

FODAcOm: An Experience with Domain Analysis in the Italian Telecom Industry

Proceedings of the Fifth International Conference on Software Reuse, Victoria, B.C., 1998

Alessandro Dionisi Vici, Nicola Argentieri
Telecom Italia
Direzione Generale
Via della Vignaccia, 45
00163 Roma - Italy

Azza Mansour
Telesoft S.p.A.
Via degli Agrostemmi, 30
00040 Roma - Italy

Massimo d'Alessandro, John Favaro
Intecs Sistemi S.p.A.
Via Gereschi, 32
56127 Pisa - Italy

Abstract

FODAcOm is a customization of the FODA domain analysis method for the Italian telecommunications authority, for application within its IT2000 restructuring program. This paper describes recent experience gained in the application of the method within a business unit of Telecom Italia. Three analysis models that were constructed for the Service Provisioning Control (SPC) domain are presented and discussed. Requirements templates, intended for reuse of user requirements across system releases, are also presented and discussed, together with FODAcOm's domain evolution strategy.

Keywords: Software reuse, telecommunications, features, domain analysis, requirements, actors.

1. Introduction and Motivation

Domain architectures are a valuable business issue in Telecom Italia. Every year the operating company produces large amounts of new or updated specifications for its Operations Support Systems, which are developed or maintained by several supplier software companies. Recently a broad program known as IT2000 has been launched to redefine most of the legacy systems, creating a unique opportunity to re-architect whole families of applications in accordance with new business processes.

Since the early 90's the controlled software company Telesoft has heavily invested in platform technology for building families of network management systems. In that context the focus was on gaining maximum advantage by implementing and massively using standard

“managed objects” and generic “management functions” across different applications; the specification of network domain-specific objects was the result of years of activity within standards bodies. Also, during the same years a joint venture with Bell Atlantic gave rise to Sodalia, a software company for development of Operations Support Systems by means of innovative processes and technologies. Sodalia, which has just attained Level 3 in the SEI CMM, is currently working on several projects, systematically applying a software reuse process to the analysis and design processes.

When all the above initiatives were launched it was felt that reuse methodologies, applicable from the earliest lifecycle phases, were scarce and difficult to adapt; therefore a project was started in 1995 with the objective to select an approach to “product line development” and investigate analysis and design methods suitable for “application families” [7]. The objective was to allow comparative analysis of requirements within a domain of related projects, which should lead to a coherent and economical specification of architecture building blocks to be developed by software suppliers. The project has resulted in the FODAcOm methodology, a customization of the SEI Feature Oriented Domain Analysis method [6]. This paper describes both the method and its application to a real telecom domain: *provisioning* for a family of network services (POTS, data and broadband) targeted at residential and business customers.

The primary experience reported in this paper is in the area of domain analysis related to requirements capture and modeling. In this regard, FODAcOm evolution with respect to FODA consists in extending the use of abstraction and refinement from “feature models” with “actors diagrams” in the context model, and “use case

models”, all of which are fundamental to the specification of inter-process integration via interactions.

Provisioning has been selected since it is a well known domain including a large and complex process, highly specialized for each different network service. Its coordination encompasses interactions with many other process: from customer service to network management. For each network service, provisioning interactions are controlled by different business rules.

Evolution planned for the chosen domain requires incremental steps of integration of the three types of network services, making it valuable to look for commonalities and reuse in requirements and possibly in architecture. The FODacom concept of “reuse checkpoints” for dealing with aspects of domain evolution within our applications is described.

It is worth noting that for Telecom Italia the capability to *reuse requirements* [8] for different releases would be valuable in itself, leading to faster, easier and better quality specifications, building incremental accuracy and completeness from existing knowledge. We describe experience with the creation of reusable *requirements templates*.

A domain architecture, including shared and also specialized building blocks, would highly improve integration, compatibility and manageability of a family of related projects in our business units, badly needed to reach market deadlines. These considerations represent ongoing work at Telecom Italia.

2. The FODacom Method

Three main sources contributed to the definition of FODacom. The FODA methodology [6] formed the point of departure. One of the primary motivations for selecting FODA was its relative maturity: FODA has a very complete definition of a full domain analysis process, together with considerable support such as seminars and course material, and a large worldwide user community. Arango notes in [2] that “FODA comes close to being the union of techniques used by other methods.”

Telecom Italia adopted Jacobson’s use case approach [4] to requirements definition several years ago, creating also an internal standard for use case description. For this reason, it was imperative to add use case modeling to the domain engineering process in FODacom. This work parallels similar efforts in other telecom organizations such as the scenario-oriented work of Fraser at Bell Northern Research [3].

Finally, the FODacom definition team has collaborated closely with Hewlett-Packard over the past two years in order to incorporate and customize several key concepts of the *Reuse-Oriented Software Engineering Business* (RSEB) of Jacobson, Griss, and Jonsson [5] in the FODacom method. The collaboration resulted in

several important extensions to “traditional” use cases for domain description, such as *explicit extension points*. RSEB concepts of layered architectural modeling were also incorporated, which have been mapped onto Telecom Italia’s IT2000 program concepts of *service modules*, *architectural components*, and *functional components*.

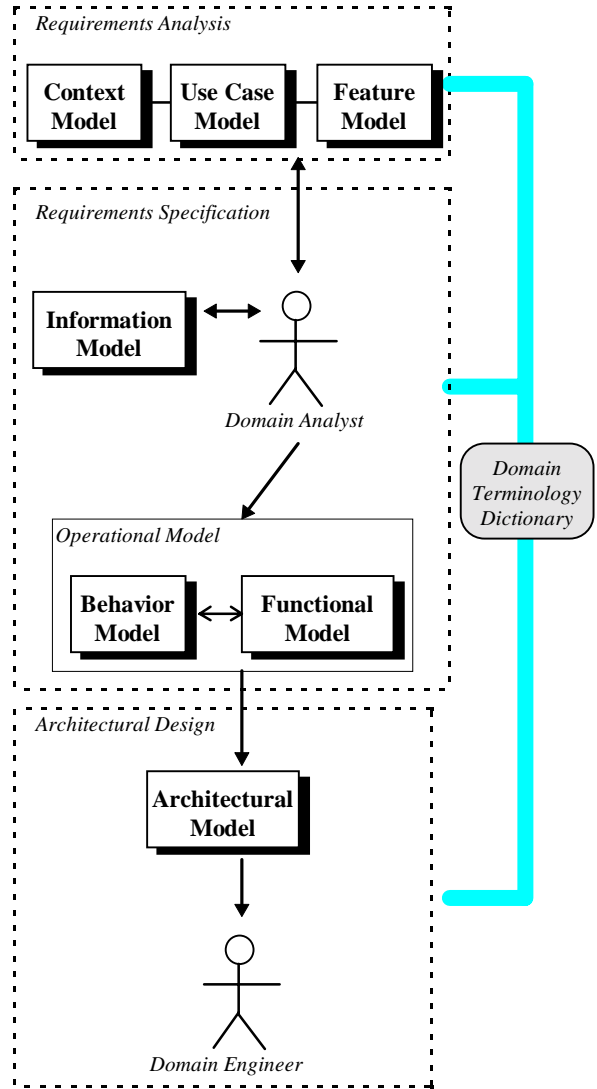


Figure 1: FODacom overview

Like its ancestor FODA, the FODacom method is a *model-driven* approach. gives an overview of the models and their relationships to each other and the principal actors. Generally speaking, FODacom has inherited the set of models associated with FODA, but customization has occurred in many places, which are highlighted in the following discussion.

During requirements analysis, the **context model** is used in the traditional way to help scope the boundaries of the domain, but in addition to the structure diagram of

FODA, an *actors diagram* is added that enforces the high-level process engineering nature of FODAcOm. An example of an actors diagram is presented later in this paper.

The **use case model** (not present in original FODA) is the primary vehicle for requirements capture. The **feature model** in FODAcOm is not subordinated to the use case model, but stands fully side-by-side, and is augmented throughout the life of the domain engineering process. Similarly, the **domain terminology dictionary** is augmented throughout the process. Examples of both use case and feature models are presented in later sections. The **information model** represents the entities of the domain and is based upon Telecom Italia corporate models.

As in standard FODA, during requirements specification the domain analyst makes use of outside sources like domain experts (not shown here), and the previous models to build the **operational model**. However, since FODAcOm was conceived in particular for the documentation and analysis of business processes within Telecom Italia, there is considerably more emphasis on behavioral rather than structural aspects of the domain. Consequently, the **behavior model** is constructed first by the domain analyst. FODAcOm has adopted a modeling formalism already in widespread use in Telecom Italia for this purpose—the Ward-Mellor formalism, which integrates the classic data flow model with synchronization mechanisms [9]. An auxiliary event model (not shown here) can be used when necessary to capture more detailed behavior with a state transition diagram. The **functional model** is integrated with the behavior model in the Ward-Mellor technique, capturing structural aspects with standard data flow modeling mechanisms.

In the architectural design phase, the **architectural model** is produced from the operational model, which has been parameterized according to the commonality and variability in the domain represented by the feature model. This phase has been customized considerably according to Telecom Italia architectural design standards and principles from the RSEB of Jacobson and Griss [5].

3. Experience with FODAcOm

In March 1997, experimentation of the FODAcOm methodology on a *real* domain was begun in Telecom Italia. A real domain in this context was of crucial importance, since the scope of the experimentation was the evaluation of the methodology with respect to the complexity of the real processes, vital to the pursuit of company business objectives.

The selected domain, Service Provisioning Control (SPC), concerns the family of systems that implement the processes of provisioning of the services offered by Tele-

com Italia to its own clients, over a variety of networks: POTS, Data, and Broadband services. More precisely, the following three systems have been considered in the experimentation: SPC/POTS, SPC/DT, and SPC/Broadband.

The SPC systems are part of different IT2000 Projects that are either related to new applications development or to legacy systems adaptation. The comparative analysis performed in the context of the experimentation on the SPC domain has been based on the requirements defined in those IT2000 Projects that belong to the domain. It is important to note that these requirements were specified in approximately 60 use cases.

The results described in the following are:

- Domain Analysis Models related to the Requirements Analysis phase;
- Requirements Templates;
- Common requirements quantification.

Before entering into details of each of the previous list items, a brief description of the SPC Domain systems, is given in the following.

The SPC Domain systems are mainly characterized by Coordination and Synchronization functionalities necessary to perform the activities needed for the provisioning of the requested services.

In this context the activities are specialized into two types: manual, performed by human actors; and automatic, executed by system actors. Each actor has a specific responsibility with respect to the activities to be performed (e.g. Network design activities, Resource Provisioning activities, etc.).

Particularly, for the manual activities, the SPC interacts, through the Operation Units Support Systems (OUSS), with different Operation Units, each responsible of a specific type of activities (e.g. Device installation, physical connections, etc.).

3.1. Domain Analysis Models

In the context of the experimentation on the SPC Domain, the following models have been produced, in accordance with the Requirements Analysis phase of the FODAcOm methodology: Context Model, Use Case Model, and Features Model. In the following sections only one model example will be shown for each of the previously stated model's types.

It is important to mention that for reasons of confidentiality and competency, the Service Provisioning Process represented in the models introduced in the following sections constitutes an intentionally altered version with respect to the real Provisioning Process adopted in Telecom Italia. Moreover, in order to facilitate the comprehension of the models, they are illustrated in a simplified version.

SPC Context Model. The context model represents

the data flows exchanged from/to the domain and its different actors. The Context Model constitutes the starting point for the definition of the domain boundary. The actors defined in the actors diagram of the context model will then constitute the basis for the definition of the services offered by the domain (Use Case Model).

Actors defined in the context model are divided into two types: abstract and specialized. SPC domain specialized actors are those external systems that actually interact with the domain systems, while abstract actors are those obtained by generalizing the specialized actors roles. Such actors are valid for the whole domain. The identification of abstract actors, performed in the context model, allows the definition of use cases that are valid for the whole domain.

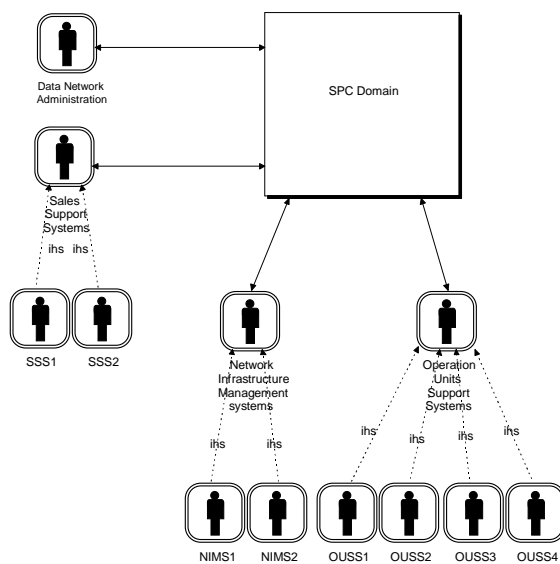


Figure 2: SPC Actors Diagram

Figure 2 shows the abstract actors obtained by the generalization of the specialized actors that correspond to the external systems that interact with the SPC domain, in particular:

- Sales Support Systems (SSS), responsible for the emission/cancellation of Service Provisioning Requests (SPR) towards SPC;
- Data Network Administration (DNA), responsible for the network technical information management that concerns the SPR;
- Network Infrastructure Management Systems (NIMS), responsible for the execution of automatic activities necessary for the provisioning of the requested service;
- Operation Units Support Systems (OUSS), responsible for the assignment, coordination, and execution of manual activities necessary for the provisioning of the requested service.

SPC Use Case Model. This model represents the

different phases of the process flow and the relative flows of events to/from the abstract actors identified in the context model.

The domain use cases are related to the abstract actors (context model actors diagram) and are obtained by abstracting the use cases of the single domain systems, as they represent common process aspects. Differences encountered in the processes of the various systems of the domain are separately represented in the Use Case Model by means of the mechanisms offered by the FODacom methodology (derived from RSEB), that is, *parameterization and extension points*.

Parameters are represented by the string {P<nb>:<parameter name>} and are inserted in specific places in the use case descriptive text, to represent a certain characteristic that varies, in the domain, from one system to another. The application of the parameterization results in a generic use case valid for the whole domain. The specialization of such a use case consists in the assignment of specific values to the defined parameters.

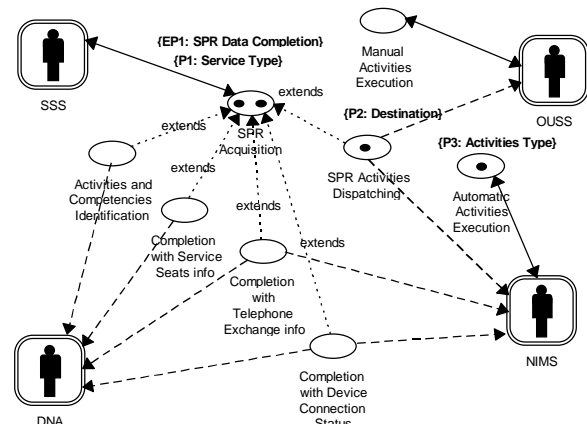


Figure 3: SPC Use Case Model

Extension Points are represented by the string {EP<nb>:<extension point name>} and are used in the use case descriptive text to express a variation in the behavior specification from one domain system to another. The application of the extension point mechanism results in a generic use case valid for the whole domain. The specialization of such a use case consists in the association (by means of the extends relationship) of specific use cases to the defined Extension Points.

Note that more than one parameter, more than one extension point, or a mix of both mechanisms can be used in the same use case.

The process defined for the SPC is activated each time an SPR (Service Provisioning Request) is emitted by SSS (Sales Support System). Figure 3 illustrates use cases that correspond to the SPC process phases and shows their associations to the different domain actors.

The SPC process consists of the following phases: acquisition of the SPRs, dispatching, synchronization and coordination of the activities that have to be performed

for the provisioning of the service requested in the SPR.

In particular, the *SPR acquisition* process phase (use case) consists of the *verification*, *analysis* and *completion* of the information provided in the SPR. The *verification* is of completeness and consistency of the SPR data with respect to the requested service type; the *analysis* will give rise to the identification of the information that must be acquired by the SPC and incorporated in the emitted SPR in order to be able to identify activities and competencies needed for the provisioning of the requested service (see related use case in Figure 3). This information refers to technical details that characterize each service, typically unknown to the SSS emitting the SPR.

Although *completion* of the SPR information is performed in all of the three domain systems under consideration (SPC/POTS, SPC/DT and SPC/Broadband), information to be incorporated in the SPR and behavior associated with the completion process varies from one system to another. If the variation only concerned the information type, it would have been modeled by means of a parameter, but this is not the case. Variation here is also in the behavior associated with the completion process of the SPR missing data. This kind of variation is modeled by means of extension points.

Figure 3 shows the ‘EP: SPR Data Completion’ that models this type of variation. Three different variants, that is, use cases, can be assigned to the extension point: *Completion with service seats info*; *Completion with telephone exchange info* and *Completion with connection device status*. The variant *Completion with service seats info* is instantiated in case the requested service is a SPC/DT; the missing data concern technical information relevant to the different seats the data network must cover; behavior related to this variant consists in capturing the completion network seat data and its incorporation in the SPR. In case the requested service is an SPC/POTS, the variant *Completion with telephone exchange info* is instantiated and is responsible for identification and acquisition of information related to the telephone exchange involved in the requested telephone line, that have to be incorporated into the SPR. The *Completion with connection device status* variant is activated if the requested service is an SPC/Broadband and is concerned with verification of the connectivity of devices that realizes the end-to-end network connection involved in provisioning of the requested service. The SPC is afterwards updated by the acquired information.

The synchronization and coordination of the activities (use cases: *Manual activities coordination* and *Automatic activities coordination*) necessary for the provisioning of the service requested in the SPR is performed by means of an exchange of events with the involved system actors, more precisely:

- SPC sends activities execution requests to responsible

systems;

- in response, the involved systems send to SPC completion notifications of the activities previously assigned.

Finally when all the activities involved in the SPR are performed, SPC communicates to the SSS the fulfillment of the SPR. At this point the process terminates.

SPC Features Model. The features model structures commonalities and differences that exist among the domain features, using abstraction and refinement mechanisms. Such structuring results in a classification of the domain features in common, alternative, and optional with respect to the domain systems.

The domain Features Model is obtained by a systematic and repeated application of the abstraction and refinement mechanisms giving rise to a classification of all the domain features in mandatory, alternative, and optional with respect the different domain systems.

The features model constitutes the main tool for the parameterization of the remaining domain analysis models. Such parameterization constitutes the basis for the design of modular architectural components, flexible, and reusable in the context of the domain.

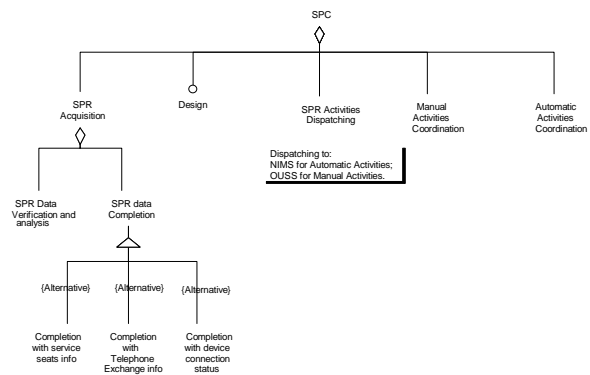


Figure 4: SPC Features Model

Figure 4 illustrates the SPC domain features model relevant to the behavioral features, that is features derived from the Domain Use Cases. As an example, only the *SPR Acquisition* feature has been detailed, in the following figure, into its component features.

Note how the extension point *SPR Data Completion* in the use case model is mapped onto a corresponding feature in the features model. Variants are typically represented by *alternative* features, Figure 4 shows the three corresponding features to the variants showed in Figure 3.

The formalism used by original FODA for the features model has been adapted to a UML-like notation in FODacom for the following reasons:

- UML is currently undergoing standardization;
- UML is supported by more and more CASE tools (consequently information can be stored in the repository);

- ER notation, of which UML constitutes an updated version, is widely used in Telecom Italia;
- UML explicitly expresses generalization and aggregation relationships.

This is in keeping with current trends in modeling notation. To represent the different types of features in the features model, the following conventions have been used for the branches representing the different kinds of features (see Figure 4):

- a simple branch is used to indicate a mandatory feature;
- a branch terminated by a hollow circle is used for an optional feature;
- a labeled branch for an alternative feature.

SPC Terminology Dictionary. The domain terminology dictionary is a collection of all the terms that occur in the domain. The dictionary ensures better communication among different teams involved in the development of the domain systems, as it allows sharing of common terminology throughout the domain and avoids problems such as using different names to express the same object or concept in different systems—or conversely, giving the same name to different concepts or objects.

Each occurrence in the dictionary is defined by the following set of elements: Name, Synonym (in the domain) and Description. An example of an entry in the data dictionary is illustrated in the following table.

Name	Synonym	Description
SPR Acquisition		Represents the SPR process phase relative to the acquisition of an SPR. Includes the verification and analysis of the SPR data, the acquisition of the information required for the completion of the SPR, the identification of the involved competencies and the related activities, necessary for the provisioning of the requested service.

3.2. Requirements Pattern and Template

A requirements pattern represents a trace of the component sub-processes characterizing the domain process or processes. The requirements pattern is directly derived from the domain Use Case model and constitutes a guide to the identification and definition of requirements templates valid for the domain. The pattern contains an introductory section that concisely describes its applicability. This section serves to facilitate the selection of the right pattern in the context of a common repository.

The pattern *SPC Service provisioning* has been defined for the process characterizing the SPC domain. This pattern divides the process into four different main sub-processes or phases: *Acquisition*; *Dispatching*; *Execution* and *Closing*.

A requirements template is a guide to the definition of new requirements, building incremental accuracy and completeness from existing requirements. The requirements template is based on both the domain use case model, from which it is directly derived, and the single-system use case models, which are used to specify variability. The requirements template is composed of a set of fixed and variable events that respectively correspond to common and different requirements related to domain use cases and single systems use cases.

The use of a requirement template consists of the following:

- reuse (as-is) of the fixed flows of events;
- specialize, if necessary, the variable flows with respect to the system under consideration;
- add, if necessary, new flows that correspond to not yet defined requirements in the domain.

The following tables illustrate an example requirements template, *SPR Acquisition*, based on the *Acquisition* sub-process of the domain pattern previously introduced. The template is illustrated in three different tables. Note that, for the sake of simplicity and due to space constraints, the behavior reported in the following tables only concerns basic event flows and no alternative paths (exceptions, faults, and so on) are reported.

The first table is mandatory and it describes the main flow of events that characterizes the generic (abstract) use case specified by the requirements template. In such a flow, variable parts are represented by means of extension points and/or parameters, derived from the use case model illustrated in Figure 4.

The second table is to be provided only if the template include extension points and describes each extension point in terms of both abstract behavior, valid for the domain, and specialized behavior, relative to the single systems of the domain. Specialized behavior corresponds as such to values that can be assigned to the extension points in order to obtain a specific use case to a particular system of the domain

The third table is to be provided only if the template includes parameters. It describes the parameters used in the template and provides possible values assignable to the parameters and derived from the domain systems.

Event No.	Description	Condition on Event
1	SSS sends to SPC an SPR execution request related to a particular SPC {P1:service type}	All data related to the requested service and necessary for the SPR acceptance are present in the SPR.
2	SPC sends to DNA a request of verification and analysis of the data included in the SPR.	
3	{EP1: Completion of SPR	

	information}	
4	SPC sends to SSS an acceptance notification relatively to the SPR previously received.	the SPR has been completed by all technical data necessary for its execution.

Extension Points

Name	Abstract Behavior	Possible Refinements
EP1	SPC asks the competent resources of information to acquire all the technical data necessary for the SPR execution; SPC asks DNA to update the SPR information with the acquired data.	<ul style="list-style-type: none"> (SPC/DT) SPC asks DNA to acquire all the technical information relative to the client connections of all the seats involved in the requested service; SPC asks DNA to update the SPR with the acquired data . (SPC/POTS) SPC asks NIMS to identify the telephone exchange responsible of the location area to which the client belongs; SPC asks DNA to update the SPR with the acquired data. (SPC/broadband) SPC asks NIMS to verify the client device connection status (“plug” or “unplug”) and in case of plug, to return the device configuration data; SPC asks DNA to update the SPR with the acquired data.

Parameters

Name	Description	Value
P1	includes information relative to the type and characteristics of the requested service.	POTS: DT: Broadband: ..

3.3. Common Requirements

The experimentation on the SPC domain described in this paper has been until now mainly focused on the SPC process. The reason for this emphasis is the nature of the systems belonging to the domain. Such systems are characterized by a significant dynamic behavior and functional aspects that dominate the information/data aspects.

The emphasis of the experimentation has been therefore concentrated on flows of activity execution; events exchanged with external systems, and business rules. The results of the experimentation were in line with our expectations and successfully confirmed a significant opportunity for reuse of dynamic and functional requirements, within the domain. This opportunity has been quantified on the basis of common requirements identified over the domain.

The number of common requirements has been computed on the basis of mandatory features identified among the domain systems and distinguished into two different types: *common features* (f_c), or features that are originally common to the systems under consideration; and *generalized features* (f_g), that are obtained by capturing, in a separate generalized feature, common parts shared among different features. This is performed by means of a generalization mechanism.

The rate of commonality over the domain has been computed on the basis of the common features (f_c and f_g) with respect to the total features characterizing the domain: f_c , f_g , f_o (optional features) and f_a (alternative features).

The rate of common process requirements over the SPC domain, as well as the equation used to compute this rate, are introduced in the following:

$$\frac{f_c + f_g}{f_c + f_g + f_o + f_a} = 30.43\%$$

It is important to state that the rate of commonality found between two of the domain systems is quite significant: 69.04%. We find it equally interesting to report the rate of commonality over the domain, calculated only on the basis of the features corresponding to originally common requirements f_c . Such a strong commonality rate confirms that SPC domain boundaries and systems have been properly defined.

$$\frac{f_c}{f_c + f_g + f_o + f_a} = 28.26\%$$

Note that the above results have been calculated on the basis of the real features model, developed within the experimentation performed on the SPC domain, of which the model illustrated in Figure 4 constitutes only a simplification.

In order to better evaluate the experimentation size and results, we provide the following relevant numbers: human resources involved in the experimentation account for a total effort of 5 man months; the total number of use cases considered in the experimentation is 108. This number refers to both the different systems as well as the domain use case models. The total number of features defined in the domain features model is 66 and refers to only a subset of the total domain use cases.

4. Domain Evolution and Reuse Check-points

As noted earlier, comparative analysis in the SPC domain presented in the previous section was carried out on the basis of requirements definitions of systems in the SPC domain that currently correspond to different individual IT2000 projects, and the analysis was carried out

in parallel with the projects.

This situation highlights another issue. In the IT2000 program, there is not always a set of pre-existing projects and system, but rather a set of projects progressing *simultaneously*. This is related to the *domain evolution* problem, which has been the subject of much recent interest [1]. The problem is addressed in FODACOM through a concept of *Reuse Checkpoints* (RCPs). These are points of synchronization along the process of system family development.

Figure 5 illustrates the concept. The boxes with rounded corners describe the development phases of the projects. The darkness of the boxes indicates the degree of coordination among the activities. The darker the box, the stricter the coordination necessary. The boxes on the right describe the domain models that are produced during the progress of the projects.

At the beginning, the projects proceed separately, each defining its own system context. Note that the domain may already possess standards before the FODACOM process is launched (e.g. quality manual, terminology dictionary, etc.). In this case, all of the projects must adhere to these standards from the beginning.

The first RCP (Context Analysis) unifies the individual context models and produces the domain terminology dictionary (which is then continuously augmented in all subsequent RCPs) and the domain context model. After context analysis the individual projects continue, performing use case analysis separately, whereby, however, all refer to the same context model and domain terminology dictionary—thus the moderate coordination indicated.

The second RCP (Use Case and Feature Analysis) unifies the individual use case models of the various projects into a single domain use case model. In addition, the domain information and feature models are produced. From here onwards the projects proceed in strict coordination, since domain analysis has identified the common requirements for all systems, as well as those that change from one system to another. The phases of modeling functional and behavioral specifications are therefore distributed among the various projects in order not to duplicate effort, with a view toward unifying the individual models into a single model.

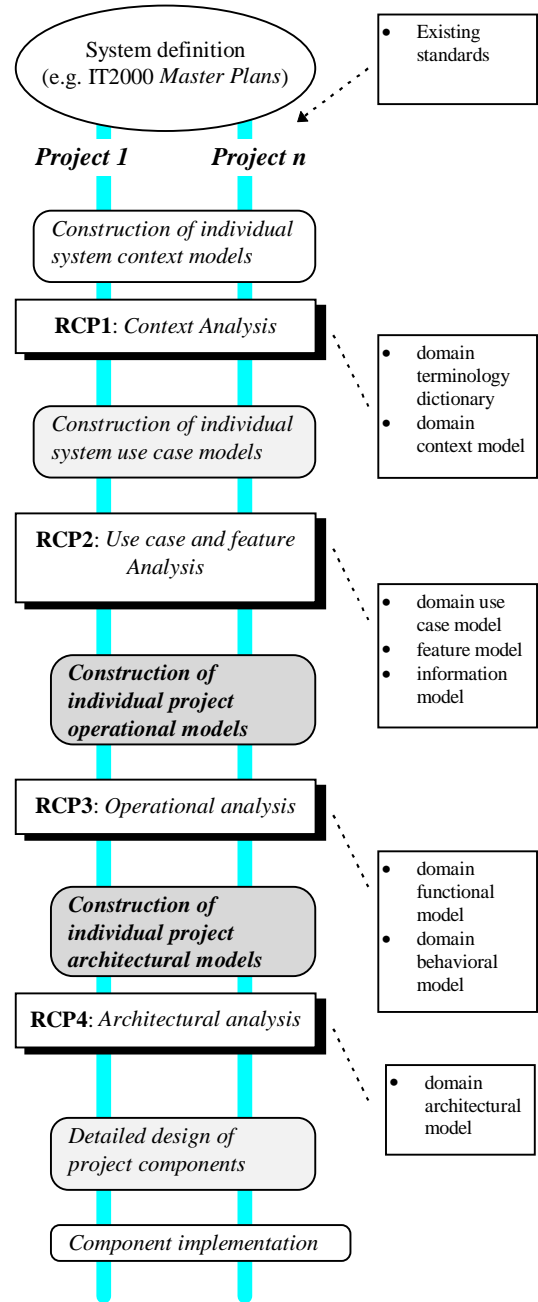


Figure 5: Forward Engineering of System Families with Reuse Check Points

The third RCP (Operational Analysis) synthesizes the work of the projects and constructs a unified operational model in which the features common to all systems have a single (functional and behavioral) specification, parameterized with respect to the variation points of the features. After this RCP the projects continue to proceed in strict cooperation, distributing among themselves the work of realizing an architectural framework for the domain operational model.

The final RCP (Domain Architectural Model) syn-

thesizes the work of the projects into a single domain architectural model. From this point onwards the projects distribute among themselves the subsequent phases of implementation. After an initial phase of detailed design of the components identified by the domain architectural model, in which the projects still work in a coordinated manner, the implementation phase continues in relative independence, having now identified in the preceding phases the common features among the various systems. Although the figure shows a linear, sequential process, in practice the phases may be repeated several times before arriving at a fixed point. For example, during the construction of the use case and feature models, it may become necessary to modify the context model.

5. Conclusions and Further Work

Even at its conception, FODA was envisioned as a methodology to be customized as necessary for specific application contexts, and our experience has confirmed this vision. FODA has provided a reliable base for the extensions and modifications that we have introduced for application in our own business units. The scope of the experimentation on the SPC domain had, from the very beginning, a set of clear objectives. These objectives have been classified according to the importance they represent with respect to the company business objectives. In the following they are listed in increasing order of importance

- Requirements Template reuse, to be applied to accelerate the process of definition of the users requirements;
- Analysis Model reuse, to support sharing of information among data and process analysts;
- Architectural Component reuse, for the development of *product lines* in the context of the same domain.

The results presented in this paper concern only the first item of the list. We discuss the others in the following.

The experimentation carried out on the SPC domain made it possible to compute the rate of common requirements within the domain. However, such a rate cannot help estimate the real reuse opportunities of design components in the domain. Results obtained from the experimentation are aimed to encourage requirements specification reuse and to influence, as much as possible, the integration of processes definitions within the domain projects. Even if not guaranteed, this influence might lead to a further revision of architectural and technological choices (if they have been already made) within the domain projects.

In this context, our experimentation taught us that the promotion of Domain Analysis products is strictly dependent on the level of coverage of FODACOM process phases. Since FODACOM's main goal is to enable the

construction of flexible architectures, if we limit reuse to requirement models, we do not fully exploit the potential to reach architectural models.

Our first experience with FODACOM highlighted interesting problems and drawbacks. The first problem occurred at the method starting point, since requirements specifications of the different systems were not organized in a use-case-like structure. A considerable amount of the total effort spent in the experimentation has been dedicated to the re-engineering of these specifications. In order to ensure a smooth introduction of domain analysis into the development process, specification models must be compatible. On the other hand, understanding process-oriented requirement models is not yet familiar to analysts.

We have also realized that, in the context of a large organization like Telecom Italia, the real problem to be faced is a matter of culture rather than methods: there should be an awareness of and agreement on the importance of the discipline behind the method. Once this is assured, method practice must be supported by tools helping to manage complexity. Having experimented with FODACOM on a real domain, we realize the need to provide and continuously enhance support for the management and maintenance of a Domain Analysis *repository*, to be implemented as an organizational support tool. In addition, OO CASE tools do not currently fully support abstraction and refinement applied to use case and feature models. This problem forced us into an awkward adaptation of a commercial tool.

The experimentation of Domain Analysis within a family of related projects has successfully exploited commonalities and captured reuse opportunities. We are now facing the challenging objective of setting up an industrial software engineering process aimed at *product line development*. The first step will be to increase the use of *requirement templates* and monitor the efficiency gains as well as the improvement in process specification quality. The second step will be to design architectural models, to measure the synergies obtained in developing product lines and to increase the core competence of our Domain Analysts and Domain Engineers.

6. References

- [1] Guillermo Arango, E. Schoen, and R. Pettengill, "Design as Evolution and Reuse," *Proc. Second International Workshop on Software Reusability*, Lucca: IEEE Press, 1993.
- [2] Guillermo Arango, "Domain Analysis Methods," in *Software Reusability*, Ellis Horwood, 1994.
- [3] Steven Fraser, "Evolving FODA: Scenarios, teams and education strategies," *OOPSLA '95*, Workshop 14, October 1995.

- [4] Ivar Jacobson, *Object-Oriented Software Engineering—A Use Case Driven Approach*, Addison-Wesley, 1992.
- [5] Ivar Jacobson, Martin Griss, Patrik Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*, Addison-Wesley-Longman, May 1997.
- [6] Kang et. al., “FODA Feasibility Study,” Software Engineering Institute, November 1990.
- [7] David L. Parnas, “On the Design and Development of Program Families,” *IEEE Transactions on Software Engineering*, VOL: SE-2, No 1, March 1976.
- [8] Suzanne Robertson and K. Strunch, “Reusing the Results of Analysis,” *Second International Workshop on Software Reusability*, Position Paper, Lucca, IEEE Press, 1993.
- [9] Paul T. Ward and S. Mellor, *Structured Development for Real-Time Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1985.