Systematic Construction and Analysis of *i** Models for assessing COTS-Based Systems Development

THESIS PROJECT

Presented to
The Departament of Llenguatges i Sistemes Informàtics of the
Universitat Politècnica de Catalunya (UPC)
Software Doctoral Program

By: Gemma Grau Colom

Advisor: Xavier Franch Gutiérrez

Gemma Grau Colom Xavier Franch Gutiérrez

Barcelona, January 2006

Table of Contents

1. Introduction

1.1. COTS-Based Systems: a New Development Paradigm

The software development field is a discipline that, although being well established, evolves quickly. This is due to the fact that the traditional software life cycle has to be adapted in order to support new development paradigms. The growing importance of Commercial-Off-The-Shelf components – hereafter COTS components – requires adapting traditional software development practices to support the assembling and integration of existent software development and the migration of existing systems towards COTS-based systems – hereafter CBS - approaches [Meyers-Oberndorf 2002], [Carney-Long 2000]. In this paradigm, software is not build from the scratch, instead different already developed COTS components are glued together. So, both the cost and the development time are reduced [Oberndorf-Brownsword 1997]. Because of this benefit, as the size and complexity of software systems grows, the interest in CBS increases.

Commonly, COTS components provide a specific functionality and are available in the market to be purchased, interfaced, and integrated into other software systems. A more formal definition of COTS is:

> *"A COTS product is a [software] product that is: (1) sold, leased, or licensed to the general public; (2) offered by a vendor trying to profit from it; (3) supported and evolved by the vendor, who retains the intellectual property rights; (4) available in multiple, identical copies; and (5) used without source code modification by a consumer."*
>
> B.C. Meyers, P. Oberndorf. *Managing Software Acquisition*, Addison-Wesley, 2001.

According to [Lauesen 2004] COTS components may be acquired in the following ways:

- Components for in-house development. The customer's IT-organization looks around for suitable COTS components, buys them and uses them in their own systems.
- Components for complex products. The customer is a product developer that integrates the COTS components into his own product.
- Business applications, company customer. The customer is a large company who wants a business application. Examples are ERP systems and banking systems.
- COTS tender for public organizations. The customer is a public organization who wants an application such as a health care system or a payroll system.

The growing importance of COTS components looks very promising for improving productivity and quality in software systems development. However, software engineering practices had to be adapted to its particular framework because some aspects of the traditional software development lifecycle are not valid in a COTS

approach. For instance, traditional approaches are intended to create the architectures needed to fulfil the native requirements of the system, but, in CBS development, not taking into account the availability of COTS components may lead to architecture not feasible with the available COTS components [Brownsword-et-al. 2000].

As a result, traditional activities such as requirements elicitation, architecture, design, implementation and testing had to be redefined in order to take simultaneous consideration of other aspects such as the system context (including system characteristics like requirements, business process involved, operating and support environments…), management issues (cost, schedule, risks), capabilities of products in the marketplace, and viable architectures and designs. Figure 1.1 shows how the traditional software development has to be reconsidered to take into account the simultaneous definition and tradeoffs of the required COTS approach.

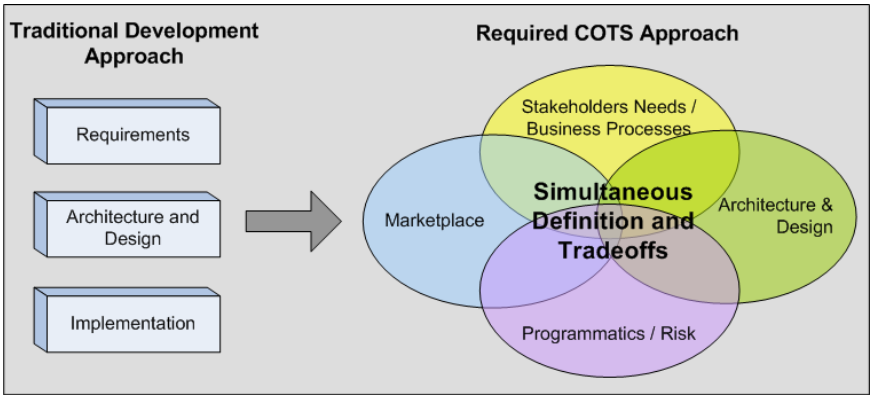

Fig. 1.1. Traditional versus COTS-based approaches [SEI-a].

The main characteristics that contribute to CBS development are:

- The increase in the quality and variety of COTS components;
- Economic pressures to reduce system development and maintenance costs;
- The emergent consolidation and growing of component integration technology;
- The increasing amount of existing software in organizations that can be reused in new systems.

However, the changes that CBS development represents in the software development paradigm entails new problems and risks [Voas 1998; Boehm-Abts 1999] that are not generally found in custom development [SEI-b], mainly because:

- The COTS marketplace grows continuously because new and improved products and technologies are continuously offered by vendors.
- Changes to products are driven by the vendors' market share goals, instead that by the unique needs of the customers.
- Each product has its own model of how the customer will use it, which may be quite different from the end user's current process.
- Licensing secures the use of the needed products; data rights and warranties protect both the customers and the vendors in the long-term use of those products.
- Vendors release product upgrades (and new products) at times that suit their marketing strategies, not following the customer priorities.
- Customers have little insight into how a COTS component has been put together and how it behaves and why.

- Just as each product has its own idea of user processes, each also has its own architectural assumptions and paradigms, making products hard to integrate.
- Some COTS components have built-in dependencies on other COTS, which amplify the impacts of product-change events.

Despite these problems and risks, the development of CBS is common in different application areas due to its economic and strategic benefits. Consequently, a wide variety of COTS components is accessible in the marketplace, which implies that COTS have to be selected and integrated to construct the new system in the most appropriated way.

As stated in [Chung-Subramanian 2001], the expected cost on CBS development is in the requirements, architecture and detailed design phases. Thus, benefits of using COTS components are in the implementation and testing phases. Because of that, the evaluation and selection of the most suitable product may be the most critical activity and it is often a non-trivial task that requires careful and simultaneous consideration of multiple criteria [Ncube-Maiden 1999; Brownsword-et-al. 2000]. Despite the selection of a single COTS component has been previously addressed [Kontio 1996; Morisio-Tsoukias 1997; Maiden-Ncube 1998; Briand 1998; Kunda-Brooks 1999], less attention has been paid in how to inform the selection of multiple components. There exists many case studies dealing with the integration of COTS components in a particular project, but it is difficult to obtain general rules on how to inform multiple components selection. As the selection of one component usually depends on the other components that have to selected, [Franch-Maiden 2003] proposes a framework for modelling dependencies between software components and applies some metrics over the generated models in order to inform their selection.

Taking this framework as the starting point, this work presents a proposal for dealing with COTS components architectures in the requirements phase in order to inform the selection of multiple COTS to be integrated in the same system. In this context, the objective of this work is twofold. On the one hand to present the state of the art of the related COTS processes and also some issues of other fields that can be applied in order to improve this processes (namely goal-oriented techniques, software architectures, agent-oriented methodologies and business process reengineering practices). On the other hand, this work presents a proposal for dealing with COTS components architectures in the requirements phase in order to inform the selection of multiple COTS components.

1.2. The Importance of the Problem

The CBS development lifecycle is composed by four main aspects that have to be defined and traded simultaneously: the stakeholders' needs and business processes; the COTS marketplace; the architecture and design of the system; and the programmatic and risks.

Some of the issues of the CBS development lifecycle are already addressed in COTS selection processes. For instance:

- Requirements vs. Marketplace. In order to deal with stakeholders requirements and the existing COTS marketplace, selection processes seeks to identify and select the

desired requirements from the already existing COTS components in the market (for instance, [Franch-Carvallo 2003]).

- Risk vs. Marketplace. One of the risks in the selection process is the growing size of the marketplace and the rapid changes of its products. This risk is mitigated by organizing the marketplace in a taxonomy such as proposed in (for, instance [Ayala-Botella-Franch 2005]).

However, some other aspects still remain an open issue in CBS development and the software engineering community:

- Requirements vs. Architecture. One key task that remains a difficult challenge for practitioners is how to proceed from requirements to architectural design [Chung-Nixon-Yu 1999]. As stated in [Bastos-Castro 2004], the terminology and concepts used for architectural decisions are quite different from those used for the requirement specification. On the other hand, is still a challenge to show that a given software architecture satisfies a set of functional and non functional requirements.
- Requirements vs. Organizational Strategy vs. Architecture. According to [Giaglis 2001], organizations should align the design of information systems with the design of the corresponding business processes, in order to obtain maximum benefits from their synergy. However, business strategy usually falls outside the scope of current requirements engineering approaches [Bleinstein-et-al. 2005].
- Architecture vs. marketplace. As mentioned before, COTS selection is a critical activity in the CBS development and, although it has been successfully addressed for the selection of an individual and independent components, few of them address the selection of interdependent components. However, in real-world applications, COTS have to be integrated with other COTS or already working systems in order to provide their functionality and, selecting a COTS without taking into account the other COTS on the CBS architecture, may end up to unsuccessful results.

Taking these open issues into account, requirements have to be market-driven; requirements also have to take into account the organization strategic needs; and requirements and architecture decisions must assist further COTS integration. This last point is especially important in the case of informing multiple components selection because, as stated in [Yakimovich-et-al. 1999], a good estimation of the integration cost can help to decide on a certain COTS solution with the most suitable COTS products, and to determine the amount and type of glueware that needs to be built.

Nowadays, system evolution is a fact of industrial life. Organizations are embedded into a competitive environment that exposes them to frequent changes. This often implies changes on their software-based systems and, as a consequence, system evolution costs are far higher than initial development costs [Rolland-et-al. 2004]. CBS are even more exposed to changes because COTS depend on vendors for upgrading and sometimes, new versions may not have the same functionalities as the previous ones, or the interoperability with the other COTS of the system may change.

In order to assess CBS development, the different aspects of the CBS lifecycle have to be considered simultaneously. Already established practices for selecting COTS components or organizing the marketplace have to be integrated in the process, but new practices are needed in order to assess the COTS architecture from the requirements stage, taking into account both the organization strategic needs and the marketplace.

1.3. Approximation to the Solution

The CBS development lifecycle takes into account multiple aspects of the process that have to be addressed simultaneously. In [Kozaczynski 2002], the relationship between requirements, architectures and risks is established: requirements drive architectures and architectures address requirements; requirements determine risks and risks prioritize requirements; and architectures mitigate risks and risks sequence architectures (see right part of figure 1.2). To that schema, we have added the marketplace and the organizational strategy to that relationship graph in order to consider all the issues of the CBS development lifecycle. Hence, in the left part of figure 1.2, we observe that the requirements influence the marketplace (e.g., the producers take into account the consumers preferences) and the marketplace constraints the requirements; requirements address organizational strategy, whilst this strategy drives the requirements process; organizational strategy influences the marketplace and evaluates risks; finally, the marketplace supports the organizational strategy (for example, ERPs) and the architecture.

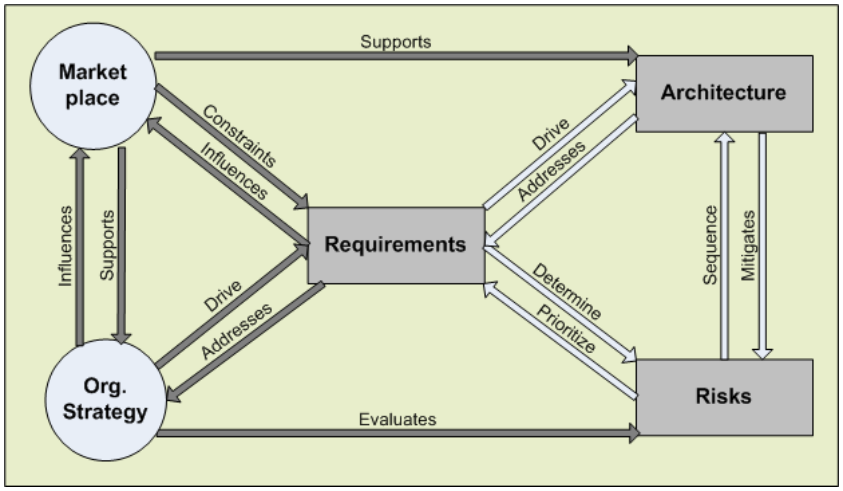

Fig. 1.2. Requirements, Architectures and Risks [Kozaczynski 2002] taking into account marketplace and organizational strategy.

The REACT framework stated in [Franch-Maiden 2003] establishes some research basis to support a process that takes into account most of the open issues mentioned in the previous section. The most significant points on the REACT framework are:

- The modelling the architecture of the resulting system by using the requirements perspective that the goal-oriented modelling provides; and
- Using the modelled architectures, REACT proposes two different treatments (at a component and architectural level) that can be applied to determine important system properties that can be evaluate over the modelled system and that will inform multiple component selection.

A software architecture can capture a system design as a set of interacting components and capture the role of COTS software in implementing certain components [Warboys-et-al. 2005]. However, traditional architectural styles tend to focus on concepts, protocols and underlying technologies but not on the business processes or the non functional requirements of the application. The *i** framework is a goal-oriented

requirements engineering approach, that allows to model a system architecture in terms of dependencies. These dependencies are established between *i** actors (e.g., COTS components, stakeholders, organizations or software) and represent goals to be achieved, softgoals to be satisfied, tasks to be completed, and provided and consumed resources. In the specific case of COTS components dependencies; [Franch-Maiden 2003] defines the following relationships:

- **Enabling their functionality.** A product from a domain requires a product from other domain to provide a given functionality (e.g., in order to follow document life-cycles, document management tools need workflow technology to define them).
- **Complementing their functionality.** A product from a domain requires a product from another domain to offer an additional feature, not originally intended to be part of its suitability (e.g., a web page edition tool can complement a web browser to facilitate the edition and modification of web pages).
- **Enhancing their quality attributes.** A product from a domain requires a product from another domain to improve its quality of service (e.g., resource utilization can be improved significantly using compression tools).

Software architecture is a field of software engineering directed at reducing costs of developing applications and increasing the potential for commonality among different members of a closely related product family [Medvidovic-Taylor 1997]. In order to achieve this goal, some techniques are needed. The REACT framework defines two different treatments:

- **Architecture-Level Analysis of the System.** To inform the selection of interdependent components, the concrete assignment of COTS component types to the candidate components modelled in *i** have to be analysed. The behaviour of the overall system is evaluated by using architecture properties that indicate feasible system architectures with respect to non-functional requirements.
- **Component-Level Analysis of the System.** To inform the selection of a particular component, concrete COTS products and dependencies for a chosen architecture have to be analysed. Three complementary treatments can be applied: first to determine feasible candidate architecture instances using component dependencies; then to explore the consequences of selecting a COTS component with respect to the system architecture and dependencies; and, finally, to investigate groups of system dependencies.

The REACT framework differs from more traditional system architecture modelling approaches because it does not require pre-emptive decision making about system architectures prior to COTS component selection. Specific contributions include:

- Highlighting the role of dependencies among types of COTS components during selection with the introduction of the *i** framework. This helps to obtain the characteristics that should be analysed in COTS candidates and to model them and explore COTS architectures.
- Separating functionality from COTS components. In the *i** framework, COTS are modelled as actors that encapsulate a certain functionality. Thus, the assignment of functions to components is systematic by exploring dependencies between agents and using some architectural properties;
- Integrating model dependencies among COTS components. Consequently, the universe of discourse for architectures and components is the same and reasoning about the result of the selection is done in a flexible way, by ranking the feasible

architecture instances, and then exploring consequences of choosing one of them by fixing and removing particular components, and so on.

The REACT framework takes into account the stakeholders needs, the COTS marketplace and the programmatic and risks that arise into the integration of COTS components; however, it stills being an ongoing work that can be extended to provide a more general framework for assessing the development of CBS. For such a framework to be general, it has to:

- include the organizational strategy as an input;
- be based on already existent requirement engineering techniques;
- be based on already existing COTS selection methodologies;
- provide a systematic method for constructing the *i** model for the CBS architecture;
- provide a set of patterns to define and apply properties over the resulting *i** model;
- define a framework for defining metrics to measure such properties; and
- provide a set of guidelines for analysing the results.

There are some works related with modelling architectures using the *i** framework and applying some measures over the resulting models. However, the goal of this works is to model and evaluate the models from an organizational point of view. In [Kolp-et-al. 2001, Kolp-et-al. 2003] software architectures are described as a social organization and some architectural styles based on organization theory and strategic alliances, are modelled. Each of the styles is evaluated with respect to a set of quality attributes by using the NFR framework [Chung-Nixon-Yu 2000]. An extension of this proposal providing a more specific methodology is proposed in [Bastos-Castro 2003; Bastos-Castro 2004]. There are some other proposals that address modelling and evaluating architectures [Clements-et-al. 2002; Kazman-et-al. 1994] from a technical point of view, but as far as we know, none of them addresses the problem as the REACT framework does. Because of that, this thesis proposal contains a method that extends the REACT method in order to solve its open issues.

1.4. Objectives of the Research

The general objective of this thesis work is:

> *To provide support for assessing the construction and analysis of i* models in order to inform the development of CBS.*

This research is aimed to advance the state of the art in the study of methodologies for developing CBS, focusing on the assessment of those CBS that require to integrate more than one COTS or the integrate COTS into already existing systems. Such assessment is performed in the requirements engineering phase and includes the generation and evaluation of alternative CBS, based on the system requirements and the most suitable solutions. Because of that, CBS best practices are taken into account, and techniques of requirements engineering and evaluation of software architecture are also considered.

From this perspective, special attention is given to goal-oriented approaches because several research performed so far [Chung-Nixon-Yu 1999; Franch-Maiden 2003], points them as a good candidate for representing architectures, generate alternative

architectures and evaluate them. Architectures are represented by using the *i** framework. As *i** is also used in agent-oriented approaches, the study of agent-oriented methodologies is useful for assessing the proposed methodology. Finally, business process reengineering techniques are analysed in order to provide a consistent basis to the whole proposed methodology.

As a result, it is expected to answer some practical questions that arise often during CBS development:

- Which are the aspects that have to be considered to assess CBS development?
- Which are the relationships among COTS marketplace, requirements, architectures and risks that can be useful to assess CBS development?
- Which lessons learned could be helpful to this particular activity?

These practical questions can be translated into the next research questions:

- How can the organizational strategy, the COTS marketplace and the user requirements on a CBS be synthesized?
- How can this information be used to build *i** models in a prescriptive way?
- How can *i** models be evaluated in a reliable way in order to assess the CBS development?

Thus, besides the general objective presented above, some side-off objectives exist for answering these research questions:

1. To explore the applicability of the *i** framework to model CBS architectures.
2. To explore the applicability of the *i** framework for generating and evaluating candidate COTS architectures.
3. Based on the previous objectives, to propose a methodology to assess CBS development by constructing and analysing *i** models of candidate CBS architectures.
4. To explore the applicability of metrics and other reasoning techniques to support the evaluation of *i** models of CBS architectures.
5. To provide a framework for the definition of metrics over *i** models.

1.5. Structure of this Document

- Section 2 summarizes the proposal solution based on the objectives and the state of the art that are stated in Section 1 and Section 3 respectively.

- Section 3 presents the state of the art related with the problem to be solved. There, the main issues related to COTS Components and Evaluation of Software Architectures are presented. Some other fields that contribute to that research are also explained, namely the *i** framework, agent-oriented methodologies and business process reengineering.

- Section 4 shows the research planning to fulfil the goals of this thesis project. The different activities are divided into a three-iteration process. An initial phase 0 is also explained, as well as the detailed results expected for the other iterations.

- Section 5 summarizes the research performed up to date and which activities will be performed for getting further research objectives.