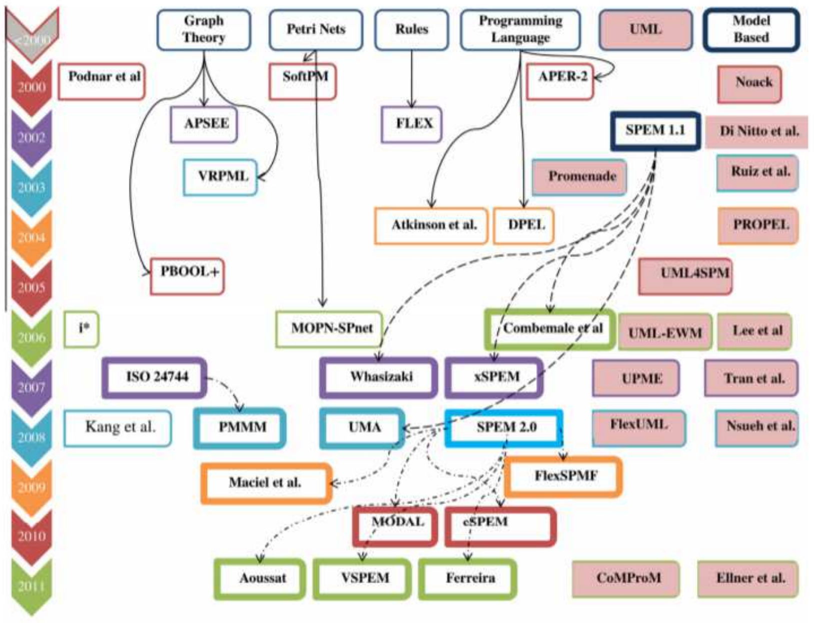


Figure 1.3. Relations and base technology of SPMLs

framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes. An implementation of this meta-model would be targeted at process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for their development organizations or individual projects. It was introduced for:

- providing a standardized representation and managed libraries of reusable method content;
- supporting systematic development, management, and growth of development processes;
- supporting the enactment of a process for development projects.

SPEM allows to describe all the elements of a software or system process. To this aim it defined all the possible elements characterizing a software and system process elements that are shown in Figure 1.4.

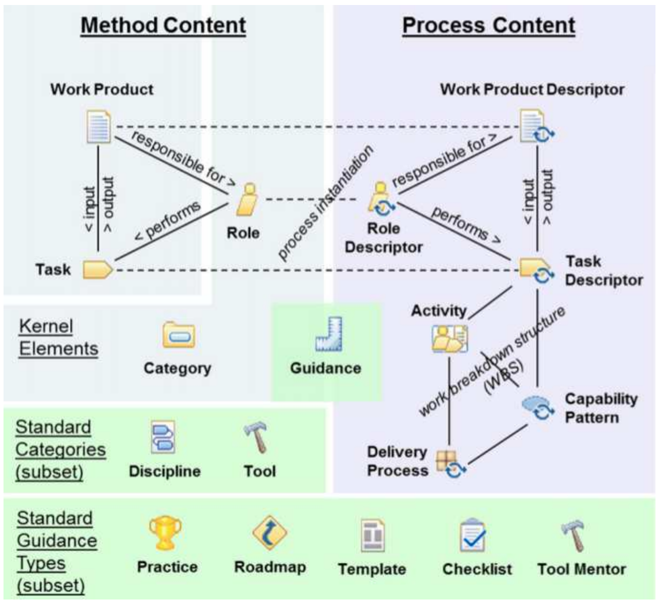


Figure 1.4. Concepts exposed by the SPEM formalism

As the Figure shows, SPEM differentiates between method content and process content. Method content concepts such as Task, Work Product and Role are used to define reusable descriptions that can be incorporated into several processes. The corresponding elements in the process content, i.e. Task Descriptor, Work Product Descriptor and Role Descriptor, are basically copies that contain the information from the method content counterparts, but can be locally adapted as part of the process description. The process content also includes Activity, Capability Pattern and Delivery Process that can be used to organize and define work breakdown structures and reusable process patterns. The method content includes Category that can be used to define custom categories, e.g. Discipline to categorize tasks and Tool to categorize tool-specific guidance. The Guidance concept can be used to define specific guidance types, e.g. Roadmap, Template, Checklist and Tool Mentor that can be associated with both method and process content. Practice is a special guidance kind for organizing elements that belong to the same practice [36].

1.1.4 Software Process Improvement and Software Process Characteristics

"Ideal" or "Standard" software process that can be applied in all organizations or for all software products of a particular type does not exist. Each company has to develop its own process depending on its size, the background and skills of its staff, the type of software being developed, customer and market requirements, and the company culture. Process improvement, therefore, does not simply mean adopting particular methods or tools or using a published, generic process. Although organizations that develop the same type of software clearly have much in common, there are always local organizational factors, procedures and standards that influence the process. It is difficult to be successful in introducing process improvements bu simply attempting to change the process to one that is used elsewhere. There is the need to consider the local environment and culture and how this may be affected by process change proposals. In Software Process Improvement initiatives, the companies have to consider what aspects of the process are willing to improve. As an example, the company might be interested in improving software quality and in introducing new process activities that change the way software is developed and tested. As another aspect, the companies may have the goal to improve some attribute of the process itself. In this case they have to identify which process attributes are the most important for them. Examples of process attributes that may be targets for improvement are listed in Table 1.1.

Table 1.1. Process Characteristics

Process Characteristic	**Key Issues**
Understandability	To what extent is the process explicitly defined and how easy is it to understand the process definition?
Standardization	To what extent is the process based on a standard generic process? This may be important for some customers who require conformance with a set of defined process standards. To what extent is the same process used in all parts of a company?
Visibility	Do the process activities culminate in clear results, so that the progress of the process is externally visible?
Measurability	Does the process include data collection or other activities that allow process or product characteristics to be measured?
Supportability	To what extent can software tools be used to support the process activities?
Acceptability	Is the defined process acceptable to and usable by the engineers responsible for producing the software product?
Reliability	Is the process designed in such a way that process errors are avoided or trapped before they result in product errors?
Robustness	Can the process continue in spite of unexpected problems?
Maintainability	Can the process evolve to reflect changing organizational requirements or identified process improvements?
Rapidity	How fast can the process of delivering a system from a given specification be completed?

Research community is devoting a great effort in proposing solutions aimed at improving software processes. A recent study by Khan *et al.* analyzed the most reviews carried out in the last decade and found out that the study on Software Process Improvement mostly focused on analyzing their success factors and possible process models [37]. Even though the great number of work in this area, research in Software Processes and their Improvements are still needed.

1.1.5 Process Quality Evaluation Frameworks

In order to guarantee the quality of software products, different standards and evaluation framework have been defined. These standards focus on the quality of the software process in order to demonstrate the quality of the produced software. To this aim they propose best practices and guidelines to follow for reaching certain levels of capability or maturity of their process.

The CMMI Evaluation framework, introduced by the Software Engineering Institute (SEI), provides guidance for improving organizations' capability to develop quality products and services that meet the needs of customers and end users. It introduces best practices for improving efficiency, speed, and product quality fueled by a lower number of defects. In order to assess the organizations' capability CMMI provides five Maturity Levels for executing a rigorous benchmark rating method that enables you to compare your organization's capability to its competitors, its industry, and itself over time. The CMMI Maturity Levels are described in Figure 1.5.

CMMI focuses on the goals rather than the way they are reached that is important. Organizations may use any appropriate practices to achieve any of the CMMI goals they do not have to adopt the practices recommended in the CMMI.

Another widespread framework is the ISO/IEC 15504 (now ISO/IEC

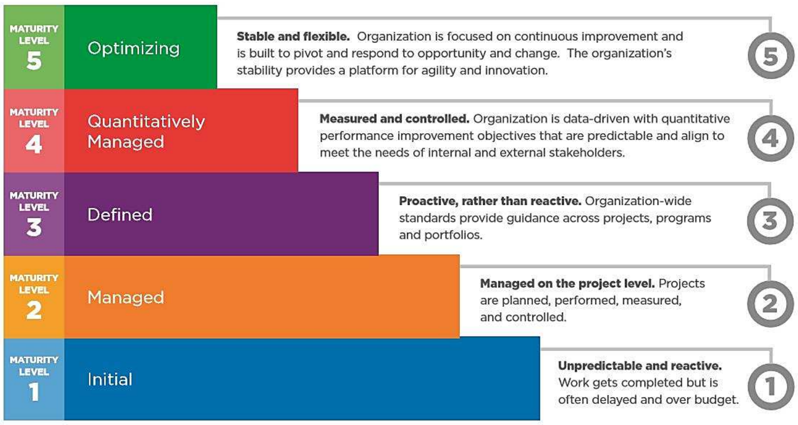

MATURITY LEVEL 5
Optimizing
Stable and flexible. Organization is focused on continuous improvement and is built to pivot and respond to opportunity and change. The organization's stability provides a platform for agility and innovation.
MATURITY LEVEL 4
Quantitatively Managed
Measured and controlled. Organization is data-driven with quantitative performance improvement objectives that are predictable and align to meet the needs of internal and external stakeholders.
MATURITY LEVEL 3
Defined
Proactive, rather than reactive. Organization-wide standards provide guidance across projects, programs and portfolios.
MATURITY LEVEL 2
Managed
Managed on the project level. Projects are planned, performed, measured, and controlled.
MATURITY LEVEL 1
Initial
Unpredictable and reactive. Work gets completed but is often delayed and over budget.

Figure 1.5. CMMI - Maturity Levels

330xx [7, 8, 38]) also known as SPICE, is the reference model for the maturity models against which is possible to determine organizations' capabilities for delivering products, such as software, systems, and IT services. The proposed capability levels, shown in Figure 1.6 are based on process attributes that specify further generic practices to apply.

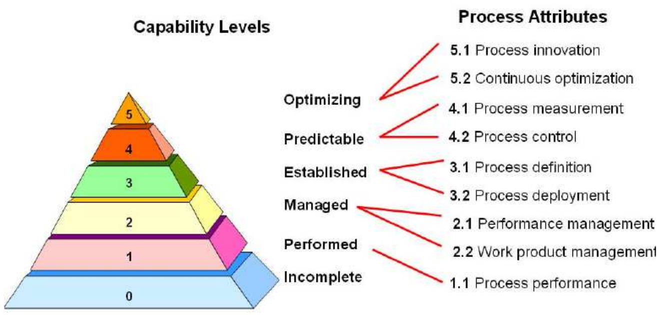


Figure 1.6. SPICE - Capability Levels

Both these Evaluation frameworks contain generic assessment models for development processes, which provide a basis to rate organizations' capability. These process models are founded on best practices proven across the entire software industry. When applying process assessment to specialized areas of software development, better results may be provided by using a tailored process model that reflects the particular practices of the industry.

1.2 Application Lifecycle Management

ALM platform has been proposed with the objective to provide a comprehensive technical solution to monitor, control and manage software development over the whole application lifecycle. The goal of an ALM platform is to make software development and delivery more efficient, lower its costs and improve the quality of produced software. However, the concept of ALM is unclear and driven by tool vendors [**?**]. As reported in Figure 1.7, ALM focused on three main aspects of software processes that are defined *the pillars of ALM*, i.e. traceability, visibility and process automation [39]. From many work emerges the idea that concepts such as traceability, process automation, reporting and tool integration are the foundations of ALM.

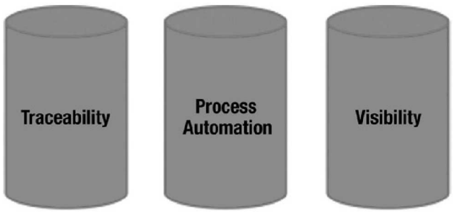


Figure 1.7. The three pillars of ALM

Traceability is born in the domain of requirements engineering where it is addressed as the ability to describe and follow the life of a requirement, in both a forwards and backwards direction, from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases [40]. Traceability is also addressed as the degree to which each element in a software development product establishes its reason for existing and to understand at a glance why an artifact has been created [41]. Traceability provides the means to track all the changes in the single artifacts and who made these changes allowing to reconstruct its history. Finally, by correcly implementing traceability the entire software development process it is possible to check if result of the process and the quality of the produced software, by identifying what tests are used to verify a requirement, by checking that all requirements are covered by test cases and by helping to identify if the implementation is in compliance with the requirements [42].

Visibility aims to offer to managers and stakeholders a closer look on the progress of development efforts. Process automation, instead, focuses on the automation of some process tasks to make the process more effective and less time-consuming. ALM can be seen as a supervisor which covers the whole development process from the initial idea to the end of the product lifecycle through different core aspects: governance, development and operations. Development is the process of creating and testing the application. Once the application is deployed it must be monitored and managed and the Operations area is the one responsible for this. It starts before the end of the development process because the operations area is also the one responsible of the deployment itself [43].

Each of the described lifecycle activities involves a multitude of technologies and tools and it's important for a software development company to successfully harness these technologies and tools. Even more important, it is need to integrate the different tools involved in the development process in order to ease the development activities and improve the management of the artifacts involved in their execution. Actors involved in the software development process adopt tools suited to their needs. However, in a modern software development process it is required to share information between different teams. From this aspect emerges the need of a consistent support for the development process as a whole by adopting a common foundation for application lifecycle management. Such common foundation let all the different teams use a common pool of tools, or only one too for those activities that are in common between the different areas, such as requirements management, build management, version control, bug tracking, test case management and dashboards and reporting, as shown in Figure 1.8.

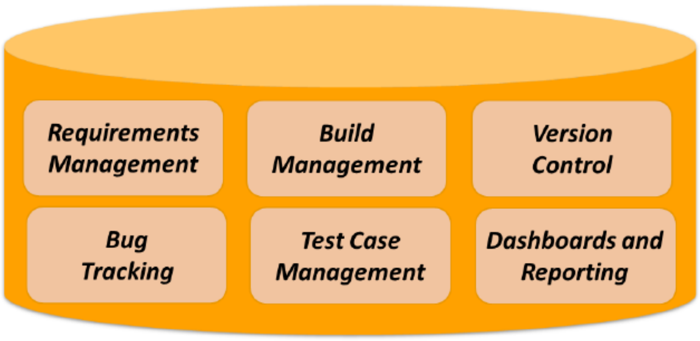


Figure 1.8. A common ALM foundation

Using a common foundation, it's possible to keep all company's development projects and their artifacts in one place making it easier to share code and development artifacts across different groups. A shared ALM foundation also helps to implement the same development process across all teams. Since everybody relies on the same underlying mechanisms, using a common process gets easier. This simplifies management, and it also allows improvements made by any group to spread quickly to all the other groups [44]. Finally, an ALM foundation also eases the communication between the company and external partners that can update the common repository to inform the company of their progress. It's easy to understand why having an ALM foundation can also improve traceability management.

Nowadays many ALM tools are available on the market (IBM Rational Team Concert[4], HP Application Lifecycle Management, Siemens Polarion ALM[5]) and all of them offer the functionality described before. However, it is not simple to introduce this ALM tools in companies, since most of them already have a de-facto development process and adopt legacy tools. The transition toward ALM can be difficult and many companies may base their choice on the features offered by the ALM platform, the ones needed by the company's development process and the work needed to tailor the platform to the company's need [45]. The selected solution should also be the one that supports integration with the highest number of tools already in use in the company. Having a direct and automatic link between the tools and the ALM platform can minimize team members manual work and chance of error.

1.3 Research Methodologies

In this section, a brief overview of research methods applied in Software Engineering domain are reported, focusing on qualitative and quantitative methods. Then, I motivate the choice to adopt the Case Study methodology for assessing our proposed approaches in real industrial contexts.

Research is a discipline for developing knowledge on the basis of the analysis and processing of collected data regarding a phenomenon under investigation [46]. There are two main research paradigms:

- the positivist paradigm considers a phenomenon is measurable by using statistical instruments such as surveys, and observable by experiments [47]

[4] *http://www-03.ibm.com/software/products/it/rtc*
[5] *https://polarion.plm.automation.siemens.com/products/polarion-alm*

- the interpretivist paradigm focuses on the researchers viewpoint for understanding the social reality [48] other than seeking for generalizable truths

Scientific communities have considered both these approaches, showing that they are complementary. More in detail, Quantitative research, based on the positivist approach, consider the reality as static and observable whereas the Qualitative research follows the interpretivist paradigm and considers that there are various alternative interpretations that accommodate the scientific knowledge itself [49].

1.3.1 Research Strategies

In Software Engineering domain, different research methodologies have been adopted [50], according to the proposed categories: quantitative research where surveys, controlled experiments and simulations are performed and qualitative research such as grounded theory, ethnography, action research. There are two types of research paradigms that have different approaches to empirical studies: Exploratory and Explanatory Research [51].

Exploratory Research

Exploratory research requires the study and analysis of the phenomenon in its natural setting, letting the findings emerge from observations. Since that, Exploratory research methods is based on the application of flexible research methods, or *Qualitative research*, that can be adapted to changes of the observed phenomenon. This type of research method exploits qualitative data. It is an inductive method since it requires to interpret the phenomenon based on the information, explanations, perceptions that are reported by the subjects involved in the study. Subjects are people that are taking part the study in order to evaluate an object. Qualitative research is an approach aiming at investigating participants' actions and words for interpret patterns of meaning [52]. It relies on the collection of information from the participants' own words and definitions, and classify them from their natural work settings or environment [53]. Qualitative researchers exploit interviews, individual experiences, case studies and focus groups to capture data about the values of people for investigative observations. The collected data can be documented in a contextual framework for conducting a closer observation of words and view of the participants and further inspection [54]. Typically, qualitative methods are inductive and they are used to investigate a new or unexplored phenomena or sometime to generate a theory. Moreover, it is required when interactions with participants to seek in-depth answers or research are needed. Qualitative studies are conducted in small groups or with a limited number of participants; hence, the results in many cases are not generalizable [55].

Explanatory Research

Explanatory research is mainly concerned with quantifying a relationship or to compare two or more groups with the aim to identify a cause-effect relationship. The research is often conducted through setting up controlled experiment. This type of study is a fixed design study, requiring that all the factors involved in it are fixed before the study is executed. Fixed design research is also known as *Quantitative research*, since it exploits quantitative data. Quantitative data may be useful when there is the need to evaluate the effects of some manipulations or activities. This kind of research requires that the data are compared by means of statistical, mathematical analysis. Quantitative research foresees the investigation of a phenomenon by collecting and analyzing numerical data using mathematical methods [56] with the aim to quantify the interrelations between different types of variables (e.g. independent, dependent), mainly using statistical techniques [57]. Quantitative research is based on a conventional and a systematical process to gather data, which describes the information by cause and effects relations in a rigorous way [58]. Quantitative research poses its ground on the assumption that the world operates with a set of physical and natural laws and it provides the means to test a hypothesis and sometimes conducts studies to observe the cause-and-effect relationships among variables with empirical investigations.

1.3.2 Case Study Research

Case study has been widely adopted in the Software Engineering community, often referring to study of specific case, in contrast to a sample from a specified population. However, these studies claimed to be case studies ranged from very ambitious and well-organized studies in the field of operations (in vivo) to small toy examples in a university lab (in vitro) [17].

Case study is a commonly used research strategy in social science context (e.g., [59]), where they are conducted with the objectives of not only increasing knowledge but also bringing change in the phenomenon being studied. Software engineering research has similar high-level objectives, that is, to better understand how and why software engineering should be undertaken and, with this knowledge, to seek to improve the software engineering process and the resultant software products. Case studies offer an approach that does not require a strict boundary between the object of study and its environment. Case studies do not generate the same results on, for example, causal relationships, as controlled experiments do, but they provide a deeper understanding of the phenomena under study [17]. Case studies have been criticized for being of less value, being impossible to generalize from, being biased by researchers, and so on. This critique can be met by applying proper research methodology practices and by reconsidering that knowledge is more than statistical significance [60].

Software engineering is also distinctive in the combination of diverse topics. Glass et al. [61] describe software engineering as an intellectually intensive, nonroutine activity, and Walz et al. [62] describe software engineering as a multi-agent cognitive activity. Many of the interim products are produced either intentionally by the actors (e.g., the minutes of meetings) or automatically by technology (e.g., updates to a version of control system). Therefore, one of the distinctive aspects of software engineering is the raw data that are naturally, and often automatically, generated by the activities and technologies. There are clear overlaps with other disciplines, such as psychology, management, business, and engineering, but software engineering brings these other disciplines together in a unique way, a way that needs to be studied with research methods tailored to the specifics of the discipline. Case studies investigate phenomena in their real-world settings, for example, new technologies, communication in global software development, project risk and failure factors, and so on. Hence, the researcher needs to consider not only the practical requirements and constraints from the researcher's perspective, but also the objectives and resource commitments of the stakeholders who are likely to be participating in, or supporting, the case study. Also, practitioners may want to intervene in future projects – that is, change the way things are done in future projects – on the basis of the outcomes from the case studies, and often software engineering managers are interested in technology interventions, such as adopting a new technology. This includes both software process improvement (SPI) work [63] and design of solutions. There are, therefore, distinctive practical constraints on case study research in software engineering.

Case Study Process

As for other kind of empirical study, compare, for example, to the processes proposed by Wohlin *et al.* [64] and Kitchenham *et al.* [65], Case

Studies can be structured in the following five major steps:

1. Case study design – objectives are defined and the case study is planned.
2. Preparation for data collection – procedures and protocols for data collection are defined.
3. Collecting evidence – data collection procedures are executed on the studied case.
4. Analysis of collected data – data analysis procedures are applied to the data.
5. Reporting – the study and its conclusions are packaged in feasible formats for reporting.

However, differently from other empirical studies, the case study is a flexible design strategy, and can require an amount of iteration over the steps [66], which is explicitly modeled on a process by Verner *et al.* [67]. Also the data collection and analysis may be conducted incrementally. If insufficient data are collected for the analysis, more data collection may be planned. However, the case study should have specific objectives set out from the beginning. If the objectives change, it is a new case study rather than the existing one, though this is a matter of judgment like all other classifications. Empiricists in software engineering often complain about the lack of opportunities to study software development process in real settings. This really implies that we must exploit to collect and analyze as much data of as many different types as possible. Qualitative data is richer than quantitative data, so using qualitative methods increases the amount of information contained in the data collected [19, 18]. To these reasons I decided to exploit Case Study Research and qualitative research methods to evaluate how the proposed approaches are applied in real industrial contexts.

Chapter 2

Using Application Lifecycle Management and Model Driven Engineering for supporting Gap Analysis Processes

In this Chapter I investigate on Questionnaire based Gap Analysis process that are executed for assessing whether a company complies with the requirements prescribed by Standards or Quality Evaluation framework. The main issues affecting Questionnaire-based Gap Analysis processes in industrial practices were identified through a survey conducted in real industrial settings. Moreover, I evaluate the feasibility of adopting state-of-the-art software engineering technologies for executing such processes and overcoming the identified issues. Then, the Chapter describes a novel approach based on Application Lifecycle Management for configuring and enacting Questionnaire-based Gap Analysis processes. The approach exploits Model Driven Engineering for configuring and implementing the Application Lifecycle Management system. This configuration activity is aided by a tool, named Gap Analysis Design and GEneration Tool (GADGET), developed for modeling the process and automatically transforming it towards the Application Lifecycle Management technology. The feasibility of the proposed approach has been evaluated by a case study conducted in the automotive domain. Two different Questionnaire Based Gap Analysis processes have been configured and implemented in an Application Lifecycle Management system with the support of the GADGET tool. The resulting ALM system was used to perform the Gap analysis processes. Semi-structured interviews with the involved industrial personnel were conducted to carry out a qualitative evaluation. The case study results show that the introduction of ALM improves the quality of the Questionnaire Based Gap Analysis processes. Moreover, the adoption of MDE approach implemented by the GADGET tool provides a viable solution for configuring ALM systems.

2.1 Introduction

Nowadays, software development companies continually strive to refine and improve the adopted development practices in order to maintain and increase the competitiveness in their markets [5]. To show the quality of their development processes, or to improve them, these companies may be appraised with respect to well-known quality frameworks, such as the Capability Maturity Model Integration (CMMI) [6] or the Software Process Improvement and Capability Determination (SPICE) [7, 8]. Software organizations have to comply with the highest possible levels of maturity and capability defined by these frameworks, if they want to excel in competitions for software projects, win and retain more and more customers [68].

On the other hand, in safety-critical system domains, such as automotive, railway, or aerospace, the software organizations are even obliged to demonstrate that they do not pose undue risk to people, property, or the environment, showing their compliance with a Standard Development Approach [9]. To cite just a few examples, Standard approaches for developing safety critical systems exist for automotive (ISO 26262 - Road Vehicles Functional Safety [10]), Medical (IEC 62304 Medical device software - Software life cycle processes [11]), and Nuclear (IEC 61513 Nuclear power plants - Instrumentation and control important to safety [12]) industries. The standard approach is to carefully code, inspect, document, test, verify and analyze the systems being developed.

Before exposing themselves to a third party certification authority for a formal appraisal, companies usually need to perform a preliminary internal assessment to understand how well their processes compare to these standard methods or frameworks and to identify areas where improvement can be made [69]. Perform Gap Analysis processes is a common approach adopted in the practice for addressing this problem. Gap Analysis was introduced in the business context as a technique to identify discrepancies with respect to the achieving of an objective by Parasuraman *et al.* [70].

More in detail, the aim of Gap Analysis is to identify the *"gaps"* between the existing *'as-is'* state of the company and the target *'to-be'* situation to reach [71]. This analysis may rely on two different models that represent the existing *'as-is'* situation of the company and the desired *'to-be'* one, respectively.

However, modeling the *'as-is'* situation is usually not straightforward due to well-known difficulties in obtaining reliable, correct, and up-dated documentation about the processes implemented by the company.

As to the *'to-be'* model, there is the issue of obtaining it because Standards do not usually define the target models, but they just provide a set of requirements that must be satisfied in the *'to-be'* situation. As a consequence, the organization has the issue of defining this model, carrying out an expensive process of analysis and interpretation of the Standard, and designing its solution according to its own interpretation of the standard and its own perspective about it.

The task of defining the *'to-be'* model may prove to be very difficult to be automatized since it requires the analysis of large textual documents written in natural language that are not easily understood by everyone and are usually open to different interpretations when they are examined by different people [72, 73, 74]. This task may be even more difficult when the Gap Analysis is performed with respect to a Standard that allows specific requirements or methods to be not implemented by the company, provided that a valid rationale is supplied.

Since that, a more pragmatic approach to Gap Analysis processes that is usually applied in industrial settings is based on interviews or questionnaires [75].

The basic idea of Questionnaire Based Gap Analysis (QBGA), is to assess the "gaps" by means of questionnaires that helps the company in collecting evidences showing to what extent it is far from implementing all, or subsets of, the requirements defined by the Standard. To implement this approach, an organization just needs questionnaires that will be answered by process experts in the company, and then the collected answers will be used to evaluate the existing gaps.

QBGA processes present several open issues. A first issue regards the setup of the questionnaires that are usually not defined by the Standard, but need to be crafted by the company according to the specific goals of the gap analysis and adapted to the industrial context where they will be used. Designing a useful and usable questionnaire is usually not straightforward and requires a considerable manual effort, as already reported in the literature [76, 77, 78].

Further issues may regard the activities of questionnaires completion, analysis, and gaps evaluation. Implementing these activities requires the cooperation between different involved actors according to specific rules defined by the company. These activities are even more difficult when they are carried out without a proper tool support. The tools adopted in the practice usually are spreadsheets, or simple word processors [79, 80] that do not provide advanced or ad-hoc features like the ones provided by tools designed for managing the questionnaire compilation and interpretation process, such as Google Forms, SurveyMonkey. Finding more effective technologies and tools to support the execution of these processes and validating them in real industrial contexts is thus a relevant research issue.

ALM is a technology commonly used in companies for managing the software development process. ALM systems focus on the products to be developed and all their lifecycle activities, offering an integrated environment and tools for working on the products, managing their versions and the access by different persons and roles, automatically tracking and notifying the product progress, and so on [81]. A relevant characteristic of ALM systems is the possibility they offer to customize specific ALM projects, defining the characteristics of the managed products, their lifecycle rules, and possible constraints.

In this Chapter I focus on QBGA processes and I investigate the possibility of introducing ALM systems to support the implementation of such processes. To this aim, with the collaboration of my research group I preliminary carried out a survey in several industrial settings, in order to obtain the most critical issues experienced by companies in the execution of QBGA processes. Thanks to the survey, we were able to define the requirements of a tool, based on the ALM technology, that provides the built-in features for overcoming the most critical and common issues of QBGA processes.

To support the development of this tool I decided to rely on the MDE approach that allows systems to be developed starting from models at different degrees of abstraction and by applying automatic transformation techniques [82].

In order to implement the MDE approach for developing the ALM-based tool, I defined different metamodels that abstract the characteristics of a generic QBGA process and of ALM systems. Moreover, I developed a tool, named GADGET (Gap Analysis Design and GEneration Tool), that supports both the modeling and the transformation steps of my MDE process. The feasibility of the proposed approach has been evaluated in a case study I performed in the automotive industrial context of the Fiat Chrysler Automobiles (FCA) company. The results of the study showed that the proposed GADGET tool well supported all the steps foreseen by my proposed MDE process. Moreover, it showed that my approach aids the execution of a QBGA processes, positively affecting several of their quality characteristics.

This work contributes to the literature in software process quality and improvement in four main aspects:

- it presents a survey conducted in real industrial settings for understanding how QBGA processes are executed and their main issues;
- it proposes the adoption of the ALM technology for supporting QBGA processes execution;
- it defines a tool-supported MDE approach for generating an ALM-based tool supporting the QBGA process execution;
- it presents a case study conducted in a real industrial setting, showing the feasibility of the proposed approach for QBGA processes.

The reminder of the Chapter is organized as it follows: Section 2.2 reports the survey on gap analysis processes I carried out in industrial settings. Section 2.3 presents the proposed approach based on ALM, the MDE approach for supporting the design and development of the ALM based tool supporting QBGA process execution. Moreover, it illustrates the tool, named GADGET, I developed to support the design of QBGA processes and its automatic implementation in an ALM-based tool. Section 2.4 describes the industrial case study I carried out in collaboration with my research group in order to evaluate the feasibility of the proposed approach in supporting the development and execution of QBGA processes. Section 2.5 discuss related Work whereas Conclusions & Future Work are reported in Section 2.6.

2.2 The addressed Problem: QBGA processes in industrial settings

My work on QBGA started with a preliminary study that I carried out with my research group in real industrial settings with the aim of understanding:

- how QBGA processes are executed in real industrial scenarios;
- which are the most critical issues real companies face when these processes are performed.

To this aim, we conducted an industrial survey among employees of selected companies involved in QBGA processes, which focused on five different aspects: the performed activities, the actors involved in these activities and their roles, the artifacts they exploited and the ones they produced, the adopted supporting tools, and the issues that were most frequently faced in the execution of these processes.

The survey we conducted relied on the execution of the following steps:

1. choice of the methodology for conducting the survey,
2. selection of the employees to be surveyed,
3. design and production of the survey forms,

4. conduction of the survey,

5. collection and analysis of the answers,

6. validation of the results.

As for the methodology, we decided to carry out the survey through semi-structured interviews. Unlike the structured interview that has a rigorous set of questions which does not allow one to change or submit new questions during the survey execution, a semi-structured interview is open, allowing new ideas to be brought up during the interview as a result of what the interviewee answers. This freedom can help interviewers to tailor their questions to the interview context/situation and to the people they are interviewing.

As regards the people involved in the study, we chose to select personnel working in three different companies. The companies were selected since they were involved in research projects carried out by our group and they performed their activities in different industrial sectors, i.e. automotive and embedded systems. They had different sizes in term of number of employees and revenues. More precisely, the study involved three sub-units of these companies that were involved in process assessment activities. The three sub-units had different sizes in terms of their employees number, being composed by 8, 15 and 40 employees, respectively. We selected these sub-units since they had been involved in the assessment of different Standards. For each sub-unit, we selected five employees that had more than two years of experience in process assessment and that had been involved in QBGA processes related to different Standards or Evaluation Frameworks. The selected employees agreed to participate in the study.

We designed and prepared an interview guide that is actually an informal grouping of topics and questions that the interviewer can ask in different ways for different participants. We organized the guide in different sections. Each section contained questions about one of the aspects of the Gap Analysis we intended to focus on that are the roles of involved actors, the involved artifacts, the adopted tools, the performed activities, and the emerged issues.

Then we conducted the semi-structured interviews with the selected subjects and collected their answers. Lastly, the data collected in each company was analyzed and coded in order to gain findings about the gap analysis process execution. The findings were validated through data triangulation and allowed us to answer the reported questions. The results that emerged from this study are reported in the following subsections.

2.2.1 Roles of the Involved Actors

In the considered companies we abstracted the following roles of the actors involved in the execution of QBGA processes:

- *Questionnaire Developer* that is an actor involved in the development of the questionnaire for executing the QBGA process.
- *Questionnaire Responsible* that is in charge of assigning to appropriate employees the responsibility to compile parts of the questionnaire and to analyze them.
- *Respondent* who has the responsibility to answer subsets of the questions composing the questionnaire.
- *Reviewer* that is involved in the analysis and evaluation of the answers provided by the Respondents.
- *Internal Assessor* that is in charge of assessing the gaps of the company with the considered Standard according to the answers given by the Respondents to the questionnaire.

2.2.2 Involved Artifacts

The following artifacts are produced and/or exploited for conducting QBGA processes:

- *Standard* or *Evaluation Framework* document that contains the Standard/Evaluation Framework the company is willing to comply with.
- *Questionnaire* that is an artifact containing the questions designed to understand the gaps of the company with respect to the requirements reported in the considered Standard.
- *Employees Questionnaire Assignment Form* that reports the questions assigned to each selected employee involved in the QBGA process
- *Gap Analysis Report* that lists the gaps identified by executing the QBGA process.

2.2.3 Adopted Tools

The features offered by the following tools are exploited in the considered companies for executing QBGA processes:

- *Spreadsheet tool* for crafting the Questionnaires, the Employee Questionnaire Assignment Forms and the Gap Analysis Reports and for supporting their completion.
- *Word Processor tool* for crafting the Questionnaires and supporting their completion.
- *Version control system* for managing the shared repository storing and versioning the Questionnaire artifacts

2.2.4 Performed Activities

Even if in the considered companies no proper process was defined to carry out QBGA, we were able to understand that it was usually carried out in three sequential steps, where three activities we named *Questionnaire Development*, *Questionnaire Completion*, and *Questionnaire Analysis* were performed.

The *Questionnaire Development* activity was devoted to the implementation of the questionnaires related to a considered standard. Usually, these questionnaires were appositely crafted by companies employees who had knowledge about the Standard.

The *Questionnaire Completion* activity consisted in the distribution of the questionnaire to the employees who were in charge of providing answers to the questions related to specific aspects reported in the Standard.

In the *Questionnaire Analysis* activity, further personnel analyzed the answers provided to the questionnaires, in order to identify the existing gaps with respect to the considered Standard.

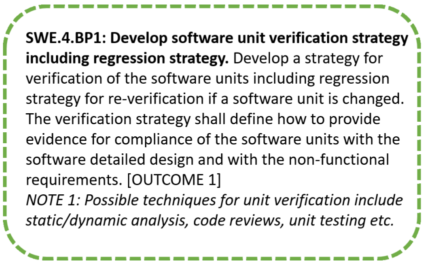
In the following sections I report specific details about these three activities, which emerged through the survey.

Questionnaire Development activity

This activity was usually conducted by employees who had knowledge about the considered Standard.

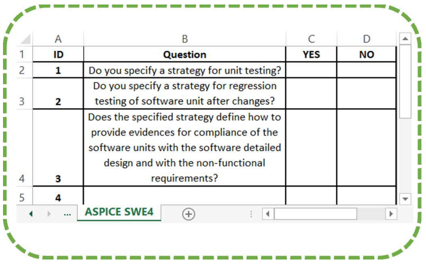
We interviewed a number of employees having been assigned with the Questionnaire Developer role in the considered companies. Questionnaire Developers declared that they did not use any specific approach or tool to carry out this activity, but usually they proceeded by analyzing the parts of the Standard and developing questions, or checklists for each of its relevant parts. The parts of the Standard to consider were defined according to the specific goals defined by the company. Depending on the specific Standard and its considered requirements, either multiple-choice or essay questions were developed.

As an example, Figure 2.1a reports an excerpt of the *SWE.4.BP1* requirement from the Automotive SPICE (ASPICE) Standard. It regards the strategy applied for software unit testing. Figure 2.1b shows three two-choice questions defined in one of the considered company to assess how this requirement is implemented.



SWE.4.BP1: Develop software unit verification strategy including regression strategy. Develop a strategy for verification of the software units including regression strategy for re-verification if a software unit is changed. The verification strategy shall define how to provide evidence for compliance of the software units with the software detailed design and with the non-functional requirements. [OUTCOME 1]
NOTE 1: Possible techniques for unit verification include static/dynamic analysis, code reviews, unit testing etc.

(a) ASPICE SWE.4.BP1 Requirement

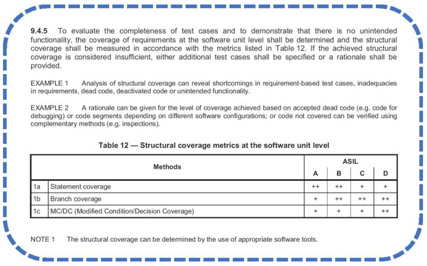


ID	Question	YES	NO
1	Do you specify a strategy for unit testing?		
2	Do you specify a strategy for regression testing of software unit after changes?		
3	Does the specified strategy define how to provide evidences for compliance of the software units with the software detailed design and with the non-functional requirements?		
4			

(b) Questions related to the ASPICE SWE.4.BP1 Requirement

Figure 2.1. Excerpt of a Requirement reported in the ASPICE and its related questions

In another company, essay questions were considered more adequate to gain as much information as possible to assess the level of compliance of the company with specific requirements of a Standard. Figure 2.2a shows the 9.4.5 Requirement from the Part 6 of the ISO 26262 standard related to the coverage of requirements at software unit level. This requirement reports the possible coverage metrics to be adopted in testing at software unit level and the required applicability for the four safety levels, also known as Automotive Safety Integrity Level (ASIL), specified by the Standard. For this requirement, four essay questions were defined in a spreadsheet document. This was needed since the company wanted to obtain more information about the applied methods and the rationale that motivated their choices.



9.4.5 To evaluate the completeness of test cases and to demonstrate that there is no unintended functionality, the coverage of requirements at the software unit level shall be determined and the structural coverage shall be measured in accordance with the metrics listed in Table 12. If the achieved structural coverage is considered insufficient, either additional test cases shall be specified or a rationale shall be provided.

EXAMPLE 1 Analysis of structural coverage can reveal shortcomings in requirement-based test cases, inadequacies in requirements, dead code, deactivated code or unintended functionality.

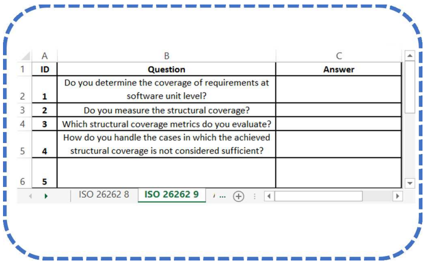
EXAMPLE 2 A rationale can be given for the level of coverage achieved based on accepted dead code (e.g. code for debugging) or code segments depending on different software configurations; or code not covered can be verified using complementary methods (e.g. inspections).

Table 12 — Structural coverage metrics at the software unit level

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

NOTE 1 The structural coverage can be determined by the use of appropriate software tools.

(a) ISO 26262 9.4.5 Requirement



	A	B	C
1	**ID**	**Question**	**Answer**
2	**1**	Do you determine the coverage of requirements at software unit level?	
3	**2**	Do you measure the structural coverage?	
4	**3**	Which structural coverage metrics do you evaluate?	
5	**4**	How do you handle the cases in which the achieved structural coverage is not considered sufficient?	
6	**5**		

(b) Questions related to the ISO 26262 9.4.5 Requirement

Figure 2.2. Excerpt of a Requirement reported in the ISO 26262 Standard and its related questions

The tools used for developing the questionnaires mostly included word processors and spreadsheets that typically do not provide any explicit feature for traceability management. As a consequence, the interviewed subjects of all the considered companies complained about difficulties in tracking the parts of the standard already accounted for, as the complexity of the questionnaire grew up. They also reported difficulties in managing the questionnaires as their size increased.

Questionnaire Completion activity

Once the questionnaires have been produced, they were distributed to selected employees assigned with the *Respondent* role. In two out of the three considered companies we found other additional actors, the *Reviewers*, involved in this activity. When all the questions had been answered, the Reviewers analyzed and validated them. When they did not accept a given answer, a change request was usually sent to its Respondent that had to analyze and answer the question again.

The overall Questionnaire Completion activity was usually managed by an employee having the *Questionnaire Responsible* role. Usually these tasks were not supported by specific tools. In two of the considered companies, the actors having the Responsible role sent multiple copies of the same questionnaire by email to the respondents. The Respondents had to (1) identify in the questionnaire they received via mail their assigned questions, (2) answer them and (3) sent the compiled questionnaire back to the Responsible. All the Responsibles complained about their difficulties in observing the Questionnaire completion progresses, reporting that they were constrained to contact via email the Respondents and ask them about the status of the assigned tasks.

In another company the Responsible made available to the Respondents in a shared file repository a single questionnaire. The files could be accessed by more respondents at the same time, thus resulting in frequent version conflict errors due to multiple concurrent accesses. In this case, the Responsible compiled the Employees Questionnaire Assignment Form, reporting which questions an employee had to answer, and made it accessible in the same shared file repository.

All the considered companies disregarded the adoption of more effective solutions, such as web-based systems for conducting surveys, because these tools did not adhere with the company's confidentiality and security policies.

Another relevant issue regarded the questionnaire review management process that was triggered when the Reviewer did not accept the answers given by respondents and required further modifications. Whether the answers were not accepted, the Reviewers had to identify the involved Respondents and to contact them by email, asking to improve the answers. In all the companies, no specific tool was exploited to track and manage this process.

From the analysis of the executions of this activity, we understood that the questionnaire and its elements showed a lifecycle that was not effectively managed during the QBGA process execution.

Questionnaire Analysis activity

In all the considered companies, this activity was performed by the *Internal Assessor* who interpreted and analyzed the answers collected by the compiled questionnaires in order to assess the gaps with respect to the considered Standard. One of the problems that arose from the survey regarded the difficulty of the assessors in gathering answers that were scattered among several copies of the questionnaire and in obtaining aggregate views about the collected data. This work was even more difficult when many versions of the same answers existed, since they had been reviewed multiple times. Moreover, almost all the Assessor complained about the lack of features for aggregating data related to specific parts of the Standard they wanted to analyze.

Since all these issues, the Assessors used to perform the analysis only when the Questionnaire Completion activity was completed. Nevertheless, in all the considered companies the Assessors expressed the need for features aiding the analysis that allowed them to easily aggregate answers related to the different parts composing the Questionnaire and that eased the application of the rating methods they had chosen.

As another aspect, almost all the interviewed assessors requested features for gain visibility of the questionnaire completion process to carry out the analysis of each part of the Questionnaire as soon as possible for detecting possible gaps and to plan appropriate improvement actions.

2.2.5 Emerged Issues

According to the data obtained through the conducted interviews, we were able to identify the main issues affecting QBGA processes in real industrial settings that I reported in Table 2.1.

Table 2.1. QBGA Process - Emerged Issues

QBGA Issue	Description
I_1	Managing the traceability between the developed Questionnaire and the document of the Standard
I_2	Observing the questionnaire completion progress
I_3	Managing multiple versions of the Questionnaire
I_4	Handling Questionnaire version conflict errors
I_5	Defining the lifecycle of the Questionnaire elements
I_6	Managing the Questionnaire review process
I_7	Merging different questionnaire versions
I_8	Gathering questionnaire data
I_9	Aggregating questionnaire data
I_{10}	Visualizing questionnaire data
I_{11}	Handling the assignments and permissions of the actors on the Questionnaire
I_{12}	Managing the communication among the involved actors

2.3 The proposed QBGA Approach based on ALM

The industrial survey provided me the evidence that the approaches and the tools typically adopted to implement, to compile and to analyze the Questionnaires were considered largely unsatisfactory from the point of view of the actors involved in the process. Tools providing ad-hoc features for supporting the QBGA process execution were needed.

2.3.1 Adopting ALM for supporting QBGA processes

To overcome the emerged limitations, I considered the possibility of exploiting the ALM technology. ALM is the process of managing the entire lifecycle of a software product. It encompasses requirements management, software design, programming, software testing, software mainte-

Table 2.2. ALM Features

ALM Feature	**Description**
F_1	Manage the lifecycle of work items and software artifacts via customized workflows
F_2	Store the artifacts in version control repositories, so every modification produces version history record
F_3	Enable real-time communication among involved actors by means of threaded discussions, wikis, notifications, and alerts
F_4	Implement and assure the traceability links among the work items and software artifacts involved in the process
F_5	Aid the collaborative work through concurrent access to all the work items and software artifacts
F_6	Manage the roles of the actors involved in the process and their privileges and permissions on the work items and software artifacts workflows
F_7	Monitor real-time the progresses of the process execution via customized dashboards, reports and rich views
F_8	Enable comment on all work items, approve them, and verify approvals with digital signatures

nance, change management, continuous integration, project management, and release management. Throughout the ALM process, each of these steps is closely monitored and controlled, followed by proper tracking and documentation of any changes to the application.

Nowadays, several commercial and Open Source ALM systems, such as Tuleap[1], IBM Rational Team Concert[2], Siemens Polarion ALM[3], are available and are effectively exploited in different industrial settings. Table 2.2 presents the main built-in features they provide.

The ALM tools are highly configurable, allowing to adapt their exposed features to the specific context and process to support. Their built-in

[1] *https://www.tuleap.org/*
[2] *https://www.ibm.com/software/products/it/rtc*
[3] *https://polarion.plm.automation.siemens.com/products/polarion-alm*

features are usually exploited to aid the execution of software development activities [83] and to support process improvements [84].

In this Chapter, I propose the adoption of ALM systems for supporting QBGA process executions, since I observed that the ALM features reported in Table 2.2 could be exploited to overcome the QBGA process issues listed in Table 2.1. Table 2.3 shows for each ALM feature the QBGA process issues they allow to solve.

Table 2.3. ALM Features and the QBGA Process Issues they solve

ALM Feature	**QBGA Process Issues**
F_1	I_5,
F_2	I_3, I_4, I_7
F_3	I_{12}
F_4	I_1
F_5	I_7
F_6	I_{11}
F_7	I_2, I_8, I_9, I_{10}
F_8	I_6

My approach for supporting QBGA processes is based on four main steps:

1. design the QBGA process;
2. develop the ALM-based tool for supporting the QBGA process execution;
3. questionnaire completion exploiting the implemented ALM-based tool;
4. questionnaire analysis through the features offered by the ALM-based tool.

The first step of the process is devoted to the design of the QBGA process and the configuration of the ALM system in order to implement all the features required to support its execution. Any ALM system must be preliminary configured for introducing it in a specific context and it need to be tailored for supporting a given process. This configuration step aims at specifying relevant properties of the supported process, such as the artifacts involved in the process, their structural characteristics, the worflows, the rules and the roles of actors involved in the different workflow activities. This information is usually provided by an ALM project. By deploying such ALM project, it is possible to realize an ALM-based tool supporting the specific process execution.

This configuration step requires a high manual effort to manually define, through appositely user interfaces exposed by the ALM system, all the properties of the QBGA process for the considered Standard [85], resulting in time consuming activities. In order to carry out this configuration step, a deep knowledge about the configuration features implemented by the adopted ALM system and on how to map them with the QBGA process concepts are needed.

In order to aid this configuration step, I developed a tool, named GADGET, that offers a viable and more user-friendly solution to implement an ALM project supporting QBGA processes. The tool allows a user to describe the subject QBGA process using just models of the process to be implemented and of the Standard it relies on. Once the models have been defined, they will be automatically translated into an ALM project using transformation engines, according to the implementation details of the target ALM technology. All this approach is coherent with the MDE principles. MDE simplifies the development of systems by focusing on models of the problem domain rather than on implementation details. MDE approaches are being effectively applied in different contexts [86, 86, 87] among which software processes [88]

More in detail, MDE combines Domains-Specific Modeling Languages with Transformation engines/generators. Within a specific domain, the Domains-Specific Modeling Languages are used to formalize the internal structure of the software system to be implemented and its behavior, independently from the target technologies used for developing it. On the other hand, the Transformation engines/generators are platform dependent since they are able to synthesize automatically, starting from the analysis of models, various types of artifacts, such as source code, XML deployment descriptors and others [89]. In a typical MDE process [90] transformations are defined on metamodels that abstract the characteristics of the models to be developed. The transformations are aimed at translating models defined according to a source metamodel toward models complying with the target metamodel.

Using MDE, the GADGET tool will aid the definition of suitable models of the QBGA process from which it will automatically develop and configure the ALM project supporting its execution.

The models will have to define all the structural and behavioral aspects of the process, in accordance with specific metamodels that abstract characteristics and constraints of the application domain. The models will be automatically translated into the ALM project supporting the process execution, using automatic transformation rules. These rules will be also defined starting from a metamodel that describe the characteristic aspects of any ALM systems and taking into account the specific ALM platform where it is intended to be deployed.

2.3.2 Configuring the ALM using the MDE approach

The MDE process I defined for the design and development of the ALM-based tool supporting QBGA processes is described in Figure 2.3. This Figure reports a SPEM[4] diagram, representing activities, artifacts,

[4] *http://www.omg.org/spec/SPEM/2.0/*

roles, and tools of a given process using specific SPEM stereotypes.

The process comprises three sequential activities, namely *Modeling the QBGA Process*, *Generating the QBGA ALM Model*, and *Implementing the QBGA ALM project* that can be executed with the support of the GADGET tool.

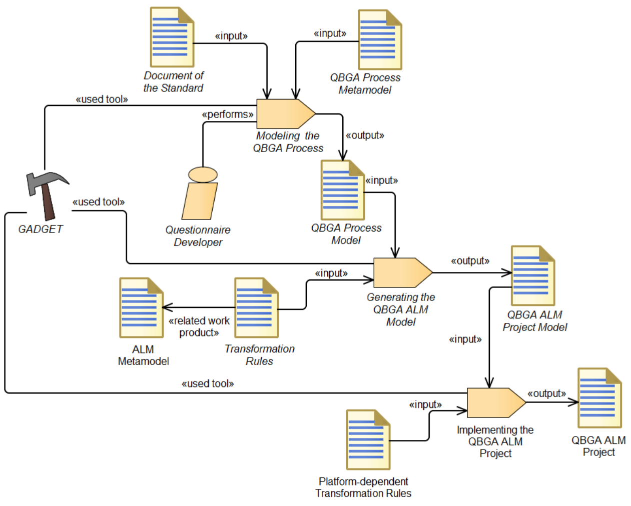


Figure 2.3. The MDE process for implementing the ALM-based tool

The *Modeling the QBGA Process* activity is carried out by the *Questionnaire Developer* who is in charge to produce the *QBGA Process Model* on the basis of three specific input artifacts, namely the *Document of the Standard*, the *Metamodel of the Standard*, and the *QBGA Process Metamodel.* In this step, the designer is aided by the GADGET tool that offers a modeling environment equipped with a rich user interface that provides features for the design of the process model.

In the *Generating the QBGA ALM Model* activity, the QBGA Process Model is automatically translated into a QBGA ALM model applying a set of *Transformation Rules* defined according to an *ALM Metamodel.*

Then, the *Implementing the QBGA ALM project* activity produces a *QBGA ALM Project* for a specific ALM technology, applying a set of *Platform-dependent Transformation Rules* to the QBGA Process Model.

Both the *Generating the QBGA ALM Model* and *Implementing the QBGA ALM project* activities are fully automated and are performed by the GADGET tool.

By deploying the ALM project on an ALM system, I obtain an ALM-based tool able to support the execution of the Questionnaire Completion and Analysis activities, as it is shown in Figure 2.4.

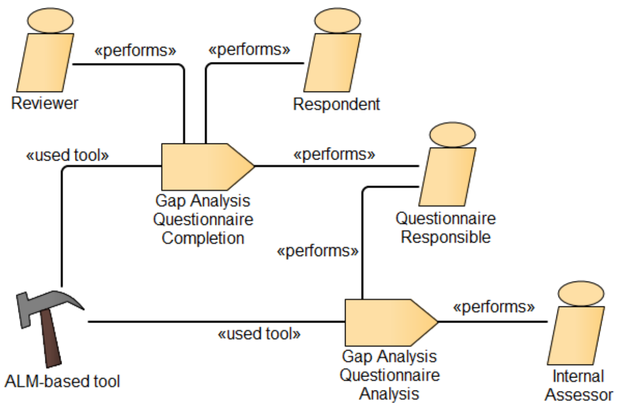


Figure 2.4. Questionnaire Completion and Analysis activities

The proposed MDE approach relies on two different metamodels: the former is used to characterize the QBGA Process application domain, whereas the latter one aims to specify the characteristics of the ALM target platform that will be used to implement the tool supporting the QBGA process execution. Details about the proposed metamodels are reported in the following.

The QBGA Process Metamodel

The first metamodel abstracts the characteristics of a QBGA process. A QBGA process metamodel should comprise the concepts related to a given Document of the Standard that the company is willing to comply with, the Questionnaire for understanding if the company can satisfy the requirements the Standard and the lifecycle of the elements of the process that defines how it is to be carried out. I report the description of these different aspects of the QBGA process Metamodel in the following.

The Document of the Standard

The first part of the QBGA process metamodel describes the Document of the Standard that contains all the requirements to be satisfied. The problem of defining models for representing the content of a specific Document of a Standard is not novel. Different works described in the literature try to identify the peculiar elements characterizing a given Standard and describe them by exploiting models, profiles or metamodels [91, 92].

However, differently from most of the works presented in the literature, I aimed at defining a generic metamodel that could be able to represent the characteristics of more than one Standard. To this aim, I considered several Standards defined for different purposes and in different domains and tried to abstract their common characteristics. Examples of Standards that I considered are the ISO/IEC 61508 (Functional Safety of Electrical/-Electronic/Programmable Electronic Safety-related Systems in a generic domain), the ISO 26262 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems in the automotive domain), the ISO 10303 (Product data representation and exchange for production manufacturing processes), the ISO/IEC 15504 (Software Process Improvement and Capability Determination in any domain) and the Automotive

SPICE (Software Process Improvement and Capability Determination in the automotive domain).

The metamodel reported in Fig. 2.5 proposes a representation of a Document of the Standard that is made of the items and relations among these items that are relevant to carry out a QBGA Process.

According to this metamodel, the *Document of the Standard* is composed by several *StructuralElements*. They represent the different parts in which the document is organized (e.g. headers, chapters, paragraphs).

As an example, the ISO 26262 standard is structured in *Parts*, composed by *Clauses*, structured in *Sub-Clauses* as reported in its Table of Contents.

A StructuralElement may contain other StructuralElements. As an example, a Document of the Standard may be composed of Sections that may be divided into more Chapters. Moreover, a Structural Element may reference other ones when the information it reports should be integrated with the one contained in the referenced Structural Elements .

Each StructuralElement contains one or more *Requirement*s requested by the Standard. A Requirement may report, among other things, constraints that need to be respected or techniques that should be applied.

The *Document of the Standard* may define several *Level*s of capability of the company to apply what requested by the Standard. These levels influence the applicability of the Requirements reported in the Standard. Depending on the Level, the applicability for each requirement may be Mandatory, optional or not required at all.

To represent this aspect, I introduced the *RequiredApplicability* class in Figure 2.5 having the attribute *applicabilityValue*.

Examples of possible RequiredApplicability values may be: (1) *Mandatory*, (2) *Optional*, (3) *Not applicable*.

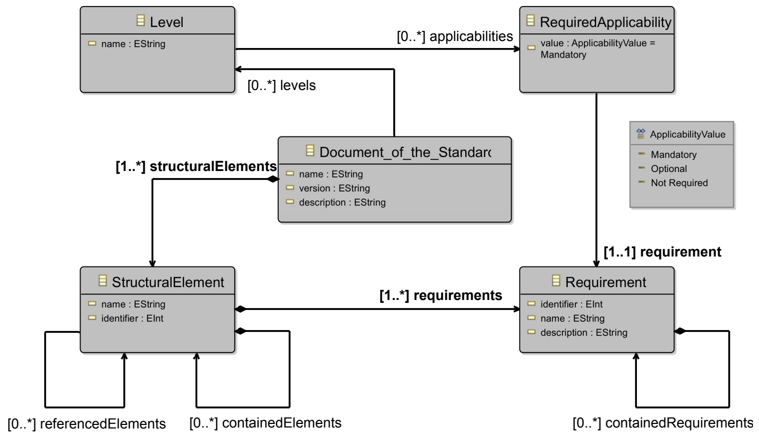


Figure 2.5. QBGA Process Metamodel - Document of the Standard

The QBGA Questionnaire

The Gap Analysis process I am interested in centers on the questionnaire that will be developed, completed, and analyzed during the QBGA process. The main aspects of the questionnaire for a QBGA process are described by the Metamodel shown in Figure 2.6.

This metamodel defines how a Gap Analysis Questionnaire can be structured. The main element of this Metamodel, the *Questionnaire*, contains all the *Questions* defined for each identified Requirement of the Standard. The Questionnaire is defined according to the *Goal* the company is willing to achieve by the execution of the QBGA process. Different questionnaires could be associated to a *Document of the Standard*, according to different goals pursued by the company.

Each *Question* reports a textual description that is defined for understanding if the company complies with a given Requirement. Questions can be either *Essay* or *MultiChoice* questions. Each Question is linked to the *Answer* that is provided to it. The *Answer* reports the given response and a *rationale* for this choice.

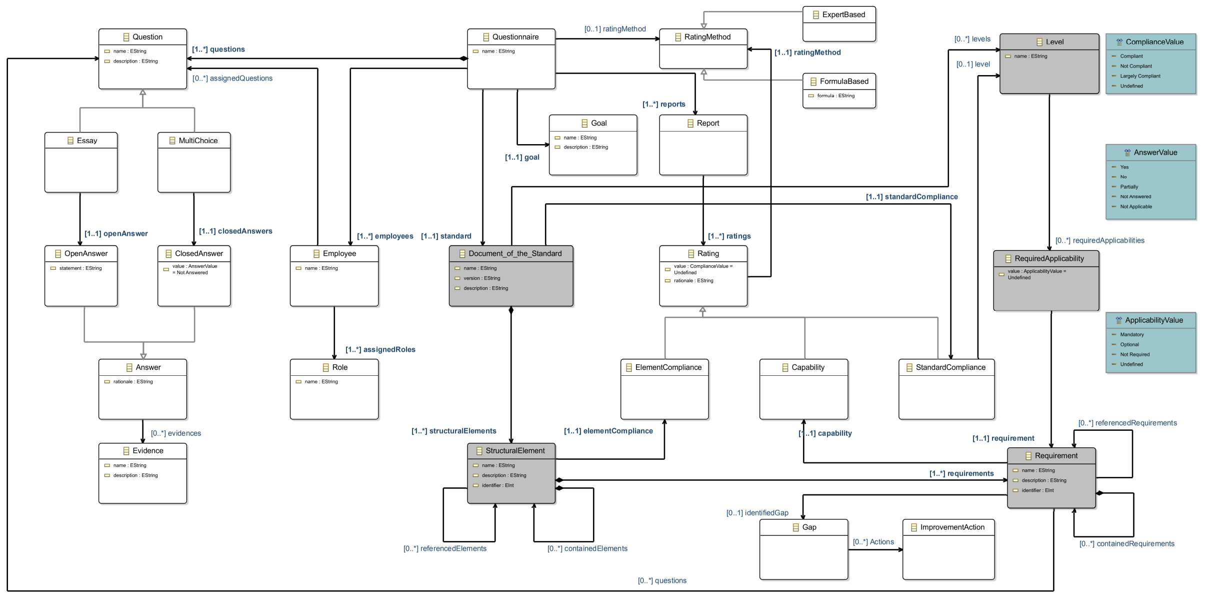


Figure 2.6. QBGA Process Metamodel - Questionnaire

Depending on the type of Question, Essay or MultiChoice, the answer either may be a *statement*, or it may assume one of its multiple predefined *values*, respectively.

Possible predefined values of answers may be: *Yes*, *Partially*, *No* or *Not Applicable* representing the cases in which the company is able to fully, partially, does not achieve, or it is not possible to apply what requested by the question, respectively. The possible AnswerValues usable for answering a MultiChoice Question may be defined according well-known appraisal methods (e.g. Standard CMMI Appraisal Method for Process Improvement (SCAMPI)), such as Yes, No, Not Applicable, Not Answered.

Each answer may be characterized by a *rationale* that is used to explain the reasons behind the given answer.

The rationale of an answer is relevant especially in case of Standards for which the compliance can be met even if one of its requirements is not satisfied, but a valid rationale is given for that choice.

The Questionnaire is carried out by different *Employees.* Each Employee may have different *Roles* in the execution of the questionnaire, such as Respondent, Internal Assessor, and so on. Each Question is assigned to an Employee that has the responsibility to answer it.

As another aspect, an Answer may be linked to different Evidences. An *Evidence* represents the proof demonstrating the fulfillment of a given requirement. Depending on the specific Standard, the evidences may be either statements provided by an actor involved in a specific process, or the produced artifacts. As an example, to demonstrate that the Unit Testing activity is carried out in the company, reports produced as output of this activity may be attached as Evidence.

The metamodel introduces also the elements that are needed to evaluate the gaps, at different degrees of detail, on the basis of the provided answers. The *Rating* represent an evaluation of the compliance. It is characterized by a complianceValue, that reports the defined value of compliance and a rationale attribute. This class has different specialization. To evaluate if and to what degree the company is able to comply with each Requirement, the Rating is specialized by the *Capability* that is associated to a Requirement.

To understand if and to what degree the company is able to comply with all the Requirements belonging to a given StructuralElement, the Rating is extended by the *ElementCompliance*. It is associated to a StructuralElement and it presents the complianceValue attribute reporting the compliance level for the associated StructuralElement.

Similarly, the Rating is also specialized by the *StandardCompliance*, having the *complianceValue* to indicate the level of compliance of the company with respect to all the considered parts of the Standard.

The values for the capability of Each Requirement, each StructuralElement and of the Document of the Standard are defined according to the RatingMethod defined for the Questionnaire.

For each Questionnaire it is possible to apply a specific RatingMethod that could be of two different types, namely *ExpertBased* and *FormulaBased*. It the first case, the evaluation is carried out by an expert. For the FormulaBased, the evaluation is automatically performed by applying the defined *formulas*.

When the analysis of the Gap Analysis Questionnaire will be accomplished, the results of this analysis will be summarized in different defined *Reports*. A Report contains information about the values of compliance reached for the considered Requirements of the Standard and the eventually identified gaps.

In addition, if a *Gap* related to a Requirement of the Standard is identified, the company may define *Improvement Actions* in order to reduce it and to reach its defined goal.

The QBGA process elements lifecycle

The proposed metamodel explicitly represents even the *Dynamic* aspects of the process that are necessary to rule the activity of Questionnaire completion. For readability, these aspects are reported in the diagram of Figure 2.7.

In order to specify how the QBGA process should be carried out, it is needed to define the lifecycle for the Question and Rating elements.

To this reason, each one of these elements has its own *Lifecycle* that is described by the *State*s that the element can assume and all the possible *Transition*s among them. A transition is characterized by its starting and ending states. Each transition can be triggered only by actors having specific *Role*. Moreover, a transition can be triggered only if peculiar *Rule*s are met. As an example, a Rule for the workflow of a Question may state that is not possible to trigger a transition from a *InProgress* state to the *Done* State if the value for its related Answer has not been set.

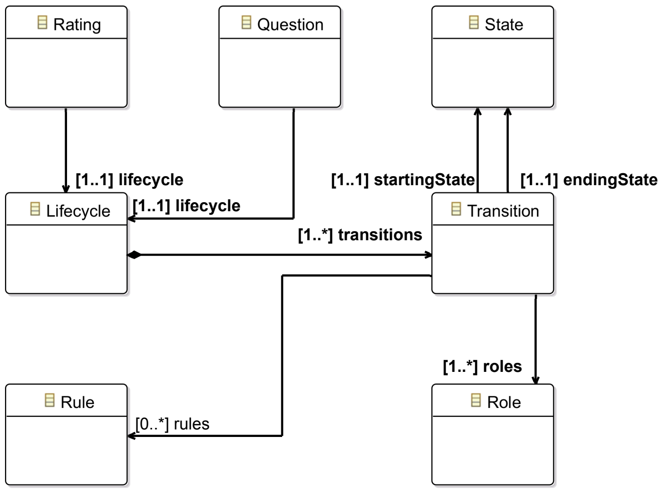


Figure 2.7. QBGA Process Metamodel - Process elements lifecycle

Possible roles that may be involved in the execution of a QBGA process

Table 2.4. Example of Question Lifecycle - States Transitions Table

State	**Initial**	**In Progress**	**Done**	**Reviewed**
Initial		startProgress		
In Progress			markDone	
Done		markUnreviewed		markReviewed
Reviewed		reOpen		

Table 2.5. Example of Roles Transitions Table

Role	**Transition**
Assignee	startProgress markDone
Reviewer	markReviewed markUnreviewed

are: (1) the *Respondent*, representing an employee to whom a question has been assigned. The Respondent, on the basis of his knowledge and experience, is the only one having the rights to answer the questions assigned to him and to provide the related rationale; (2) the *Reviewer* representing a user having the rights to review the answered questions and to evaluate the element assigned to him; (3) the *Administrator* who has the right to assign the roles to the users and to assign the questions to the assignees and the evaluation of the elements to the reviewers.

An example of a Question lifecycle is reported by the states-transitions matrix reported in Table 2.4 and by the UML State Machine diagram in Figure 2.8.

The Table 2.5 reports an example of how the roles involved in the QBGA process may handle the lifecycle of a Question.

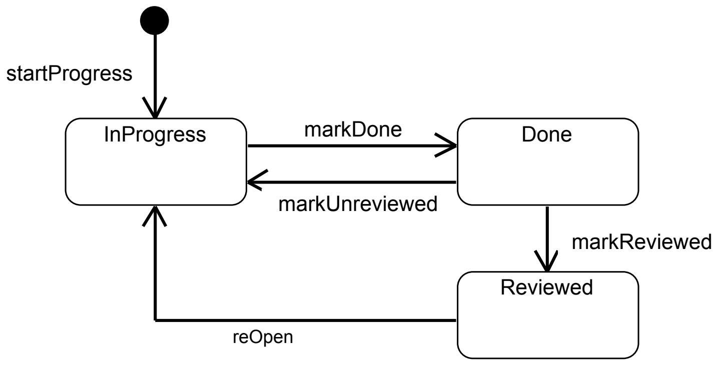


Figure 2.8. Question Lifecycle - Statechart Diagram

The Metamodel of the ALM platform

The other metamodel need by my approach describes the characteristics of ALM systems that I chose as target technology to support the execution of QBGA processes. A first attempt to model all the concepts related to ALM systems was carried out by Picha *et al.* in [93]. In this work, an ALM metamodel was defined in order to collect data for identifying project management anti-patterns. I adapted and extended the proposed metamodel taking into account the ALM features I exploit in the QBGA process execution. The ALM metamodel I introduced is reported in Figure 2.9.

An ALM is structured in *Projects*, that may be organized in different *ProjectSegments*, representing specific activities of the project. A peculiar element managed by the ALM process is the *WorkItem*. It may be specialized in *WorkUnit*, representing a task to carry out during the project, or in *Artifact* produced or needed to carry out a task. Each ProjectSegment may be structured in different WorkUnits, that may require different WorkItems. A WorkItem may have *CustomFields*, that represent its peculiar properties, such as a filename for an Artifact. Moreover, each WorkItem has a *Workflow*, representing its lifecycle. The Work-

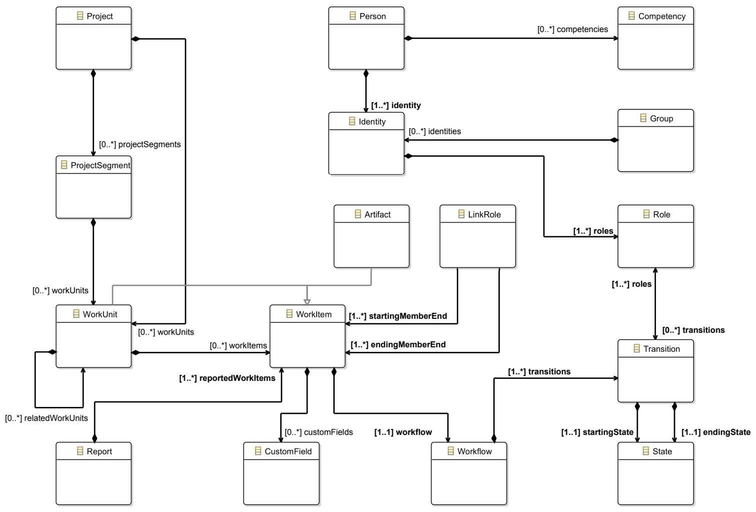


Figure 2.9. ALM Metamodel

flow is characterized by possible *Transitions* among its possible *States*. WorkItems may be related to each other through a *LinkRole*, that may be used for traceability purposes or to represent logical connections among WorkItems. A LinkRole represents a unidirectional relationship between a *startingMemberEnd* and an *endingMemberEnd* WorkItem.

An ALM may involve different *Actors*, each having its own *Competencies*. A competence represents expertise, certifications and skills of a person. An Actor may have different *Identities* on the system, representing its different account on the ALM system. An actor may have a specific Identity for each of the specific tools composing the ALM. The ALM allows to define *Groups* for aggregating different Identities. Moreover, an Identity may have several *Roles*. Each Role may have specific permissions for triggering Transitions on WorkItems Workflow.

2.3.3 The GADGET Tool

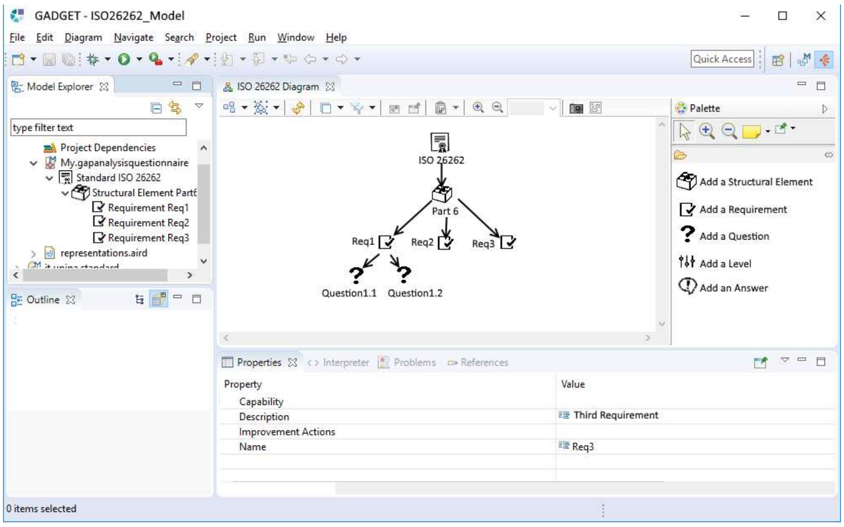
In order to support the QBGA process design and implementation, GADGET was designed to provide two different types of features: modeling features for supporting the definition of the QBGA process and transformation features to automatically produce the ALM-based tool for the QBGA process execution. GADGET was developed as a graphical modeling workbench exploiting the features offered by Sirius. Sirius is an Eclipse project which allows to easily create graphical modeling workbench by leveraging the Eclipse Modeling technologies[5].

More in detail, regarding the modeling support, GADGET provides an editor and a graphic palette that can be exploited to define models of QBGA Processes. It guides the design of these models by checking if these models are compliant with the defined metamodels at editing time. As an example, during the modeling phase, an error is highlighted if one of the defined Requirement is not linked to a Structural Element.

Figure 2.10 shows the modeling user interface provided by GADGET. It allows to design the model by exploiting the elements reported in the Palette in the right side, to populate the diagram. It is possible to define the attributes values of each selected element of the model, by means of the properties box located in the bottom. Moreover, it exposes a Tree View of the defined questionnaire in the box located in the left side of the window.

It also provides specific views for the visualization of the QBGA process dynamic aspects. As an example, in Figure 2.11 I reported the Table Views exposed by GADGET in order to visualize the QBGA Process Dynamic aspects. Fig. 2.11a shows the Transitions-States Table View provided by GADGET. It reports the starting and ending States of the transitions composing the defined Question Lifecycle. As shown in Fig.

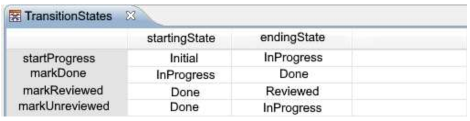
[5] *http://www.eclipse.org/sirius/overview.html*



Property	Value
Capability	
Description	Third Requirement
Improvement Actions	
Name	Req3

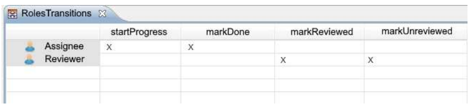
Figure 2.10. GADGET - User Interface for QBGA Process Modeling

2.11b, GADGET reports for each defined transition the roles involved in its execution through the Roles-Transitions Cross Table View .


TransitionStates

	startingState	endingState
startProgress	Initial	InProgress
markDone	InProgress	Done
markReviewed	Done	Reviewed
markUnreviewed	Done	InProgress

(a) GADGET - Transitions - States Table View


RolesTransitions

	startProgress	markDone	markReviewed	markUnreviewed
Assignee	x	x		
Reviewer			x	x

(b) GADGET - Roles-Transitions Cross Table View

Figure 2.11. GADGET - Table Views

GADGET support the fully automatic execution of the transformation process reported in Figure 2.12. This process can be easily triggered by the Questionnaire Developer through the Rich IDE offered by the tool.

The process relies on the execution of two consecutive transformation steps. The former step implements the *Generating the QBGA ALM Model* activity. It performs a *model-to-model* transformation that translates the QBGA Process model toward a generic QBGA ALM Project Model. The latter one is related to the *Implementing the QBGA ALM Project* activity. It is actually a *model-to-text* transformation aimed at translating the QBGA ALM Project Model into a QBGA ALM Project instance that is specific for a given ALM system, i.e., Siemens Polarion ALM, IBM Rational Team Concert, Microsoft Team Foundation[6].

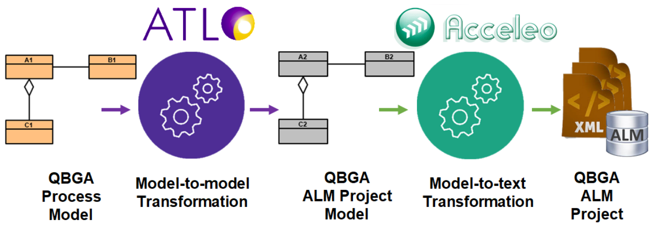


Figure 2.12. Proposed Transformation Process

In order to define the *model-to-model* transformation rules for producing an instance of a QBGA ALM Project Model starting from a QBGA Process Model instance I had to map each element of the QBGA Process metamodel with its corresponding one in the ALM metamodel. Table 2.6 reports the mapping I proposed.

As an example, the Structural Elements and Requirements defined in the modeling phase are mapped in apposite WorkItems in the QBGA ALM Project Model. Moreover, the Relationship identified among elements are translated to specific Link Roles. The Lifecycle is translated into the Workflow of the mapped WorkItem. The defined Employees are mapped to Persons and their Identities in the ALM Project Model. In the ALM Project Model the Roles are specified according to the corresponding ones defined for the QBGA Process Model.

Furthermore, in order to specify the *model-to-text* transformation rules I had to map the elements of the ALM Metamodel with the corresponding ones that are specific of the ALM system adopted in the company. As

[6] *https://www.visualstudio.com/tfs/*

Table 2.6. Mapping between QBGA Process Metamodel and ALM Metamodel

QBGA Process Metamodel	**ALM Metamodel**
Questionnaire	Project
Goal	Custom Field
Structural Element	WorkItem
Requirement	WorkItem
Relationship	LinkRole
Question	WorkItem
Answer	Custom Field
Evidence	WorkItem
ImprovementAction	WorkItem
RequiredApplicability	Custom Field
Level	Enumeration
Employee	Person
Role	Role
Lifecycle	Workflow
State	State
Transition	Transition
Report	Report

an example, I defined specific rules that take into account the specific implementation details of one ALM system, i.e. Polarion ALM.

In order to implement the needed transformation rules, different standard transformation languages are already available, such as eXtensible Stylesheet Language Transformations (XSLT), Query View Transformation (QVT), ATLAS Transformation Language (ATL), and Acceleo. I decided to adopt ATL[7] as language to define the needed model-to-model transformation rules and Acceleo[8] for supporting the model-to-text transformation.

ATL is used in a transformational pattern to transform a source model M_a into a target model M_b. The source and target models and the ATL transformation definition conform to their metamodels MM_a, MM_b, and ATL respectively [94]. These metamodels need to conform with the Meta Object Facilities (MOF) core specification[9] defined by the Object Management Group (OMG).

An example of the ATL transformation rules I defined is reported in Listing 2.1. This rule is applied to transform a *StructuralElement* of a QBGA Process Model to a *WorkItem* of a QBGA ALM Project Model.

I defined in Acceleo the platform-dependent transformation rules. Acceleo is both a model-to-text transformation language and a tool; its main purpose is to implement code generators. It is an implementation of the MOF Model-to-Text Language (MTL) standard [95]. It provides a flexible and simple environment to design and develop a variety of code generators, using simple and standard templates. An Acceleo program requires a metamodel and a model compliant with that metamodel, from which it generates text or source code according to a specified template. An example of the defined Acceleo transformation rules is reported in Listing 2.1. This rule is applied to transform the Workflow of a WorkItem into

[7] *http://www.eclipse.org/atl/*
[8] *http://www.eclipse.org/acceleo/*
[9] *http://www.omg.org/spec/MOF/2.4.2/*

Listing 2.1. Sample ATL Transformation rule

```
-- Transforms a StructuralElement into an Work Item
rule StructuralElem2WorkItem{
  from
    s : Questionnaire!StructuralElement
  to
    w : ALM!WorkItem(
      type <- s.type,
      name <- s.name,
      description <- s.getDescription()
    )
}
```

a Polarion ALM XML file. The generated XML file reports the transitions related to a WorkItem specifying for each one of them its composing States and the Roles that can trigger it.

2.4 Evaluation of the Proposed Solution: a Case Study in the Automotive Domain

I conducted in collaboration with members of my research group a validation experiment to evaluate the feasibility of the proposed approach in real industrial settings. More in detail, we designed and performed an industrial case study in the automotive domain according to the guidelines proposed by Runeson *et al.* [96] for comprehending two different aspects. The first one was the impact of the GADGET tool on the QBGA Process Development activity. The second aspect was the impact of using an ALM-based tool in executing QBGA Processes for carrying out the Questionnaire Completion and Analysis activities. The study was aimed at answering the following research questions:

Listing 2.2. Sample Acceleo Transformation rule

```
[comment encoding = UTF-8 /]
[module generateALMProject('http://it.unina.
   gapanalysisquestionnaire')]
[template public generateWorkflowXML(workItem :
   WorkItem)]
[comment @main/]
[file ('workflow.xml', false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8" standalone="
   no"?>
<workflow initial-action="init" initial-status="
   planned" prototype="WorkItem">
    <transitions>
        [for (t : Transition | workItem.lifecycle)
           ]
                <transition action=[t.name/] from
                   =[t.startingState/] to=[t.
                   endingState/]/>
        [/for]
        </transitions>
        <actions>
        [for (t : Transition | workItem.lifecycle)
           ]
                <action id=[t.name/] />
                <roles>
                [for (r : Role | t.roles)]
                        <role name=[r.name/]/>
                [/for]
                </roles>
        [/for]
      </actions>
[/file]
[/template]
```

RQ$_1$ How does the GADGET tool support the design and development of ALM-based tools for aiding the QBGA Processes?

RQ$_2$ How does ALM-based tools affect the QBGA Processes execution?

2.4.1 Objects of the Study

The *Objects* of the study were two QBGA Processes for assessing the capabilities of two different divisions of FCA Europe, Middle East, and Africa (EMEA) Region department to comply with the ISO 26262 Standard. The two QBGA processes were selected from the ones of interest for the company at the time of the study. The first QBGA process, *P1*, was planned for assessing the gaps of the software process implemented by the *D1* division with respect to the part 6 of the Standard, whereas, the *P2* QBGA process had the goal of assessing the capabilities of the *D2* division to adhere with the part 9 of the same Standard.

The ISO 26262 Standard, entitled *Road vehicles - Functional safety* is a functional safety standard an it is an adaptation of the *IEC 61508* (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems) Standard in the context of road vehicles. The *functional safety* is defined by the standard itself as *the absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems.* In order to guarantee the functional safety, the ISO 26262 imposes the V-model as a reference process model for the different phases of the product development. Moreover, the standard defines four Levels to specify requirements and safety measures, defined as ASIL. The ISO 26262 defines requirements to be applied according to the Level of the ASIL of the product being developed. The Part 6 of the Standard focuses on the product development at software level. The Part 9 of the Standard specifies the requirements for Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses.

2.4.2 Subjects of the Study

The *Subjects* of the study were FCA employees who were asked and accepted to perform the QBGA processes exploiting our approach.

To evaluate the impact of the GADGET tool, we selected two different employees, *QD1* and *QD2*, who were in charge of developing the Gap Analysis questionnaires in past QBGA processes carried out by the company. The selected employees had on average 5 years of experience on Gap Analysis and had been involved in 4 different QBGA processes, on average.

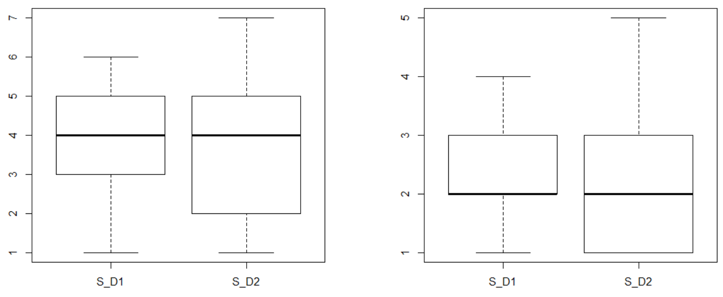
In addition, we recruited other ten employees, five in each selected division, to carry out the QBGA process. These subjects had different levels of expertise, had been involved in at least one past QBGA process and had knowledge about the software processes applied by the division they were employed in. All the subjects were familiar with ALM systems, since they adopt it in their usual industrial practice. Box-plots diagrams representing the level of expertise and the number of QBGA processes performed by the selected employees of the D1 division, *S_D1*, and D2 division, *S_D2*, are reported in Figure 2.13b.

2.4.3 Case Study Design

We decided to perform the case study through a qualitative evaluation exploiting semi-structured interviews with the involved study subjects. We decided to employ a qualitative analysis since using qualitative methods increases the amount of information contained in the data collected and its diversity allowing to increase the confidence in the results through triangulation, multiple analyses, and greater interpretive ability [18].

To carry out the evaluations, we designed two interview guides for answering the proposed RQs.

The first interview guide, structured in three different parts, was crafted



(a) Years of Expertise **(b)** Number of Projects

Figure 2.13. Involved Subjects Details

for the two subjects having assigned the Questionnaire Developer role and it was designed for answering the RQ_1. The guide parts aimed at evaluating the following aspects:

$P_{1.1}$ how the subjects adopted the features provided by GADGET tool for accomplishing their assigned tasks,

$P_{1.2}$ how the tool supported the Questionnaire Development task execution,

$P_{1.3}$ the main limitations of the tool.

In order to address RQ_2, a further interview guide was designed for the remaining ten involved subjects. The parts of this guide were aimed at understanding:

$P_{2.1}$ how the subjects exploited the ALM-based tool during the QBGA process execution,

$P_{2.2}$ how the tool features supported the subjects in their tasks execution,

$P_{2.3}$ how the adoption of the tool impacted on the quality of the QBGA process,

$P_{2.4}$ the main limitations of the tool.

To understand how the adoption of the ALM-based tool impacted the quality of the QBGA process we considered the three quality process attributes of visibility, acceptability and supportability. According to Sommerville [97], the *visibility* of a process represents to what extent the process activities culminate in clear results, so that the progress of the process is externally visible. The *acceptability* of QBGA Processes represents if they are acceptable to and usable by the subjects involved in their execution whereas the process *supportability* describe to what extent software tools are available to be adopted for supporting the process execution. To understand how the introduction of the ALM-based tool influenced the process visibility, we queried the involved subjects which process aspects they were interested in monitoring and if they were able to obtain this information. To assess how our approach had impact on the process *acceptability*, another part of the interview guide was aimed at understanding which process factors influenced the process usability from the involved subjects point of view, and whether these factors were exposed by the proposed ALM-based tool. Regarding the *supportability* of the QBGA process, specific questions were designed to understand what kind of features the different involved subjects required and if they were provided by the ALM-based tool.

2.4.4 Case Study Procedure

The *Procedure* we followed to conduct the case study consisted of seven sequential steps.

At first, we performed a two phases *Training Step* to explain our proposed approach to the subjects. The first phase was devoted to the Questionnaire Developers that were trained on the use of the GADGET tool. In the second phase the remaining subjects were introduced in the use of the ALM-based tool for accomplishing the tasks related to the Questionnaire Completion and Analysis activities. Both the training phases were carried out by one the researchers involved in the development of the approach and each one lasted 8 hours. The researcher explained with detailed examples the different features offered by the GADGET and ALM-based tools.

In the *Questionnaire Development Step*, we assigned the *QD1* and *QD2* subjects with the task of developing the *P1* and *P2* questionnaires, respectively. The two subjects carried out the assigned task exploiting the GADGET tool. At the end of their tasks, they developed and deployed the ALM-based tools configured for supporting the execution of the *P1* and *P2* processes.

In the *Questionnaire Completion and Analysis Step*, the other subjects carried out the gap analysis tasks assigned them exploiting the produced ALM-based tools. More in detail, the subjects belonging to the *D1* division carried out the *P1* process whereas the others the *P2* one.

In the *Data Collection Step* a researcher conducted the interviews with all the study subjects exploiting the designed interview guides.

Then, a *Data Analysis and Interpretation Step* was performed. The answers collected through the interviews were independently analyzed and interpreted by two different researchers in order to assess how the proposed approach had impact on QBGA process executions. In this step, they coded the interviews in order to gain the relevant information and analyzed them.

The analyzed data was submitted to a *Data Validation Step*. In the first phase of this Validation step, the interviews results obtained by each researcher were submitted to the interviewed subjects in order to find possible misinterpretations and solve them. In a second phase, a data triangulation was carried out to compare and validate the results achieved by the two researchers. The validated results were used to gain findings about the performed processes and to answer the RQs.

2.4.5 Case Study Findings

In this section, I report the finding of the study, focusing on the aspects of interest about the Questionnaire development and Questionnaire Completion and Analysis activities.

Questionnaire Development

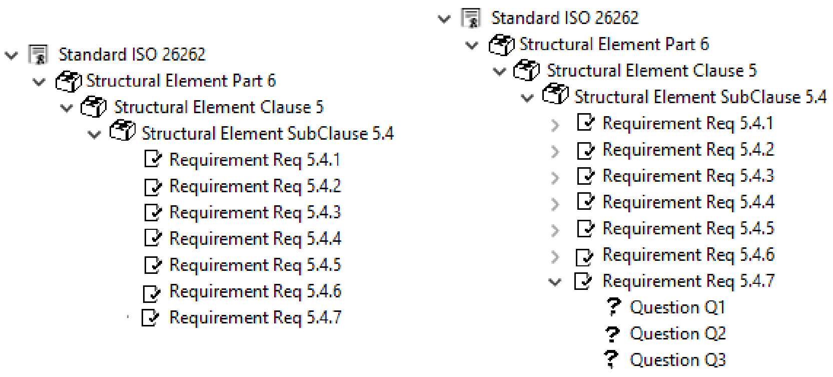
As regards to the Questionnaire Development activity, the involved QD1 and QD2 subjects were able to model the QDBA process related to the Part 6 and Part 9 of the ISO 26262 standard exploiting the features offered by the GADGET tool.

GADGET tool adoption. On the basis of the answers given to the part $P_{1.1}$ of the interview, we were able to understand that the involved subjects successfully adopted the graphic editor offered by GADGET for modeling the considered part of the Standard and the related QBGA process. As an example, Figure 2.14a reports an excerpt of the Structural Elements and the Requirements of the ISO 26262-6 Standard modeled by the *QD1* subject. Figure 2.14b also shows three questions assigned to the 5.4.7 Requirement. As another aspect, both the subjects were able to define the lifecycle of the questionnaire elements, specifying the roles and the rules for carrying out the completion and analysis activities.

Using the transformation features offered by GADGET they were able to produce the ALM-based tool. They manually deployed it on the target ALM system.

Support provided by GADGET. Thanks to the answers given to the Part $P_{1.2}$ we understood that both the subjects considered the GADGET features useful and usable. More in detail,they appreciated the visualization features offered by the tool that simplified the execution of the questionnaire development task. The QD2 subject extensively exploited the Tree view offered by the tool that gave him a clear vision of the defined questionnaire during its modeling. According to this point, he reported that *"... even if the gap analysis for the ISO 26262 required the definition of a lot of questions, I had always a clearer vision of the entire questionnaire that I was developing ... "*. Moreover, the subjects reported that thanks to the consistency checks automatically enforced by GADGET, they were able to identify errors in the model being developed and to correct them. As an example, the *QD1* subject stated that during the task execution he was able to identify some Requirements that were incorrectly not linked to any Structural Element, and Requirements having no linked Questions. The possibility to define through GADGET the lifecycle of the involved elements was considered by both the subjects as crucial for the QBGA process development. They reported that they had no means to define them in past projects due to inadequate tool support. Both the subjects considered straightforward to define these aspects exploiting the contextual menus offered by the GADGET graphical editor. Finally, they appreciated the capability of the GADGET tool of automatically translating the model into the ALM-based tool, avoiding them tedious implementation tasks.

GADGET tool limitations. According to the answers given to $P_{1.3}$, both the subjects reported the lack of tool features for optimizing the model visualization as its size grows. According to the QD2 subject, features for grouping and ungrouping the elements of the QBGA process would have made the model more readable, easing its navigation. One of the subject suggested to add a feature for automatically deploying the produced ALM-based tool on a target ALM system, avoiding the manual download and installation of the produced tool on the selected ALM



- Standard ISO 26262
  - Structural Element Part 6
    - Structural Element Clause 5
      - Structural Element SubClause 5.4
        - Requirement Req 5.4.1
        - Requirement Req 5.4.2
        - Requirement Req 5.4.3
        - Requirement Req 5.4.4
        - Requirement Req 5.4.5
        - Requirement Req 5.4.6
        - Requirement Req 5.4.7

- Standard ISO 26262
  - Structural Element Part 6
    - Structural Element Clause 5
      - Structural Element SubClause 5.4
        - Requirement Req 5.4.1
        - Requirement Req 5.4.2
        - Requirement Req 5.4.3
        - Requirement Req 5.4.4
        - Requirement Req 5.4.5
        - Requirement Req 5.4.6
        - Requirement Req 5.4.7
          - Question Q1
          - Question Q2
          - Question Q3

(a) ISO 26262 - Excerpt Part 6 Sub-Clause 5.4

(b) ISO 26262 Questionnaire - Excerpt Requirement 5.4.7

Figure 2.14. Excerpt ISO 26262 QBGA Process Model

system.

Questionnaire Completion and Analysis

Regarding the Questionnaire Completion and Analysis activities, we were able to understand the impacts of the ALM-based tool on their execution, on the considered process attributes, as well as its main limitations.

ALM-based tool adoption. The answers to the part $P_{2.1}$ of the interview allowed us to understand that the subjects involved in the Questionnaire completion and analysis activities were assigned with specific roles, according to the ones defined in the QBGA process models. Overall, for both the processes, one Responsible, three Respondents and one Internal Assessor roles were assigned to different subjects. To carry out their tasks they exploited the ALM-based tool features exposed to their roles.

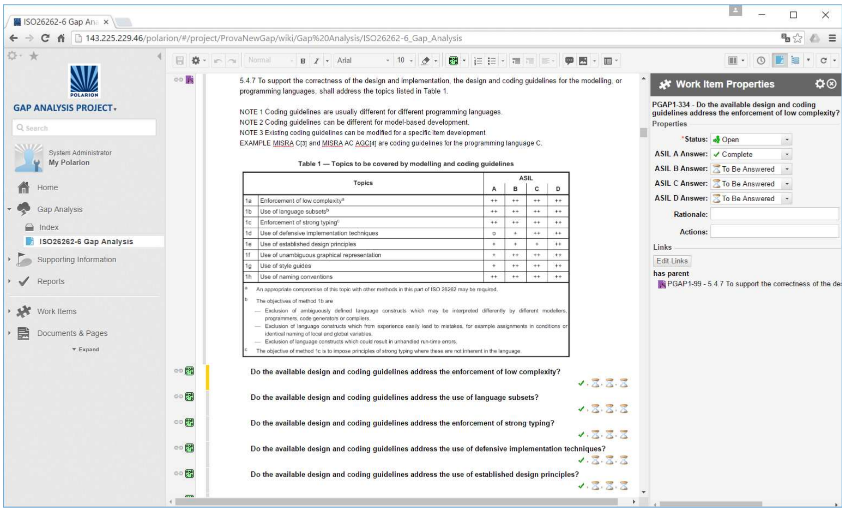
Support provided by the ALM-based tool. According to the answers given to the part $P_{2.2}$ of the interview, all the subjects were able to successfully carry out their tasks exploiting the features offered by the ALM-based tool. For nine out of ten subjects the tool simplified their work. The Responsible subjects were able to assign the defined roles to the involved subjects and were able to monitor the progress of the process execution through the ALM features. The Respondents appreciated the possibility to easily take track of their assigned questions and the ones for accessing the information related to questions they were analyzing. The Internal Assessor exploited the gathering and report features offered by the tool for carrying out their assigned tasks. One of the Internal Assessor reported that the tool eased its work thanks to the aggregated views it provided that gave him focused views on the elements he was assessing.

Impact on QBGA process attributes. The answers to the part $P_{2.3}$ of the interview gave us the evidence to state that the introduction in the QBGA process of the ALM-based tool, positively affected the considered process attributes. As for the *process visibility*, different roles were interested in different aspects of the QBGA process. The Responsible wanted to evaluate the overall progress of the questionnaire completion. More in detail, he had to track the progresses in the questions answering and rating. Almost all the respondents were mainly interested in understanding the advancement of their own assigned questions. As for the internal assessors, their main focus was the progress of the questions related to the elements they had to assess as well as their own assigned rating elements. All the involved subjects, regardless of their role, reported that the information they needed was always accessible through the ALM-based tool. One of the subjects stated that *" ... it was easy for me to have a clear idea of the part of the process I was interested about. With a simple click I was able to see how many of the questions assigned to me were In Progress, Done. In this way, I could always be aware of how many and what other questions had yet to be answered and I could better organize my work ... "*.

With respect to the *process acceptability*, the different roles involved in the process reported similar usability requirements. Specifically, almost all the subjects claimed the necessity of an integrated working environment where they could directly access to the parts of the questionnaire they were involved in. At the same time, they expected features for directly referring the documentation of the considered Standard.

Moreover, the capability for interacting with the other actors involved in the process was another factor affecting the process acceptability for seven out of ten subjects; on the other hand, the need to switch between different artifacts limited the acceptability of the process for nine out of ten subjects . According to the provided answers, we were able to understand that the usability of the process was positively affected by the introduction of the ALM-based tool that provided for nine out of ten subjects an adequate support. According to their impressions, they could easily carry out all their assigned tasks through a unique tool where they find all the information needed to answer the questions or conduct the analysis of the gaps. Moreover, the tool enabled and simplified the collaboration among the involved actors. Only one of the actors considered that the acceptability of the process was not influenced by the adoption of the ALM-based tool, since he already considered the usual QBGA process acceptable.

Considering the *process supportability*, according to the interviews answers, all the involved subjects considered that had appropriate tool support for performing each of the activities of the QBGA process they were involved in. As for the Responsible, the ALM-based tool provided specific features for giving roles to the selected subjects and assign them questions answering or rating activities. From the point of view of the Respondents, the tool offered specific views aggregating the question assigned to each subject and facilities for answering the questions. As an example, Figure 2.15 shows an excerpt of the questionnaire exposed by the ALM-based tool. Its central frame lists all the questions. By selecting one of the questions, the assigned Respondent was able to answer it or triggering one of its lifecycle transition through the panel located on the upper right side. Moreover, through traceability features, accessible through the bottom right panel, the Respondents were able to easily access the question referenced parts of the Standard.



5.4.7 To support the correctness of the design and implementation, the design and coding guidelines for the modelling, or programming languages, shall address the topics listed in Table 1.

NOTE 1 Coding guidelines are usually different for different programming languages.
NOTE 2 Coding guidelines can be different for model-based development.
NOTE 3 Existing coding guidelines can be modified for a specific item development.
EXAMPLE MISRA C[3] and MISRA AC AGC[4] are coding guidelines for the programming language C.

Table 1 — Topics to be covered by modelling and coding guidelines

Topics		ASIL			
		A	B	C	D
1a	Enforcement of low complexity[a]	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++

[a] An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.
[b] The objectives of method 1b are
— Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.
— Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
— Exclusion of language constructs which could result in unhandled run-time errors.
[c] The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

Do the available design and coding guidelines address the enforcement of low complexity?
✓, ⌛, ⌛, ⌛

Do the available design and coding guidelines address the use of language subsets?
✓, ⌛, ⌛, ⌛

Do the available design and coding guidelines address the enforcement of strong typing?
✓, ⌛, ⌛, ⌛

Do the available design and coding guidelines address the use of defensive implementation techniques?
✓, ⌛, ⌛, ⌛

Do the available design and coding guidelines address the use of established design principles?
✓, ⌛, ⌛, ⌛

PGAP1-334 - Do the available design and coding guidelines address the enforcement of low complexity?

Properties

*Status:	Open
ASIL A Answer:	✓ Complete
ASIL B Answer:	⌛ To Be Answered
ASIL C Answer:	⌛ To Be Answered
ASIL D Answer:	⌛ To Be Answered
Rationale:	
Actions:	

Links

has parent
PGAP1-99 - 5.4.7 To support the correctness of the de

Figure 2.15. ISO 26262-6 QBGA Polarion ALM Project

All the subjects assigned with the Internal Assessor role were able to evaluate the gaps through the gathering and visualization features offered by the ALM-based tool. The Report Views exposed by the ALM-based tool was used to obtain the aggregated data they needed to carry out their tasks, simplifying their activities. Example of a Report View they exploited is reported in Figure 2.16. It aggregates the Question Answers and the Compliance rating of the Structural Elements of the Standard.

ALM-based tool limitations Some limitation about the ALM-based tool adoption emerged. from the answers of the interview $P_{2.4}$ part. One of the involved subjects having assigned the Respondent role reported that even if the tool provided valuable features, it required numerous more interactions. He was more accustomed to the use of spreadsheet systems for

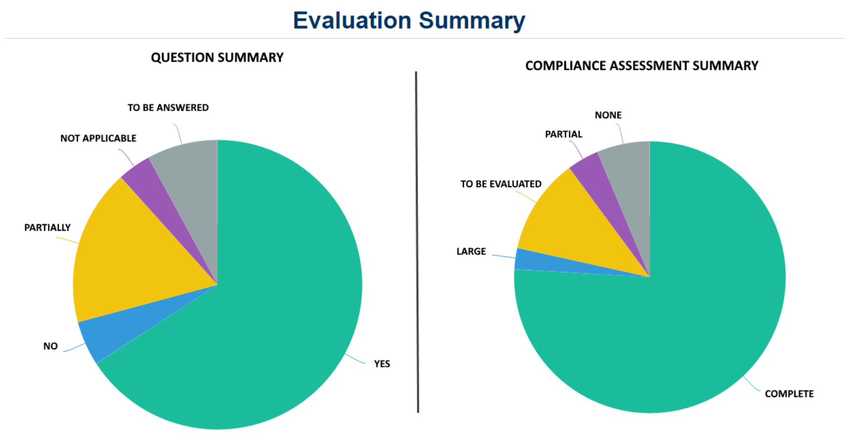


Figure 2.16. Evaluation Summary Report Page

carrying out his task and found the new features offered by the ALM-based tool not so easy to be adopted. As another factor, he would have appreciated a documentation of the ALM-based tool for better understanding the QBGA process workflow and what elements and features he needed to exploit.

2.4.6 Case Study Results

In the following I report the answers to the defined RQs according to the reported findings.

RQ_1: How does the GADGET tool support the design and development of ALM-based tools for aiding QBGA Processes?

By analyzing the answers given by the Questionnaire Developers we were able to gain evidences about the support given by the GADGET Tool for the Questionnaire Development phase. The results showed that the GADGET tool well supported the Questionnaire Developers in performing their design and development tasks. The adoption of the tool simplified the execution of the design tasks through the modeling and visualization features. Moreover, its adoption allowed a more accurate design of the QBGA process, allowing to easily define the lifecycle of its composing elements. As another aspect, the tool eased the implementation of the ALM-based tool for supporting the QBGA process execution, avoiding the Questionnaire Developers to focus on implementation details.

RQ_2: How does ALM-based tools affect the QBGA Processes execution?

Thanks to the reported findings we were able to state that the introduction of the ALM-based tool positively affected the QBGA process execution. In particular, the features offered by the ALM-based tool eased the execution of the tasks of the subjects involved in the process with different roles. Moreover, the adoption of the ALM-based tool made the QBGA process visible for all the subjects regardless of their role. As another aspect, the tool met the usability requirement of almost all the subjects, which made the process more acceptable for them. According to the judgments of the involved subjects, all the tasks foreseen by the QBGA process were well supported by the produced ALM-based tool.

2.4.7 Threats to Validity

Internal Validity

This aspect of validity needs to be evaluated when causal relationships are examined. It defines how sure I can be that the treatment actually caused the outcome. In our case, also the subjects experience could be another factor influencing the outcomes of the study. To mitigate this threat, we selected, according to their availability, subjects with different levels of expertise and that have been involved in a different number of QBGA processes in the past. To further mitigate this threat, I plan to execute a wider experimentation involving a large sample of subjects having different levels of expertise on QBGA processes.

External Validity

External validity is related to what extent it is possible to generalize the findings and if these are of interest to people outside the reported case. A possible threat may be related to the application of the approach in a specific industrial domain and for a specific standard. In order to minimize this threat, a more extensive experimentation in different industrial executing several QBGA processes with respect to different standards will be performed.

Reliability

The reliability is concerned with to what extent the data and analysis are dependent on the specific researchers having carried out the study. Hypothetically, if another researcher later on conducts the same study, the result should be the same. In order to mitigate this threat, we defined and conducted a data validation steps aimed at avoiding possible misinterpretations. Moreover, we reported in this Chapter the experimental procedure we followed for allowing replications of the case study.

2.5 Related Work

The use of Gap Analysis, as a technique to identify discrepancies with respect to the achieving of an objective, can be tracked back in 1980s when Parasuraman et al. [70] developed a service quality model based on analysis of the gap between customers' expectations and their perceptions of the service. Since then, the use of gap analysis spread in many different domains, especially business process management [98] and supply chain management [99].

Gap analysis has been used in several other contexts. In Service Oriented Architectures gap analysis has been exploited to determine how available software services can be traced to new business models that best meet organization goals [100, 101]. Moreover, it has been adopted in frameworks for the evolution, modernization and management of legacy software systems [102, 103].

Also in *security* domain, gap analysis techniques and methods are proposed to evaluate the compliance of companies with information security standards. In [104], a gap analysis methodology was proposed with the aim of evaluating the compliance with the ISO 17799 Standard. This methodology relied on the use of surveys. In this work, a case study in a governmental organization that showed the accuracy and efficiency of the proposed methodology was also presented. Another approach based on gap analysis was proposed for assessing the compliance of information systems in the context of SMEs with respect to the ISO 27001 standard [105]. Their proposed approach was supported by an appositely developed tool. An experimentation was conducted in different SMEs to show that the tool helped in reducing the time needed to perform the gap analysis. Moreover, Al-Mayahi and Monsoor reported in [80] results of four case studies related to the execution of gap analysis performed to determine the compliance of different organizations within the UAE e-goverment with the requirements of the ISO 27001 Standard. Unlike ours that is intended to be applicable to any Standard, all these approaches are strictly tied to a specific Standard.

In the context of Knowledge Management, gap analysis is used as a tool to evaluate enterprise knowledge management performance [106].

In the context of Software Processes, Gap analysis is used as a tool for supporting organizations in assessing their processes, in seeking process improvement or in planning compliance with a certain Standard [75].

Amaral et al. proposed a gap analysis methodology for assessing an organization against the practices requested by the Team Software Process (TSP) that was applied in one Portuguese organization [107]. This methodology was specific for the TSP process and was based on elements of the *SCAMPI* and *ISO/IEC 15504* assessment methodologies. Ceccarelli and Silva proposed in [69] a framework for gap analysis to measure the compliance of company's practices, knowledge and skills against the requirements of a standard for the development of safety-critical systems. They exploited the proposed framework to rate a company maturity in the usage of the avionic standard DO-178B. In the last years several MDE approaches have been proposed in the literature for supporting process assessment tasks, and many of them targeted safety Standards. Falessi et al. [108] introduced a model-based and tool-supported approach to assist suppliers and certifiers in developing formal agreement to demonstrate compliance with a safety standard. The approach was not formally evaluated, but it is considered a more effective way to do what it has already been done manually. In [74] an approach based on model-driven engineering to analyze the evidence necessary to demonstrate compliance with a safety standard was also proposed. They proposed an approach and web-based supporting tool to systematically negotiate a consistent agreement between suppliers and certifiers about the information to collect for safety certification. The tool relies on an information model that captures core concepts and their relations of a given safety standard. The approach was evaluated through a case study of a sub-sea production control system and a survey of industry practitioners. Our work differs from these ones since it does not focus on the agreement between certifiers and suppliers but it aims at supporting internal self-assessment processes regardless of the considered Standard.

2.6 Conclusions and Future Work

Gap Analysis is a widespread technique used in industry to assess the implemented development processes with respect to the requirements prescribed in Process Quality Frameworks or Standards. Thanks to an industrial survey we conducted in several industrial settings, we became aware of the main issues and limitations affecting gap analysis processes carried out in the practice.

In order to address the emerged shortcomings, in this chapter I proposed a tool-supported approach for aiding the execution of QBGA processes. The approach exploited ALM, a technology widely adopted in the practice for supporting the execution of software development processes. ALM needs to be properly configured for introducing it in a specific context for enacting a given process.

To guarantee that the ALM can be easily configured and adapted for supporting different QBGA processes, I developed the GADGET tool that provides features for modeling QBGA processes and for automatically translating such models towards ALM systems, according to the MDE paradigm.

The feasibility of the proposed approach was evaluated with an industrial case study carried out in the automotive domain. In this context, the approach was adopted to support two QBGA processes related to different Parts of the ISO 26262 Functional Safety Standard. The study showed that the execution of the QBGA process can be positively impacted by the proposed approach. In particular, the GADGET tool aided the introduction of the approach simplifying the design and the development of the ALM tool. Moreover, the adoption of the latter tool improved the visibility, acceptability and supportability of the QBGA processes. These preliminary results suggest that the approach may be successfully adopted in industrial practices.

As future work, I plan to extend the validation of the approach by case studies that will involve more companies and QBGA processes targeting different Standards. Moreover, we intend to extend the features offered by the GADGET and ALM-based tools according to the comments collected during the case study execution.

This page intentionally left blank.

Chapter 3

Improving Traceability Management in Software Processes through Tool Integration

In this Chapter I present a software architecture I designed for integrating ALM platforms with the tools used to carry out the testing process. The architecture aimed at fully automating the execution of the process and at automatically generating the appropriate traceability links when they are established. It exploits the features offered by Continuous Integration Engines that allowed the development of a modular, evolvable and reconfigurable integration architecture. The new architecture was validated by an experiment performed in the automotive domain that showed its capability in correctly and completely generating and handling traceability links between artifacts involved in the testing process. The experiment demonstrated that the integration solution produced also beneficial effects on other quality attributes of the process.

3.1 Introduction

In 1994 Gotel et al. [40] defined the Requirement traceability as *the ability to describe and follow the life of a requirement, in both a forwards and backwards direction.* Intuitively, this definition is realized in practice by establishing relationships, called trace links, between the requirements and one or more artifacts of the system [109].

During the years this definition was extended and nowadays more in general I talk about Software traceability for indicating the creation and the use of links (or connections) between different kinds of software artifacts such as requirements, models, source code, test cases, or test results. These connections are called *trace links* and connect in a bidirectional way a source artifact to a target artifact. Many authors addressed the topic of the *traceability management* that is the planning, the organization, and the coordination of all the activities related to traceability, including the creation, maintenance, and use of trace links [110].

Traceability is required by different software development standards (SPICE, CMMI, ISO 26262, etc.) and its importance is well-recognized by software engineering community [111]. Traceability leads to improvements in software systems by supporting tasks related to maintenance, evolution, reuse and more.

In practice, it is very difficult to guarantee an effective traceability management since software projects are often developed by distributed teams, both software artifacts and trace links undergo constant change, and multiple stakeholders with different backgrounds are involved [110]. Moreover, traceability support in contemporary software engineering tools is not satisfactory [112]. Nowadays, an effective solution for managing the traceability between software artifacts is offered by ALM platforms. An ALM platform consists of a set of tools, technologies, or techniques that attempt to provide support for monitoring, controlling and managing software development over the whole application lifecycle [113]. At the same time ALM platforms offer a unified storage and management of every software artifact that is simply stored in version control repositories. They may also provide the automatic implementation and management of traceability relations among artifacts that are guaranteed via automatic change control of every requirement.

ALM tools certainly offer a valuable support to the software lifecycle. Unfortunately, the existing implementations of ALM do not provide tool support for every lifecycle process, neither the existing ALM are able to automatically integrate with any legacy tool and the artifacts it may produce [114]. As a consequence, in many organizations ALM are just used for storing the traceability links between artifacts, while traceability generation and update are manually performed by the personnel involved in the software lifecycle. This habit causes a great waste of time and resources and may yield many inconsistencies, errors, and problems with the manually produced links.

To overcome these limitations, more and more organizations are forced to address the problems of integrating the adopted ALM platform with their legacy software tools and of implementing the automatic generation and update of the traceability links.

A feasible solution to these problems requires the creation of toolchains that implement pipelines of existing tools. Several approaches have been proposed in the literature to implement toolchains, such as [115] but they are still not mature enough for being adopted.

In the practice, the organizations often implement ad-hoc solutions, based on glue code, wrappers, adapters, and other software integration mechanisms, which often result in scarcely maintainable and efficient systems.

In this Chapter I present an industrial experience I performed in collaboration with the FCA company within the research project APPS4SAFETY.

The main aim of the APPS4SAFETY research project was to improve the traceability management in a software testing process performed in FCA.

In this project, I designed a software architecture for integrating the existing ALM platform with the tools used in the testing process. This solution aimed at fully automating the execution of the testing process and at generating the appropriate traceability links when they are established (in-situ creation). These links had to be automatically stored in the ALM used in the company.

To implement this toolchain, I exploited a Continuous Integration Engine that allowed me to develop a modular, evolvable and reconfigurable integration architecture, satisfying the design principles of low coupling and location transparency.

In order to validate the proposed solution, I conducted in collaboration with my research group, an experiment in the industrial organization. The experiment involved real software testing projects conducted in the company and the personnel involved in it. The experimental results showed the capability of the proposed solution of correctly and completely handling traceability links between software artifacts involved in a testing process. Moreover, it demonstrated that the integration solution produced beneficial effects on several quality attributes of the process.

The remainder of the Chapter is organized as follows. Section 3.2 presents the software testing process where I addressed the traceability management and the integration problems. Section 3.3 illustrates the software architecture of the proposed integration solution. Section 3.4 presents the validation experiment. Section 3.5 provides background information about ALM and solutions for software toolchain implementation. Section 3.6 finally reports conclusive remarks.

3.2 The Addressed Problem

In this Section I present the tool integration problem I addressed in the context of embedded software development process. More precisely, I focused on the Model-In-the-Loop (MIL) testing process performed by FCA EMEA SWF, the unit responsible for the development of embedded software for two Electronic Control Units of the vehicle, i.e., the Instrument Panel Cluster (IPC) and the Body Computer Module (BCM).

In the last years, the automotive field has been interested by the introduction of Model Based (MB) [116] approaches and technologies in the software development processes.

MB technologies yield many advantages to the software processes, which become mostly focused on producing high-level models of the SoftWare Component (SWC) that can be used for simulation in very early stages of the development process. These technologies enable the engineers to test the SWCs models in a virtual environment at a stage of the process where they are inexpensive to be fixed, i.e. before the code is actually implemented or integrated on the final hardware, i.e. ECU [117].

This first stage of testing is called MIL and consists of a testing activity where the model and its environment are simulated (interpreted) in the modeling framework (usually MATLAB/Simulink Stateflow) without any physical hardware component [117].

The MIL testing process executed in FCA requires several steps and involves six different testing artifacts having heterogeneous formats, i.e., Microsoft Word Files, Microsoft Excel Files, and MATLAB files. The process exploits MATLAB as the virtual environment where the models of the SWCs are tested and involves the use of an ALM platform. The ALM stores all the artifacts necessary for and produced by the testing process, manages their lifecycle and the traceability relationships between artifacts. Moreover, it handles the roles of users involved in the process

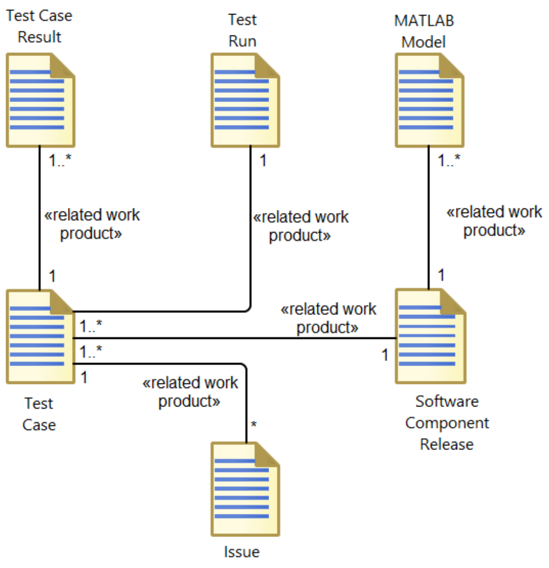


Figure 3.1. Artifacts Relationships in SPEM

and regulates how they can interact with the artifacts.

Figure 3.1 reports the model of the artifacts (i.e. documents or work products) stored in Polarion ALM and the ≪*related work product*≫ relationships that may exist among them. Instances of these relationships are actually the traceability links that should be stored in the ALM.

As figure shows, a *Test Run* is a composition of one or more *Test Case* documents. Each *Test Case* relates to the *Software Component Release* work product to be tested. Any *Software Component Release* is linked to the *MATLAB Model* artifacts simulating its behavior. A *Test Case* document has its own *Test Case Result* work products. *Issue* work products are associated to the *Test Case* founding them.

The considered MIL testing process is described in detail by the SPEM [1] model reported in Figure 3.2.

[1] *http://www.omg.org/spec/SPEM/*

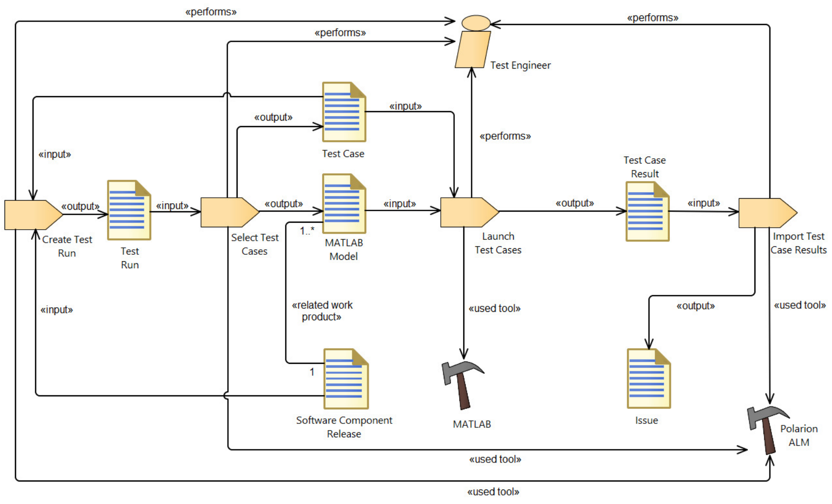


Figure 3.2. The MIL Testing Process in SPEM

The *Test Engineer* manually ≪*performs*≫ the four activities required for the process completion. Polarion ALM and MATLAB are the ≪*used tool*≫ exploited for the accomplishment of these activities.

The MIL process starts with the *Create Test Run* activity. Here the *Test Engineer* exploits Polarion ALM for creating a new Test Run. The Test Cases needed for simulating a given Software Component Release are linked to the newly created Test Run and these links are stored into the ALM.

After, the *Select Test Cases* activity is performed. In this activity, the *Test Engineer* exploits the ALM for selecting the actual Test Case artifacts and choosing the MATLAB Models implementing the behavior of the SWC Release under test.

These Test Cases are executed for simulating the MATLAB Models in the *Launch Test Cases* activity, where the *Test Engineer* has to configure, manually, the MATLAB testing environment before starting the Test Cases execution. At the end of this activity a document containing the

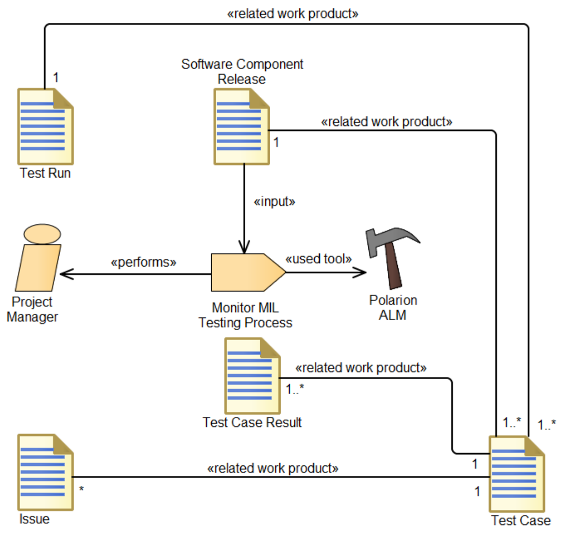


Figure 3.3. The MIL Monitoring Process in SPEM

results of the Test Cases, i.e. the Test Case Results, is produced.

The last activity of the MIL testing process is the *Import Test Case Results*. Here the *Test Engineer* manually stores, into the ALM, all the results of the test case executions, along with the traceability links between Test Cases and Test Case Results. Polarion ALM is configured to automatically produce an Issue work item for each failed Test Case, i.e., a Test Case that found at least a failure on a MATLAB Model. Polarion ALM automatically links Test Cases and Issues.

The MIL Testing process is monitored by the Project Manager, using the ALM tool. This monitoring process is performed according to the SPEM model reported in Figure 3.3. The Project Manager accesses the information about a SWC Release navigating through its related artifacts. In this way, he is able to evaluate the process progress.

As the above descriptions highlighted, many tasks of the MIL process were manually executed by the Test Engineer, essentially due to the lack of integration between the ALM and the remaining testing environment. This caused many obvious disadvantages for the overall process. I was able to identify its most important drawbacks:

- The *slowness* of the MIL process, due to the many manual and time-consuming activities (of interacting with the ALM repository) performed by the Test Engineer.
- The *ineffectiveness* of the traceability links management that are manually introduced into the ALM by the Test Engineers. The responsibility for the completeness and the correctness of these links was left to the Test Engineers. A tracebility link could be incorrect when it erroneously related two artifacts. Traceability is incomplete when there is a missing link between two artifacts that need to be related.
- The *scarce visibility* of the process from the point of view of the Project Manager. This issue derived from the improper habit of the Test Engineers of failing or delaying the traceability links updates in the ALM repository. As a consequence, the current and actual advancement state of the process could not be instantly visible to the Project Manager.
- The process was *scarcely accepted* by its stakeholders. The Test Engineers bothered that they had to spend a lot of manual effort for accomplishing their tasks.

3.3 The Proposed Solution

To overcome the limitations of the current MIL testing process, it was necessary to find a solution for reducing the manual intervention of the Test Engineer in the process. To this aim, I decided to improve the integration between the tools used in the process and to automate the most tedious activities conducted by the Test Engineer, e.g., *Select Test Cases*, *Launch Test Cases* and *Import Test Results*.

More in detail, the new testing process had to be executed carrying out the following activities:

1. the Test Engineer will create the Test Run,
2. the Test Engineer will exploit a single point of access provided by the Graphical User Interface (GUI) of Polarion ALM, where it:
    - configure the MATLAB testing environment,
    - launches the execution of the Test Cases composing the Test Run.
3. the test case results will be automatically imported into Polarion ALM at the end of the test cases execution,
4. the traceability links between test cases, test case results and issues will be automatically created and stored into Polarion ALM.

To satisfy these requirements I had to find a solution for integrating the ALM tool with the testing environment. According to Wasserman [118] and Thomas [119], tools integration can be achieved at five different levels:

- *presentation integration:* this integration solution provides a similar look to the tools that have to be integrated or a single point of access to all tools, to deliver a better user experience;
- *data integration:* such a solution manages data sharing between tools and format and semantic transformations to make data available to different tools;

- *control integration:* it makes available the functions offered by a tool to other tools in the environment;
- *process integration:* it provides process management tools with data from development tools;
- *platform integration:* it provides network and operating systems transparency for tools based on heterogeneous hardware.

My solution should provide four of these levels of integration. The presentation integration was necessary because the Test Engineer had to interact only with the GUI provided by the ALM. The data integration was needed to guarantee the automatic import of the test results from the MATLAB testing tool into the ALM and the automatic creation of the traceability links. The control integration was required to allow the automatic cooperation of the two tools.

Finally, the solution had to guarantee the aspect of process integration since the data of the process management tool, i.e., Polarion ALM, had to be automatically obtained from the development tool, i.e., MATLAB testing tool.

I did not have to satisfy the platform integration aspect since the involved tools did not run on heterogeneous hardware nodes.

3.3.1 Architectural Design

For developing the integration architecture, I had to choose a strategy able not only to interconnect the ALM to the testing tool, but also of automatically coordinating the execution of the process activities by the supporting tools.

A first option consisted in exploiting the mechanism offered by Polarion ALM for interconnecting it with external tools and for extending the ALM platform default features. This mechanism consists in adding new plugins to the ones already present in the standard distribution of the tool. Polarion ALM plugins are actually Java projects exploiting the Application Programming Interface (API)s[2] provided by Polarion ALM. Through these APIs is it possible to programmatically execute all the features of Polarion ALM, such as selecting an artifact, adding a new artifact, adding a new traceability link between two artifacts, etc. Moreover, using Java commands for launching executable files, these plugins have the capability of controlling third party tools.

However, I did not decide to follow this strategy that implements in the ALM all the business logic of the process and couples the ALM directly to the external tools. Vice versa, I decided to introduce an intermediate *Coordinator Component* between ALM and more testing tool and to embed all the business process logic into this component. Figure 3.4 describes the proposed integration solution. The coordinator component will exploit specific connectors for interfacing the ALM with one or more MATLAB testing environments, respectively.

Such an alternative solution is more flexible and less invasive since it does not require any modification neither in the ALM nor in the testing tool. This architecture can be easily extended or adapted when new tools have to be integrated with the ALM platform or the process workflow needs to change.

3.3.2 Implementing Components and Connectors

The Coordinator Component has the responsibility for the automatic execution of the two tasks of *Launch Test Cases* and *Import Test Case Results*. Basically, it acts as a broker between Polarion ALM and MATLAB testing environment and vice versa.

Thanks to this component, Polarion ALM launches automatically the selected Test Cases in the MATLAB testing environment. Then, the Test

[2] *http://almdemo.polarion.com/polarion/sdk/doc/javadoc/index.html*

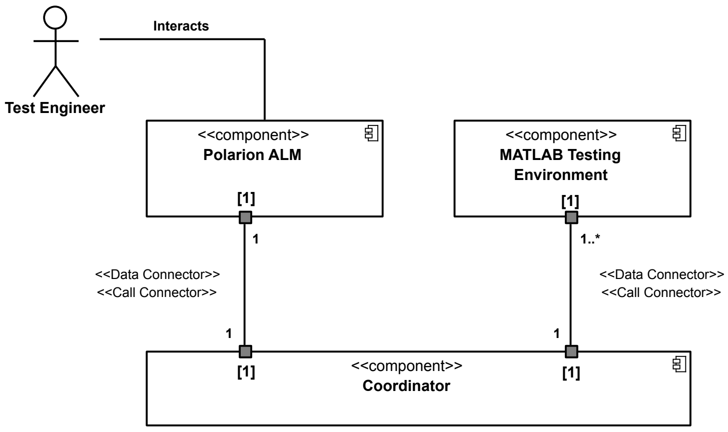


Figure 3.4. Proposed Tool Integration Architecture

Case Results are automatically imported into the ALM as soon as the test cases execution terminates.

I decided to implement this component by exploiting the features provided by the Jenkins automation engine[3]. Jenkins is a valid support for building pipelines involving the execution of heterogeneous third party tools in scenarios of continuous integration, automated testing, or continuous delivery. Polarion ALM itself suggests to exploit Jenkins for building automated testing toolchains, to this aim Polarion ALM provides a suitable connector, named *PJ Connector*[4], allowing the automatic interaction of the ALM with Jenkins.

Figure 3.5 shows details about the design of the overall architecture. The *Coordinator* component has been implemented as a Jenkins based Master/Slave architecture. The Jenkins Master component interacts with the *Polarion ALM* component by means of two connectors [120]. The *Execute Test Run* connector is a call connector exposed by the Jenkins Master component and was realized exploiting the Jenkins Remote access

[3] *https://jenkins.io/*
[4] *http://www.emerasoft.com/pj-polarionjenkins-connector/*

API[5]. This connector is exploited by the ALM for launching a Test Run. The Jenkins Master requires the data access connector *Export Test Run* provided by Polarion ALM for querying the Test Cases belonging to the Test Run to be launched.

The Jenkins Master delegates the execution of all the Test Cases belonging to a Test Run to one of the Jenkins Slaves by exploiting the *Dispatch* Jenkins connector. Each slave component is responsible for interacting with just one testing environment. This is guaranteed by the two connectors exposed by each testing environment. The *Execute Test Cases* call connector allows the Jenkins slave to launch one or more test cases on the MATLAB platform. The *Read Test Results* data access connector, is used by the slave to load, from the testing environment, the results of the tests execution. The slave exploits the two data connectors *Update Test Runs* and *Update Test Case* provided by the ALM for updating the status of both the Test Runs and the Test Cases.

All the connectors exposed by Polarion ALM are data access connectors and were developed exploiting the APIs exposed by Polarion ALM itself[6] to access its stored elements.

On the other side, the MATLAB testing environment was a legacy tool that did not expose any APIs exploitable to control its execution nor to access its produced Test Case Results. As a consequence, in order to define the Execute Test Cases call connector and the Read Test Result data connector I had to implement a solution for wrapping the legacy tool. To this aim, I carried out a reverse engineering process [121, 122] to understand the features needed to allow and configure the tool control and to access its produced results.

[5] *https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API*
[6] *http://almdemo.polarion.com/polarion/sdk/index.html*

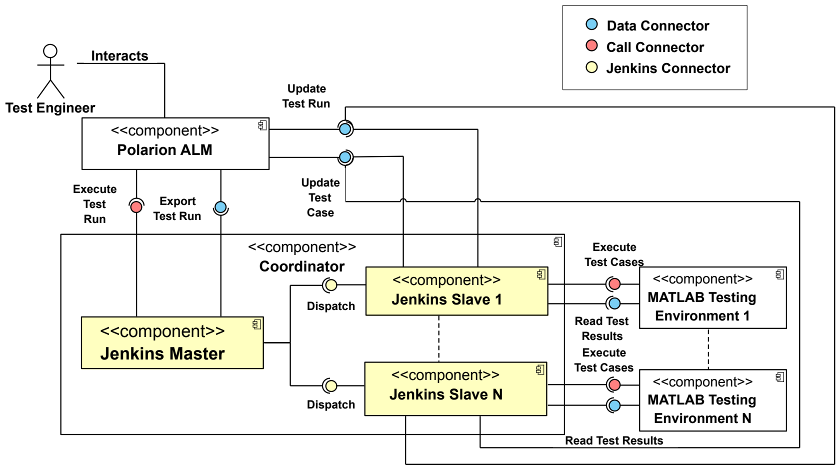

Data Connector
Call Connector
Jenkins Connector
Test Engineer
Interacts
<<component>> Polarion ALM
Update Test Run
Update Test Case
Execute Test Run
Export Test Run
<<component>> Coordinator
<<component>> Jenkins Master
<<component>> Jenkins Slave 1
<<component>> Jenkins Slave N
Dispatch
Dispatch
Execute Test Cases
Read Test Results
Execute Test Cases
Read Test Results
<<component>> MATLAB Testing Environment 1
<<component>> MATLAB Testing Environment N

Figure 3.5. Detailed Component Diagram of the proposed architecture

3.3.3 Extending the ALM GUI

In order to overcome the limitations of the previous process, where the Test Engineer had to interact with the GUIs of two different environments, I implemented a new GUI in the ALM offering a single point of access for the execution of the MIL testing process.

To this aim, I exploited the Wiki-based mechanism provided by Polarion ALM[7] to extend its default Test Run GUI, used in the original process for defining the Test Run and for manually importing the Test Case Results.

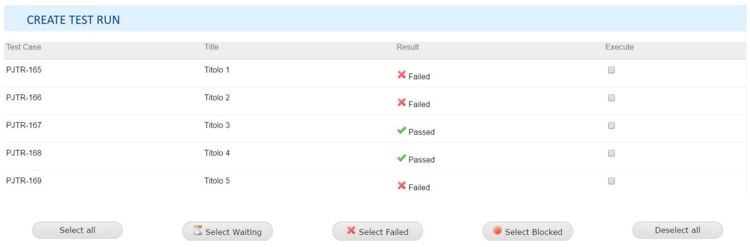
I enriched this GUI by two additional interaction panels.

The *"Create Test Run"* panel is reported in Figure 3.6a. It shows the Test Cases available for a given SWC Release and allows the Test Engineer to select the Test Cases to be run. Moreover, the panel shows the status (i.e., Failed, Passed, Blocked, Waiting) of all these Test Cases.

The *"Test Run Execution"*, shown in Figure 3.6b, is structured in sev-

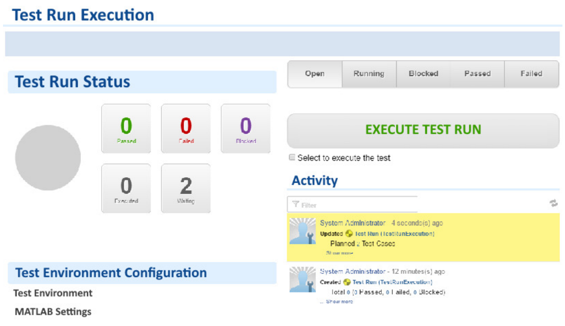
[7] *http://almdemo.polarion.com/polarion/help/?topic=/com.polarion.xray.doc.user/ugchAppendix.html*

eral sections. The testing environment can be configured through the "Test Environment Configuration" section and the Test Run execution can be launched by the "Execute Test Run" button. In addition, its "Test Run Status" section offers an overview about the status of all the launched Test Cases and the "Activity" section reports the last activities performed by the users on the current Test Run.


CREATE TEST RUN

Test Case	Title	Result	Execute
PJTR-165	Titolo 1	✗ Failed	☐
PJTR-166	Titolo 2	✗ Failed	☐
PJTR-167	Titolo 3	✓ Passed	☐
PJTR-168	Titolo 4	✓ Passed	☐
PJTR-169	Titolo 5	✗ Failed	☐

(a) Create Test Run Panel



☐ Select to execute the test

System Administrator 4 seconds(s) ago
Updated Test Run (TestRunExecution)
Planned 2 Test Cases
Show more

System Administrator - 12 minutes(s) ago
Created Test Run (TestRunExecution)
Total 0 (0 Passed, 0 Failed, 0 Blocked)
... Show more

(b) Test Run Execution Panel

Figure 3.6. The extended Test Run GUI

3.4 Evaluation of the Proposed Solution

I conducted in collaboration with members of my research group a case study [17] in the considered automotive context to evaluate whether the proposed solution was able to mitigate the process drawbacks highlighted in Section 3.2. More in detail, the study aimed at answering the following Research Questions:

RQ$_1$ How does the adoption of the proposed solution affect the process rapidity?

RQ$_2$ How does the adoption of the proposed solution influence the effectiveness of the traceability links management related to the process?

RQ$_3$ How does the adoption of the proposed solution impact on the process visibility from the point of view of the Project Manager?

RQ$_4$ How does the adoption of the proposed solution impact the process acceptability from the point of view of the Test Engineers?

To answer these RQs, I executed with my research group the MIL Testing process using the new toolchain (hereafter "new process") on a number of SWC Releases that had been already tested in the past in the former infrastructure (from now on "original process"). Archival data about these process executions was available, such as the time needed for the execution of each activity of the process, or the number of traceability links produced every day.

3.4.1 Objects

As objects of the study we considered SWC Releases belonging to the IPC of a segment B mass market vehicle produced by FCA.

The selected SWCs were: the *Speedometer* responsible for filtering and presenting to the driver the vehicle speed information; the *Tachometer* that filters and presents to the driver the engine speed information and the *Trip* providing features for calculating and presenting to the driver some information related to one or more "trips", i.e. segments of traveling selected by the driver, including travel duration and length, travel average speed and estimation of distance that can be traveled until the fuel tank will be empty. Table 3.1 shows for each SWC the number of the available Test Cases for its testing.

Table 3.1. Case study selected objects

Software Component	Number of Available Test Cases
Speedometer	34
Tachometer	75
Trip	150

3.4.2 Subjects

The subjects of the study were employees of the EMEA SW Factory. We selected a Project Manager and three Test Engineers involved in the past execution of the original process on the IPC considered in the study. However, they worked on SWC Releases different from the ones we considered in the case study. The selected Project Manager had more than 15 years of experience managing more than 15 different projects. The selected test engineers had on average more than 20 years of experience in the MIL testing process.

3.4.3 Considered Metrics

Now I report the metrics we used to collect the evidences necessary for answering the RQs.

Rapidity

According to the definition of process rapidity provided by Sommerville [1], the rapidity of the MIL testing process represents how fast can the process deliver the test case execution results.

To evaluate this process characteristic, we measured the *execution time* of the whole testing process and of its composing activities.

The execution time of the overall MIL Testing process is $T = \sum_{i=1}^{4} T(A_i)$, where $T(A_i)$ represent the A_i process activity execution time.

Effectiveness of Traceability Links Management

To evaluate the effectiveness of the traceability link management in a given MIL process execution, we adopted the following two metrics:

- #ITL: the number of incorrect traceability links between Test Cases and Test Case Results stored into the ALM
- #MTL: the number of missing traceability links between Test Cases and Test Case Results.

These metrics were measured by a company employee who analyzed all the traceability links stored in the ALM during the MIL Testing process executions.

Visibility

According to Sommerville, the visibility of a process represents to what extent the process activities culminate in clear results, so that the progress of the process is externally visible. Usually the progress of a process can be evaluated by observing the availability of specific artifacts delivered by the process and measuring the quantity of these artifacts that are available at specific points of the process execution.

To evaluate the process visibility, we designed a semi-structured interview and submitted it to the Project Manager.

The interview aimed at understanding which artifacts of the MIL testing process the Project Manager exploits to monitor the MIL testing process and if this information can be obtained when needed.

Acceptability

The acceptability of the MIL Testing Process represents if the defined process is acceptable to and usable by the engineers responsible for its execution.

To evaluate the acceptability of a given MIL process, we designed a semi-structured interview with the Test Engineers. The interview aimed at understanding the usability requirements of the Test Engineers about the MIL process, which process factors impacted the process usability from their point of view, and whether these usability conditions were met by the considered process.

3.4.4 Case Study Procedure

The case study was performed following a four steps procedure.

Subjects Allocation Step

The Project Manager randomly assigned to each Test Engineer the execution of the MIL Testing process related to one of the selected SWC Releases.

Execution Step

Each Test Engineer executed the testing process exploiting the proposed tool architecture. They were asked to complete the assigned process execution in a fixed time frame of five days, according to the standard company practices. For the new process executions, they had just to Create the Test Runs and to launch their execution through Polarion ALM.

Data Collection Step

At the end of the new process executions, the data needed to answer the defined Research Questions was collected.

The data needed to calculate the T, $T(A_i)$, $\#ITL$ and $\#MTL$ metrics were available in Polarion ALM. One of the authors queried the ALM to obtain the data regarding the new process executions. Archival data, about the original process executions, stored in the ALM were also queried.

Moreover, another author conducted the designed interviews with the study subjects and collected their answers.

Data Analysis Step

The collected data was analyzed to obtain the evidences to answer the defined Research Questions. The values of the defined metrics for the original and the new process executions were compared. To evaluate how the rapidity of the process was influenced we calculated the *Speedup Percentage* for the entire process, *SP*, defined according to the following formula:

$$SP = \frac{T_{original} - T_{new}}{T_{original}} * 100 \tag{3.1}$$

Moreover, the he *Speedup Percentage*, $SP(A_i)$, for each A_i activity was evaluated too. $SP(A_i)$ is defined as:

$$SP(A_i) = \frac{T_{original}(A_i) - T_{new}(A_i)}{T_{original}(A_i)} * 100 \tag{3.2}$$

$T_{original}$ and $T_{original}(A_i)$ represent the execution times for the entire process and for its A_i composing activities in the original process obtained through the archival data whereas T_{new} and $T_{new}(A_i)$ refers to the ones related to new process executions.

Table 3.2 shows the comparison results we obtained regarding the entire MIL Testing process (MIL) and the Create Test Run (CTR), Select

Test Cases (STC), Launch Test Cases (LTC) and Import Test Case Results (ITCR) activities.

Table 3.2. Rapidity Comparison Results

Software Component	**(MIL)**	**(CTR)**	**(STC)**	**(LTC)**	**(ITCR)**
Speedometer	7%	0 %	73%	0%	79%
Tachometer	9%	0.1%	81%	0%	90%
Trip	14%	0.1%	91%	0.3%	96%

To estimate the differences in effectiveness between the original and the new process two metrics named *Incorrect traceability links reduction percentage (ITLRP)*, and *Missing traceability links reduction percentage (MTLRP)* were proposed.

$ITLRP$ is given by the formula reported below and it measures the percentage reduction of the number of incorrect traceability links of the new process, $\#ITL_{new}$, with respect the ones of the original process, $\#ITL_{original}$.

$$ITLRP = \frac{\#ITL_{original} - \#ITL_{new}}{\#ITL_{original}} * 100 \tag{3.3}$$

$MTLRP$ measures the percentage reduction of the number of missing traceability links of the new process, $\#MTL_{new}$, with respect the ones of the original process, $\#MTL_{original}$. It is represented by the following formula.

$$MTLRP = \frac{\#MTL_{original} - \#MTL_{new}}{\#MTL_{original}} * 100 \tag{3.4}$$

Whether these comparison metrics assume negative values it means that the proposed solution decreases the process performance.

Table 3.3 reports the comparison results we obtained at the end of the study.

Table 3.3. Traceability Management Effectiveness Comparison Results

Software Component	**ITLRP**	**MTLRP**
Speedometer	100%	100%
Tachometer	100%	100%
Trip	100%	100%

The data collected through the interviews carried out to compare the visibility and the acceptability of the two processes was submitted to a data triangulation step. More in detail, the interviews data was independently analyzed by different authors and the results were compared.

3.4.5 RQ Answers

On the basis of the collected data I was able to answer the proposed RQs.

RQ_1: How does the adoption of the proposed solution affect the process rapidity?

I was able to answer RQ_1 on the basis of the data reported in Table 3.2.

The data highlights that the execution time of the overall process was not essentially affected by the adoption of the proposed architecture since the SP was always not greater than 14%

On the other hand, analyzing Speedup Percentage of each process activity, I was able to understand that the *Select Test Cases* and *Import Test Results* are the activities affected since they shows SP values always not less than 73% and 79%, respectively. No substantial differences in terms of execution time were measured for the other two activities of the process.

These results showed us that the adoption of the proposed architecture leads to process rapidity improvement.

RQ_2: How does the adoption of the proposed solution influence the effectiveness of the traceability links management related to the process?

Thanks to the analysis of the data reported in Table 3.3 I answered to RQ_2. All the incorrect and missing links introduced in the original process due to their manual insertion were avoided in the new process executions thanks to the automation of the Import Test Case Results activity.

The adoption of the proposed architecture leaded to 100% reduction for both Incorrect and Missing traceability links for all the objects of the study, improving the effectiveness of the process traceability links management.

RQ_3: How does the adoption of the proposed solution impact on the process visibility?

The interview showed that the Project Manager usually exploits the percentage of the executed test cases for monitoring the progresses of the MIL testing processes involving the Software Component Releases of its own responsibility.

By analyzing his answers, I was able to understand that in the original process, the Project Manager was able to monitor the progresses only on two defined milestones (i.e., the third and fifth day of the testing process). The other time he wanted to obtain the needed information he had to directly inquire the assigned Test Engineers.

On the other hand, in the new process the Project Manager is able to have a real time monitoring since the data are stored into the ALM as soon they are produced. I report in the following some excerpts of the

Project Manager statements that lead us to our conclusions:

"... with the original process, I could not check the information about component testing progress in real time because the information in the ALM did not reflect the real testing progress. To get real time information I had to directly inquiry the involved Test Engineers...".
"... Test Engineers should report test results every time they finish a test execution, but often they don't have the time to do so due to strict deadlines. So we have established some milestones to enforce in the ALM the executed test case results. When the milestone is reached, all the test cases executed until that point must be reported in Polarion ALM. This happens two times a week..."

The results of this qualitative analysis showed us the evidences that the adoption of the proposed architecture leads to significant improvements in the visibility of the MIL Testing process from the point of view of the Project Manager.

RQ_4: How does the adoption of the proposed solution impact the process acceptability?

From this analysis of the collected answers I understood that the main factors affecting the acceptability of the process from the point of view of the Test Engineer are the number of different tools to exploit and the tedious manual activities they have to perform for the accomplishment of their work.

The data showed us that in the original process the Tester Engineers need to switch between two different tools (Polarion ALM and MATLAB Testing environment) and had to execute many time consuming and boring tasks. On the contrary, thanks to the new architecture they use a single tool and the most tedious tasks are automatically executed without any manual intervention.

To give evidence of this, I report excerpts of statements given by one of the interviewed Test Engineer: *"... I have to use two tools for selecting and executing test cases ... the import of the results is fully manual .... this needs a great effort and distract me from the analysis of test results... With the new approach I select the test cases to execute in Polarion and launch their execution in the same tool... at the end of the test execution, I do not need to insert the test results back in Polarion since they are stored automatically."*

The results confirmed us that the adoption of the proposed architecture was successfully accepted by the Test Engineers.

3.4.6 Threats to Validity

Internal Validity

This aspect of validity needs to be evaluated when causal relationships are examined. It defines how sure we can be that the treatment actually caused the outcome. In our case, the subjects experience could be another factor influencing the outcomes of the study. To mitigate this threat, we selected subjects with different levels of experience. Moreover, a wider experimentation with other subjects should be carried out.

External Validity

External validity is related to what extent it is possible to generalize the findings and if these are of interest to people outside the reported case. A possible threat could be related to the application of the approach in a specific industrial domain and for a specific process. In order to minimize this threat a further experimentation in different industrial domains and for different processes will be performed.

Reliability

The reliability is concerned with to what extent the data and analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducts the same study, the result should be the same. To minimize this threat, the case study protocol and the artifact produced was reviewed by a group of researchers of the university. I reported, to the extent possible, the followed case procedure in this Chapter. Furthermore, data collected through the interviews and their analysis were reviewed by case subjects in order to avoid possible misinterpretations

3.5 Related Work

Application Lifecycle Management has been proposed as a solution for monitoring, controlling and managing software process over the entire application lifecycle [45]. ALM can be seen as a supervisor which covers the whole development process from the initial idea to the end of the product lifecycle through different core aspects: governance, development and operations [43]. They combine the supporting tool such as Version Control Systems (VCS) to handle the artifacts to issue tracking applications [123]. Nowadays many companies adopt Application Lifecycle Management (ALM) platforms to follow the entire lifecycle of their software product. The goal of an ALM platform is to make software development and delivery more efficient, lower its costs and improve software quality. ALM is based on three main concepts, called *the pillars of ALM*: *traceability*, *visibility* and *process automation* [39]. In order to truly reach these goals, an ALM platform should provide a holistic view on all the software process. All the tools involved in the development process should be able to interact among them through the ALM platform.

Although ALM tool support has increased during the past decades, there is still a tool support ecosystem largely fragmented [114]. Many of the ALM tools available on the market (e.g. IBM Rational Team Concert, HP Application Lifecycle Management, Polarion ALM, etc.) are able to reach the defined goals, but only through the integration of tools of their ecosystems. Most companies already have a de-facto development process and tools. For these reasons ALM platforms offer mechanisms to interconnect most commonly used tools like Microsoft Excel, Eclipse, IBM DOORS and they also offer an Application Programming Interface (API) for the integration with other tools.

There is still the need to provide solution able to integrate the different tools used during the software development in order to automate its process and to interconnect artifacts created through the adopted heterogeneous tools.

The need to link tools that address different aspects of the development process, creating so called toolchains is emerged since 90s [118].

Tool integration is often achieved building automated tool-chains that take care of all the needed interactions between tools, from data exchange and manipulation, to the tools execution itself. Creating a toolchain is not an easy task since every toolchain is specific for a particular development process and environment. Based on the number of tools to integrate and on the breadth of the development sub-process that must be automated, it can be hard to properly describe, design and develop the needed toolchain.

Solutions to tool integration are based on the same principles: the presence of a central repository for data management, a framework based on a service oriented architecture or a component architecture for tool orchestration and a set of transformations to make data available to different tools. In [124], in the context of MOGENTES project (a project aimed at enhancing testing and verification of dependable embedded systems), a service-oriented, model-driven, process-centric approach for the definition of toolchains was proposed. Their approach leads to the creation of a tool manager (the broker) to which the tools should be registered to offer their functionality as services to the final user. However, each tool must expose a well-defined interface to provide homogeneous programming interfaces and hide heterogeneity.

In [115], a way to properly describe a toolchain defining a Domain Specific Modeling Language (DSML), named Tool Integration Language, was proposed. This language aims at mapping tool integration domain concepts to language ones in order to fully describe a toolchain and to allow its automatic generation using a Service Component Architecture (SCA). Anyway, such approach is not used in practice where usually ad-hoc solutions are developed, instead.

The tool integration was highly investigated in the automotive domain, where several research projects have been funded to investigate this problem. These projects aim to create a toolchain that covers the entire embedded software development process, starting from requirements specification, to model design, to its implementation and testing.

The Cesar Project [125] offers a toolchain that is divided in two interconnected parts, one for the requirements management and one for the system design, safety analysis and V & V. The two parts use different integration technologies and different repositories for their artifacts, but the toolchain is able to offer full traceability of the application. The integration is achieved by using a defined meta-model and transformations technologies.

The Amalthea Project [126] goal was to create an open source toolchain for embedded software development in the automotive domain based on the Eclipse Software development Framework and leaded to the Amalthea4Public tool[8]. The proposed platform integrates a lot of existing tools already supported by the Eclipse Framework. These tools can be plugged in and executed automatically, but Amalthea platform adds to the standard framework the possibility to define models using the Eclipse Modeling

[8] *http://www.eclipse.org/community/eclipse_newsletter/2015/march/article2.php*

Framework and the presence of a high performance model repository to store development models so that every tool in the framework can access project data.

Many tools were proposed to help companies keeping traceability between the artifacts involved in the development process and choosing which one to use is a complex task. The usually adopted tools to express and maintain traceability were general purpose tools such as word processors or, even better, spreadsheets. They are adopted to create a Traceability Matrix, a matrix showing dependencies between artifacts. This matrix is easily readable by an user and has the advantage of being a single repository for documenting both forwards and backwards traceability across all of the work products [127]. Traceability was also managed using Entity-Relationship (ER) Models and database technology, with the development artifacts represented by entities, while relationship instances represent links between artifacts [128]. But keeping such a matrixes or tables always updated requires considerable manual efforts. With the advent of hypertext technology and special-purpose tools it became easier to provide full traceability to a software project.

The importance of automatic traceability management tool has been extensively investigated by the authors in [129] who conducted a series of interviews with industrial and academic partners and software development stakeholders to study the impact of such tool on the quality of traceability links. The data from these interviews reveals two main challenges: traceability links are still mostly created manually and users must cope with interconnected but highly heterogeneous artifacts across tool boundaries.

3.6 Conclusions and Future Work

The ecosystem of artifacts, tools and platforms used in software development processes executed in industrial organizations can be very complex and heterogeneous. Software traceability represents a feasible solution for managing the complexity of these ecosystems and ALM tools provide a professional solution for storing and maintaining the traceability links among artifacts.

In the practice, ALM and legacy tools adopted in real software processes are not coordinated and integrated, and software organizations are forced to adopt tool integration solutions if they want to automate and improve the traceability management in their processes.

In this Chapter I reported our experience in developing a solution for integrating an ALM tool with the software tools used in MIL testing processes executed in FCA. The proposed solution exploits a Continuous Integration platform to implement a toolchain aimed at automating the manual activities of the process and at automatically creating the traceability links between the involved artifacts. This architectural solution is based on Master/Slave components, where the Master implements the process workflow, while each Slave interfaces a single tool of the toolchain. This architecture can be easily extended and reconfigured both to integrate new tools with the ALM, both to implement new software processes. The integration of another tool will require an additional Slave component and the design of the connectors needed for interfacing the tool. The implementation of a new process will require the modification of the Master component too.

The proposed architecture was evaluated by a case study conducted in the FCA EMEA SW Factory where I compared the results of the MIL testing process execution before and after the introduction of the new toolchain. The experimental results showed that the traceability management, the testing process rapidity, visibility and acceptability improved thanks to the new toolchain.

In future work, I plan to extend our architecture by integrating additional tools with the ALM and to reuse the architecture for implementing different processes carried out both in the same context and in different industrial domains.

Chapter 4

Approaches and tools for supporting the analysis and comprehension of spreadsheet based artifacts

This Chapter presents a reverse engineering process and a tool I designed to support the comprehension of spreadsheet based artifacts, even the ones developed exploiting the VBA programming language. I decided to focus on spreadsheet based artifacts, since they are widely adopted to support different phases of the software development process and their comprehension is still an open issue. The heuristic-based reverse engineering process I designed allows the inferring of data models from a corpus of spreadsheet-based artifacts. The process is fully automatic and it is based on seven sequential steps. Regarding the tool, named EXcel Application Comprehension Tool (EXACT), it was developed for supporting the comprehension of spreadsheet based artifacts, even the one implemented exploiting the VBA programming language. The tool has been implemented as an add-in that extends the Excel working environment by providing analysis and visualization features. It provides features for the automatic extraction of information about the elements composing the analyzed Excel spreadsheet artifact, the functionality it exposes through its User Interface and the dependencies among its cells. This information is provided by means of interactive views. Both the applicability and the effectiveness of the proposed reverse engineering process have been assessed by an experiment I conducted with the support of my research group in the automotive industrial context. The process was successfully used to obtain the UML class diagrams representing the conceptual data models of three different corpora of spreadsheet based artifacts. Finally, the validity of the EXACT tool in supporting real comprehension tasks has been assessed by another case study performed with professional end users from the same automotive industrial domain.

4.1 Introduction

Spreadsheets are interactive computer application programs for organization, analysis and storage of data in tabular form. Their usage is diffused in a community made by many millions of end user programmers [130] and it has been estimated that 90% of desktop computers are equipped with Excel [131], the most popular spreadsheet system belonging to the Microsoft Office suite. Spreadsheets are widely adopted within companies and, although such systems were originally developed to support calculation and financial analyses, now they are widely used as organizational performance tools [132] and as collaborative technology [133], in scientific, business and industrial domains. Spreadsheet based artifacts are widely used in software processes for supporting different phases of the development. Hence, their comprehension has been not widely investigated by the research community.

In the most widely accepted sense, a spreadsheet is simply considered as grids of cells filled with data or formulas to perform calculations. Nevertheless, a spreadsheet is usually more than that. Spreadsheet systems provide the possibility to enrich both the user interface and the functionality provided by a basic spreadsheet through the employment of programming languages, such as Visual Basic for Applications (VBA) used by Excel. VBA is the most popular scripting language usable for extending any host software application belonging to the Microsoft Office Suite[1]. It provides mechanisms and programming constructs for defining both rich and interactive GUIs that allow end users to generate user events, and procedures that will manage these events. These procedures may even implement complex business rules that manipulate the data stored in spreadsheet cells. By means of these mechanisms, end users are able to develop real software applications that often turn out to be used for storing and processing increasing amounts of data and for supporting growing numbers of users over long periods of time [134]. In addition, these applications are usually submitted to frequent maintenance interventions aiming to extend and evolve their content and behavior.

Although additional and more modern technologies, such as C#, F# or Javascript, are available for extending Excel spreadsheet applications, VBA is still the most widely and frequently used one in this context. Moreover, there is a huge amount of legacy spreadsheet applications having a large part of VBA scripts that are still largely used. This kind of applications are usually developed and maintained by end users who are not software engineers and often do not have a full knowledge of the Excel development environment too. As reported in [135], end users learn the Excel system by using it or with the help of a colleague. They typically develop the applications performing no preliminary planning, analysis or

[1] *http://msdn.microsoft.com/en-us/library/dd361851.aspx*

design activities. These users alternate creative activities like thinking and mechanical activities like typing, pointing cells or copying them. This development process may lead to an uncontrolled growth of the complexity of these applications, the frequent introduction of bugs and the lack of any kind of development documentation.

The resulting scarce quality and comprehensibility problems of spreadsheet applications are well known in the literature. Moreover, the *horror stories*[2] presented by the European Spreadsheet Risk Interest Group report common problems with uncontrolled use of spreadsheet applications and their impacts. They show the relevance of effectively comprehending and analyzing Excel applications.

Reverse engineering techniques and tools may provide a valuable support to end users involved in software development processes in the tasks of adopting, maintaining and evolving existing spreadsheet based artifacts. Unfortunately, the Visual Basic Editor integrated in Excel lacks valid support for the comprehension and reverse engineering of VBA code. In addition, the software engineering research community did not devote adequate attention to this field. While many problems regarding the management of the data embedded in spreadsheet cells have been widely addressed in the literature [136, 137] since the nineties, other relevant aspects concerning the analysis, comprehension, and reverse engineering of the scripting code components belonging to these applications have been disregarded.

I believe these topics are worth to be investigated, especially due to the economic importance and diffusion of spreadsheets in many productive, industrial and business environments, especially in the software process domain. In addressing this issue, it can not be neglected that spreadsheets are peculiar software applications that pose a number of different challenges. As an example, they show the comprehensibility issues of an hybrid software architecture, including the characteristics of both event-

[2] *http://www.eusprig.org/stories.htm*

based and data-shared software architectural styles [138]. Moreover, since the core components of spreadsheets are their cells, which may contain either data, or formulas, or even user interface components, it is not easy to directly locate the different parts of interest of the application. Lastly, their analysis and comprehension is aggravated by the existence of many dependency relationships between cells that may be scattered in different parts of the spreadsheet and that may not always be clear to end users.

In order to address these issues, I developed a reverse engineering process and a reverse engineering tool, named EXACT (EXcel Application Comprehension Tool), aimed at supporting the analysis and comprehension of Excel spreadsheet artifacts, even the ones based on VBA. The reverse engineering process is based on the application of different heuristic rules I defined through analysis of different spreadsheet based artifacts corpora. The application of the process aims at producing, through subsequent refinements, a conceptual UML class diagram abstracting the data model of the artifacts. On the other hand, the EXACT tool has been implemented as an add-in that extends the Excel working environment by providing analysis and visualization features. It is able to extract information about the elements composing the analyzed Excel spreadsheet application, the functionality it exposes through its User Interface and the dependencies among its cells. This information is provided by means of interactive graph-based and list-based views. The usefulness of the process and tool has been assessed with case studies that I performed, supported by my research group, with professional end users from an automotive industrial domain where several spreadsheet based artifacts are used to support and manage different phases of the embedded software development process.

The remainder of the Chapter is organized as it follows: Section 4.2 describes background information about spreadsheet artifacts and reports the main comprehension issues affecting them. Section 4.3 reports the solution I proposed to solve the reported issues. More in detail, it describes the reverse engineering process to recover conceptual models from spreadsheet based artifacts, a conceptual model defined to represent a VBA-based spreadsheet artifact and the characteristics of the EXACT tool. Section 4.4 reports the case studies I carried out to validate the proposed solutions in the automotive domain. In Section 4.5 related works about reverse engineering techniques applied to spreadsheet based artifacts are discussed. Eventually, section 4.6 presents conclusions and future work.

4.2 The Addressed Problem: Comprehension issues of Spreadsheet based artifacts

Excel offers end users a simple and intuitive working environment for implementing calculation sheets or simple data stores in tabular format. From a structural point of view, an Excel application is made of a *Workbook*, that usually includes a number of *Worksheets*, grids of *Cells* that end users can fill in with data and formulas. In the same working environment of Excel, end users can personalize and extend the GUI of the application either by drawing graphical objects, named *Shapes*, over the cells or by defining advanced *User Forms*.

Moreover, end users can also extend the application behavior by means of *Procedures* written in VBA code. The Excel environment also provides the *Record Macro* feature allowing end users to automate repetitive tasks without directly writing a single line of code.

This feature permits to capture and record the user interactions with the spreadsheet environment transforming them into a *Macro*, a re-executable VBA code. End users can execute a recorded macro whenever needed. Although this feature is considered really helpful, it produces inefficient code [139], since each interaction of the user with the Excel environment is directly translated into VBA code. As a result, the automatically produced code may result very difficult to be comprehended.

In addition, Excel provides a dedicated IDE, the VBA Editor, that allows both to define user forms and to write VBA Procedures. The execution of Procedures is driven by user events, since Excel supports the event-handling mechanism. In particular, a *Procedure* can be defined as event handler of a given Excel item (such as Worksheets, shapes, etc.) and will be executed whenever a specific event occurs on this item. By means of these features, end users can easily develop complex spreadsheet applications, made by many data cells, organized in multiple sheets, and embedding even complex business rules for managing user events fired on the rich GUI of the application.

Analyzing and comprehending this type of applications are affected by many specific issues. A first problem derives from the lack of a clear separation between the presentation, business logic and data management components of a spreadsheet application. The cells can be filled in with either data, either formulas, or User Interface elements indistinctly. The flexibility of the spreadsheet environment, mixing data and calculation, may be beneficial for end users, simplifying their work. At the same time, it may complicate the comprehension of the application, since end users may feel confused and struggle with identifying the different parts of interest during the accomplishment of comprehension tasks.

As another issue, it should be considered that Excel applications are characterized by a hybrid software architecture that is a combination of shared-data and event-based architectural styles [138]. The application provides indeed a central data store, composed by all the data cells included in its Worksheets, and a set of independent business logic components given by all the Procedures of the application, whose execution is driven by user events. Comprehending the behavior of event-based architectures is affected by the problem of finding and firing all the events that may be executed. On the other hand, shared data architectures suffer from the well-known problem that a change in shared data may imply changes in all modules using it. As an example, whenever the format or content of a cell changes, all the VBA Procedures relying on this data cell will have to be detected and changed too. This task may be hard and time-consuming, if we consider that data cells can also have hidden dependency relationships with other cells that are due either to formulas or other VBA Procedures. The necessity for an end user or for a maintainer of the application to take into account these hidden inter-dependencies is well-known in the literature [140]. Unfortunately, analyzing and comprehending this web of interconnections may be a complex and time-consuming task, even harder if we consider that Excel applications are often made by many thousands of cells, scattered across many different worksheets, and they often lack of any kind of effective internal and external documentation.

Another complication derives from the fact that the content and the structure of a spreadsheet application are not fixed and may be varied at run-time. An end user can add or delete, in a very easy way, elements such as Worksheets or Shapes and content in different cells while using the application. The end users may not have a clear vision of the impacts of such modifications. Even a simple insertion of a value in a cell of a given worksheet may cause implicit modifications of other elements of the application such as VBA procedures and/or trigger the execution of other events that the end user may not be aware of or informed about.

In order to address these comprehension issues, a feasible approach is provided by reverse engineering processes and tools. According to [141], there are different kinds of feature offered by these tools that can aid the comprehension tasks. *Browsing and Visualization features* may support the analysis, comprehension, and navigation inside the structural organization of the application. *Locating features* could provide a valid support for recovering and accessing the different types of component making up the application and for tracing their inter-relationships. Moreover, *De-*

pendency Analysis features may ease the identification of dependencies among data cells. *Analysis and Abstraction features* may support the comprehension of the functions provided by the application through its GUI. Consequently, I decided to develop a tool for supporting the execution of comprehension tasks on VBA-based spreadsheet applications by providing features for:

- recovering and reporting the elements composing these applications and the relationships existing among them in order to aid the comprehension of their different parts and to overcome the lack of a clear separation between their different components
- retrieving and listing the event-handlers related to User Interfaces graphical objects and all the components involved in their execution. This feature is needed to support the comprehension of the user functionality offered by means of event handling mechanisms
- recognizing and showing the dependencies that may be established between several cells through different mechanisms like formulas, data validation features and VBA code. Such a feature is needed for supporting end users in comprehension tasks related to maintenance, migration or knowledge transfer of spreadsheet applications as reported in [140]

4.3 The Proposed Solutions

In this section I report the solutions I developed for supporting the comprehension of spreadsheet based artifacts. Precisely, I defined a Reverse Engineering process for recovering Data Models from them and a Reverse Engineering tool, named EXACT, providing visualization and analysis features. More details about these solutions are reported in the following.

4.3.1 A Reverse Engineering Process to Recover Data Models from Spreadsheet based artifacts

The industrial context of the work reported in this Chapter included a large number of spreadsheet based artifacts, implemented by Excel files, used in the development process of ECU in an automotive company. They supported the Verification & Validation (V&V) activities and the management of Key Performance Indicators (KPI) about the development process. The spreadsheets inherited from a same template and included some VBA functionalities providing data entry assistance. Moreover, their cells followed well-defined formatting rules (i.e., font, colors, and size of cells) that improved the readability and usability of the spreadsheets, and complied to the following layout rule: all the data concerning the same topic in a single spreadsheet file were grouped together in rows or columns separated by empty cells or according to specific spreadsheets patterns [142]. Taking the cue from other works proposed in the literature, I founded my data model reverse engineering process on a set of heuristic rules. I defined a process made of seven steps that can be automatically performed in order to infer, with gradual refinements, the UML class diagram of the considered spreadsheet based artifacts composing a specific artifacts corpus. In each step, one or more heuristic rules are executed. Each rule is based on the analysis of one or more spreadsheet based artifacts belonging to the corpus of spreadsheet files composing the subject information system. In the following I describe the steps of the proposed process.

In the *Step 1* I preliminary apply Rule 1 that abstracts a class named *Sp* whose instances are the spreadsheet based artifact that comply with a same template.

In the *Step 2* I exploit the Rule 2 that is executed on a single spreadsheet file of the corpus. This rule associates each non empty sheet of the spreadsheet based artifact with a class S_i, having the same name of the corresponding sheet. Moreover, an UML composition relationship between the *Sp* class and each S_i belonging to the file is inferred. The multiplicity of the association on each S_i side is equal to 1. Figure 4.1 shows an example of the application of this rule on an example spreadsheet based artifact.

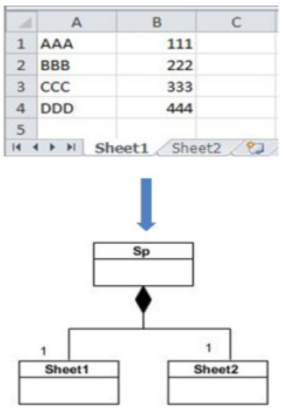
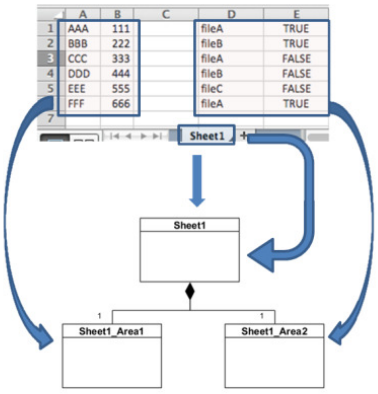


Figure 4.1. Example of Step 2 execution

The *Step 3* exploit the *Rule 3* that is executed on a single spreadsheet file of the corpus. This heuristic, according to [143] and [142], associates a class A_j for each non empty cell area $Area_j$ of a sheet already associated to a class S_i by the Rule 2. Each class A_j is named as follows: $Nameofthesheet\_Area_j$. Moreover, an UML composition relationship between the S_i class and each A_j class is inferred. The multiplicity of the association on each A_j side is equal to 1. In Figure 4.2 an example of Step 3 execution is reported.

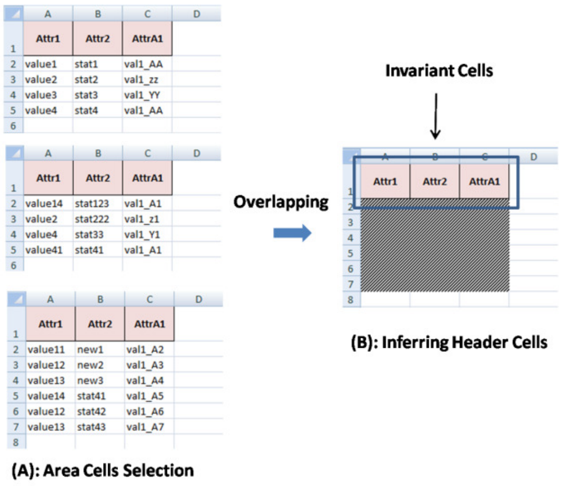
Step 4 foresees the sequential application of two different rules, i.e. *Rule 4.1* and *Rule 4.2*. The Rule 4.1 is applied to discriminate the header cells [144] of each area $Area_j$ that was inferred in the previous step. Rule 4.1 analyzes the content of the spreadsheet files belonging to the corpus in order to find the invariant cells for each area $Area_j$. An invariant cell of an area is a cell whose formatting and content is the same in all the analyzed spreadsheets. The set of invariant cells of an area $Area_j$ composes the header of that area. Figure 4.3 shows how the Rule 4.1 works. In a first phase, all the spreadsheets are queried to select, for each of them, the cells of a given area.



	A	B	C	D	E
1	AAA	111		fileA	TRUE
2	BBB	222		fileB	TRUE
3	CCC	333		fileA	FALSE
4	DDD	444		fileB	FALSE
5	EEE	555		fileC	FALSE
6	FFF	666		fileA	TRUE
7					

Figure 4.2. Analysis of non-empty cell areas belonging to Sheet1 executed in Step 3

In the next phase, all the selected cells are analyzed to recognize the invariant cells for the considered area. Intuitively, overlapping the contents of all the spreadsheets for a given area $Area_j$, then the header is given by the cells that, for the considered area, are invariant in all the files. Rule 4.2 is executed on the headers inferred by Rule 4.1. For each area $Area_j$, this heuristic analyzes the style and formatting properties of the cells composing its header. It permits to discriminate subareas $SubArea_m$ having header cells satisfying specific patterns. Rule 4.2 associates a class SA_m for each $SubArea_m$. Each class SA_m is named as follows: $Nameofthesheet\_SubArea_m$, if no name could be associated to the class according to the recognized patterns. The names of the attributes of the class are inferred from the values contained into the header cells. Moreover, an UML composition relationship between the class S_j and each related SA_m class is inferred. The multiplicity of the association on each



	A	B	C	D
1	Attr1	Attr2	AttrA1	
2	value1	stat1	val1_AA	
3	value2	stat2	val1_zz	
4	value3	stat3	val1_YY	
5	value4	stat4	val1_AA	
6				

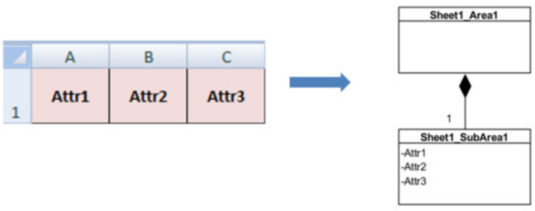
	A	B	C	D
1	Attr1	Attr2	AttrA1	
2	value14	stat123	val1_A1	
3	value2	stat222	val1_z1	
4	value4	stat33	val1_Y1	
5	value41	stat41	val1_A1	
6				

	A	B	C	D
1	Attr1	Attr2	AttrA1	
2	value11	new1	val1_A2	
3	value12	new2	val1_A3	
4	value13	new3	val1_A4	
5	value14	stat41	val1_A5	
6	value12	stat42	val1_A6	
7	value13	stat43	val1_A7	
8				

	A	B	C	D
1	Attr1	Attr2	AttrA1	
2	▨	▨	▨	
3	▨	▨	▨	
4	▨	▨	▨	
5	▨	▨	▨	
6	▨	▨	▨	
7	▨	▨	▨	
8				

Figure 4.3. Example of Step 4 execution

class SA_m side is equal to 1.



	A	B	C
1	Attr1	Attr2	Attr3

Figure 4.4. Example of header cells pattern inferring a single class and its attributes

Some examples of Rule 4.2 executions are reported. As an example, in Figure 4.4 an UML class, having the default name $Sheet1\_Area_1$, is inferred since three consecutive header cells have the same formatting style. The attributes of the class are named after the values contained into the header cells. Figure 4.5 shows a variant of the example reported in Figure 4.4. In this case two UML classes are inferred since two groups of consecutive cells having the same formatting characteristics were found.

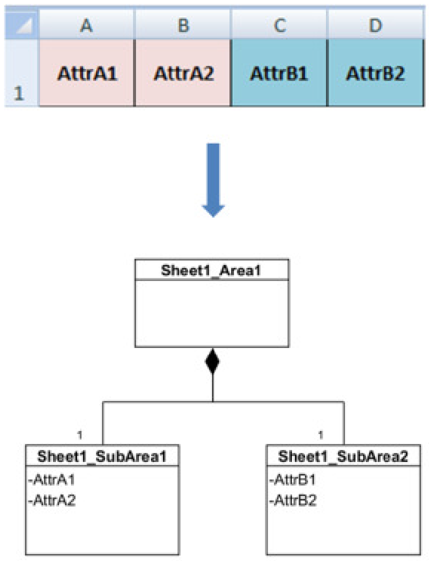


Figure 4.5. Example of header cells pattern inferring two classes and their attributes

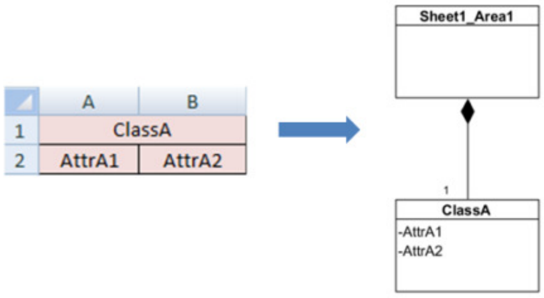


Figure 4.6. Example of header cells pattern inferring a single class with attributes and class name

A further pattern is shown in Figure 4.6 where the header is structured in two different levels. The upper level, composed of merged cells, permits to infer a class named ClassA, while its attributes are named after the val-

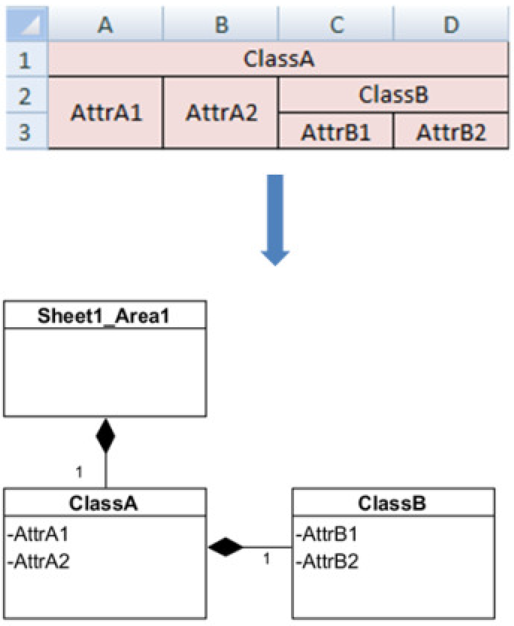
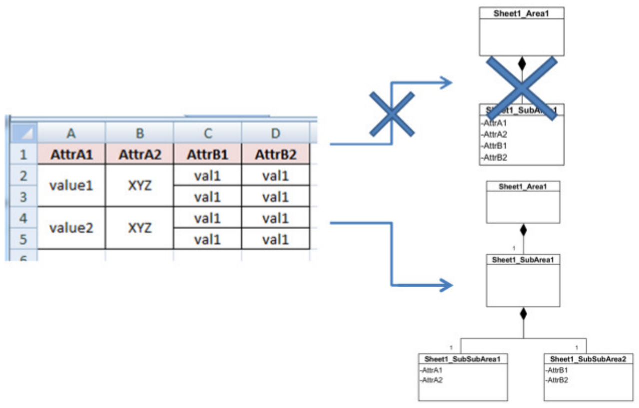


Figure 4.7. Example of header cells pattern inferring two classes with attributes, class names and their composing relationship

ues contained in the header cells of the lower level. In the example shown in Figure 4.7, the previous pattern is applied twice and a composition relationship is inferred between the two obtained classes.

In the *Step 5* I exploit *Rule 5* that is applied on the whole corpus of spreadsheets. Rule 5 is applied to all the subareas $SubArea_m$ to find possible sub-subareas made by groups of data cells having the same style and formatting properties. If a subarea $SubArea_m$ is composed by two or more sub-subareas then the heuristic associates a class, SSA_k for each sub-subarea. Each class SSA_k is named as follows: $Nameofthesheet\_$-$SubSubArea_k$. Moreover, the new classes substitute the class associated to the $SubArea_m$, and an UML composition relationship between the A_i class and each SSA_k class belonging to the area is inferred. The multiplicity of the association on each class SSA_k side is equal to 1.

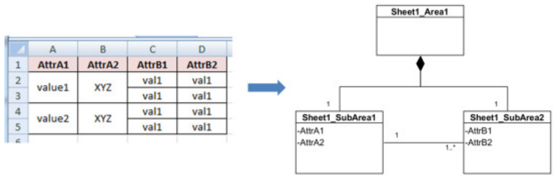


	A	B	C	D
1	AttrA1	AttrA2	AttrB1	AttrB2
2	value1	XYZ	val1	val1
3			val1	val1
4	value2	XYZ	val1	val1
5			val1	val1

Figure 4.8. Example of Step 5 Execution

Figure 4.8 shows an example of how Step 5 works.

Step 6 requires the application of *Rule 6*. This heuristic is applied to the overall corpus of spreadsheets. It analyzes the formatting properties of the cells belonging to consecutive subareas and sub-subareas in order to infer association relationships and multiplicities between the classes that were associated to these areas in the previous steps. An example of how this rule works is reported in Figure 4.9. In this case two consecutive subareas are considered, i.e., $SubArea_1$ related to columns A and B and $SubArea_2$ corresponding to columns C and D. The classes named $Sheet1\_SubArea_1$ and $Sheet1\_SubArea_2$ were associated to the $SubArea_1$ and the $SubArea_2$, respectively. Since for each spreadsheet a tuple of $SubArea_1$ is composed by a number of merged cells that is an integer multiple of the number of merged cells related to a tuple of $SubArea_2$, then Rule 6 infers a UML association between the two classes related to the two subareas. The multiplicity of the association on the side related to the class $Sheet1\_SubArea_1$ is equal to 1 whereas the one on the other side is 1..*.



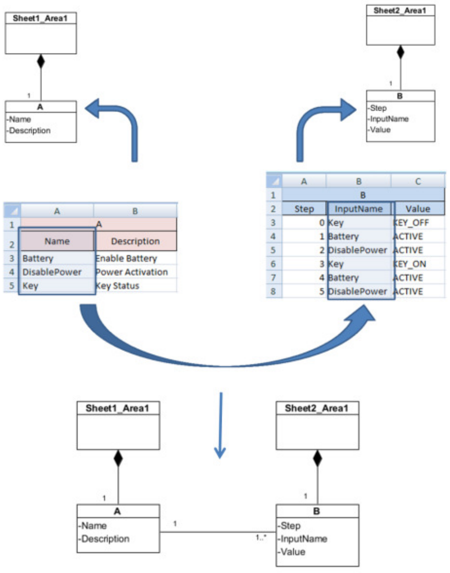
	A	B	C	D
1	AttrA1	AttrA2	AttrB1	AttrB2
2	value1	XYZ	val1	val1
3			val1	val1
4	value2	XYZ	val1	val1
5			val1	val1

Figure 4.9. Example of Step 6 Execution

In the *Step 7* the *Rule 7* is exploited. This heuristic is applied to the overall corpus of spreadsheets. It analyzes the value of the cells in order to infer association relationships between classes. As an example, if in all the spreadsheets, for each cell of a column/row that was exploited to infer an attribute of a class A there is at least a cell of a column/row that was exploited to infer an attribute of a class B having the same value, then it is possible to define a UML relationship between the UML Class A and the UML Class B. Moreover, if the cells of the column A have unique values then the A side and B side multiplicities of the inferred UML relationship are 1 and 1..* respectively, as shown in Figure 4.10.

4.3.2 Conceptual Model of VBA-based Spreadsheets

Moreover, I intended to develop a reverse engineering tool, I named EXACT, for supporting the comprehension of spreadsheet based artifacts. In order to implement the reverse engineering features provided by the EXACT tool, I preliminarily defined a *Conceptual Data Model* representing a VBA-based Spreadsheet artifact in terms of its entities and the relationships existing among them. More in detail, I focused on: (1) the peculiar elements belonging to any VBA-based spreadsheet artifact such as Workbook, Worksheet, VBProject, Code Module, etc., (2) the entities that are responsible for rendering their rich user interfaces, (3) the different elements needed for implementing the event-based mechanisms of VBA-based spreadsheet applications, (4) the entities that are able to de-



	A	B
1	A	
2	Name	Description
3	Battery	Enable Battery
4	DisablePower	Power Activation
5	Key	Key Status

	A	B	C
1	B		
2	Step	InputName	Value
3	0	Key	KEY_OFF
4	1	Battery	ACTIVE
5	2	DisablePower	ACTIVE
6	3	Key	KEY_ON
7	4	Battery	ACTIVE
8	5	DisablePower	ACTIVE

Figure 4.10. Example of Step 7 Execution

fine dependencies among different cells such as Formulas, Data Validation Feature and VBA code. The proposed model does not provide a comprehensive representation of all the elements of any spreadsheet application but just the ones required for implementing the feature of EXACT. In order to define this Conceptual Data Model I considered the Excel Object Model that reports the COM-based object model of the Excel application. This model is accessible exploiting the *Office Primary Interop Assemblies* (PIAs) library[3].

The proposed *Conceptual Data Model* is reported by two UML Class Diagrams presented in the following.

[3] *http://msdn.microsoft.com/en-us/library/15s06t57.aspx*

Modeling the structural components of a VBA-based Excel application

The Class Diagram shown in Figure 4.11 reports the structural entities of an Excel application along with their properties, composition and generalization relationships.

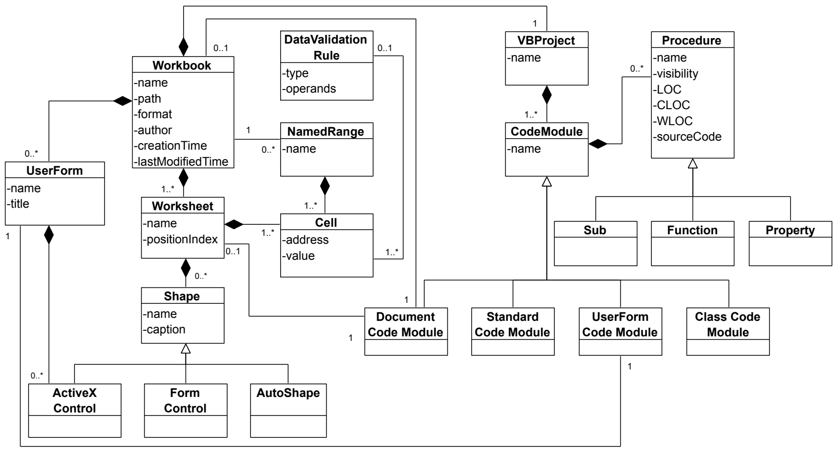


Figure 4.11. Conceptual Data Model - Structural Relationships

The main entity is represented by the *Workbook* class, associated with the Excel file of the application. The properties of a Workbook include its *name*, the *path* where it is stored, its *author*, the *fileFormat* that is the format of the analyzed Excel file (e.g. .xls, .xlsx, .xslm, etc.), the date-time of its *creation* and the one of its *last modification*. A Workbook can be composed by one or more *Worksheet* instances. Each Worksheet is identified by its name and index related to its position, is composed by 1 or more cells (*Cell*) where each Cell instance is characterized by its *address*, identifying the location of the cell in the worksheet, and may contain a *value* that can be a numeric one, a text, a formula, etc. Moreover, it is possible to assign a mnemonic name to a cell or set of cells, defined as *Range*, in order to reference them in formulas or in the VBA code. Each

Workbook may be composed of zero or more *NamedRange* instances. In addition, Excel provides a data validation feature that can be exploited to define restrictions on what data can be entered in a cell. These restrictions can be defined through several types of data validation rules involving additional operands. As an example, a Data Validation Rule of type *List* indicates that a cell can be filled with one of the values specified by the *operands* that may be *Ranges* of cells. In order to represent this aspect of Excel, in the proposed conceptual data model a Cell may be associated with a *DataValidationRule* Class that is characterized by a *type* and a list of operands.

All these classes represent entities that logically implement the 'Data Layer' of the application. In addition, each Worksheet may also include items responsible for implementing the User Interface offered by the spreadsheet. In particular, a Worksheet may include 0 or more *Shapes*, i.e. specific widgets enriching the User Interface offered by the spreadsheet. Each *Shape* is characterized by its *name* (for its identification) and, eventually, by a *caption* representing the text shown above the shape. Three specific types of shape are represented by the *ActiveX Control*, *Form Control* and *Auto Shape* classes.

The *FormControl* shape class represents the original GUI objects that are compatible with earlier versions of Excel and can be used either to interact with cell data without using VBA code, or to run a Macro. The *ActiveX Control* class contains GUI objects more complex than the Form Control ones since they provide richer interactions to the final user having more events that can be handled. The *AutoShape* class represents drawing objects that may overlay the Excel GUI.

Each Workbook may have its User Forms too. The *UserForm* class, characterized by its *name* and *title* caption, represents the custom windows containing zero or more ActiveX Controls.

Finally, the model includes several classes representing the VBA code included in the application. The *VBProject* class represents the overall VBA code belonging to the application. It is organized into one or more *Code Module* instances representing VBA code containers. There are four types of VBA code modules, represented by the classes *Document Code Module*, *User Form Code Module*, *Class Code Module*, and *Standard Code Module*, respectively. The *Document Code Modules* class is meant to enclose the event handlers belonging either to the Workbook or to its Worksheets. The *User Form Code Module* class contains the VBA code related to the event handler belonging to the User Forms. Each User Form is associated to its User Form Code Module class. The *Class Code Module* includes the VBA code defining custom classes and the *Standard Code Module* includes code snippets related both to custom functions and macros. A Code Module may include zero or more Procedures. A *Procedure* class represents a VBA code block and is characterized by its *name*, visibility, LoC and Commented Lines of Code (CLoC). VBA provides three types of procedures that are Subs, Functions and Properties represented by *Sub*, *Function* and *Property* classes respectively. A Sub is a procedure that does not have any returned value whereas the Function does have a returned value and can be used inside a cell as a user-defined formula. A Property procedure is meant to be used in a Class Code Module to provide both set and get methods for its attributes.

Modeling the relationships between VBA-based Excel application components

The class diagram reported in Figure 4.12 shows how different spreadsheet entities can be associated with each other due to relationships implemented either by means of VBA code or other Excel mechanisms (such as *formulas* or *data validation rules*).

The main entity shown in this model is the *Procedure* class that may present three types of association relationship with *User Form* entities that are due to specific VBA code statements included in the procedure. These relationships are reported in the left part of Figure 4.12. The *shows* association relationship indicates that a *Procedure* may instantiate and render zero or more User Forms. The *unloads* association tells that a *Procedure* is potentially able to close zero or more *User Forms*, while the *hides* relationship indicates that a *Procedure* may make invisible a set of *User Forms*. These three associations have a bidirectional navigability. In other words, a *User Form* may be loaded, unloaded or hidden by more *Procedures*.

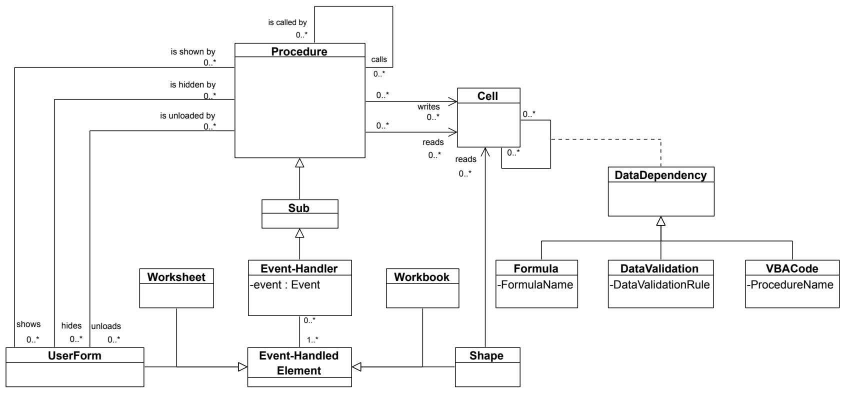


Figure 4.12. Conceptual Data Model - Excel Specific Relationships

The self-association *calls* between *Procedures* indicates the well-known calling relationship between VBA procedures. Moreover, the model reports that a *Procedure* may have *reads* and *writes* associations with zero or more *Cells*. The former association means that a *Procedure* may load the value of a set of *Cell*, whereas the latter one indicates that a *Procedure* potentially changes the value of a group of *Cells*. Specific types of *Shapes*, like ComboBoxes and Labels, may have a group of Cells as data source. This relationship is represented by the *reads* association that may exist between a *Shape* and zero or more *Cell* entities.

The model also reports some relationships concerning the event handling.

An *Event-Handler* is a specific type of *Sub* procedure. Each *Event-Handler* is triggered whenever a specific *event* is fired on its *Event-Handled Element*. The *Event-Handled Element* class generalizes the set of entities that may have a registered *Event-Handler*, such as *Workbook*, *Worksheet*, *UserForm* and *Shape*. Eventually the model reports some data dependencies between spreadsheet entities.

The *Data Dependency* association class between *Cells* indicates an inter-cells dependency that may be specialized in three different classes: *Formula*, *Data Validation*, or *VBA Code*.

A *Formula* dependency exists between a cell C and n ranges of cells $(R_1, \ldots, R_n)$, if C contains an Excel formula having $(R_1, \ldots, R_n)$ as operands. There is a *Data Validation* dependency relationship between a cell C and a range of cells R, if C has a Data Validation rule of type *List* having R as operand. A *VBA Code* dependency relationship exists between a cell C_i and a cell C_j if a VBA procedure P_i writes in C_i a value based on the computation of the value contained in the cell C_j.

4.3.3 The EXACT Tool

The reverse engineering tool EXACT[4] has been designed to analyze a running VBA-based Excel application and to recover an instance of the Conceptual Data Model presented in Section 4.3.2. Moreover, the tool exposes the information contained in this model by means of several rich and interactive views about the subject application that describe it at different levels of detail.

In order to provide the end user with an intuitive and simplified access to the tool functionality, EXACT has been implemented as an Excel add-in component that extends the native Excel interface by a ribbon menu composed by several buttons. Each button can be used to generate a

[4] *http://github.com/reverse-unina/EXACT*

specific type of view about the application. The tool has been implemented using C# technology. In the following, I show further details about the analysis implemented by the tool and the views it provides.

The reverse engineering technique

The analysis technique implemented by EXACT is based on three steps that are performed while the application is in execution in the Excel environment. Fig. 4.13 reports, for each one of these steps, the artifacts required as input and the ones produced as output. In the first step, named *Structural Analysis*, the Excel Object Model of the *Running Excel Application* is analyzed in order to get information about its structure. This step is performed exploiting the APIs offered by the *PIAs* library. The extracted information is used to reconstruct a partial instance of the conceptual model shown in Fig. 4.11. This instance will be stored into an XML file. Each node of this file represents instance of a retrieved element of the conceptual model with its properties. Moreover, the VBA code embedded in the application is extracted and maintained by the tool.

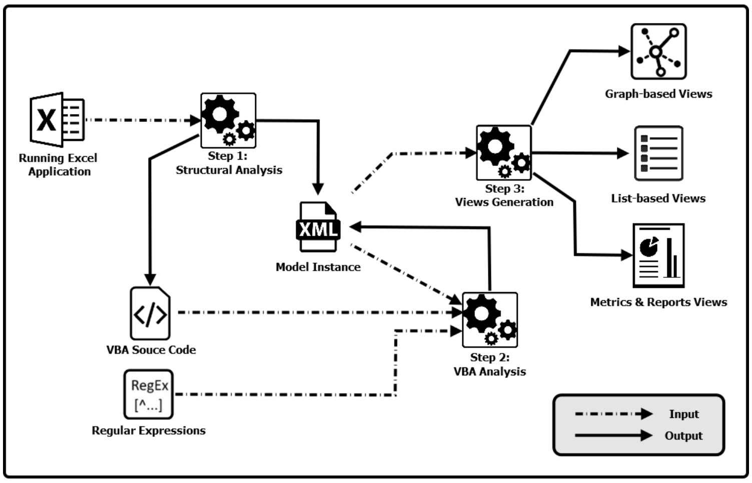


Figure 4.13. The Reverse Engineering Process

The subsequent *VBA analysis* step is performed to refine the partial model instance obtained as output of the previous step. An island parsing [145] of the VBA source code is executed, i.e., only source code fragments of interest are parsed [146]. In this step, the information of the partial model instance is also exploited. The parser is structured in several procedures aimed at recovering further elements and relationships. These procedures were designed by taking into account peculiar Excel and VBA characteristics. As an example, the pseudo code shown in Algorithm 1 describes one of the procedures I implemented for inferring the out-coming relationships starting from the *Procedure* entities. The algorithm exploits an instance of the `ExcelModel` class that provides the methods for creating, querying, updating and persisting the model instance.

The algorithm retrieves all the *CodeModule* objects belonging to the model instance. All the procedures composing the retrieved code modules are then analyzed. In particular, the tool applies specific string pattern matching rules on the signature of the analyzed procedures for comprehending if they are event handlers of a GUI element. As an example, the `Procedure.signature.equals(GuiElementName_EventName)` rule is one of them. Other string pattern matching rules are used to identify potential procedure call relationships. Moreover, specific regular expressions are aimed at identifying if the analyzed procedure *shows*/*hides*/*unloads* User Forms, *reads* data from cells or *writes* data in cells. All the retrieved relationships are used to refine the model instance. To this aim, the `update` methods provided by the `ExcelModel` class are exploited.

Algorithm 1 Algorithm inferring dependency relationships out-coming from Procedure entities

```
$CodeModules[] \leftarrow excelModel.getCodeModules();$
for all $CodeModule\ c \in CodeModules[]$ do
  $Procedures[] \leftarrow c.getProcedures();$
  for all $Procedure\ p \in Procedures[]$ do

    {(1) check if p is an event handler}
    $eventHandler \leftarrow excelModel.checkEventHandler(p);$

    {(2) find the procedures called by p}
    $calledProcedures[] \leftarrow findCalledProcedures(p);$

    {(3) find the user forms shown, hidden and unloaded by p}
    $shownUserForms[] \leftarrow executeRegExp(p,$ "([']*)$.Show$(['])*)[s|(]?");
    $hiddenUserForms[] \leftarrow executeRegExp(p,$ "([']*)$.Hide$([']*)[s|(]?");
    $unloadedUserForms[] \leftarrow executeRegExp(p,$ "([']*)$.Unload$([']*)[s|(]?");

    {(4) find the cells read by p}
    $readCells[] \leftarrow executeRegExp(p,$ "([']*)$.Cells$([']*)[s|(]?");

    {(5) find the cells written by p}
    $writtenCells[] \leftarrow executeRegExp(p,$ "([']*)$.Cells$([']*)[s|(]?");

    {Update the Model Instance}
    $excelModel.updateEventHandler(p, eventHandler);$
    $excelModel.updateCallsRelationships(p, calledProcedures[]);$
    $excelModel.updateShowsRelationships(p, shownUserForms[]);$
    $excelModel.updateHidesRelationships(p, hiddenUserForms[]);$
    $excelModel.updateUnloadsRelationships(p, unloadedUserForms[]);$
    $excelModel.updateReadsRelationships(p, readCells[]);$
    $excelModel.updateWritesRelationships(p, writtenCells[]);$
  end for
end for
```

The *Views Generation* is the last process step, where the current model instance is queried to abstract and to produce several types of view about the subject application.

Views produced by EXACT

EXACT generates high-level interactive and interconnected views about the Excel application in execution. These views can be browsed to obtain information about the application at different levels of detail. The tool exposes both graph-based and list-based views.

The graph-based views may result familiar to Excel end users, since they are very similar to the ones provided by the Excel environment for representing inter-cell dependencies (e.g. *Excel Inquire*[5]). These views are rendered in Microsoft Windows Forms exploiting the *NodeXL Class Libraries*[6] for the graph generation. The list-based ones are intended to provide an enumeration of the elements of the application that can be further analyzed by the end user. They are rendered through Microsoft Custom Task Panes. A *Navigation* feature was implemented in order to allow the end users to easily navigate between different interconnected views. Moreover, EXACT offers a *Flying* feature that allows to navigate between an element of the view and the corresponding part of the Excel application. In the following I present the provided views with reference to a real Excel Application containing VBA code. This application is named *GolfTeeOffForm*[7] and handles the booking of a golf course.

[5] *https://support.office.com/en-us/article /What-you-can-do-with-Spreadsheet-Inquire-ebaf3d62-2af5-4cb1-af7d-e958cc5fad42*

[6] *http://nodexl.codeplex.com*

[7] *http://www.contextures.com/GolfTeeOffForm.zip*

Graph-based Views

EXACT produces four different types of graph-based views named Structural View, Relationships View, Event Handling Graph View and Cell Dependencies Graph View, respectively. Each type of view focuses on different aspects of the application.

Structural View. This view is intended to provide an overview description of how the application is organized. It renders the analyzed application as a tree whose nodes represent its components (and each different type of component has a different graphical icon as reported in the legend of Fig. 4.14), while edges indicate containment relationships between components. Its root node represents the Excel Workbook.

The view can be used to support systematic comprehension strategies, typically adopted by end users who do not have any knowledge about the subject application. Fig. 4.14 reports the Structural View recovered from the *GolfTeeOffForm* application.

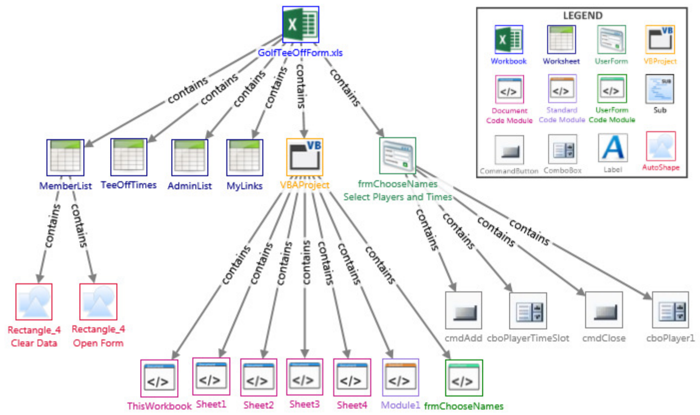


Figure 4.14. GolfTeeOffForm Structural View

Relationships View. It is a graph showing, for a selected component of the application, the other components it is related with. The types of edge reported in the view coincide with the relationships included in the Conceptual Model presented in Section 4.3.2. In particular, each edge reports the role name of the relationship it represents. On the other hand, the relationships between an element and the procedures handling its events are represented by black edges reporting the event name.

The aim of this view is to support 'in-depth' comprehension of how a given component of the application relates with other components, which are potentially scattered in many different parts of the spreadsheet application. As an example, if an end user is interested in understanding additional details about the *frmChooseNames* User Form, he can exploit its related *Relationships View*, shown in Figure 4.15. The view allows the end user to understand the procedures responsible to: (1) show the User Form or (2) to hide it and (3) possibly the procedures calling them.

This view presents an edge labeled *shows* from the *ShowPlayerForm* procedure towards the *frmChooseNames* User Form, indicating that the User Form is rendered when the *ShowPlayerForm* procedure is executed. The edge labeled *onClick* toward the ShowPlayerForm represents that this procedure handles the *Click* event related to the Rectangle_4 (*Open Form*) Shape contained in the *MemberList* Worksheet. The view also shows that ShowPlayerForm is called by the *Workbook_Open* procedure that handles the *Open* event of the Workbook.

Eventually, the view describes that the User Form is closed by the *cmdClose_Click* procedure that handles the *Click* event related to the *CmdClose* Button.

The same type of view also provides further details about the graphical elements composing the GUI. If the selected graphical element is, as an example, a ComboBox, the related *Relationships View* will show the handled events along with the respective event handler procedures and possibly the range of cells used to fill the items of the ComboBox. For instance, Figure 4.16 shows the *Relationships View* about the *cboPlayer1*

ComboBox (reported in Figure 4.16a) contained in the *frmChooseNames* User Form.

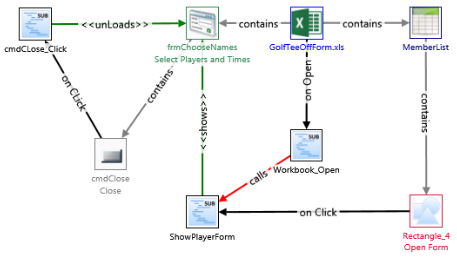
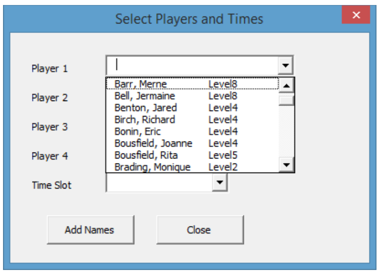


Figure 4.15. frmChooseNames Relationships View


Select Players and Times

Player 1

Player 2

Player 3

Player 4

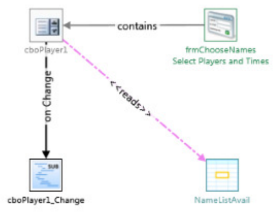
Time Slot

Barr, Merne	Level8
Bell, Jermaine	Level8
Benton, Jared	Level4
Birch, Richard	Level4
Bonin, Eric	Level4
Bousfield, Joanne	Level4
Bousfield, Rita	Level5
Brading, Monique	Level2

Add Names

Close

(a) cboPlayer1 ComboBox

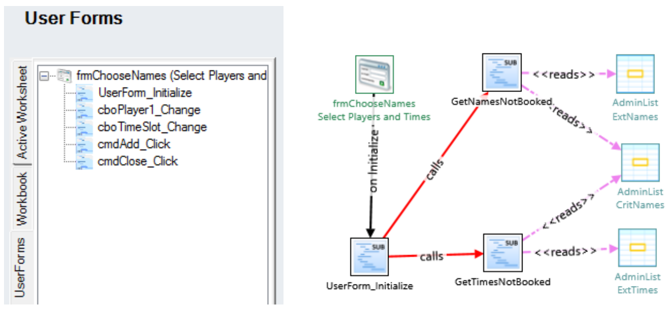


(b) cboPlayer1 Dependencies View

Figure 4.16. cboPlayer1 ComboBox Analysis

Event Handling Graph View. This view is useful for the comprehension of event-based GUIs since it provides a link between the GUI elements and the code implementing the business logic of the application, as well as a link between the business logic and the involved data [147]. It is a direct graph whose nodes represent either (1) the graphical objects which the event handler is attached to through an event handler, or (2) the call graph of the procedures called by the event handler, or (3) the data cells read or written by each procedure, and hence potentially involved in the event handler execution. The edges represent possible relationships between the nodes, according to the model of VBA-based relationships reported in Figure 4.12 . This view is accessible through the *Event Handlers List View* (that will be presented hereafter).

As an example, Figure 4.17b reports the *Event Handling Graph View* related to the *userForm_Initialize* event handler. This view shows that the event handler is called each time the *Initialize* event is triggered on the *frmChooseNames* User Form. Moreover, the graph displays that the event handler calls two other procedures, each one having two *reads* dependencies with different cell ranges.



(a) User Forms Event Handlers List View **(b)** frmChooseNames_Initialize Event Handling Graph View

Figure 4.17. User Forms Event Handlers List and Event Handling Graph Views

Cell Dependencies Graph View. This view is a direct graph showing inter-cells dependencies. Comprehending how cells are related to each other is a well-known issue in the field of spreadsheets and it could be considered as a classical comprehension problem where end users ask questions like *"How did that data get into that field?"* [147]. Cell dependencies are recovered by exploiting both features offered by the *PIAs* library and appositely crafted parsing features for the analysis of the VBA code. The nodes of the graph represent either single cells or ranges of cells. The edges describe the existing relationships among the nodes. Edges may be labeled in three different ways. A *Formula* dependency existing between cells is represented as an edge reporting the formula name. As an example, the view reported in Figure 4.18a illustrates that the cell *J2* belonging to the *TeeOffTimes* worksheet depends on the cells *B2*, *D2*, *F2* and *H2* of the same worksheet by means of the formula *COUNTA*.

A *data validation* stereotyped edge represents a *Data Validation* dependency between cells. For instance, thanks to the *Cell Dependencies Graph View* reported in Figure 4.18b it is possible to understand that the cell *J1* of the *MemberList* worksheet depends on the range of cells named *TimeList*. Moreover, by clicking on the node representing the TimeList cells range, EXACT reports that it corresponds with the cells *A2:A32* belonging to the *TeeOffTimes* worksheet.

The *VBA Code* relationships between two cells are represented by edges labeled as *VBA*. The cells involved in a VBA Code dependency may be precisely identified by the tool when the VBA procedure contains an explicit cell reference. In other cases, when the code presents implicit or dynamically defined cell references, the cells cannot be identified by static analysis. In the latter case, EXACT just gives the pointers to the involved code that can be further analyzed by the end user through the *VBA Code Dependency Highlighting* feature. Thanks to this feature, by clicking on a node representing a cell involved in a VBA Cell Dependency, a view that highlights the part of code that allows the abstraction of the dependency is reported.

List-based Views.

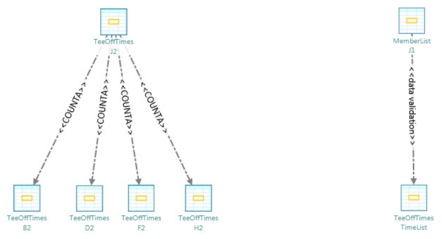
EXACT provides two different List-based views, called *Event Handlers List View* and *Cell Dependencies List View*, respectively. The former view lists all the event handlers defined through the VBA code whereas the second one lists all the dependencies that may exist between cells ranges.

Event Handlers List View. Since the GUIs of Excel applications are event-based systems where event handlers may be attached to each graphical object of the GUI, a possible GUI comprehension strategy requires the end users to interact with each widget for firing events on them and to observe the resulting behavior of the application. The proposed *Event Handlers List View* aims at simplifying this process, by providing a direct access to event handlers that may be attached to different items of the application. This list-based view is organized into three different tabs. The *Workbook* tab shows all the event handlers included in the Workbook. The *Active Worksheet* tab displays the event handlers attached to the worksheet being analyzed at run-time and the GUI elements it contains. In the *User Forms* tab the event handlers that may be attached to the user forms of the application and their GUI elements are listed. Figure 4.17a lists the event handlers related to the *frmChooseNames* User Form. This view is interactive and by selecting one of the listed event handlers, the related *Event Handling Graph View* is rendered.

Cell Dependencies List View. This view enumerates all the dependencies between cells grouped into three different tabs. The *Formula* tab reports all the *Formula dependencies*. The second tab, named *Data Validation*, lists all the *Data Validation dependencies* given by data validation rules. The *VBA Code* tab reports all *VBA Code dependencies*. Once the end user selects one of the listed cell-dependencies, the related *Cell-Dependencies Graph View* is rendered.

Metrics & Reports View.

Finally, EXACT provides a view reporting eleven different metrics about the analyzed spreadsheet application, related either to its structural components (such as # Worksheets, # User Forms, # Procedures, # Event Handlers, # Used Cells, # Formula Cells, etc.), and to its VBA code, such as VBA LOCs, CLOCs and the counts of the different types of Procedures. This view also provides features to generate graphical and



(a) Formulas Dependencies

(b) Data Validation Dependencies

Figure 4.18. Different types of Cell Dependencies

textual reports (in html format) listing summary data and including the views offered by the tool.

4.4 Evaluation of the Proposed Solutions

In order to evaluate the effectiveness of the proposed solutions, namely the reverse engineering process for recovering Data Models and the EXACT tool, I carried out different case studies in the automotive domain. The conducted case studies are detailed in the following.

4.4.1 Evaluation of the Reverse Engineering Process

To analyze the effectiveness of the proposed process, I performed a series of case studies involving different spreadsheet based artifacts corpora from the same industrial domain. The goal of these case studies was to evaluate the applicability of the process and the acceptability and precision of the inferred models. In the first case study, I used the process to reverse engineer the conceptual data model from a legacy Spreadsheet based artifacts corpus implemented in Microsoft Excel. This system was used by the Hardware-In-the-Loop (HIL) Validation Team of FCA and provided strategic support for the definition of high-level testing specifications, named Test Patterns. Test Patterns represent essential artifacts in the overall testing process [148] since they allow the automatic generation of test cases, the Test Objects, necessary to exercise the ECUs of automotive vehicles. The generation process is carried out thanks to a lookup table that embeds both the translation of high-level operations to the corresponding set of low-level instructions, and the mapping between input and output data to the set of ECU pins. Test Patterns were implemented by means of a predefined Excel template file organized into 7 worksheets, referred to different phases of the testing process.

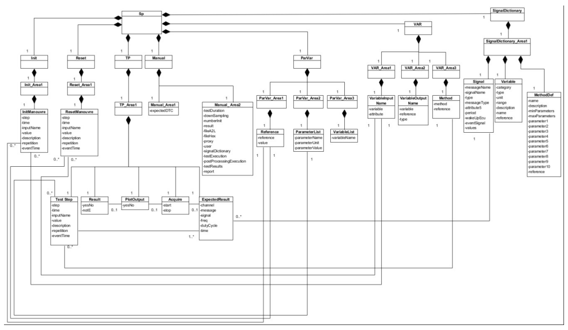


Figure 4.19. Inferred Conceptual UML class diagram for the Test Pattern Information System

The worksheets embedded the model of the data. In the considered context, such template has been adopted to instantiate a set of 30,615 different artifacts, constituting the overall spreadsheet-based corpus. Each spreadsheet contained on average 2,700 data cells. All of these files inevitably shared the same structure (i.e., the data model) with a high rate of replicated data (about 20% of data cells recurred more than 1,300 times, while about the 50% more than 100 times). This high rate of replication was mainly due to the fact that data were scattered among multiple spreadsheets. Therefore, the underlying model was not normalized, with any information about related data. In addition, the automatic verification of data consistency and correctness was not natively supported. At the end of the process execution I obtained a UML class diagram composed of 36 classes, 35 composition relationships and 23 association relationships. Seven of these classes are associated with the worksheets composing each spreadsheet file, while the remaining ones are related to the areas and subareas of each worksheet. Figure 4.19 shows the conceptual data model class diagram that was automatically inferred by executing the process. For readability reasons, I have not reported all the association relationships and the associations multiplicities, whereas I reported all the classes and the composition relationships that were inferred. Figure 4.20 shows an example of rules execution on the Test Pattern sheet that is the most complex one. By executing the rules, I was able to infer five classes, 4 association relationships and 5 composition relationships. Moreover, it shows how the Rule 4 was able to infer the two classes Test Step and Expected Results and Rule 6 proposed the association relationship between them.

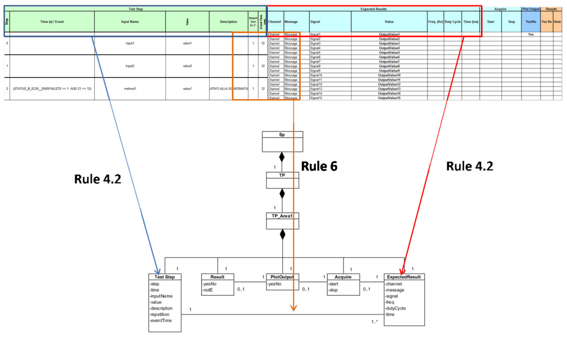


Figure 4.20. Example of class proposal and its mapping with the spreadsheet file

In the second and third case study, I analyzed two further systems used by the company. The systems included a Software Factory Test Case (called SoftWare Factory Test Cases (SWFTC)) repository and a

KPI repository (hereafter KPI). The SWFTC repository contains essential artifacts of MIL andSoftware-In-the-Loop (SIL) testing process in the considered company, since they allow the automatic generation of executable MATLAB test scripts necessary to validate both the models and the source code of the software components that will be deployed on the ECUs. This information system is composed by 14,000 Excel files inheriting from a common template composed by 10 sheets. Each spreadsheet contained 4,000 data cells on average. About the 75% of data cells are replicated in the spreadsheets. KPI repository is a spreadsheet-based information system used to manage the key performance indicators of each development project regarding a software component belonging to a specific type of vehicle. The information system is composed by 1,500 Excel files inheriting from a common template composed by 8 sheets. These spreadsheets contained 1,370 data cells on average. About the 77% of data cells are replicated in the spreadsheets.

To support the process execution a prototype software application was developed. It is implemented in C# and takes as input the location of the spreadsheets composing the information system under analysis. It accesses to the underlying Excel object model of each spreadsheet in order to extract the raw data that are needed for the automatic execution of the steps, i.e., spreadsheet's structure, data content, properties of cells, etc.. The tool is able to provide as output an XMI file representing the inferred Class Diagram. To implement the Step 7, the prototype exploits some features offered by the Data Quality tool Talend Open Studio[8].

Table 4.1 shows the results of the conceptual data model reverse engineering process related to the three case studies. It reports the number of classes and relationships that were automatically inferred from each information system. After the analysis, I performed a validation step in order to assess the acceptability and the precision of the inferred models.

[8] *https://www.talend.com/products/talend-open-studio/*

Table 4.1. Reverse Engineering Results

Information System	**# Classes**	**# Association Relationships**	**# Composition Relationships**
SW TC	49	33	48
Test Pattern	36	23	35
KPI	25	12	24

To this aim I enrolled three experts from the company belonging to the application domains of the three information systems.

I submitted them: (1) the inferred data models, (2) a report containing the description of each inferred UML item and one or more traceability links towards the spreadsheet's parts from which it was extracted, (3) a set of questions in order to collect the expert judgments about the validity of the UML item proposals. I asked the experts to answer the questions and thus I was able to assess the effectiveness of the overall reverse engineering process by means of the Precision metric reported below:

$Precision = \frac{|V.I.E.|}{|I.E.|} * 100$

$I.E.$ is the set of the UML elements, i.e., classes, class names, class attributes, relationships between classes and multiplicities of the relationships that were inferred by the process. $|I.E.|$ is the cardinality of the $I.E.$ set.

$V.I.E.$ is the number of the inferred UML elements that were validated by the industrial expert and $|V.I.E.|$ is the cardinality of this set.

Table 4.2 shows the precision values I obtained for each information system. The precision values reported showed that more than 80% of the inferred elements were validated by the experts. In the remaining cases, the experts proposed some minor changes to the candidate classes. As an example, with respect to the first system, the Test Patterns one, the expert decided to candidate additional classes by extracting their attributes from the ones of another candidate class, or to merge some of the proposed

Table 4.2. Evaluation Results

Information System	**Precision (%)**
SW TC	81
Test Pattern	84
KPI	92

classes into a single one.

Lessons Learned

I analyzed in depth the results of the validation steps in order to assess the applicability of the rules used throughout the process. In this way, I was able to learn some lessons about the effectiveness of the heuristic rules. In particular, with respect to the class diagram reported in Figure12, I observed that the expert proposed (1) to discard the PlotOutput class and to move its attributes into the ExpectedResult one and (2) to extract a new class, named Repetition, having the attributes repetition and eventTime that were given from the candidate class named Test Step. Similar changes were proposed also in the other two case studies I performed. I observed that: (1) when the expert decided to merge two candidate classes into a single one, these classes had been considered as different by Rule 4.2, since they derived from two subareas having headers with different colors but actually belonging to the same concept. Whereas, (2) when the expert decided to extract an additional class from a candidate one and assigned it a subset of the attributes of the inferred one the Rule 4.2, Rule 5 and Rule 6 were not able to identify this extra class since the cells associated to these two concepts had the same formatting and layout style. As a consequence, the proposed process was not able to discriminate between them. In both cases the process failed because the spreadsheet did not comply with the formatting rules exploited by the process. I studied in detail the cases in which the experts proposed the changes in order to understand if it was possible to introduce new heuristics. In the case (1) I was not able to find any common property between cells belonging to the classes that were proposed to be merged by the expert, as a consequence no new heuristic was introduced. In case (2) I observed that the expert proposed to extract the columns/rows of a candidate class into a new one when they presented a high level of data replication percentage (>80%). On the basis of this observation I proposed the following heuristic rule that can be applied as a further Step of the proposed process, named Step 8. In this step, the Rule 8 is exploited. This heuristic is applied to the overall corpus of spreadsheets. It analyzes the data contained in the columns/rows belonging to the classes that were inferred at the end of Step 7. If two or more columns/rows related to a given class Ci have a data replication percentage that is higher than 80% then a new class is extracted from the original one. The attributes of the extracted class have the same name of the considered columns. The extracted class is named as follows: $NameofCiExtracted_i$. A relationship association is inferred between the classes. The multiplicity of the association on the side related to the class Ci is equal to 1 whereas the one on the other side is 1..*. The data quality tool Talend was employed to implement this heuristic. As an example, by applying the Step 8 on the Test Step class I was able to obtain a result similar to the one proposed by the expert. In detail, the Rule automatically extracted a class, named $TestStep\_Extracted_1$, having the attributes related to the repetition and eventTime columns. The new class was inferred since the level of data replication of the two considered columns was higher than 80%. Figure 4.21 shows one of the data analysis views provided by Talend. The histogram reports the results of the analysis about the data cells belonging to the columns repetition and eventTime. By applying the formula reported below it is possible to observe that in this case the percentage of data replication is equal to 93,8%.

$$DataReplicationPercentage = \frac{RowCount - DistinctCount}{RowCount} * 100$$

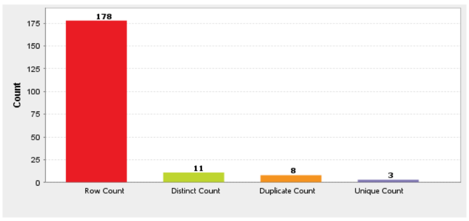


Figure 4.21. Example of a histogram reporting a data replication analysis on the class Test Step of an example spreadsheet

Figure 4.22 shows an example of the application of the Step 8 on the candidate class Test Step. To confirm the validity of the Rule 8, I applied the Step 8 to the models that were inferred by the previous process and measured the precision of the resulting class diagrams. The results reported in Table 4.3 show that by applying this step the effectiveness of the whole process increases. As to the rules I used to associate the candidate classes with a name, only in 21 over 110 cases they failed, since the spreadsheets did not include meaningful information to be exploited for this aim. Furthermore, I observed that in some cases the expert decided to discard some of the proposed composition relationships between the inferred classes. This fact occurred when the proposed conceptual class diagram presented a particular pattern, as reported in Figure 4.23. In this case, the expert decided to move the attributes of the leaf class ($Sheet\_SubArea_1$) to the Sheet1 class and to remove the remaining classes. This specific pattern occurred 8 times and in 6 cases the expert decided to make these changes. This result showed the need to introduce new rules aimed at reducing the complexity of the class diagram that may be used in the occurrence of this particular pattern.

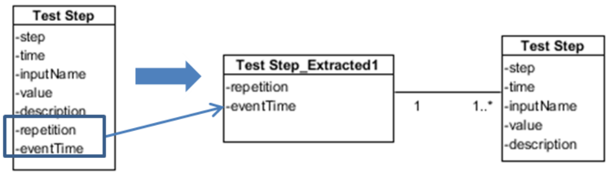


Figure 4.22. Example of Step 8 execution

Table 4.3. New Process Results

Information System	**Precision (%)**
SW TC	90
Test Pattern	92
KPI	95

4.4.2 EXACT Tool Evaluation

To validate the proposed reverse engineering tool, I decided to assess its usefulness in aiding the comprehension of real VBA-based Excel spreadsheet applications. To this aim, I performed a qualitative case study [149] in a real automotive industrial context, with the support of members of my research group. More in detail, this study was carried out in a Unit of an automotive company having approximately one hundred employees devoted to the development of Electrical and Electronic systems. In this context, several VBA-based spreadsheet applications are used to support different phases of the model-based embedded software development process. The study was carried out to answer the following research questions:

RQ$_1$ How does EXACT support professional end users to comprehend VBA-based spreadsheet applications?

RQ$_2$ What are the main limitations of EXACT according to the end-users'

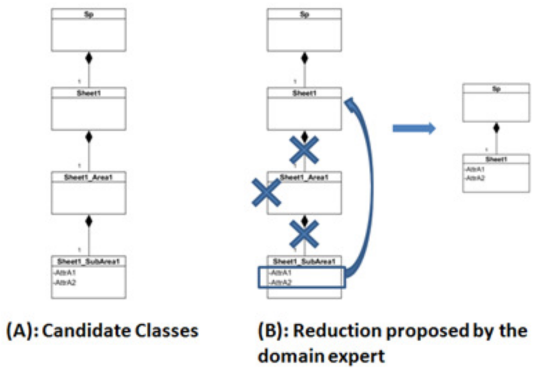


Figure 4.23. Example of Class Diagram reduction proposed by the domain expert

point of view?

The proposed RQ_1 will guide in comprehending whether or not EXACT can be considered a valuable support for professional end users in the correct accomplishment of comprehension tasks involving real VBA-based spreadsheet applications. On the other hand, the results of RQ_2 will help to understand the aspects of EXACT that do not work well and need to be improved.

Study design

Comprehension is not an end goal, but it is a step needed to carry out different tasks: maintenance, software inspection, quality evaluation and testing [141]. According to this statement, we decided to evaluate the use of EXACT in supporting maintenance activities whose execution needs the comprehension of spreadsheet applications. Moreover, we intended to explicitly assess the correctness of the comprehension results obtained with the support of the tool.

Objects.

As objects of the study we selected real maintenance projects from the ones of interest for the company and that met the following criteria:

- the project had to involve a nontrivial VBA-based spreadsheet application;

- the project had not yet started and no staff had been assigned to it.

On the basis of these criteria we selected three real projects.

Table 4.4 reports for each selected project a brief description and the involved application.

The applications involved in the selected projects had been used in the company for years and underwent several maintenance interventions. They were mainly used as data storage and they provided several features of data entry, data analysis and data validation. Table 4.5 shows some complexity metrics about the applications involved in the selected projects. These applications were nontrivial, since they presented a rich and complex GUI and included a relevant part of VBA code composed by more than one thousand lines of code. Moreover, the applications were hard to be analyzed and comprehended because they did not present external documentation and their internal documentation was inadequate. For all the applications, the percentage of CLOC (Comment Lines Of Code) over the number of LOC (Lines Of Code) was lower than 3%, as evidenced by Table 4.5. Many of the GUI widgets implemented in VBA code did not present a meaningful name, since they had been automatically generated by the VBA Editor. For each application the percentage of Shapes having default names was 38%, 67% and 54%, respectively.

Table 4.4. Selected Industrial Projects

Project ID	**Involved Application**	**Project Description**
P1	**A1**	*Reverse Engineering*: this project involved the migration of the functionalities provided by **A1** towards an MVC-based Web Application. The main objective of this project was the reverse engineering of the business logic and the user functionality exposed by the application.
P2	**A2**	*Re-Engineering*: this project required the execution of a re-engineering process involving **A2**. The application had to be re-engineered towards a Web-based architecture. Both the structure of the data contained in the spreadsheet and the functionalities and the user interfaces defined through the use of VBA code needed to be comprehended.
P3	**A3**	*Re-Documentation*: This project was carried out with the aim of re-documenting **A3**. This application was completely undocumented since no description of the functionality it offers was available in the company. The main aim of this project was to produce a User Manual listing all the functionalities it exposes.

Subjects.

To conduct the case study, we recruited company employees. We considered employees having different roles in the company. The inclusion criteria we defined to select the subjects are described below.

- Each subject had at least one year of experience using and designing Excel applications.
- Each subject had been involved in more than one project related to VBA-based spreadsheet applications.

Table 4.5. Metrics about the Applications

Metrics	***A1***	***A2***	***A3***
# Worksheets	7	10	8
# User Forms	16	6	6
# Shapes	279	54	26
# Default Named Shapes	105	36	14
# Used Cells	17902	45298	5210
# Formula Cells	0	2107	508
# Code Modules	29	34	16
# Procedures	164	72	34
# Event Handlers	87	35	21
# LOC	8020	4168	1320
# CLOC	80	97	12

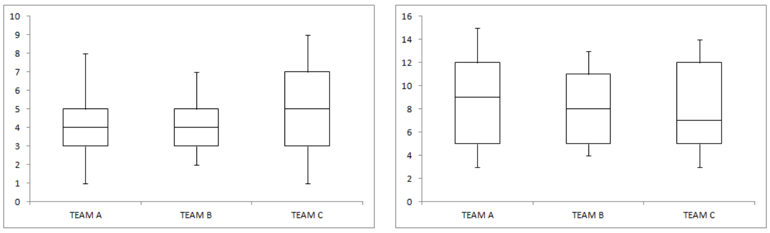
- Each subject had not to be familiar with any of the applications A1, A2, and A3.

We selected 15 electronic engineers who agreed to participate in the case study and who met the defined inclusion criteria. The subjects had different levels of expertise on the use of spreadsheet applications: on average a participant had 4 years of expertise working on 8 projects. None of the recruited subjects had skills in software engineering.

The participants were grouped in 3 different teams (named Team A, Team B and Team C) composed of five participants. Each team mixed employees from several departments having different years of experience and number of projects in which they had been previously involved.

Box-plot diagrams related to the years of experience and the number of the projects are reported in Figure 4.24. As the diagrams show, we distributed participants in order to guarantee teams with similar distributions.

Moreover, for each subject application we recruited in the company



(a) Years of experience **(b)** Number of projects

Figure 4.24. Teams Statistics

an end user who had been involved in the maintenance of the application in the past and had a deep knowledge about it. Each end user could be considered as an expert of one of these applications.

Case Study Procedure.

I report the procedure we followed to carry out the case study. It was performed in five different steps.

As *first step*, we assigned each project to one of the teams: P1, P2 and P3 were assigned to Team A, Team B and Team C, respectively. We did not impose the teams any time limitation to accomplish the projects. However, we recommended them to follow the ordinary company standards in the execution of their projects. Moreover, we did not impose them any specific work assignment or to apply any particular comprehension process or strategy. This choice is coherent with the work presented in [150] that shows how each professional follows its own process to comprehend an application. We just asked them to organize their work and to use EXACT as best as they considered. Each team planned the tasks needed for accomplishing its project goal. According to the team organization, one or more tasks were assigned to each participant that executed them individually.

In the *second step* one of the authors explained to all the participants how the features provided by EXACT can be used. To this aim, a four-hours training session was performed. More in detail, the researcher, expert in the use of EXACT, explained through examples how to use the different features offered by the tool. During the training, the *GolfTeeOff-Form* application was exploited for showing the use of EXACT. After the training, each team carried out its assigned project.

When the teams accomplished the projects, we performed a *data collection step.* We conducted a semi-structured interview with each participant to gather information about the use of EXACT. The questions regarding the use of EXACT are reported in Table 4.6.

Table 4.6. Questions

Q1	Which comprehension tasks did you perform to carry out the assigned project?
Q2	For each comprehension task executed exploiting EXACT, which results did you obtain?
Q3	For the execution of each comprehension task, how did you use EXACT?
Q4	Which views provided by EXACT did you exploit for the accomplishment of each comprehension task ?
Q5	How do you rate the views that were exploited for the accomplishment of each comprehension task ?
Q6	Did you face difficulties in the use of the views provided by EXACT? Explain these difficulties, if any.
Q7	What limitations of EXACT did you observe in the accomplishment of the tasks?

These semi-structured interviews guided us to obtain the evidences that allowed us to answer the research questions. Since in the study we did not ask to execute a specific comprehension process, the answers to *Q1* and *Q2* allowed us to discover which comprehension tasks were executed by each participant and which results they achieved. *Q3*, *Q4* and *Q5* were intended to understand how EXACT supported the end users in the execution of the comprehension tasks. *Q6* and *Q7* gave us information about the difficulties faced by the participant in using EXACT and its main drawbacks.

In the fourth step, a *data analysis* was performed. In order to improve the reliability of the study, two different researchers analyzed the collected data independently. They coded the interviews in order to gain the relevant information and to analyze them. The results of the analysis were compared and no significant anomalies were evident. Moreover, to avoid misinterpretations, the results of the analysis were also reviewed by the participants.

Finally, in order to evaluate the correctness of the comprehension results, an *Assessment step* was performed. In this step, we submitted to the experts of the subject applications the answers given to *Q2* by the professional end users. We asked them to validate the comprehension results and to rate them according to the following discrete scale: 1 if the result was judged as **incorrect**, 2 if the result was considered as **partially correct**, and 3 if the result was considered as **correct**. When the same task was performed more time in the same project, we evaluated the average of the expert judgments.

Findings

In this section I report the findings of the study related to: (1) the comprehension tasks executed with the tool, (2) the judgment about the correctness of the comprehension results given by the experts, (3) the features and views of EXACT exploited by the subjects, (4) the judgments related to the features offered by the tool given by the end users and (5) its main limitations. Table 4.7 reports, for each project, the comprehension tasks that were executed with the support of the tool. We considered that a comprehension task was executed in a project if it at least one of the team participants performed it with the support of EXACT. Table

4.8 summarizes, for each comprehension task, the average of the expert judgments about the correctness of the comprehension results. Moreover, an additional finding of the study regarded the comprehension strategies followed by the participants using the tool. These strategies were collected and described according to the classifications reported in [147, 151].

P1 - Reverse Engineering Project.

As reported in Table 4.7, the team A performed overall 12 different comprehension tasks for accomplishing *P1*. Moreover, the data reported in Table 4.8 indicate that 10 of the comprehension results were judged as correct by the expert and the remaining ones were considered as partially correct. More in detail, the result of the execution of the *T14* comprehension task was considered as not completely correct by the expert since some dependencies of a specific cell range were not found by the participants. In particular, several dependencies defined by means of VBA procedures that needed a manual investigation were not correctly comprehended by the participants. Similarly, the *T15* comprehension task was also considered not completely correct. In this case, the participants were not able to identify some of the cells involved in one of the application features.

Four participants used the *Structural View* as starting point of their comprehension process. Thanks to this view they easily succeeded in understanding the general structure of the application and its composing elements. Three of them needed to deeply investigate the relationships among the elements of the application. To this aim, they exploited the *Navigation feature* for rendering the *Relationships View* of each composing element. The analysis of these views allowed them to understand how the elements of the application were interconnected. The fifth subject followed a different approach starting the comprehension process by exploiting the *Event Handlers List View*. Thanks to this view he was able to gather a list of all the event handlers defined on each Worksheet composing the appli-

Table 4.7. Executed Comprehension Tasks

CT	**Comprehension Task Description**	**P1**	**P2**	**P3**
T1	What are the worksheets composing the application ?	✓	✓	✓
T2	What are the widgets included in a worksheet ?	✓	✓	✓
T3	What are the User Forms included in the application ?	✓	✓	✓
T4	What are the widgets composing a User Form ?	✓	✓	✓
T5	What are the procedures included in the application?	✓		✓
T6	What is the list of the features accessible through the GUI of the application?	✓		✓
T7	What procedures depend on a specific procedure?		✓	
T8	What events are handled by a specific GUI widget?	✓	✓	✓
T9	What procedures may be executed when a specific event is fired on a GUI widget?	✓	✓	
T10	What is the fan-in of a specific procedure?		✓	
T11	What is the fan-out of a specific procedure?		✓	
T12	Which procedures allow the rendering of a specific User Form ?	✓	✓	✓
T13	Which procedures allow the hiding of a specific User Form ?	✓	✓	✓
T14	What are the dependencies of a specific range of cells?	✓	✓	
T15	What are the cells needed in the execution of the features provided by the application?	✓	✓	

Table 4.8. Average correctness judgments *(AJ)* about the comprehension results

CT	**AJ - P1**	**AJ - P2**	**AJ - P3**
T1	3	3	3
T2	3	3	3
T3	3	3	3
T4	3	3	3
T5	3	-	3
T6	3	-	3
T7	-	3	-
T8	3	3	3
T9	3	3	-
T10	-	3	-
T11	-	3	-
T12	3	3	3
T13	3	3	3
T14	2	2	-
T15	2	2	-
3 correct; 2 partially correct; 1 incorrect			

cation. Since the subject was interested in comprehending which elements of the application were involved in the execution of each event handler, he exploited the *Event Handling Graph Views.* He adopted the same approach for comprehending the behavior of the event handlers offered by the User Forms of the application.

All the participants appreciated the *Flying* feature since it allowed them to easily map the elements reported in the exploited views with the corresponding parts of the application without navigating it. As an example, one of the subject reported that *"To better understand the application, the graph-based views were not always enough for me. Sometimes, I needed to know which parts of the application were actually referred by the elements of the graph I was analyzing. Thanks to this feature I was able to rapidly navigate the application by exploiting the graph representing it."* The *Cell Dependencies List View* was exploited by all the subjects involved in *P1* for reconstructing the inter-cell dependencies. Thanks to this view they were able to understand that no formula was used in the application. As a consequence, they used both the *VBA Code* and *Data Validation Cell Dependencies Views.* Three out of five participants also exploited the Metrics and Reports View, but just to review the proposed metrics. No participant exploited the report proposed by EXACT.

According to these finding, I could conclude that all the participants followed a systematic comprehension strategy aiming at achieving an high level comprehension of the application structure.

As to the answers given to *Q5*, all the participants gave EXACT a positive evaluation. Three of them reported that the views they exploited supported at least sufficiently their tasks. The *Structural View* was considered really helpful and its rate was more than sufficient on average. This view made them aware of some aspects of the application not immediately visible. One of the participants stated: *"... the Structural View was really helpful. Some of the elements present on the Worksheet were very difficult* *to be identified without the support of this view. In fact, I was able to discover both an hidden Worksheet and that the application had a Shape that was difficult to be identified since it was positioned close to the row* $9,000$ *..."*. The three subjects that exploited the *Event Handling Graph Views* stated that these views guided them in the exploration and in the understanding of the features provided through the GUI of the application. They considered them useful since their use avoided the execution of tedious tasks and allowed them to easily understand the impact of a feature on different parts of the application. Only the support given by the *Cell Dependencies Views* were considered just sufficient, since they were considered too chaotic in some cases.

Different limitations were pointed out by the participants: (1) the lack of a proper search mechanism related to the Cell Dependencies Views and (2) difficulties in reading the *Event Handling Graph Views* related to procedures having several data dependencies.

P2 - Re-Engineering Project.

As Table 4.7 shows, the accomplishment of *P2* required the execution of 13 different comprehension tasks. Moreover, data reported in Table 4.8 indicate that 11 of the comprehension results were judged as correct by the expert and the remaining ones were considered as partially correct. The *T14* and *T15* comprehension tasks were not considered completely correct, according to the expert's judgment. The dependencies of two different cell ranges were not recovered correctly. They were VBA dependencies that needed a further manual investigation. Moreover, the participants were not able to identify all the cells involved in two different features offered by the application.

Three participants started their comprehension process from the Structural View that was exploited to detect the elements of interest. Each element was further analyzed through its *Relationships View* in order to understand its dependencies with the other elements of the application. The other two participants, interested in comprehending the User Forms composing the GUI, started their analysis from the *Event Handlers List View*. In particular, they focused on the User Forms tab (see Figure 4.17a) and used this tab for rendering the *Event Handling Graph Views* of each handler provided by the forms. End users analyzed these views for understanding the dependencies between data cells and procedures that were involved in the event handlers executions. The *Cell Dependencies Views* were used by three subjects who were asked to modify the data structure of the application. They mainly analyzed the ones due to VBA Code. Subjects widely adopted the *VBA Code Dependency Highlighting* feature for better understanding the VBA Code dependencies that were not statically defined. Only two participants exploited the *Metrics and Reports View* to review the complexity of the application.

On the basis of the answers they gave, I could deduce that all the participants followed an *as-needed* comprehension strategy. No subject systematically navigated through all the elements composing the application, but they focused only on the elements involved in their maintenance task.

Analyzing the answers to *Q5* I could state that all the subjects rated more than sufficient the support given by EXACT. The *Event Handling Graph Views* were considered the most helpful views. As an example, one of the subject reported that *"The Event Handlers List View and the Event Handling Graph Views allowed me to explore the features offered by the application in a more targeted and efficient way. I could easily identify the elements potentially involved during the execution of a feature. This allows me to test the feature of interest knowing which part of the application needed to be examined."* . The *Structural View* along with the *Relationships Views* gave end users a more than sufficient support. Also the support given by the Cell Dependencies View was considered sufficient. Although, two subjects though that this kind of view need some improvements. The *Navigation feature* was heavily adopted by all the participants and was considered useful in the accomplishment of their tasks.

Regarding the limitations of EXACT, two participants observed that when a procedure had an high number of dependencies with data cells, the *Relationships Views* and the *Event Handling Graph Views* became quite difficult to be read. One of them remarked: *" ... in these cases the tool needs a feature to group the cell dependencies in some way, in order to make the views more readable".* Three subjects stated that *VBA Code Cell Dependencies Views* should be improved avoiding the need to manually investigate the VBA source code. The analyzed application presented two different *Excel Charts*. EXACT does not have any feature for analyzing the charts, since I did not consider them in the proposed conceptual model. Two participants stated that they needed features for analyzing the charts contained in the application and the data cells involved.

P3 - Re-Documentation Project.

The accomplishment of *P3* needed the execution of 10 different comprehension tasks, as shown in Table 4.7. Moreover, data reported in Table 4.8 indicate that all the comprehension results were judged as correct by the expert. Four participants started this project by preliminarily generating the report provided by the *Metrics and Report View*. They verified if the report could be considered adequate as User Manual of the application. Since the Report lacked some useful information, like screenshots of the user interfaces, dependencies among cells ranges, etc. they needed to exploit other features provided by EXACT. The *Structural View* and the *Relationships Views* were exploited by the participants to comprehend the different parts of the application. Three participants were interested in re-documenting the features offered by the application through its

GUI elements. To this aim the *Event Handling Graph Views* were widely adopted. All the participants exploited the *Flying* feature during the accomplishment of their tasks, since in some cases they needed to capture a screenshot of specific parts of the application related to the views they were analyzing. Four participants exploited the *Cell Dependencies Views* in order to document them into the User Manual. They were asked to describe for each Worksheet of the application: (1) the cells whose values change as the values of other cells vary; (2) the cells that may be filled in only with specific values. To this aim they widely analyzed the *Cell Dependencies Graph Views* related to the Formula and Data Validation dependencies.

Thanks to the analysis of the process followed by the subjects involved in this project, I was able to understand that all the subjects used EXACT according to a *systematic* comprehension strategy.

Regarding the judgment of the features given by the answers to *Q5* and views provided by EXACT, all the participants rated them more than sufficient. The Structural View was the most appreciated one. Also the support given by the *Event Handlers List Views* and *Event Handling Graph Views* and by the *Navigation* and *Flying* features were rated more than sufficient by all the participants.

Although, some limitations arose. Three participants stated that EXACT lacked a mechanism for editing the provided views and updating the generated reports. They also pointed out the need of a feature for selecting the elements of interest and for releasing them into the report. Four participants recognized the need for a screen scraping feature to capture and add screenshots of the application GUI to the report. Two subjects stated that EXACT should provide a global view about the dependencies among cells in order to ease their analysis.

Conclusions

The results of the case study allowed to answer the proposed research questions.

RQ_1: *How does EXACT support professional end users to comprehend VBA-based spreadsheet applications?*

The end users exploited all the features produced by the EXACT tool to approach the comprehension of unknown Excel applications. They choose the views to be used depending on the maintenance task to be accomplished and according to either systematic or ad-hoc comprehension strategies. Overall, almost all the comprehension tasks performed with the support of EXACT were carried out correctly by the end users.

On the basis of the end-users judgments, I could conclude that the Structural View was able to orient them at an initial approach with an unknown spreadsheet, offering an overview and comprehensive description of all the different parts composing the application. On the other side, the list-based views and the associated graph-based views were used to address the specific comprehension issues that affect wide Excel applications, such as the comprehension of the event management offered by the application, or the detection of inter-dependencies between its different parts. In conclusion, the tool avoided the execution of repetitive and tedious tasks to gather information and knowledge about the subject applications.

RQ_2: *What are the main limitations of EXACT?*

The end users did not observe any particular or specific problem concerning the functionality, the views, or the features offered by the EXACT tool. They mostly indicated additional features of the tool they would appreciate. These features included: 1) an advanced search functionality (such as a guided search) in order to better support the cell dependencies analysis; 2) a more comprehensive view showing the different types of dependency relationships; 3) the capability of analyzing further Excel elements (such as Pivot Tables or Excel Charts) not yet addressed by the tool; 4) some additional features for aiding the comprehension of not explicitly defined cell dependency relationships; 5) clustering mechanisms for improving the readability of Event Handling Graph Views, as the the size of these graphs grows.

Threats

Internal Validity. This aspect of validity needs to be evaluated when a causal relationship is examined. To minimize the threat that the obtained results could depend on the characteristics of the involved subjects, we enrolled in the study subjects involved in different areas of the company and having different levels of expertise. Moreover, this threat can be further mitigated by considering a larger group of subjects from different contexts. This aspect may be addressed in future experiments.

External Validity. External validity is related to what extent it is possible to generalize the findings and if these are of interest to people outside the reported case. A possible threat could be the real representativeness of the subjects and objects we choose to conduct the study. However, both the subjects and objects we choose were very similar to others from different contexts described in the literature [140] or in the stories reported by the European Spreadsheet Risks Interest Group (EuSpRiG)[9]. To mitigate this threat, a wider experimentation involving different types of subjects and objects should be carried out.

[9] *http://www.eusprig.org/stories.htm*

Reliability. The reliability is concerned with to what extent the data and analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducted the same study, the result should be the same. To minimize this threat, the case study protocol and the artifact produced was reviewed by a group of researchers of the university. The case procedure we followed was reported, to the extent possible, in this chapter. Furthermore, data collected and their analysis were reviewed by case subjects in order to avoid possible misinterpretations. To increase the realism of the study we used real spreadsheets adopted by our industrial partner that cannot be released due to confidentiality matters. Although similar studies can be conducted on different VBA-based spreadsheet applications.

4.5 Related Work

A considerable number of works in the literature investigate problems related to the comprehension of spreadsheets. Many of them focus on the difficulties of understanding the global structure of a spreadsheet and point out the necessity of evaluating the dependencies among its cells, such as [152]. Other authors identify the need for different visualizations to support spreadsheet comprehension, like Hodnigg et al. [153] who propose the use of complexity measures as indicators to choose the proper visualization.

As to the tools supporting the visualization of the spreadsheet structure, Davis et al. [137] present an arrow tool aimed at supporting the audit of a spreadsheet application representing data dependencies between cells using arrows, Shiozawa et al. [136] propose a three-dimensional visualization of inter-cell dependencies to increase users comprehension of spreadsheets and Breath et al. [154] implemented a three-dimensional visualization for spreadsheet based on the WYSIWYG (*What You See Is*

What You Get) paradigm.

Many contributions address the problem of inferring abstract data models of the information extracted from a spreadsheet that may support comprehension tasks. In [155] Mittermeir et al. introduce the concepts of logical and semantic equivalent classes to support the comprehension of a spreadsheets, while Hung et al. [156] present a technique based on explicit extraction and transformation rules proposed by the users. Recently Chen et al. [157] propose automatic rules to infer some information useful in the migration of a spreadsheet into a relational database. They evaluated the effectiveness of the proposed rules on a very large set including more than 400 thousand of spreadsheets crawled from the Web. The works of Abraham et al. [144, 143, 158, 159] instead propose techniques for automatically inferring shared templates underlying a given spreadsheet corpus that could be exploited for safe editing of the spreadsheets by end users, avoiding possible errors. Also Ahmad et al. [160] propose similar approaches that exploit specific information about the type of data contained in the spreadsheet cells. Clermont [161] instead introduces a technique to analyze large spreadsheet programs decomposing them into smaller units. Cunha et al. propose a technique to infer relational models from spreadsheets that exploits methods for finding functional dependencies between data cells [162]. They also present techniques to infer ClassSheets Models, previously introduced by Abraham and Erwig, from spreadsheets [163]. They present a tool for embedding the visual representation of ClassSheets Models into a spreadsheet system. In some successive works, [164, 165, 166, 167], they propose a framework for the employment of a Model Driven Development approach for the spreadsheet context. In addition, they evaluated the impact of the proposed Model Driven approach on users' productivity through an empirical study [168] and in [169] they perform a further empirically study aimed at comparing the quality of automatically generated relational models against the ones provided by a group of database experts.

Another set of relevant contributions is due to Hermans et al. who propose a technique to recover a conceptual data model from spreadsheets that is based on two-dimensional patterns proposed by the same authors [142], as well as on other ones found in the literature (e.g. [170, 171, 172]). The technique has been validated with respect to the spreadsheets corpus proposed by Fisher and Rothermel [173]. Later, Hermans et al. presented an approach to support users in understanding spreadsheet applications by means of leveled dataflow diagrams [140]. The proposed diagrams were intended to represent calculations within a spreadsheet. They evaluate the proposed approach in a large financial company. In a successive work [174] they propose a technique to detect data clones within spreadsheets and to visualize them by means of dataflow diagrams. They quantitatively evaluate the proposed data clone detection algorithm on the *EUSES Corpus* [173] and the implication of data clones on real-life spreadsheets that were used in a Dutch organization and in academia. Roy and Hermans [175] presented a first study to evaluate the relevance of visual dependence tracing techniques to understand cell-to-cell data flows. In the same work they highlighted the need for a wider validation of visualization tools in order to assess their effectiveness on real-life spreadsheet applications and their usability.

Comprehension processes have been also presented for finding errors and bad smells in spreadsheet applications. Hermans et al. investigate the applicability of code smells to spreadsheet applications. In [176] they analyze inter-worksheet smells, whereas in [177] they focus on formula smells that were analyzed on the EUSES corpus [173] and on spreadsheets developed by professional users. Abreu et al. [178] propose a technique to automatically detect faults in spreadsheets using a catalog of spreadsheet smells taken from [177, 179, 180].

The only contributions found in the literature addressing the analysis of the code of procedures included in spreadsheets are due to Cheng and Rival. These authors propose a static analysis techniques to infer type constraints over spreadsheets cells to avoid type errors [181] and to detect run-time type unsafe operations [182].

4.6 Conclusions and Future Work

In this Chapter I proposed the reverse engineering tool EXACT that was designed to address the main comprehension issues that trouble VBA-based Excel applications. This tool extends the Excel IDE offering analysis and visualization features that can be exploited by end users involved in the tasks of comprehending an existing application. Although the tool has been developed just for VBA-based spreadsheet applications, many of its features may be adapted for aiding the comprehension of spreadsheet applications implemented by different technologies. To this aim, the EXACT conceptual data model should be adapted and specific code parsers should be developed in order to correctly consider the specific characteristics of those applications.

EXACT has been preliminary validated through a qualitative case study that was conducted in a real automotive industrial context. The study showed the usefulness of the tool in supporting the execution of comprehension tasks carried out in real maintenance projects. The features offered by the tool aided the end users in the execution of both top-down and bottom-up comprehension processes. The experiment also showed some limitations of the proposed tool and the need for additional features.

In future work, I plan to extend the features offered by the tool and to address some of its limitations emerged from the study. Moreover, I intend to assess the effectiveness of the tool in a controlled experiment aimed at comparing comprehension processes of Excel applications performed with and without the tool. Eventually, an empirical study involving applications from different domains and end users from different contexts will be also performed for validating the proposed tool.

Chapter 5

Supporting the adoption of Software Product Lines in Software Processes using Reverse Engineering

In this Chapter, I report a software infrastructure I implemented for supporting the introduction of SPL in industrial settings where Model Based Design (MBD) software processes are adopted. The proposed architecture, named AutoMative, exposes features for the semi-automatic generation of Product Architecture (PA) from specification documents. The feasibility of the architecture in supporting the introduction of SPL approach was evaluated through an experience I performed in collaboration with the FCA company for the application of the SPL in one of its MBD processes.

5.1 Introduction

Nowadays, the automotive market presents a huge number of vehicle models and new models are placed on the market every year. In 2015 there were 222 vehicle models available in the U.S. and 43 new models were introduced [1]. Vehicle models are tailored according to customer requirements and preferences besides the different laws and the cultural preferences of the market in which they are sold. This tailoring not only impacts on the aesthetic design of the vehicle, but also on the functionalities it provides. Functionalities typically range from basic safety functions, like brake and airbag control, to driver assistance systems, like adaptive cruise control, automatic parking and infotainment system. They are provided by specific control software made by almost 100 million lines of code that are executed on 70 to 100 microprocessor-based ECUs networked throughout the body of the car [183]. The variability of these functionalities has hence a great impact on the control software implementing them.

In this scenario, automotive companies need to adopt cost-effective software development processes in order to manage the variability of the produced software. A case-by-case basis approach, where the software variability is managed at the end of the development process, can no longer be considered suitable to resolve these issues. Since that, more systematic solutions need to be introduced [14].

Software Product Lines (SPL) approach has proven to be a successful solution for handling the complexity and variability of the software developed in different domains as shown by several success stories reported in [15]. According to Clements and Northrop [16], *"a software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a*

[1]http://www.statista.com/statistics/200092/total-number-of-car-models-on-the-us-market-since-1990/

prescribed way".

SPL were profitably applied in the automotive domain too, as reported in [184, 185]. These works describe experiences at General Motors on the application of SPL in the automotive context, for managing products complexity and variability across the entire development lifecycle. To this aim, a specific SPL aware software tool, i.e., BigLever Gears[2], was exploited. Even if a complete adoption is far from being reached, they already getting value out of their efforts.

Although the SPL bring actual benefits to the software variability management, they strongly impact on the overall software life cycle. As a consequence, one of the main challenges, for both academic and industrial communities, is to find methodologies aiding the companies in the adaption of their software development processes to the SPL [186].

In this Chapter, I describe the industrial experience performed in collaboration with a SoftWare Factory (SWF) of the FCA company. I report the results of this experience in terms of: (1) the approach I applied for adapting the current SWF development process to the SPL, (2) the infrastructure I designed and implemented for exploiting the SPL, (3) the costs for introducing the approach, and (4) a brief discussion on the benefits gained by SWF after the introduction of SPL.

The Chapter is structured as follows: Section 5.2 reports the industrial context where I applied the approach presented in Section 5.3. Section 5.4 shows the feasibility of the approach and the results I obtained from its application in the industrial automotive domain. Section 5.5 reports work related to mine. Finally, Section 5.6 summarizes conclusions and possible future work.

[2] *http://www.biglever.com/solution/product.html*

5.2 The Addressed Problem

I addressed the problem of introducing the use of SPL in one of the embedded software development processes followed in FCA. More precisely, I adapted the usual development process carried out by its SWF in a decoupled project for the implementation of the Instrument Panel Cluster (IPC) control software.

The IPC, or Vehicle Dashboard, is the main component of the Human Machine Interface (HMI) existing between the vehicle and the driver. It is controlled by a dedicated *IPC-ECU* that elaborates the different signals coming from the other ECUs composing the overall electric/electronic architecture of the vehicle. Signals are processed in order to render on the IPC information about the vehicle status.

Fig. 5.1 shows an IPC that is assembled on the dashboard of the Fiat Punto[3]. It presents (A) a speed indicator, (B) a fuel level gauge, (C) an engine coolant temperature gauge, (D) a tachometer and (E) a multi-function display. In addition, it contains LED or lamp telltales and acoustic signaling.

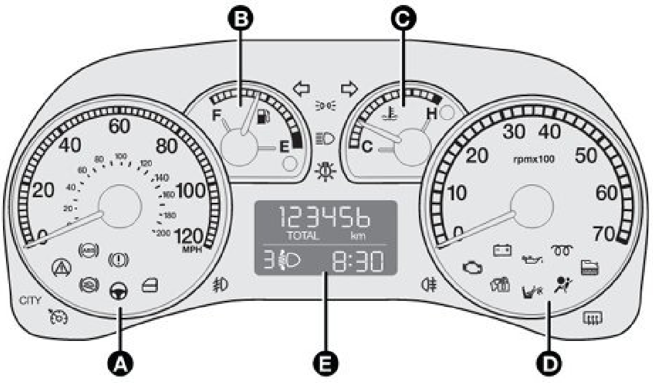


Figure 5.1. *Fiat Punto IPC Example*

[3] *http://www.fiatia.com/fman-431.html*

The FCA SWF, involved in the development of the application software for the *IPC-ECU*, exploits a *MBD* approach [187]. SWF software engineers receive requirements specification documents in different formats, such as IBM DOORS and Microsoft Excel files, from teams of system and components requirement engineers. Starting from these specifications, they develop prescriptive MATLAB/Simulink software architectural models. Production embedded code is then automatically generated from these models by exploiting the *dSpace TargetLink* automatic code generator tool[4].

I analyzed the adopted MBD process and I observed that it suffered from two main issues since no systematic reuse approaches were exploited. As for the first issue, new prescriptive architectural models were developed for each new vehicle, even if they implement same or very similar features. Software engineers eventually just exploited opportunistic reuse strategies such as clone-and-own or copy-paste-and-modify ones. It is well known that these approaches are both error-prone and inefficient and they could lead to resources waste. The second issue was related to the difficulties in maintaining links and/or specific relationships between these models, given that possible bug-fixes or modifications in one of their common parts should be applied in all the different models separately.

I decided to introduce the use of SPL to overcome these issues, since it is a well-known solution for these problems.

The introduction of SPL would require an adaption of every single phase of the development process. This drastic change was considered too risky. Instead, an incremental approach was undertaken, where I decided to initially introduce SPL in the architectural design phase.

I defined an approach that allowed to adapt the MBD development process followed in SWF towards the SPL. In order to exploit the SPL, I designed and implemented an SPL infrastructure, named *AutoMative*,

[4] *https://www.dspace.com/en/pub/home/products/sw/pcgs/targetli.cfm*

able to produce prescriptive MATLAB/Simulink architectural models tailored for specific vehicles starting from their requirements specification documents, in a semi-automatic way.

The approach I followed to introduce SPL in the architectural model development phase is reported more in detail in the next section.

5.3 The Proposed Solution

Similarly to [188], my approach relayed on the execution of two different processes: (1) a *Domain Engineering* process aimed at developing the SPL Infrastructure and (2) an *Application Engineering* process needed to generate specific products by exploiting this infrastructure. In the following, I report more details about how these processes were carried out.

5.3.1 Domain Engineering

The main aim of this process was to identify the features of the IPC and to define its Product Line Architecture (PLA). PLA provides the design that is common to all SPL products, hence it describes all the mandatory and varying features of the SPL domain [189]. A PA is obtained by tailoring the PLA for a specific product. The tailoring is defined by means of a Feature Profile (FP) representing the configuration of the overall architecture in terms of the features to provide and the values for configuring them. The Domain Engineering Process required the execution of the following activities:

Identify Features

it was carried out to infer a Feature Model of the IPC, where the Feature Model is a widely used approach to describe commonalities and variabilities in SPL. To this aim, I manually analyzed the IPC specification documents of different vehicle models and identified the main features and the relationships existing among them.

Design the PLA

it was executed for designing the PLA of the IPC. PLA was actually realized as a composition of configurable MATLAB/Simulink architectural models. Each model was specific for one of the identified features and was defined in terms of:

- composing parametric subsystems and their Variation Point (VP)s, where a VP indicates a location in the model where a variation can occur,
- configurable interconnections among parametric subsystems,
- constraints to be satisfied by both subsystems and interconnections.

In order to design the configurable MATLAB/Simulink architectural models, commonalities and variabilities among different vehicle models were identified. This can be done by employing a reverse engineering process on the architectural models developed in previous projects and/or by analyzing specification documents related to different vehicles. Once identified, commonalities and variabilities were exploited to define the fixed and parametric parts of the generic architectural model.

Moreover, I specified the transformation rules needed to produce IPC PAs. These rules define how instantiate and configure both the parametric subsystems and the interconnections among them according to a given FP.

Define the Extraction Rules

it is executed to produce a FP starting from specification documents of a given vehicle. This guarantees the semi-automatic generation of IPC

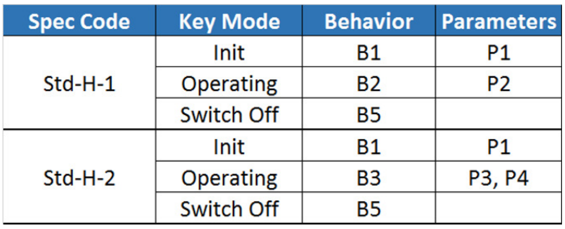
PAs. Reverse engineering activity on various types of specification documents adopted by the company was performed in order to infer their conceptual models. To this aim, I exploited the process defined in [190] to recover models of the specifications expressed as spreadsheet information systems. Exploiting these models, I defined specific extraction rules aimed at analyzing the specification documents in order to identify the features VPs and extract their values. Moreover, I also defined the rules needed to map the extracted values and fill in a FP instance.

As an example, for the *F1* feature, VPs and their values are identified and extracted from different types of specification documents as shown in Figure 5.2a and Figure 5.2b. The specification document of Figure 5.2a reports for each behavior specification, its code, its key operating modes. Each key operating mode has its behavior and configuration parameters. The specification document of Figure 5.2b is another document reporting for each indication, the identifier of the associated Vehicle Function, the identifier of the required behavior and its parameters. The values extracted from the specification documents are used to fill in the FP, as shown in Figure 5.2c. The FP reports for each feature, the involved indications, the required behaviors and their configuration parameters.

Develop the SPL Infrastructure

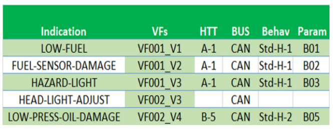
to support the automation of the application engineering process, I developed an infrastructure, named *AutoMative*, that can be exploited to semi-automatically produce IPC PAs tailored for specific products. This infrastructure allows: (1) to handle the PLA, the configurable MATLAB/Simulink models, the extraction and the transformation rules to be applied and the implemented parametric subsystems defined in the domain engineering process and (2) to automatically produce IPC PA starting from the vehicle specification documents.

Figure 5.3 reports the AutoMative architecture. All the operations are



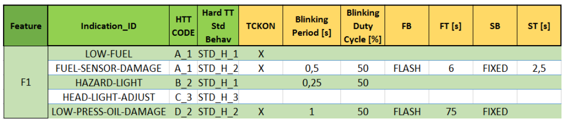
Spec Code	Key Mode	Behavior	Parameters
Std-H-1	Init	B1	P1
	Operating	B2	P2
	Switch Off	B5	
Std-H-2	Init	B1	P1
	Operating	B3	P3, P4
	Switch Off	B5	

(a) IPC Specifications



Indication	VFs	HTT	BUS	Behav	Param
LOW-FUEL	VF001_V1	A-1	CAN	Std-H-1	B01
FUEL-SENSOR-DAMAGE	VF001_V2	A-1	CAN	Std-H-1	B02
HAZARD-LIGHT	VF001_V3	A-1	CAN	Std-H-1	B03
HEAD-LIGHT-ADJUST	VF002_V3		CAN		
LOW-PRESS-OIL-DAMAGE	VF002_V4	B-5	CAN	Std-H-2	B05

(b) IPC Behaviors Specs



Feature	Indication_ID	HTT CODE	Hard TT Std Behav	TCKON	Blinking Period [s]	Blinking Duty Cycle [%]	FB	FT [s]	SB	ST [s]
F1	LOW-FUEL	A_1	STD_H_1	X						
	FUEL-SENSOR-DAMAGE	A_1	STD_H_2	X	0,5	50	FLASH	6	FIXED	2,5
	HAZARD-LIGHT	B_2	STD_H_1		0,25	50				
	HEAD-LIGHT-ADJUST	C_3	STD_H_3							
	LOW-PRESS-OIL-DAMAGE	D_2	STD_H_2	X	1	50	FLASH	75	FIXED	

(c) IPC Feature Profile

Figure 5.2. Analyzed Specifications and Generated Feature Profile

managed by a *Coordinator*, that performs an orchestration of the components in the architecture. The core of the architecture is a *Model Transformer*, that executes the transformation rules in order to produce a PA starting from a FP. The *Profile Manager*, semi-automatically fills in a FP according to the specification documents applying the defined extraction rules. All the elements needed to generate the PAs are stored and managed by the *Repository* component.

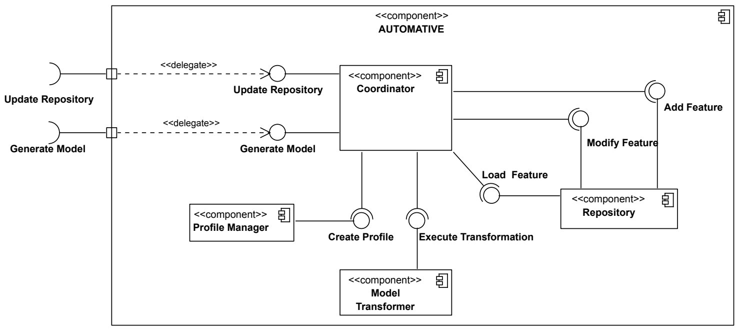


Figure 5.3. *AutoMative Architecture*

5.3.2 Application Engineering

The main goal of the Application Engineering process is to generate PAs according to given vehicle specification documents by exploiting the SPL infrastructure defined in the Domain Engineering Process. Two different activities need to be executed in this process:

Feature Profile Definition

it is performed to generate the FP of a specific vehicle starting from its specification documents. If these documents comply with the models inferred in the Extraction Rules Definition activity, then AutoMative is able to automatically generate a FP. Otherwise, an additional manual effort is required to fill in the FP values not automatically retrieved. In this case, AutoMative provides features aiding the manual editing of the missing values.

Product Architecture Generation

it is executed to automatically produce an IPC PA tailored to a specific vehicle model on the basis of its FP. The automatic generation of PA is performed by AutoMative that is able to: (1) analyze the FP, (2) generate the MATLAB/Simulink models of the required features by instantiating and configuring the needed parametric subsystems and the interconnections among them.

5.4 Evaluation of the Proposed Approach for the development of a real IPC

To show the feasibility of the proposed approach, I adopted it for introducing the use of SPL in the MBD development process for a subset of the IPC features.

Table 5.1. SPL complexity metrics

Feature	***# Implemented Parametric Subsystems***	***# Identified Variation Points***
F1	18	93
F2	17	76
F3	13	104
F4	3	196

Table 5.2. PA complexity metrics

# Instantiated Parametric Subsystems	***# Filled Variation Points***
40	164
124	418
208	1664
3	196

More in detail, the execution of the Domain Engineering Process allowed us to implement the AutoMative infrastructure that was exploited in executing the Application Engineering process for developing the IPC PA for the Fiat TIPO.

The execution of the Domain Engineering process allowed us to identify four main features of the IPC and to obtain its PLA. Table 5.1 reports, for each feature, the number of the implemented parametric subsystems and the overall number of identified variation points.

Regarding the costs of the Domain Engineering process, seven man-months were spent for the *AutoMative* infrastructure development. On the other end, the execution of the remaining activities required eleven man-months.

The execution of the application engineering process allowed us to obtain IPC PA tailored to the FIAT Tipo. Table 5.2 reports the complexity of the generated PA in terms of instantiated parametric subsystems and filled variation points. The developed parametric subsystems were instantiated by opportunely configuring their variation points. A parametric subsystem could be instantiated multiple times, each time with a specific configuration, for realizing a given Feature.

The effort for executing the Feature Profile Definition activity depends on the nature of the specifications: if they comply with the model obtained with the Extraction Rules Definition activity, *AutoMative* is able to automatically produce a FP in tens of seconds. Otherwise, this activity can be performed by an expert developer in a matter of hours, based on the complexity of the specification documents. In the reported experience, about a 30% of the variation point values had to be manually defined.

The proposed approach properly suited the currently adopted development process. The experience in its application showed us its feasibility related to both its implementation and application costs and the efficiency improvements achieved.

5.4.1 Discussions and lessons learned

Our experience showed that it is possible to apply the approach for introducing the SPL in a software development process based on MBD paradigm.

Moreover, it was possible to introduce the SPL without any impact on the entire development cycle. Our approach did not require any modification of the specification documents and it produced MATLAB/Simulink prescriptive models that can be processed by code-generation tools.

At the expense of the initial effort for the introduction of the SPL and in particular for the implementation of the AutoMative infrastructure, the overall development process has increased its effectiveness since few hours are needed for implementing the IPC-ECU application software.

Another benefit of the introduction of the SPL is the semi-automatic generation of the MATLAB/Simulink models from the specification documents. In this way, a reduced level of expertise in MATLAB/Simulink software modeling is required during the development process.

5.5 Related Work

Software Product lines has been widely investigated by the Software Engineering community. In this section I report the main literature work related to SPL, according to the bibliometric analysis carried out by Heradio *et al.* [191]. According to this analysis, the work on SPL proposed in the literature focused on different topics, i.e Software Architecture, Feature Modeling, Software Design, Variability Management, Software Quality, Software Reuse, SPL Testing, and Product Derivation. In this work I focused mainly on Software Architecture. More precisely, I focused on PLA that represents a special kind of software architecture. It is designed to describe the software architecture of a set of similar software products that are developed in the context of an SPL [16]. In the literature, many definitions have been presented to define PLA. These definitions consider PLA as a core architecture that captures the variability of a set of software products at the architecture level. However, they differ in terms of the variability definition. Gomaa tried to specify the nature of the architecture variability linking it with the architectural-elements [192]. Thus, in his definition, PLA defines the variability in terms of mandatory, optional, and variable components, and their connections.

Different approaches have been proposed to recover PLA. Koschke *et al.* proposed an approach aiming at recovering PLA that identifies component variants based on the detection of cloned code among the products [193]. Acher *et al.* proposed an approach to reverse engineering architectural feature model [194]. This is based on the software architect's knowledge, the architecture dependencies, and the feature model. The feature model is extracted based on a reverse engineering approach presented by She et al. [195]. Pinzger *et al.* defined an approach for recovering PLA of a family of software product variants [196]. For each software product, it recovers a set of architecture views from the source code. Duszynski *et al.* described an approach to visually analyze the distribution of variability and commonality among the source code of product variants [197]. The analysis includes multi-level of abstractions (e.g. line of code, method, class, etc.) for facilitating the identification of reusable entities.

Differently from these proposed approaches, my work focused on models rather than on code. The proposed approach aims at recovering commonalities and variability from MATLAB/Simulinks models.

5.6 Conclusions and Future Work

In this Chapter, I report the industrial experience I carried out in a SWF of FCA to introduce a more systematic reuse approach based on SPL for the development of architectural models. In order to support this approach, I developed a SPL infrastructure, named AutoMative, aimed at semi-automatically producing MATLAB/Simulink architectural models tailored to given vehicles starting from their specification documents. In order to evaluate the feasibility of the approach, I reported the experience of its application to develop architectural model tailored to a specific vehicle and the lessons I gained.

The application of the approach, preliminary showed its feasibility and allowed me to identify different points of improvement that will be addressed in future work. I plan to apply the SPL to the other phases of the development process. Moreover, in order to guarantee the correct integration of the SPL within the current development process, the AutoMative infrastructure should be opportunely integrated with the tools exploited in FCA for the execution of the change management process.

This page intentionally left blank.

Chapter 6

Conclusions and Future Work

Software process is a complex phenomenon that is aimed at producing a software product. It involves several actors having different roles that are asked to perform a number of complicated activities involving a multitude of artifacts, with the support of different tools.

Even if the Software Engineering community has been devoting a great effort for proposing methodologies, approaches and tools for supporting software process assessment and improvement, challenges and issues are still open and new solutions have to be proposed that can be actually applied in real industrial settings.

In this Thesis work, I propose different solutions, i.e. approaches and tools, that can be applied in real industrial contexts for addressing some of the open issues still affecting software processes. More in detail, I present an approach that exploit both ALM and MDE for supporting the design and execution of Gap Analysis processes that are planned and executed by companies as Internal assessment against the practices and requirements prescribed by Software Process Quality Standards or Quality Evaluation Frameworks they are willing to comply with. In collaboration with my research group, I identified the main issues affecting this kind of process through a survey we conducted in real industrial settings. According to the results of this study, I identified the requirements of a tool for supporting the execution of Gap analysis processes and I proposed to adopt the features of ALM for overcoming the identified issues and supporting, in an effective way, the execution of this kind of processes. Moreover, for supporting the tailoring of the ALM tool to the specific gap analysis process with respect to a given Standard, I defined and implemented in a tool, named GADGET, an MDE approach providing features for easing the design of the Gap Analysis process and automatically producing the tailored ALM based tool.

As another contribution of this work, I developed an approach for the automatic management of artifacts involved in industrial software process that leverages on a tool integration architecture. More in detail, the proposed architecture integrates ALM platforms with the software tools adopted for supporting software testing processes. It was realized by exploiting the features offered by Continuous Integration platforms with the aim of implementing a toolchain that is able to automate the manual activities of the process and to automatically create the appropriate traceability links between the involved artifacts. The architecture was designed in order to be easily extended and reconfigured both to integrate new tools with the ALM, both to implement new software processes. The integration of other tools will require an additional component and the design of the connectors needed for interfacing the tool with the ALM. The architecture can be also adapted to support new processes requiring to develop a component enacting the required workflow.

With respect to the issues tied to the comprehension of the artifacts involved in the software process, I defined both a reverse engineering process and a tool, able to automatically extract the needed information from them and abstract the models required for easing their comprehension. More in detail, I defined a heuristic based approach for reconstructing Conceptual

Data Models of spreadsheet based artifacts. The process requires the execution of seven sequential steps that allow to obtain, through subsequent refinements, the information needed for automatically reconstructing its conceptual Data Model. Moreover, I developed a tool, named EXACT, providing visualization and analysis features for supporting the comprehension of VBA based spreadsheets. I decided to focus on these peculiar artifacts since they are widely adopted for supporting different activities of software development processes in several industrial domains.

Finally, in order to handle the variability that may characterize software processes and to introduce in companies the adoption of a more systematic reuse approach, I developed an SPL infrastructure, named AutoMative. This infrastructure provides features for semi-automatically producing MATLAB/Simulink architectural models tailored to a given software product, starting from its specification documents.

All the proposed solutions for addressing the considered software process issues were validated through case studies conducted in industrial domains, involving personnel employed in the execution of real software processes. The case studies were conducted following the guidelines defined by Runeson *et al.* in [17]. I chose to leverage on qualitative evaluations, involving interviews and surveys with the involved subjects. In this way, I was able to gain more information about the executed software processes, the impacts of the proposed solutions on the processes and on the involved personnel. According to the results of these case studies I conducted in collaboration with members of my research group, I was able to evaluate the feasibility and the validity of the proposed approaches and tools. The case studies showed that our proposed approaches had positive impacts on the software process leading to improvements in the considered process characteristics. Moreover, according to the qualitative evaluations that were carried out, I was able to identify possible limitations and points for their improvement.

As future work, I plan to improve the proposed approaches and tools and to further evaluate their validity through case studies involving software processes in different domains, considering a greater number of subjects and objects. In this way, I aim at assessing their generalization. Moreover, I plan to design and develop a collaborative and configurable environment, based on MDE and ALM technologies for integrating the proposed and future solutions for supporting different Software process Improvement initiatives, according to specific goals pursued by companies.

Bibliography

[1] I. Sommerville, *Software Engineering.* USA: Addison-Wesley Publishing Company, 9th ed., 2010.

[2] S. Bandinelli, A. Fuggetta, L. Lavazza, M. Loi, and G. Pietro Picco, “Modeling and improving an industrial software process,” *IEEE Trans. Softw. Eng.*, vol. 21, pp. 440–454, May 1995.

[3] M. Kuhrmann and S. Beecham, “Artifact-based software process improvement and management: A method proposal,” in *Proceedings of the 2014 International Conference on Software and System Process*, ICSSP 2014, (New York, NY, USA), pp. 119–123, ACM, 2014.

[4] M. Kuhrmann, P. Diebold, J. Münch, and P. Tell, “How does software process improvement address global software engineering?,” in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, pp. 89–98, Aug 2016.

[5] F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öhman, “A practitioner’s guide to light weight software process assessment and improvement planning,” *J. Syst. Softw.*, vol. 81, pp. 972–995, June 2008.

[6] "CMMI - Capability Maturity Model Integration," standard, CMMI Institute, Pittsburgh, US Mar. 2015.

[7] "ISO/IEC 15504:2012 - Information technology – Process assessment," standard, International Organization for Standardization / International Electrotechnical Commission, Geneva, CH, Feb. 2012.

[8] "ISO/IEC 330xx:2015 - Information technology – Process assessment," standard, International Organization for Standardization / International Electrotechnical Commission, Geneva, CH, Mar. 2015.

[9] J. L. de la Vara, A. Ruiz, K. Attwood, H. Espinoza, R. K. Panesar-Walawege, Ángel López, I. del Río, and T. Kelly, "Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel," *Information and Software Technology*, vol. 72, pp. 16 – 30, 2016.

[10] "ISO 26262:2011 - Road vehicles – Functional safety," standard, International Organization for Standardization, Geneva, CH, Nov. 2011.

[11] "IEC 62304 Medical device software – Software life cycle processes," standard, IEC, Geneva, Switzerland Mar. 2006.

[12] "IEC 61513 Nuclear power plants - Instrumentation and control important to safety," standard, IEC, Geneva, Switzerland Mar. 2011.

[13] D. M. Fernández, B. Penzenstadler, M. Kuhrmann, and M. Broy, "A meta model for artefact-orientation: Fundamentals and lessons learned in requirements engineering," in *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems: Part II*, MODELS'10, (Berlin, Heidelberg), pp. 183–197, Springer-Verlag, 2010.

[14] S. Thiel and A. Hein, "Modeling and using product line variability in automotive systems," *IEEE Softw.*, vol. 19, pp. 66–72, July 2002.

[15] S. Apel, D. Batory, C. Kstner, and G. Saake, *Feature-Oriented Software Product Lines: Concepts and Implementation.* Springer Publishing Company, Incorporated, 2013.

[16] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns.* Addison-Wesley Professional, 2001.

[17] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples.* Wiley Publishing, 1st ed., 2012.

[18] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, and J. Sillito, "Qualitative research in software engineering," *Empirical Softw. Engg.*, vol. 16, pp. 425–429, Aug. 2011.

[19] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, pp. 557–572, Jul 1999.

[20] D. Amalfitano, V. De Simone, A. R. Fasolino, and S. Scala, "Improving traceability management through tool integration: An experience in the automotive domain," in *Proceedings of the 2017 International Conference on Software and System Process*, ICSSP 2017, (New York, NY, USA), pp. 5–14, ACM, 2017.

[21] D. Amalfitano, A. R. Fasolino, P. Tramontana, V. De Simone, G. Di Mare, and S. Scala, *A Reverse Engineering Process for Inferring Data Models from Spreadsheet-based Information Systems: An Automotive Industrial Experience*, pp. 136–153. Cham: Springer International Publishing, 2015.

[22] D. Amalfitano, V. De Simone, A. R. Fasolino, and P. Tramontana, “Exact: A tool for comprehending vba-based excel spreadsheet applications,” *Journal of Software: Evolution and Process*, vol. 28, no. 6, pp. 483–505, 2016.

[23] D. Amalfitano, V. D. Simone, A. R. Fasolino, M. Lubrano, and S. Scala, “Introducing software product lines in model-based design processes: An industrial experience,” in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pp. 287–290, April 2016.

[24] A. Fuggetta, “Software process: A roadmap,” in *Proceedings of the Conference on The Future of Software Engineering*, ICSE ’00, (New York, NY, USA), pp. 25–34, ACM, 2000.

[25] W. W. Royce, “Managing the development of large software systems: concepts and techniques,” in *Proc. IEEE WESTCON*, IEEE Press, August 1970. Reprinted in Proc. Int’l Conf. Software Engineering (ICSE) 1989, ACM Press, pp. 328-338.

[26] H. D. Benington, “Production of large computer programs,” *Annals of the History of Computing*, vol. 5, pp. 350–361, Oct 1983.

[27] T. E. Bell and T. A. Thayer, “Software requirements: Are they really a problem?,” in *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE ’76, (Los Alamitos, CA, USA), pp. 61–68, IEEE Computer Society Press, 1976.

[28] B. W. Boehm, “A spiral model of software development and enhancement,” *Computer*, vol. 21, pp. 61–72, May 1988.

[29] M. Aoyama, “Agile software process and its experience,” in *Proceedings of the 20th International Conference on Software Engineering*, pp. 3–12, Apr 1998.

[30] L. Rising and N. S. Janoff, “The scrum software development process for small teams,” *IEEE Software*, vol. 17, pp. 26–32, Jul 2000.

[31] “ISO/IEC 12207:2008 - Systems and software engineering – Software life cycle processes,” standard, International Organization for Standardization / International Electrotechnical Commission, Geneva, CH, Feb. 2008.

[32] L. García-Borgoñón, M. Barcelona, J. García-García, M. Alba, and M. Escalona, “Software process modeling languages: A systematic literature review,” *Information and Software Technology*, vol. 56, no. 2, pp. 103 – 116, 2014.

[33] C. Wohlin and R. Prikladnicki, “Systematic literature reviews in software engineering,” *Information and Software Technology*, vol. 55, no. 6, pp. 919 – 920, 2013.

[34] K. Zamli, “Process modeling languages: A literature review,” *Malaysian Journal of Computer Science*, vol. 14, no. 2, pp. 26–37, 2001. cited By 14.

[35] R. Bendraou, J. M. Jezequel, M. P. Gervais, and X. Blanc, “A comparison of six uml-based languages for software process modeling,” *IEEE Transactions on Software Engineering*, vol. 36, pp. 662–675, Sept 2010.

[36] B. Elvesæter, G. Benguria, and S. Ilieva, “A comparison of the essence 1.0 and spem 2.0 specifications for software engineering methods,” in *Proceedings of the Third Workshop on Process-Based Approaches for Model-Driven Engineering*, PMDE ’13, (New York, NY, USA), pp. 2:1–2:10, ACM, 2013.

[37] A. A. Khan, J. Keung, M. Niazi, S. Hussain, and H. Zhang, *Systematic Literature Reviews of Software Process Improvement: A Ter-*

tiary Study, pp. 177–190. Cham: Springer International Publishing, 2017.

[38] K. E. Emam, *Spice: The Theory and Practice of Software Process Improvement and Capability Determination.* Los Alamitos, CA, USA: IEEE Computer Society Press, 1st ed., 1997.

[39] J. Rossberg, *Beginning Application Lifecycle Management.* Berkely, CA, USA: Apress, 1st ed., 2014.

[40] O. C. Z. Gotel and C. W. Finkelstein, “An analysis of the requirements traceability problem,” in *Proceedings of IEEE International Conference on Requirements Engineering*, pp. 94–101, Apr 1994.

[41] “Ieee standard glossary of software engineering terminology,” *ANSI/ IEEE Std 729-1983*, pp. 1–40, Feb 1983.

[42] V. Kirova, N. Kirby, D. Kothari, and G. Childress, “Effective requirements traceability: Models, tools, and practices,” *Bell Labs Technical Journal*, vol. 12, no. 4, pp. 143–157, 2008.

[43] D. Chappell, “What is application lifecycle management?,” tech. rep., 2014.

[44] D. Chappell, “Adopting a common ALM Foundation: why it makes sense in a heterogeneous world,” tech. rep., 2011.

[45] J. Jwo, T. Hsu, and Y. C. Cheng, “Jumpstarting application lifecycle management: A new approach with tool support,” *J. Inf. Sci. Eng.*, vol. 29, no. 3, pp. 475–492, 2013.

[46] H. Coolican, *Research methods and statistics in psychology.* Hodder Education, 2009.

[47] C. H. L. Blaxter and M. Tight, *How to research.* Open University Press, 2010.

[48] L. Given, *The Sage encyclopedia of qualitative research methods.* SAGE Publications Ltd, 2008.

[49] C. Goulding, *Grounded theory: A practical guide for management, business and market researchers.* SAGE Publications Ltd, 2002.

[50] R. D. Galliers and F. F. L, "Choosing appropriate information systems research approaches: A revised taxonomy," in *In Proceedings of the IFIP TC8 WG8.2*, pp. 317–335, 1990.

[51] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering.* Springer Publishing Company, Incorporated, 2012.

[52] R. Stake, *Qualitative Research: Studying How Things Work.* The Guilford Press, 2010.

[53] A. Holliday, *Doing and writing qualitative research.* SAGE Publications Ltd, 2007.

[54] S. Hesse-Biber and P. Leavy, *The practice of qualitative research.* SAGE Publications Ltd, 2010.

[55] J. Maxwell, *Qualitative research design: An interactive approach.* SAGE Publications Ltd, 2004.

[56] D. Muijs, *Doing quantitative research in education with SPSS.* SAGE Publications Ltd, 2010.

[57] M. Balnaves and P. Caputi, *Introduction to quantitative research methods: An investigative approach.* SAGE Publications Ltd, 2001.

[58] J. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches.* SAGE Publications Ltd, 2009.

[59] R. K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods).* Sage Publications, fourth edition. ed., 2008.

[60] J. Miller, "Statistical significance testing: A panacea for software technology experiments?," *J. Syst. Softw.*, vol. 73, pp. 183–192, Oct. 2004.

[61] R. L. Glass and I. Vessey, "Software tasks: intellectual, clerical ... or creative?," in *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, vol. 4, pp. 377–382, Jan 1994.

[62] D. B. Walz, J. J. Elam, and B. Curtis, "Inside a software design team: Knowledge acquisition, sharing, and integration," *Commun. ACM*, vol. 36, pp. 63–77, Oct. 1993.

[63] H. E. Thomson and P. J. Mayhew, "Approaches to software process improvement," *Software Process: Improvement and Practice*, vol. 3, no. 1, pp. 3–17, 1997.

[64] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction.* Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[65] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, pp. 721–734, Aug 2002.

[66] C. Andersson and P. Runeson, "A replicated quantitative analysis of fault distributions in complex software systems," *IEEE Trans. Softw. Eng.*, vol. 33, pp. 273–286, May 2007.

[67] J. M. Verner, J. Sampson, V. Tosic, N. A. A. Bakar, and B. A. Kitchenham, "Guidelines for industrially-based multiple case studies in software engineering," in *2009 Third International Conference on Research Challenges in Information Science*, pp. 313–324, April 2009.

[68] F. S. Silva, F. S. F. Soares, A. L. Peres, I. M. de Azevedo, A. P. L. Vasconcelos, F. K. Kamei, and S. R. de Lemos Meira, "Using cmmi together with agile software development: A systematic review," *Information and Software Technology*, vol. 58, pp. 20 – 43, 2015.

[69] A. Ceccarelli and N. Silva, *Computer Safety, Reliability, and Security: SAFECOMP 2015 Workshops, ASSURE, DECSoS. ISSE, ReSA4CI, and SASSUR, Delft, The Netherlands, September 22, 2015, Proceedings*, ch. Analysis of Companies Gaps in the Application of Standards for Safety-Critical Software, pp. 303–313. Cham: Springer International Publishing, 2015.

[70] L. L. B. A. Parasuraman, Valarie A. Zeithaml, "A conceptual model of service quality and its implications for future research," *Journal of Marketing*, vol. 49, no. 4, pp. 41–50, 1985.

[71] J. Cadle, D. Paul, and P. Turner, *Business Analysis Techniques: 72 Essential Tools for Success.* British Comp Society Series, British Computer Society, 2010.

[72] F. Redmill, "Installing iec 61508 and supporting its users – nine necessities," in *5th Australian Workshop on Safety Critical Systems and Software*, 2000.

[73] R. Feldt, R. Torkar, E. Ahmad, and B. Raza, "Challenges with software verification and validation activities in the space industry," in

2010 Third International Conference on Software Testing, Verification and Validation, pp. 225–234, April 2010.

[74] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand, “Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation,” *Inf. Softw. Technol.*, vol. 55, pp. 836–864, May 2013.

[75] P. E. McMahon, *Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement.* Addison-Wesley Professional, 1st ed., 2010.

[76] W. E. Saris and I. N. Gallhofer, *Design, Evaluation, and Analysis of Questionnaires for Survey Research.* Wiley Publishing, 2007.

[77] C. E. Wilson, “Designing useful and usable questionnaires: You can’t just "throw a questionnaire together",” *interactions*, vol. 14, pp. 48–ff, May 2007.

[78] D. A. Dillman, J. D. Smyth, and L. M. Christian, *Internet, Mail, and Mixed-Mode Surveys: The Tailored Design Method.* Wiley Publishing, 3 ed., 2008.

[79] M. Conrad, “Artifact-centric compliance demonstration for ISO 26262 projects using model-based design,” in *Informatik 2012, 42. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 16.-21.09.2012, Braunschweig*, pp. 807–816, 2012.

[80] I. Al-Mayahi and S. P. Mansoor, “Iso 27001 gap analys - case study,” in *WorldComp 2012 - Proceedings of the World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2012.

[81] M. Gatrell, “The value of a single solution for end-to-end alm tool support,” *IEEE Software*, vol. 33, pp. 103–105, Sept 2016.

[82] S. Kent, "Model driven engineering," in *Proceedings of the Third International Conference on Integrated Formal Methods*, IFM '02, (London, UK, UK), pp. 286–298, Springer-Verlag, 2002.

[83] A. Fuggetta and E. Di Nitto, "Software process," in *Proceedings of the on Future of Software Engineering*, FOSE 2014, (New York, NY, USA), pp. 1–12, ACM, 2014.

[84] H. Lacheiner and R. Ramler, "Application lifecycle management as infrastructure for software process improvement and evolution: Experience and insights from industry," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011, Oulu, Finland, August 30 - September 2, 2011*, pp. 286–293, 2011.

[85] J. Jwo, T. Hsu, and Y. C. Cheng, "Jumpstarting application lifecycle management: A new approach with tool support," *J. Inf. Sci. Eng.*, vol. 29, no. 3, pp. 475–492, 2013.

[86] F. Ciccozzi, D. D. Ruscio, I. Malavolta, and P. Pelliccione, "Adopting mde for specifying and executing civilian missions of mobile multi-robot systems," *IEEE Access*, vol. 4, pp. 6451–6466, 2016.

[87] N. Ferry, H. Song, A. Rossini, F. Chauvel, and A. Solberg, "Cloudmf: Applying mde to tame the complexity of managing multi-cloud applications," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 269–277, Dec 2014.

[88] J. A. Hurtado Alegría, M. C. Bastarrica, A. Quispe, and S. F. Ochoa, "An mde approach to software process tailoring," in *Proceedings of the 2011 International Conference on Software and Systems Process*, ICSSP '11, (New York, NY, USA), pp. 43–52, ACM, 2011.

[89] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *Computer*, vol. 39, pp. 25–31, Feb. 2006.

[90] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, "Atl: A model transformation tool," *Science of Computer Programming*, vol. 72, no. 1, pp. 31 – 39, 2008. Special Issue on Second issue of experimental software and toolkits (EST).

[91] R. Panesar-Walawege, M. Sabetzadeh, L. Briand, and T. Coq, "Characterizing the chain of evidence for software safety cases: A conceptual model based on the iec 61508 standard," in *Software Testing, Verification and Validation (ICST), 2010 Third International Conference on*, pp. 335–344, April 2010.

[92] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand, "Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation," *Information and Software Technology*, vol. 55, no. 5, pp. 836 – 864, 2013.

[93] P. Picha and P. Brada, "Alm tool data usage in software process metamodeling," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 1–8, Aug 2016.

[94] F. Jouault and I. Kurtev, "Transforming models with atl," in *Proceedings of the 2005 International Conference on Satellite Events at the MoDELS*, MoDELS'05, (Berlin, Heidelberg), pp. 128–138, Springer-Verlag, 2006.

[95] J. Mtsweni, "Exploiting uml and acceleo for developing semantic web services," in *2012 International Conference for Internet Technology and Secured Transactions*, pp. 753–758, Dec 2012.

[96] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples.* Wiley Publishing, 1st ed., 2012.

[97] I. Sommerville, *Software Engineering: (Update) (8th Edition) (International Computer Science).* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[98] N. Palmer and L. Mooney, "Building a business case for BPM – a fast path to real result," tech. rep., OpenText Corporation, 2007.

[99] P. Bolstorff and R. Rosenbaum, *Supply Chain Excellence: A Handbook for Dramatic Improvement Using the Scor Model, Second Edition.* New York, NY, USA: Amacom, 2007.

[100] D. K. Nguyen, W.-J. van den Heuvel, M. P. Papazoglou, V. de Castro, and E. Marcos, *GAMBUSE: A Gap Analysis Methodology for Engineering SOA-Based Applications*, pp. 293–318. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[101] M. Postina, I. Sechyn, and U. Steffens, "Gap analysis of application landscapes," in *2009 13th Enterprise Distributed Object Computing Conference Workshops*, pp. 274–281, Sept 2009.

[102] J. Moratalla, V. de Castro, M. L. Sanz, and E. Marcos, "A gap-analysis-based framework for evolution and modernization: Modernization of domain management at red.es," in *2012 Annual SRII Global Conference*, pp. 343–352, July 2012.

[103] A. D. Lucia, A. R. Fasolino, and E. Pompelle, "A decisional framework for legacy system management," in *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, pp. 642–651, 2001.

[104] B. Karabacak and I. Sogukpinar, "A quantitative method for iso 17799 gap analysis," *Computers & Security*, vol. 25, no. 6, pp. 413 – 419, 2006.

[105] T. Valdevit and N. Mayer, "A gap analysis tool for smes targeting ISO/IEC 27001 compliance," in *ICEIS 2010 - Proceedings of the 12th International Conference on Enterprise Information Systems, Volume 3, ISAS, Funchal, Madeira, Portugal, June 8 - 12, 2010*, pp. 413–416, 2010.

[106] M. Cao, J. J. Fan, H. P. Lv, and J. L. Chen, "Evaluation of enterprise knowledge management performance based on gap analysis," in *Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference on*, pp. 894–897, June 2010.

[107] L. Amaral and J. Faria, "A gap analysis methodology for the team software process," in *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pp. 424–429, Sept 2010.

[108] D. Falessi, M. Sabetzadeh, L. Briand, E. Turella, T. Coq, and R. Panesar-Walawege, "Planning for safety standards compliance: A model-based tool-supported approach," *Software, IEEE*, vol. 29, pp. 64–70, May 2012.

[109] A. Murugesan, M. W. Whalen, E. Ghassabani, and M. P. E. Heimdahl, "Complete traceability for requirements in satisfaction arguments," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 359–364, Sept 2016.

[110] R. Wohlrab, J. P. Steghöfer, E. Knauss, S. Maro, and A. Anjorin, "Collaborative traceability management: Challenges and opportu-

nities," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pp. 216–225, Sept 2016.

[111] G. Regan, M. Biro, D. Flood, and F. McCaffery, "Assessing traceability—practical experiences and lessons learned," *Journal of Software: Evolution and Process*, vol. 27, no. 8, pp. 591–601, 2015. JSME-15-0062.

[112] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, Sept. 2007.

[113] H. Lacheiner and R. Ramler, "Application lifecycle management as infrastructure for software process improvement and evolution: Experience and insights from industry," in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 286–293, Aug 2011.

[114] M. Gatrell, "The value of a single solution for end-to-end alm tool support," *IEEE Software*, vol. 33, pp. 103–105, Sept 2016.

[115] M. Biehl, J. El-Khoury, F. Loiret, and M. Törngren, "On the modeling and generation of service-oriented tool chains," *Software & Systems Modeling*, vol. 13, no. 2, pp. 461–480, 2014.

[116] F. Franco, M. Mauro, S. Stevan, A. B. Lugli, and W. Torres, "Model-based functional safety for the embedded software of automobile power window system," in *2014 11th IEEE/IAS International Conference on Industry Applications*, pp. 1–8, Dec 2014.

[117] E. Bringmann and A. Krämer, "Model-based testing of automotive systems," in *Proceedings of the 2008 International Conference on*

Software Testing, Verification, and Validation, ICST '08, (Washington, DC, USA), pp. 485–493, IEEE Computer Society, 2008.

[118] A. I. Wasserman, "Tool integration in software engineering environments," in *Proceedings of the International Workshop on Environments on Software Engineering Environments*, (New York, NY, USA), pp. 137–149, Springer-Verlag New York, Inc., 1990.

[119] I. Thomas and B. A. Nejmeh, "Definitions of tool integration for environments," *IEEE Softw.*, vol. 9, pp. 29–35, Mar. 1992.

[120] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice.* Wiley Publishing, 2009.

[121] G. Canfora, A. R. Fasolino, G. Frattolillo, and P. Tramontana, "A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures," *J. Syst. Softw.*, vol. 81, pp. 463–480, Apr. 2008.

[122] D. Bovenzi, G. Canfora, and A. R. Fasolino, "Enabling legacy system accessibility by web heterogeneous clients," in *Seventh European Conference onSoftware Maintenance and Reengineering, 2003. Proceedings.*, pp. 73–81, March 2003.

[123] P. Picha and P. Brada, "Alm tool data usage in software process metamodeling," pp. 1–8, Aug 2016.

[124] B. Polgár, I. Ráth, and I. Majzik, *Model-based Integration Framework for Development and Testing Tool-chains*, pp. 227–235. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[125] E. Armengaud, M. Biehl, Q. Bourrouilh, M. Breunig, S. Farfeleder, C. Hein, M. Oertel, A. Wallner, and M. Zoier, "Integrated tool-chain

for improving traceability during the development of automotive systems," in *Proceedings of the 2012 Embedded Real Time Software and Systems Conference*, 2012.

[126] C. Wolff, L. Krawczyk, R. Höttger, C. Brink, U. Lauschner, D. Fruhner, E. Kamsties, and B. Igel, "Amalthea - tailoring tools to projects in automotive software development," in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, pp. 515–520, Sept 2015.

[127] L. Westfall, "Bidirectional requirements traceability," tech. rep., 2006.

[128] R. Wieringa, "An introduction to requirements traceability," tech. rep., 1995.

[129] S. Maro, A. Anjorin, R. Wohlrab, and J.-P. Steghöfer, "Traceability maintenance: factors and guidelines," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp. 414–425, ACM, 2016.

[130] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pp. 207–214, Sept 2005.

[131] L. Bradley and K. McDaid, "Using bayesian statistical methods to determine the level of error in large spreadsheets.," in *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pp. 351–354, May 2009.

[132] R. Panko and D. Port, "End user computing: The dark matter (and dark energy) of corporate it," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, pp. 4603–4612, Jan 2012.

[133] W. Hussain and T. Clear, "Spreadsheets as collaborative technologies in global requirements change management," in *Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on*, pp. 74–83, Aug 2014.

[134] B. Hofer, A. Perez, R. Abreu, and F. Wotawa, "On the empirical evaluation of similarity coefficients for spreadsheets fault localization," *Automated Software Engg.*, vol. 22, pp. 47–74, Mar. 2015.

[135] P. Mireault, "Structured spreadsheet modeling and implementation," pp. 32–38, 2015.

[136] H. Shiozawa, K. Okada, and Y. Matsushita, "3d interactive visualization for inter-cell dependencies of spreadsheets," in *Information Visualization, 1999. (Info Vis '99) Proceedings. 1999 IEEE Symposium on*, pp. 79–82, 148, 1999.

[137] J. Davis, "Tools for spreadsheet auditing," *International Journal of Human-Computer Studies*, vol. 45, no. 4, pp. 429 – 442, 1996.

[138] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice.* Wiley Publishing, 2009.

[139] R. Bovey, D. Wallentin, S. Bullen, and J. Green, *Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft Excel, VBA, and .NET.* Addison-Wesley Professional, 2nd ed., 2009.

[140] F. Hermans, M. Pinzger, and A. van Deursen, "Supporting professional spreadsheet users by generating leveled dataflow diagrams," in

Software Engineering (ICSE), 2011 33rd International Conference on, pp. 451–460, May 2011.

[141] M. Storey, "Theories, methods and tools in program comprehension: past, present and future," in *Program Comprehension, 2005. IWPC 2005. Proceedings. 13th International Workshop on*, pp. 181–191, May 2005.

[142] F. Hermans, M. Pinzger, and A. van Deursen, "Automatically extracting class diagrams from spreadsheets," in *ECOOP 2010 – Object-Oriented Programming* (T. D'Hondt, ed.), vol. 6183 of *Lecture Notes in Computer Science*, pp. 52–75, Springer Berlin Heidelberg, 2010.

[143] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, (New York, NY, USA), pp. 182–191, ACM, 2006.

[144] R. Abraham and M. Erwig, "Header and unit inference for spreadsheets through spatial analyses," in *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pp. 165–172, Sept 2004.

[145] L. Moonen, "Generating robust parsers using island grammars," in *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, pp. 13–22, 2001.

[146] G. Canfora, M. Di Penta, and L. Cerulo, "Achievements and challenges in software reverse engineering," *Commun. ACM*, vol. 54, pp. 142–151, Apr. 2011.

[147] S. Wiedenbeck and A. Engebretson, "Comprehension strategies of end-user programmers in an event-driven application," in *Visual*

Languages and Human Centric Computing, 2004 IEEE Symposium on, pp. 207–214, Sept 2004.

[148] H. Shokry and M. Hinchey, "Model-based verification of embedded software," *Computer*, vol. 42, pp. 53–59, April 2009.

[149] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[150] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?," in *Software Engineering (ICSE), 2012 34th International Conference on*, pp. 255–265, June 2012.

[151] A. Karahasanović, A. K. Levine, and R. Thomas, "Comprehension strategies and difficulties in maintaining object-oriented systems: An explorative study," *Journal of Systems and Software*, vol. 80, no. 9, pp. 1541 – 1559, 2007. Evaluation and Assessment in Software EngineeringEASE06.

[152] B. A. Nardi and J. R. Miller, "The spreadsheet interface: A basis for end user programming," in *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, (Amsterdam, The Netherlands, The Netherlands), pp. 977–983, North-Holland Publishing Co., 1990.

[153] K. Hodnigg and R. T. Mittermeir, "Metrics-based spreadsheet visualization: Support for focused maintenance," *CoRR*, vol. abs/0809.3009, 2008.

[154] R. Brath and M. Peters, "Excel visualizer: One click wysiwyg spreadsheet visualization," in *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pp. 68–73, July 2006.

[155] R. Mittermeir and M. Clermont, "Finding high-level structures in spreadsheet programs," in *Reverse Engineering, 2002. Proceedings. Ninth Working Conference on*, pp. 221–232, 2002.

[156] V. Hung, B. Benatallah, and R. Saint-Paul, "Spreadsheet-based complex data transformation," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, (New York, NY, USA), pp. 1749–1754, ACM, 2011.

[157] Z. Chen and M. Cafarella, "Automatic web spreadsheet data extraction," in *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*, SS@ '13, (New York, NY, USA), pp. 1:1–1:8, ACM, 2013.

[158] R. Abraham, M. Erwig, and S. Andrew, "A type system based on end-user vocabulary," in *Visual Languages and Human-Centric Computing, 2007. VL/HCC 2007. IEEE Symposium on*, pp. 215–222, Sept 2007.

[159] R. Abraham and M. Erwig, "Mutation operators for spreadsheets," *Software Engineering, IEEE Transactions on*, vol. 35, pp. 94–108, Jan 2009.

[160] Y. Ahmad, T. Antoniu, S. Goldwater, and S. Krishnamurthi, "A type system for statically detecting spreadsheet errors," in *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pp. 174–183, Oct 2003.

[161] M. Clermont, "Analyzing large spreadsheet programs," in *Reverse Engineering, 2003. WCRE 2003. Proceedings. 10th Working Conference on*, pp. 306–315, Nov 2003.

[162] J. Cunha, J. Saraiva, and J. Visser, "From spreadsheets to relational databases and back," in *Proceedings of the 2009 ACM SIGPLAN*

Workshop on Partial Evaluation and Program Manipulation, PEPM '09, (New York, NY, USA), pp. 179–188, ACM, 2009.

[163] J. Cunha, M. Erwig, and J. Saraiva, "Automatically inferring classsheet models from spreadsheets," in *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pp. 93–100, Sept 2010.

[164] J. Cunha, J. Fernandes, J. Mendes, H. Pacheco, and J. Saraiva, "Bidirectional transformation of model-driven spreadsheets," in *Theory and Practice of Model Transformations* (Z. Hu and J. de Lara, eds.), vol. 7307 of *Lecture Notes in Computer Science*, pp. 105–120, Springer Berlin Heidelberg, 2012.

[165] J. Cunha, J. Fernandes, J. Mendes, and J. Saraiva, "Mdsheet: A framework for model-driven spreadsheet engineering," in *Software Engineering (ICSE), 2012 34th International Conference on*, pp. 1395–1398, June 2012.

[166] J. Cunha, J. Fernandes, J. Mendes, and J. Saraiva, "Towards an evaluation of bidirectional model-driven spreadsheets," in *User Evaluation for Software Engineering Researchers (USER), 2012*, pp. 25–28, June 2012.

[167] J. Cunha, J. Saraiva, and J. Visser, "Model-based programming environments for spreadsheets," in *Programming Languages* (F. de Carvalho Junior and L. Barbosa, eds.), vol. 7554 of *Lecture Notes in Computer Science*, pp. 117–133, Springer Berlin Heidelberg, 2012.

[168] J. Cunha, J. Fernandes, J. Mendes, and J. Saraiva, "Embedding, evolution, and validation of model-driven spreadsheets," *Software*

Engineering, IEEE Transactions on, vol. 41, pp. 241–263, March 2015.

[169] J. Cunha, M. Erwig, J. Mendes, and J. Saraiva, “Model inference for spreadsheets,” *Automated Software Engineering*, pp. 1–32, 2014.

[170] D. Janvrin and J. Morrison, “Using a structured design approach to reduce risks in end user spreadsheet development,” *Information Management*, vol. 37, no. 1, pp. 1 – 12, 2000.

[171] R. Panko and J. Halverson, R.P., “Individual and group spreadsheet design: patterns of errors,” in *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, vol. 4, pp. 4–10, Jan 1994.

[172] B. Ronen, M. A. Palley, and H. C. Lucas, Jr., “Spreadsheet analysis and design,” *Commun. ACM*, vol. 32, pp. 84–93, Jan. 1989.

[173] M. Fisher and G. Rothermel, “The euses spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms,” *SIGSOFT Softw. Eng. Notes*, vol. 30, pp. 1–5, May 2005.

[174] F. Hermans, B. Sedee, M. Pinzger, and A. van Deursen, “Data clone detection and visualization in spreadsheets,” in *Software Engineering (ICSE), 2013 35th International Conference on*, pp. 292–301, May 2013.

[175] S. Roy and F. Hermans, “Dependence tracing techniques for spreadsheets: An investigation,” in *Software Engineering Methods in Spreadsheet (SEMS), 2014, First Workshop on*, July 2014.

[176] F. Hermans, M. Pinzger, and A. van Deursen, “Detecting and visualizing inter-worksheet smells in spreadsheets,” in *Software Engineer-*

ing (ICSE), 2012 34th International Conference on, pp. 441–451, June 2012.

[177] F. Hermans, M. Pinzger, and A. van Deursen, "Detecting code smells in spreadsheet formulas," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pp. 409–418, Sept 2012.

[178] R. Abreu, J. Cunha, J. Fernandes, P. Martins, A. Perez, and J. Saraiva, "Smelling faults in spreadsheets," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, pp. 111–120, Sept 2014.

[179] J. Cunha, J. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *Computational Science and Its Applications – ICCSA 2012* (B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. Rocha, D. Taniar, and B. Apduhan, eds.), vol. 7336 of *Lecture Notes in Computer Science*, pp. 202–216, Springer Berlin Heidelberg, 2012.

[180] J. Cunha, J. Fernandes, P. Martins, J. Mendes, and J. Saraiva, "Smellsheet detective: A tool for detecting bad smells in spreadsheets," in *Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on*, pp. 243–244, Sept 2012.

[181] T. Cheng and X. Rival, "An abstract domain to infer types over zones in spreadsheets," in *Static Analysis* (A. Miné and D. Schmidt, eds.), vol. 7460 of *Lecture Notes in Computer Science*, pp. 94–110, Springer Berlin Heidelberg, 2012.

[182] T. Cheng and X. Rival, "Static analysis of spreadsheet applications for type-unsafe operations detection," in *Programming Languages and Systems* (J. Vitek, ed.), vol. 9032 of *Lecture Notes in Computer Science*, pp. 26–52, Springer Berlin Heidelberg, 2015.

[183] R. N. Charette, "This Car Runs on Code," *IEEE Spectrum*, Feb. 2009.

[184] R. Flores, C. Krueger, and P. Clements, "Mega-scale product line engineering at general motors," in *Proceedings of the 16th International Software Product Line Conference - Volume 1*, SPLC '12, (New York, NY, USA), pp. 259–268, ACM, 2012.

[185] L. Wozniak and P. Clements, "How automotive engineering is taking product line engineering to the extreme," in *Proceedings of the 19th International Conference on Software Product Line*, SPLC '15, (New York, NY, USA), pp. 327–336, ACM, 2015.

[186] A. Leitner, R. Mader, C. Kreiner, C. Steger, and R. Weiß, "A development methodology for variant-rich automotive software architectures," *e & i Elektrotechnik und Informationstechnik*, vol. 128, no. 6, pp. 222–227, 2011.

[187] M. Schulze, J. Weiland, and D. Beuche, "Automotive model-driven development and the challenge of variability," in *Proceedings of the 16th International Software Product Line Conference - Volume 1*, SPLC '12, (New York, NY, USA), pp. 207–214, ACM, 2012.

[188] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[189] A. C. Contieri, G. G. Correia, T. E. Colanzi, I. M. S. Gimenes, E. A. Oliveira, S. Ferrari, P. C. Masiero, and A. F. Garcia, *Software Architecture: 5th European Conference, ECSA 2011, Essen, Germany, September 13-16, 2011. Proceedings*, ch. Extending UML Components to Develop Software Product-Line Architectures: Lessons

Learned, pp. 130–138. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[190] D. Amalfitano, A. R. Fasolino, P. Tramontana, V. De Simone, G. Mare, and S. Scala, *Data Management Technologies and Applications: Third International Conference, DATA 2014, Vienna, Austria, August 29-31, 2014, Revised Selected papers*, ch. A Reverse Engineering Process for Inferring Data Models from Spreadsheet-based Information Systems: An Automotive Industrial Experience, pp. 136–153. Cham: Springer International Publishing, 2015.

[191] R. Heradio, H. Perez-Morago, D. Fernandez-Amoros, F. J. Cabrerizo, and E. Herrera-Viedma, "A bibliometric analysis of 20 years of research on software product lines," *Information and Software Technology*, vol. 72, no. Supplement C, pp. 1 – 15, 2016.

[192] H. Gomaa, "Designing software product lines with uml," in *29th Annual IEEE/NASA Software Engineering Workshop - Tutorial Notes (SEW'05)*, pp. 160–216, April 2005.

[193] R. Koschke, P. Frenzel, A. P. J. Breu, and K. Angstmann, "Extending the reflexion method for consolidating software variants into product lines," *Software Quality Journal*, vol. 17, pp. 331–366, Dec 2009.

[194] M. Acher, A. Cleve, P. Collet, P. Merle, L. Duchien, and P. Lahire, *Reverse Engineering Architectural Feature Models*, pp. 220–235. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[195] S. She, R. Lotufo, T. Berger, A. Wąsowski, and K. Czarnecki, "Reverse engineering feature models," in *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, (New York, NY, USA), pp. 461–470, ACM, 2011.

[196] M. Pinzger, H. Gall, J.-F. Girard, J. Knodel, C. Riva, W. Pasman, C. Broerse, and J. G. Wijnstra, *Architecture Recovery for Product Families*, pp. 332–351. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[197] S. Duszynski, J. Knodel, and M. Becker, "Analyzing the source code of multiple software variants for reuse potential," in *2011 18th Working Conference on Reverse Engineering*, pp. 303–307, Oct 2011.