Requirements Engineering for COTS based Systems

Colette Rolland

Requirements Engineering for COTS Based Systems

Colette Rolland
Université de Paris1 Sorbonne
UFR27, CRI
90 rue de Tolbiac
75013 Paris

Abstract

In spite of the increasing use of COTS products for system development, there is little consideration on how to acquire requirements for COTS products, how to select COTS components and how to assemble them to comply to these requirements. The paper addresses the issue of the requirements engineering process for COTS components acquisition and assembly. It proposes an approach based on the notion of requirements maps and assembly strategies and demonstrates the approach with the selection of a CASE tool.

1. Introduction

In the hope of reducing risks and costs associated with software development, organisations that rely on software systems are increasingly shifting from bespoke development to purchasing commercial off-the-shelf (COTS) products. As pointed out in [Finkelstein96] and [Ncube99], there is little or no systematic support available to guide selection and assembly of COTS components. Whereas attention being paid to COTS acquisition focuses on considerations such as functional capabilities, interfaces, architectural issues, and interoperability with legacy systems [Feblowitz99], another important consideration is the requirements engineering process to acquire customer requirements to select and assemble COTS components in order to comply to these requirements.

This paper emphasises the issue of the requirements engineering process for COTS based systems and proposes an approach to address this issue. The approach departs from the traditional process for identifying suitable components based on an indexing mechanism which allows to query component descriptors (meta-classes) and retrieve the relevant ones for the problem at hand. According to Frakes [Frakes94] indexing mechanisms can be broken into two main categories : with a *controlled vocabulary* and based on an *uncontrolled vocabulary*. The former limits the terms that can be used to index components and the way terms can be combined as well. Acceptable and unacceptable terms are listed in a thesaurus. In the latter, enumerated values, facets and keywords are used for indexing components In an enumerated classification, the subject area is broken down into hierarchical classes. The hierarchical organisation helps users to understand the relationships among indexing terms, and facilitates searching. In a faceted classification, the subject area is analysed to determine discriminating facets [Prieto-Diaz87] and components are indexed with facets values. Indexing components with a set of uncontrolled keywords was experimented with in the Ithaca project [Ader90, Bellinzona93].

There are several drawbacks of this way of working. First, the retrieval process is component based whereas it should be requirements driven thereby making it difficult to ensure that requirements are met. Second, the selection is performed on an individual component basis which makes it difficult to evaluate how components fit together and fit globally the full system requirements. Third, the global picture of the system is achieved a posteriori, i.e. when all components have been retrieved and assembled whereas the process should be driven by a full picture of what is expected from the future system.

Our view is that of *a requirements driven process*. On the one hand the organisation has system requirements which are to be engineered. On the other hand, there are requirements imposed by the usage of components. Thus, we propose to view the process as a matching process between the requirements posed on the system on one hand and the requirements (constraints) imposed by the usage of components on the other hand. First the application engineer has to get a picture of these two sets of requirements and then proceed to the selection of components whose usage requirements match the system requirements. The approach proposes to represent both kinds of requirements as directed labelled graphs that we call *requirements maps.* The map uses two fundamentals notions, intention and strategy. An intention captures the notion of a service that the system or the component intends to provide whereas the strategy is the manner in which the intention can be achieved. The

nodes of the map are intentions whereas its edges are labelled with strategies. The directed nature of the map identifies which intention can be done after a given one. When used to express the system requirements the map provides a strategic view of what is expected from the system in terms of services and strategies to achieve them. When used to describe the component requirements, the map provides a strategic view of the process for using these components. We believe that this strategic view helps in the matching process to evaluate how components fit into the requirements of the system to be developed. Furthermore, maps can be defined recursively, a map can be refined in several other maps. This shall help mastering the matching process step by step. If a component meets the requirements at a level *i*, a refined view through a map at *level i+1*, shall help identifying finer grained matching or absence of complete match, then requiring adjustments and adaptations of the components and/or the development of missing components. The remainder of the paper is in three sections. The next outlines the requirements driven process. Section 3 introduces the notion of a map whereas section 4 demonstrates the use of the approach in the case of scenario based COTS products.

2. Overview of the requirements driven process

Figure 1 is a graphical depiction of the process showing the four key goals for effective COTS components retrieval and assembly compliant to the organisation requirements. For achieving each of these goals, the approach prescribes four processes : (a) *Construct As-Is Map*, (b) *Construct To-Be Map*, (c) *Construct COTS Map* and, (d) *Integrate Maps*. The approach is inspired and extends the traditional view of change handling [Jackson95]. Whereas it recognises the role of the *As-Is* model (describing according to Jackson, the indicative properties) and the *To-Be* model (describing the optative properties), it introduces the *COTS* model and the *Integrated* model, all represented as *maps*. The *As-Is map* abstracts from the organisation current practice to describe the currently achieved goals/requirements. It serves as a support for critiquing the current situation and for identifying requirements for the future captured in the *To-Be map*. This map reflects what the organisation would like to achieve by the COTS based system acquisition whereas the *COTS map* expresses the goals/requirements that can be achieved by the COTS components usage. Finally, the *Integrated map* reflects what the future system will really be; it tells which requirements will be fulfilled by the COTS components assembly, which will be fulfilled by specific, 'in-house-made'components and which will be left out.

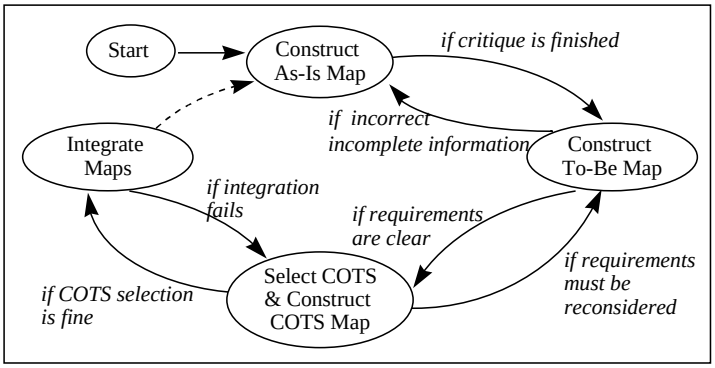

Figure 1 : The requirements driven process

As shown in the figure, the process might be iterative, each cycle corresponding to the selection and assembly of COTS components matching more or less the requirements. Iterations can be justified either because the match does not comply enough with the requirements or because more information is required in terms of requirements.

3. Maps and Components

A map is a process model in which a non-deterministic ordering of *intentions* and *strategies* has been included. It is a labelled directed graph with intentions as nodes and strategies as edges between intentions. The directed nature of the graph shows which intentions can follow which one. An intention is a goal that can be achieved by the performance of the process. For example, in the L'Ecritoire COTS map shown in Figure 2, there are two intentions (in addition to the *Start* and *End* intentions that starts and ends the process, respectively), *"Elicit a Goal"* and *"Conceptualise a Scenario"*. L'Ecritoire is a software product to support scenario-driven requirements engineering [Rolland98]and the two intentions of the map represent the goals that can be achieved by using the product. A *strategy* is an approach, a manner to achieve an intention. The strategy S_{ij} characterises the flow from the source intention I_i to the target intention I_j and the way I_j can be achieved. For example, *"initial*

goal identification strategy", *"alternative discovery strategy"*, *"variant discovery strategy"* are strategies representing three different manners to achieve the intention *"Elicit a Goal"* in the L'Ecritoire map.

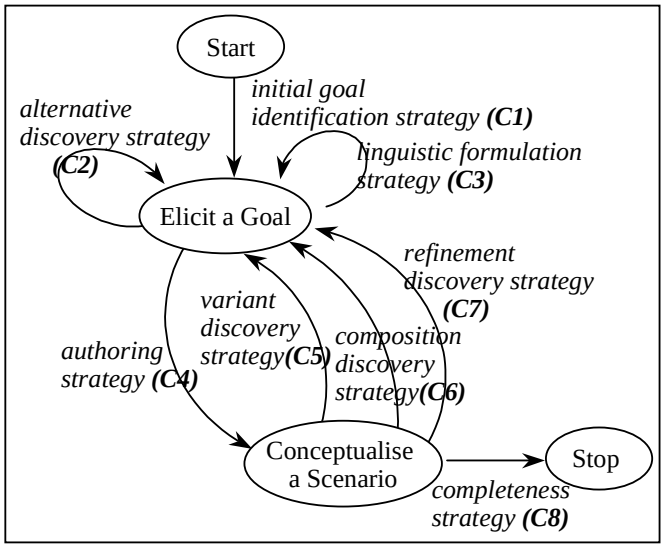

Figure 2 : L'Ecritoire COTS Map

These strategies have different source intentions but the same target intention. A triplet $<I_i, I_j, S_{ij}>$ in the map is called a *section*. The triplet *<Elicit a Goal, Conceptualise a Scenario, authoring strategy>* is an example of section in the L'Ecritoire map. Each section of the map captures the specific manner in which an intention can be achieved and is associated with a component. Table 1 briefly describes the eight components of the L' Ecritoire COTS product associated with the eight sections of the map as indicated (with component's reference) in Figure 2. The description of the eight components in Table 1 shows that each component has an *interface* and a *body*. The body is what the component really does whereas the interface is the visible part of the component. The body of the component C2 for example, generates semi-automatically alternative design options formulated as goals. Its interface is a couple <situation, intention> stating the precondition for the component to be used (the situation that an initial goal has been elicited) and the intention that can be fulfilled in that situation (to 'Elicit a Goal') following a given strategy, namely the 'alternative discovery strategy'. Thus, there is a tight connection between the components and the map : (a) each section in the map is associated to a component; (b) the interface intention of the component is the target intention of the section completed by the name of the section strategy and (c) the interface situation refers to the state resulting of the fulfilment of the section source intention.

Reference	Component Name	Component Interface	Component Body
C1	Initialisation	<(Pb.St.[1]), Elicit a Goal with initial identification strategy>	Proposes to the Requirements Engineer (RE) heuristics to identify the initial system goals.
C2	Design options Discovery	<(Goal), Elicit a Goal with alternative discovery strategy>	Suggests to the RE the following process : (1)Rephrase the informal description of the given goal according to the provided goal template, (2) Identify the possible alternative values for each parameter, (3)Compute all possible combinations of parameters, (4) Evaluate the new goals and select the goals of interest.
C3	Linguistic Formulation of a Goal	<(Goal 'elicited'), Elicit a Goal with linguistic formulation strategy>	Guides the RE rephrasing the informal formulation of the given goal according to a goal template
C4	Scenario Authoring	<(Goal' formulated'), Conceptualise a Scenario with authoring strategy>	See Figure 4
C5	Variations Discovery	<(Goal 'formulated', Scenario' conceptualised'), Elicit a Goal with variant discovery strategy>	Applies two techniques to semi automatically discover goals; one based on flow conditions of the scenario, the other one using generic

			exception types.
C6	Composition Discovery	<(Goal 'formulated', Scenario' conceptualised'), Elicit a Goal with composition discovery strategy>	Uses two ways for discovering goals ANDed to a given goal G. The first is based on the final and initial states of the scenario describing a behaviour to fulfil G. The second reasons about the scenario resources and their production and consumption by the system.
C7	Refinement Discovery	<(Goal 'formulated', Scenario 'conceptualised'), Elicit a Goal with refinement discovery strategy>	Performs two techniques, one based on scenario action completion and the second considering every action in a scenario as a goal.
C8	Completeness Verification	<(Goal, Scenario), Stop with completeness strategy>	Performs tests to check the completeness of the req. specification and guide their correction.
[1] Pb.St.	Problem Statement		

Table 1 : List of L'Ecritoire Components

The map provides a strategic view of what components can achieve individually and assembled together as well. The former is captured in a section of the map and the related component interface whereas the latter is captured in the flows from intention to intention via strategies. The view is strategic in the sense that it abstracts from the detail on how tactically, the component is able to achieve the intention. This eases we believe, to concentrate first during the requirements engineering process at the level of intention matching (to understand if a component fits the requirements) and strategy matching (to understand if the strategy to assemble components in order to meet a global objective is valid or not). In order to allow such strategic reasoning at different levels of abstraction a section in a map (a component) might be itself deployed as a map. This is illustrated in Figure 3 with the L'Ecritoire section <*Elicit a Goal*, *Conceptualise a scenario*, *authoring strategy*>, i.e. the C4 component.

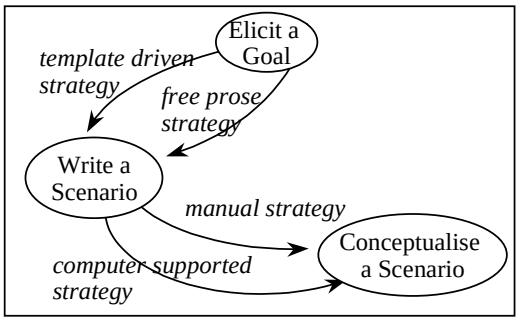

Figure 3 : The C4 component map

This might help refine the selection once the C4 component for scenario authoring has been selected. One option could be the *free prose strategy* to "*Write a scenario"* (i.e. a narrative prose) followed by *the computer supported strategy* to "*Conceptualise* it" (provides automated linguistic devices for the analysis, verification and transformation of narrative scenarios). Another option would be the *template driven strategy* to "*Write a scenario"* followed by a *manual strategy* to "*Conceptualise"* it. The former can be justified if requirements engineers do not want to be constrained even if getting a complete and unambiguous scenario at the end is required. The latter will be appropriated to the case of experienced requirements engineers for example. This brief example shows the potential of having explicit strategies in the map to reason on the assembly of components matching organisation requirements. The matching process is further considered in the next section with an illustration of the entire proposed requirements driven process.

4. Illustration of the requirements driven process

Figures 4a and 4b show respectively the *As-Is map* and *To-Be map* of our case study. Figure 2 discussed above and its extension in Figure 3 correspond to the *COTS map* while Figure 4c gives the *Integrated map*. The case study deals with the acquisition of a COTS to support a team of requirements engineers in their activity. As the

company mainly develops socio-technical systems, it has already oriented its choice towards use-case based approaches in order to take into account the interactive nature of these systems.

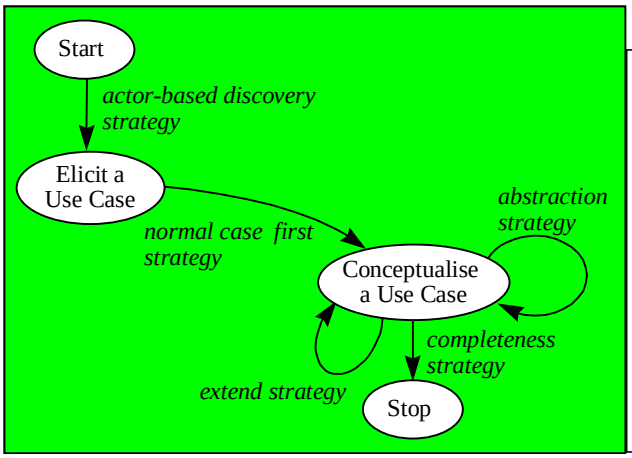

Figure 4a : The As-Is map (OOSE map)

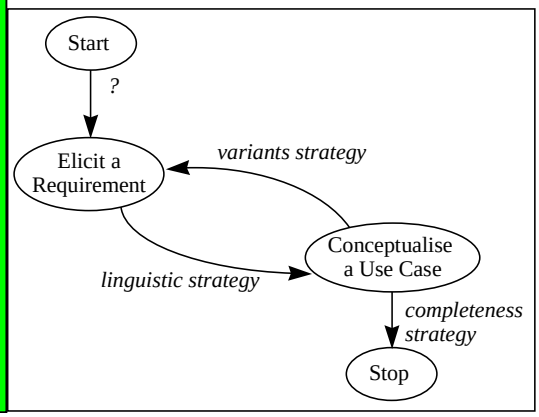

Figure 4b : The To-Be map

In the current practice, the requirements engineers use the OOSE approach [Jacobson92] abstracted in the As-Is map of Figure 4a. The small number strategies available in the map reflects the sequential nature of the process suggested by the approach and the monolithic dimension of the OOSE component. This was found to be a limitation in practice. Furthermore, the limited support provided to "*Conceptualise a Use Case*" creates difficulties and should be improved by the acquisition of the COTS product. In particular, support should be provided to identify the different variants of a use-case including the exceptional cases as well. The narrative form of use-case scenarios was found adapted to the profile of the stakeholders involved in projects but a linguistic support is required to check and improve them. Finally, the team finds necessary to distinguishing requirements and use cases. All these critiques based on the As-is practice lead to identify the requirements for a future COTS selection. These requirements are expressed in the To-Be map of Figure 4b.
Based on these requirements, the L'Ecritoire COTS product was found to be a candidate for acquisition. Its map is shown in Figure 2 and its integration with the To-Be map including some relevant parts of the As-Is map is shown in Figure 5.

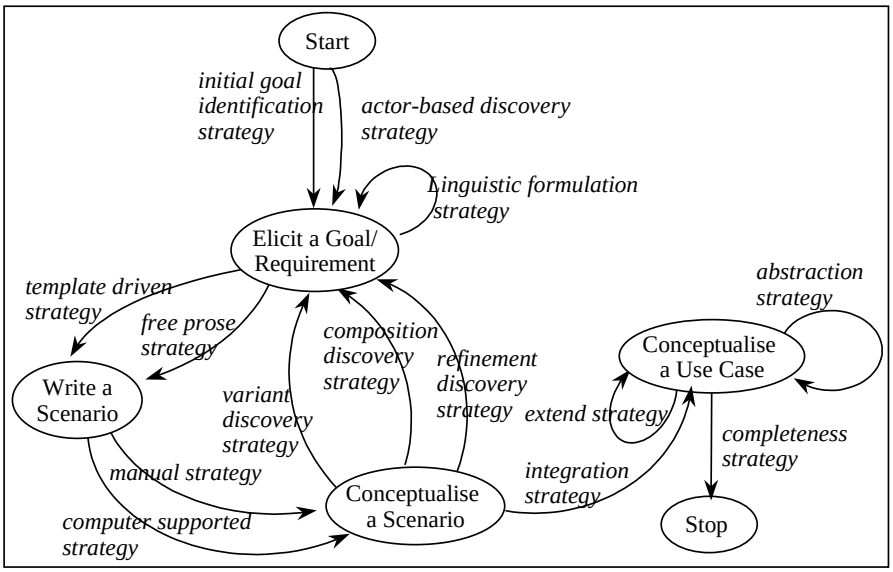

Figure 5 : Integrated map

Assuming that a requirement may be formulated as a goal, the intentions "*Elicit a Requirement*" (To-Be map) and "*Elicit a Goal*" (COTS map) were integrated into "*Elicit a Goal/Requirement*". The in depth analysis of the *authoring strategy* of the COTS map (cf. Figure 3) confirms its matching with the *linguistic strategy* requirement in the To-Be map. This led to the introduction of the intentions "*Write a Scenario*" and "*Conceptualise a Scenario*' in the integrated map. The computer support offered by the COTS product to deal with the interpretation, verification and transformation of narrative scenarios was appreciated and the *computer supported*

strategy and *free prose strategy* were introduced in the integrated map. However, even if most of the situations the requirements engineers deal with correspond to the free prose strategy and computer supported strategy, the flexibility of carried out by alternative ones was found useful and thus, the *template driven strategy* and *manual strategy* were kept in the integrated map.

Now, clearly, "*Conceptualise a Scenario"* is not equivalent to "*Conceptualise a Use-Case"*. To understand the difference it was necessary to reason on the concept model underlying the COTS product on one hand, and the To-Be map on the other hand. The finding was that the set of couples <goal, scenario> called RCs (Requirement Chunks) related through OR relationships of L'Ecritoire is equivalent to the use-case concept (see Figure 6). However whereas the various scenarios of a use-case are integrated in a single description in the OOSE and To-Be map, there are treated as separate entities in L'Ecritoire. Therefore, in order to keep the integrated use-case view, the "Conceptualise a Use-Case" intention was connected to the "Conceptualise a Scenario" intention with the integration strategy in the integrated map. Obviously this strategy is neither provided in the COTS product nor in the OOSE map. The corresponding component namely, the one associated to the new section < *Conceptualise a Scenario, Conceptualise a Use-Case, integration strategy* > will have to be 'home made'.

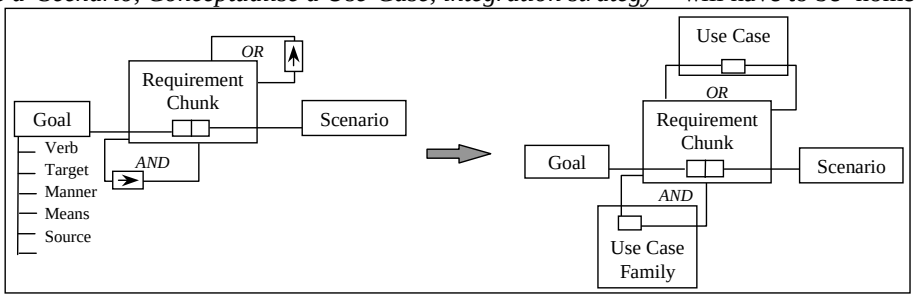

Figure 6 : Equivalence of concepts

A close look at the three strategies : the *refinement discovery strategy*, the *composition strategy* and *the variant discovery strategy* of the COTS map confirms that not only they match the *variants strategy* of the To-Be map but provides more than expected. The *composition strategy* in particular seems to solve a recurrent practical problem which was to be sure that the full functionality of the system was described in the OOSE use-case model.

Merging the "*Start''* and "*Stop"* intentions led to (1) keep the *initial goal identification strategy* and *actor-based strategy* as alternative way of getting the first goal identified and, (2) the *completeness strategy* as the means to end the requirements engineering process.

Finally, the *linguistic formulation strategy* was kept in the integrated map as support for getting goals/requirements clearer as early as possible in the process was felt necessary. Vice-versa the *alternative discovery strategy* was not maintained in the integrated map as the requirement for reasoning about design options is not of a high priority.

This informal discussion of the process to *'Integrate Maps'* intends to demonstrate how maps support reasoning on how COTS services expressed in the COTS map match organisational requirements captured in the To-Be map. It was also shown that the integration activity helps refining initial requirements while discovering functionality offered by the COTS product (e.g. the decomposition and refinement strategies offered By the COTS product were not part of the To-Be map and nevertheless were found relevant by the RE team). On the other hand, the construction of the integrated map may raise new requirements that imply 'home-made' development (e.g. the integration strategy to "*Conceptualise a Use-Case*").

In order to support formally this process, we have defined two sets of operators. The first one includes operators such as RENAME-INTENTION, MERGE-INTENTION, ADD-SECTION, REMOVE-SECTION, MERGE-SECTION etc. applicable to the integration of maps. Operators of the second set such as RENAME-CONCEPT, MERGE-CONCEPT, OBJECTIFY-LINK, GENERALISE-CONCEPT, RENAME-CONCEPT etc. operates on concepts underlying maps. For example, the OBJECTIFY-LINK was used in Figure 6 to demonstrate the equivalence between the Use-Case concept and the set of ORed RCs of the COTS product. These operators are developed in detail in [Ralyte99].

5. Conclusion

Requirements engineering for COTS components selection and assembly is an issue that has been neglected by current methods of developing systems from commercial off-the-shelf software. In addressing this issue, the paper intends to demonstrate that the selection and assembly of components should be performed in an

interleaved manner to ensure that the matching with organisation requirements is not restricted to the individual components but occurs also, globally. To support the matching process, the paper proposes the notion of requirements map that is a graph of intentions and assembly strategies. Matching COTS and organisation intentions ensure that components meet the requirements whereas matching strategies help understanding the global match. The matching process is placed within a framework inspired from change management comprising the As-Is model, the To-be model, the COTS model and the match (integrated) model, all represented through maps. The approach is a first attempt to reason systematically about the selection and assembly of a set of components meeting the requirements of an organisation. A weakness of the approach is in the process guidance. To overcome this shortcoming a set of situated rules which will be fired to 'situate' guidance are being specified. In the long term, we hope to abstract from experience a set of generic requirements critiquing strategies and assembly strategies together with a set of generic matching patterns that we could make publicly available in a library that could be improved over time.

6. References

[Ader90] Ader M., Nierstrasz O., McMahon S., Mueller G., Proefrock A-K., The ITHACA technology: A landscape for object-oriented application development, Proc. Esprit'90 Conf., Kluwer Academic Publisher, 1990

[Bellinzona93] Bellinzona R., Fugini M.G., de Mey V., Reuse of Specifications and Designs in a Development Information System, Information System Development Process (A-30), N. Prakash, C. Rolland and B. Pernici (Editors) IFIP, Sept. 1993.

[Feblowitz99] Feblowitz M., Greenspan S. Reubenstien H. Walford R., ACME/PRIME : Requirements acquisition for process-driven systems. In : Proc. Of the 8th International workshop on Software Specification and Design, IEEE Computer Society Press, Washington, DC, 1996, pp 36-45.

[Finkelstein96] Finkelstein A., Spanoudakis G. Ryan M., Software package requirements and procurement. In : Proc. of the 8th International workshop on Software Specification and Design, IEEE Computer Society Press, Washington, DC, 1996, pp 141-145.

[Frakes94] Frakes W., Pole T., An empirical Study of Representation Methods for Reusable Software Components, in IEEE Transactions on Software Engineering, Vol. 20, No 8, August 1994.

[Jackson95] Jackson, M. (1995) Software requirements and specifications - A lexicon of practice, principles and prejudices. Addison Wesley Press, 1995.

[Jacobson92] Jacobson, I., Christerson, M., Jonsson, M., and Oevergaard, G., Object Oriented Software Engineering: a Use Case Driven Approach , Addison-Wesley, 1992.

[Ncube99] Ncube, C., Maiden,N., Guiding parallel requirements acquisition and COTS software selection, Int. IEEE Conference on Requirements Engineering, Limerick, Ireland, 1999.

[Prieto-Diaz87] Prieto-Diaz R., Freeman, P., Classifying software for reusability, IEEE Software, Vol. 4, No. 1, Jan. 1987

[Ralyte99] Ralyté, J., Rolland C; Plihon V., Method Enhancement by Scenario Based Techniques", in Proceedings of the 11th Conference on Advanced Information Systems Engineering, Springer Verlag, Heidelberg, Germany, June 14-18, 1999.

[Rolland98a] C. Rolland, C. Souveyet, C. Ben Achour. "*Guiding Goal Modelling using Scenarios*", IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, 1055- 1071, Dec. 1998.