How Agile COTS Selection Methods are (and can be)?[1]

Fredy Navarrete, Pere Botella, Xavier Franch
Universitat Politècnica de Catalunya
{fjnavarrete, botella, franch}@lsi.upc.edu
http://www.lsi.upc.edu/~gessi

Abstract

Agile methods are proposed nowadays as a way to support software systems procurement. Most of the existing proposals such as eXtreme Programming or Scrum seem to conceive software procurement as an exercise of software development. However, a great deal of software systems are Commercial Off-The-Shelf (COTS)-based systems, in which the focus changes from bespoke software development to COTS selection and integration. Many proposals for COTS selection have been issued and therefore one may wonder how do they behave from the agile point of view. In this paper, we study the agile principles in the context of COTS selection and we analyze some of the most widespread existing methods. As a result, we identify some practices that would help in making COTS selection processes more agile.

1. Introduction

Agile methods [1, 2] are playing an increasingly important role in today software engineering practices. Methods such as eXtreme Programming (XP) [3], Scrum [4] and others have been adopted by a great deal of organizations and teams, and reported to be successful in many experiences (and not so successful in others, as happens with all methods).

However, in our opinion agile methods currently suffer from a bias problem: they focus mainly on in-house software systems, that is systems that are developed by a team of programmers in which reusability is limited to software component repositories basically handled by the team itself (or another part of the same organization). If we consider for instance XP, practices such as pair programming can be difficult to extrapolate to a world other than software-development-intensive systems.

This perspective leaves out a big portion of the software market: as reported by professional consultant companies such as Gartner or IDG, today's software systems procurement is mostly an activity of: searching one or more appropriate software packages (which are called Commercial Off-The-Shelf –COTS– components) in the marketplace; writing the contracts for their acquisition; customizing and integrating them; and handling the marketplace constant evolution by integrating new releases of selected packages, updating technologies, etc. In fields such as cooperative information systems or communication infrastructure, it is hard to think about developing systems in-house instead of following this acquisition-based process.

Therefore, a question that immediately arises is whether agile methods can be applied in the COTS world and therefore the benefits presented in [1, 2] achieved. Our paper is a contribution for solving this open issue. We have identified two stages in our research: first, we focus on the agility of local COTS-related processes, and next on the agility of COTS-based development as a whole. In this paper we concentrate in the first part of COTS-based development, namely COTS selection. In fact, it is natural to tackle first this stage not only for the temporal ordering but also because it is the COTS-related activity in which we may found more contributions in the form of comprehensive methods.

The rest of the paper is structured as follows. We first identify which agile principles have to be with COTS selection and study them one by one (section 2). Next, we analyze some of the most widespread COTS selection methods in the light of these principles and identify their agile and non-agile practices (section 3). Then we list some practices whose adoption could improve the agile perspective of COTS selection methods (section 4). We finally give some related work and the conclusions of our work (section 5).

[1] This work has been partially supported by the Spanish project TIN2004-07461-C02-01.

2. Agile principles in the context of COTS selection

In order to examine the most commonly accepted agile principles, we take as a basis the so-called "Manifesto for Agile Software Development" [5], more precisely, the "Principles behind the Agile Manifesto". After a first revision of those 12 principles, we discard some that either do not apply to, or not depend on, the COTS selection context:

"*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*".
"*The most efficient and effective method of conveying information to and within a development team is face-to face conversation*".
"*Working software is the primary measure of progress*".
"*The best architectures, requirements, and designs emerge from self-organizing teams*".

The first and third principles, are non-applicable in this paper because we are focusing on COTS selection and not the whole development cycle, whilst the second and fourth principles, seem not to be influenced by the COTS-based nature of the system.

In the rest of the section we examine the other agile principles that apply to COTS selection. The words in ***bold*** are considered to be the key words of the principles.

P1 *"Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage"*

COTS components are usually acquired in (or licensed from) the marketplace, and the marketplace is huge, with a great deal of information that is discovered whilst selection progresses, and is in constant evolution and change, even during the selection process itself if it takes months. This will force us to contemplate requirements for COTS-based systems to be flexible in order to capture the current state of the marketplace. In [6], it is mentioned that the 31% of the studied projects point out the need to make flexible the requirements in the definition phase.

Flexibility can be supported in several ways. On the one hand, besides considering the "what" of the features required on the COTS-based system, it is convenient to consider the "why" [7, 8], i.e. the goals behind the requirements. In COTS-based projects, the goals remain more stable throughout the project and the requirements, which can change, are elaborated to satisfy those goals [9].

On the other hand, the selection process should recognize explicitly the intertwining among requirements engineering and marketplace exploration: new requirements force the exploration of a bigger part of the marketplace, and in this process some interesting features may be discovered and incorporated to the system requirements.

P2 *"**Deliver** working software **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter time scale."*

Larman and Basili [10] showed that the idea of Iterative and Incremental Development (IID) is not something exclusive of the agile world, since IID has been present in several well-known process paradigms dating from several years (as the spiral model or the prototyping approaches). The central idea in the Unified Process (UP), the UML "official" process model, is the iterative development [11], in which each iteration includes several disciplines (Business Modeling, Requirements, Analysis&Design, Implementation, Test, and Deployment), at different percentages, in a way that every artifact produced evolves in maturity trough the iterations.

To apply this principle to the development of COTS selection, we can see the selection process as iteration-based, including, as in the UP, several disciplines in each iteration at different percentages, as: marketplace exploration, requirements analysis, COTS evaluation, and so on. In each iteration, we can progress either by selecting better or by selecting more.

Integration is an obstacle in this iterative view of COTS selection processes. An important problem that collides with the iteration is the possible existence of strong dependencies between different COTS components when selection is multiple. In that case, the incremental iterations have to take into account those dependencies whilst the architecture is being defined, and integration requirements play an important role [12].

P3 *"Business people and developers must **work together** daily throughout the project."*

In a conventional in-house software development project we have two main actors that cooperate: business people and developers, but in COTS-based systems a third actor appears: the COTS vendor (or supplier). A high percentage of the functionality of the system will come from the COTS components. The strong dependency that exists on the vendor, seems to point out his/her inclusion within the development team (see section 4 for more details). This possibility may not be feasible for components distributed massively but possible, and in fact a current practice, in other cases. Several methods such as those reported in section 3 refer to the importance of this dependency on the vendor. We think that this inclusion can be a win-win situation: the vendor can obtain benefits on learning about our project and about the integration capability of his/her product [6] and, on the other hand, the organization that delivers the system (hereafter, system provider organization) has the option to customize the COTS component and to obtain better assistance [13] (specially if the client is important). Some characteristics, such as the type of COTS component and/or vendors, the importance of the client, the budget of the project, etc., may or may not allow this inclusion.

The nature of the process also makes other actor important: the lawyer or, at least, some expert in regulations and laws. COTS selection ends up with the writing of licenses and contracts for the selected components, which must protect as much as possible the client (and also the system provider) when the selected products show some ill-functioning feature as well as making clear how product evolution will be handled. Having this actor will make the selection process a true team game.

P4 "*Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done"*

The information systems are designed and used by humans, which causes the human factor to play a preponderant role. The people and the culture of the organization are crucial on the use of the system [14], the good relations within a work team, the internal ability of communication and the different interactions with other team components. All of this influences the business process, so, we must worry to understand how human and organizational factors affect the development of our project [15]. One of the basic characteristics about the agile team is the emphasis in human factors such as: amicability, talent, skill, and communication [2]. To do so, in addition to those factors that are not specific of the COTS world, another key factor is to identify the appropriate roles that play a part in the process. In the case of COTS selection, the processes and the activities involved generate new responsibilities and new roles, making significant and very important the interaction of these roles within a team. The team must be based on the ability and verified knowledge of its members. Also, the COTS selection process should be adaptable to the specific characteristics of the system provider; for instance, using advanced techniques with not-so-skilled technicians may have serious consequences.

P5 "*Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely*."

The job stream during selection does not have to be excessive, to maintain a constant charge that neither debilitate nor deteriorates the internal pace of the team. Therefore the flows of internal processes must be constant, iterative, and allowing constant feedbacks, that is, to make iterative selections of components, iterative evaluations, iterative refresh and updates, applying feedbacks in each phase.

Two artifacts that may play an important role here are the system architecture and a repository of information. System architecture may be used as the cornerstone around which selection takes place. Repositories may contain lots of different information: about suppliers, components and requirements, but also about the processes themselves, as remarked in principle P8 below.

P6 "*Continuous attention to **technical excellence** and good design enhances agility."*

High quality is the key to high speed [1]. In each phase of selection, the involved technical people must be committed to give results of high quality, being clear in the specifications of the user requirements, in the characteristics of the component candidates, in the results of the evaluations, among other possible results. All of these fields have lots of background: techniques like goal-oriented modeling [7, 9] and win-win negotiation [16] in requirements engineering; multi-criteria decision techniques, AHP [17] and others for prioritizing requirements and also evaluation criteria; etc. Needless to say, quality of the COTS products themselves must be assessed appropriately for them being accepted as final result of the selection.

P7 "***Simplicity**--the art of maximizing the amount of work not done--is essential.*"

This is traditionally a principle difficult to reconcile with others. Consider for instance P6. Of course, technical excellence means the use of rich models that may be difficult to write down. The only way to put simplicity and technical excellence together is to focus on the appropriate candidates to invest most of the effort on their thorough evaluation and not on non-competitive ones, and to focus on the relevant requirements that really discriminate among candidates and therefore to discard irrelevant evaluation criteria.

Another conflict appears when considering sustainability (P5) and reflection (P8). These two principles require somehow to invest an effort beyond the simple selection process. Documentation, data gathering, and so on, requires some extra work that seems to hamper the simplicity principle. The key point as usual is: think on the future just when this future may happen. One-shot selections should not require the heavy use of documentation, for instance, since it is not needed for the immediate benefit of the stakeholders. In the COTS world, another point against doing much is the high evolvability of the marketplace, that can make existing descriptions of COTS components become obsolete very quickly.

P8 "*At* ***regular*** *intervals, the team* ***reflects*** *on how to become more effective, then tunes and adjusts its behavior accordingly*"

This principle is crucial in the COTS world, since its roles are relatively new, requiring more experimentation and accumulated knowledge. Until a satisfactory point is reached, the system provider organization should tune and adjust its behavior frequently. The use of repositories similar to those used in the context of the COCOTS model [18] may help to reflect as required. Applying these feedbacks at regular intervals would allow us becoming more effective in the process of fixing the role behavior.

3. An agile-oriented analysis of current COTS selection methods

In our research we have investigated 8 of the most widespread COTS selection processes: SCARLET [19], OTSO [20], CARE [21], PECA [22], CRE [23], STACE [24], COTS Score [25] and that proposed by the SEL [26]. Due to lack of space, in this section we analyze in detail the first three selection methods under the light of the 8 agile principles identified in section 2. We provide a rationale for this evaluation with a subsection for each method. Each subsection includes an item for each principle and a table relating the main issues of the method to the principles, either positively ('+'), negatively ('–') or both ('+ –').

3.1. SCARLET

SCARLET [19] (formerly named BANKSEC) is the successor of the PORE method [27]. It adapts PORE to the banking domain and enables multiple selection. SCARLET:

P1. recognizes the changing nature of requirements by defining an iterative requirements process tightly intertwined with product evaluation;

Table 1. SCARLET issues affecting agile principles.

	P1	P2	P3	P4	P5	P6	P7	P8
Intertwining processes	+	+						
Several types of templates		+					+	
Vendor left outside			–		–			
AHP, Volere and other techniques						+	–	
Process guidance					+			
No repository					–		+	–
Roles not defined				–		–		
Specialized process						+		
Contracts and supplier managed			+ –		+			
Architecture exists	–				+			

P2. processes discard components gradually, but no partial result is given in the case of multiple selection;

P3. the barrier between the technical team and the marketplace seems very rigid (suppliers are not part of the team and relationships are taken in a defensive manner), which in some cases could be unnecessary;

P4. does not handle human factors, apart from using requirements and knowledge engineering techniques that may refer to them, the method just mentions that humans act as agents of the system;

P5. provides process guidance for procurement teams during a concurrent system development process, in which stakeholder requirements, the system architecture and solution components are all determined at the same time;

P6. integrates methods, artifacts and techniques such as AHP, Volere templates, etc. that provide a high degree of technical excellence; the banking context is explicitly handled in SCARLET, with specific types of requirements that make the process more reliable;

P7. distinguishes three different types of templates to be filled depending on the amount of work to be invested in evaluation;

P8. seems to be primary a one-shot method; however, the existence of tool support and evaluation stories can act as a medium for "intelligence" and prospective reflection. But in fact, a real repository is not mentioned except in [28] as future work.

3.2. OTSO

OTSO [20, 29] can be considered the first widespread selection method. It formulated the basic principles that the subsequent methods also incorporated, such as requirements and evaluation intertwining, use of formal techniques such as AHP for founding the selection, etc. OTSO:

P1. runs concurrently the evaluation criteria definition process during the search, screening and evaluation phases. It results on a baseline (a meaningful set of evaluation criteria) derived from the requirements to be used in the evaluation and analysis phases;

P2. conducts the search and discovering activities in small increments (e.g. a few days) and review the frequency of discovering new alternatives at each increment;

P3. recognizes the marketplace and the transfer of benefits between parties (suppliers and selection team) but does not constitutes an integrated team;

P4. considers organizations' reuse infrastructure and maturity for calibrating the final form of the process, customizing then the effort to the particular context;

P5. promotes sustainability by encouraging reuse through a well-defined process with the help of a repository. The baseline mentioned in P1 provides also a skeleton used during evolution;

P6. defines formally the evaluation criteria so that the evaluation of alternatives can be conducted efficiently and consistently.

P7. invests more effort in evaluating a limited number of alternatives that appear as the best candidates, documenting systematically the results;

P8. is a long-term oriented method, using a repository for organizing knowledge and including an assessment phase at the end of the process, devoted to obtaining feedback for future selection processes. Selection is heavily based on knowledge (and evaluation) reuse.

Table 2. OTSO issues affecting agile principles.

	P1	P2	P3	P4	P5	P6	P7	P8
Baseline fixed early	–				+			
Small increments		+						
Screening phase		+					+	
Marketing and contractual issues outside			–	–				
Concurrent phases	+	+						
Organization maturity considered				+	+			+
Alternatives conducted efficiently				–	+			
Repository					+		–	+
Intensive reuse					+		–	+
AHP & detailed evaluation						+	–	
Assessment phase								+

3.3. CARE

CARE [21, 30] is a method defined as both goal- and agent-oriented. CARE:

P1. recognizes that objectives and requirements can be changed and negotiated whilst the system is under development;

P2. organizes the processes of eliciting, analyzing, correcting, and validating goals as iterative, but it is not clear that this allow to deliver value early and frequently;

P3. although maintains and stores vendors' data in its repositories, it does not include any kind of interaction with vendors;

P4. considers humans as agents in its models, as done with software and hardware. Of course, as such agents, they are intentional (i.e., they play roles and have responsibilities), but in fact human factors are not addressed in the method;

P5. is architecture-centric, which provides a means of sustainable development, also supported by the existence of technical roles (see below) that interact in a logical predefined sequence and the continuous requirements negotiation;

P6. recognizes three different technical roles, namely requirements engineer, system architect and component engineer. Having experts in this profile is a way to support technical excellence. The use of notations such as NFR and i* is also a step beyond this goal;

P7. suggests several process that require heavy documentation, in particular the NFR framework and the i* language. This sacrifices simplicity for the sake of future reuse;

P8. has no documented processes on reflection for effectiveness, although one could argue that the information repository could contribute to tuning and adjustments to the operation of the organization.

Table 3. CARE issues affecting agile principles.

	P1	P2	P3	P4	P5	P6	P7	P8
Goal negotiation	+							
Arquitecture exists	–				+			
Iterative process		+ –						
Vendor data gathered			+					
No interaction with vendors			–		–			
3 technical roles			+ –	+ –		+		
Logical interact-tion sequence guided by roles					+			
NFR, *i**		–				+	–	
Repository					+		–	+
Human factors not considered			–	–				

3.4. Final observations

Table 4 summarizes the result of this analysis. We identify the 8 principles using the *Pi* identifiers introduced in section 2. For each method *M* and principle *Pi*, we rank the degree of coverage of *M* for *Pi* using the following rationale:

- A mark 'a' means that the principle is explicitly recognized as a design principle of the method.
- A mark 'b' means that although not intended explicitly, the method manages well the principle. Also we mark with a 'b' when the principle is explicitly mentioned but it is not clear that it is handled appropriately.
- A mark 'c' means that the method does not work well with the principle, although some spare practices have to be with it.
- A mark 'd' means that the method does not cover the principle at all.

Table 4. Comparative of COTS selection methods

	P1	P2	P3	P4	P5	P6	P7	P8
SCARLET	a	c	c	c	b	a	c	b
OTSO	b	a	c	d	b	b	d	b
CARE	a	c	c	b	a	a	d	a

From table 4, some observations can be drawn:

- Principle P1 is very well covered by virtually all methods, probably because the seminal OTSO method and more remarkably PORE already recognized the importance of overlapping requirements acquisition and product evaluation. The same happens with principle P6, probably because most of the methods proposed come from the academia. In fact, it has been also reported recently [31] that formal techniques recommended by the methods presented here are often neglected in the industry due to time pressures.
- Principles P5 and P8 are reasonably well covered by the methods. Sustainability come from the existence of well defined processes, whilst reflection is supported by repositories.
- The rest of the principles are not very well covered by current methods. It is difficult to reconcile simplicity with other principles, also it is not obvious that COTS selection may deliver value frequently. Last, motivating individuals has not been usually a goal since methods have focused in technical issues. The last sentence is especially relevant for understanding that principle P3 is bad covered by most of the methods. Often the crucial importance of legal advice remains hidden by the relevance given to technical issues; in fact, the result of COTS selection is defined mainly as an ensemble of products instead of a set of contracts as in fact is. On the other hand, suppliers are usually seen more as adversaries than as potential collaborators.

4. Some practices and a research agenda to improve agility

As a final step, from the analysis carried out in the previous section, we enumerate here a set of good practices that may improve agility in current COTS methods. Furthermore, we outline a research agenda for some relevant issues that require a thorough study before being converted into practices.

4.1. Practices proposed

- Identify technical and non-technical roles specific of COTS selection.
  - The technical roles should include: *requirements engineer*, able to elicit, analyze and define the different goals and requirements of stakeholders; *market watcher*, to classify the types of products available in the marketplace and the different substantial changes that emerge that can have an impact or influences within the information system (e.g. new versions, withdrawing of suppliers from the market, etc.); *component screener*, able to look for components candidates that match the requirements, which need a more detailed analysis; *component evaluator*, with a high technical profile to be able to apply techniques and processes that allow to rank the candidate products. Also, some more classical roles as a *quality engineer* and a project manager [32] are needed. Last, a *component customizer* from the supplier side able to customize the COTS component when required.
  - The non-technical roles should include: *system client*, for stating and validating requirements; *COTS supplier*, for providing detailed information and demos of components during detailed analysis; *manager*, for sharing responsibilities with the technical team in the system provider organization; *lawyer*, for providing assistance in the writing of the contracts and the study of the licenses.

  This distinction of roles helps to clarify the activities and responsibilities that take part during COTS selection. Considering agile philosophy, every team member should be equally knowledgeable and qualified to play all of them, although the variety and specifies of some of them (e.g., requirements engineer or lawyer) may not allow that.
- Maintain a project repository. The idea of repository, although somehow opposite to agility (requires work not strictly needed) is central to be

able to improve processes, to reuse knowledge and therefore learning from the past. The repository would require another role for its maintenance. With this repository, we may reuse from the initial goals and requirements, to COTS evaluation carried before. Also the repository should store rationale behind decisions taken (why a component was rejected, why a particular technique was selected for driving evaluation, …).

- Pair evaluation. This is an easy practice that makes a parallelism among selection-based and development-based software engineering. Since selection plays the part of programming, the same arguments apply for supporting this technique.
- Component metaphor. Again taken from the agile world [1], metaphors of components allow gaining in understanding and perception of what the component that must be integrated in the system does. This metaphor shall be constructed taking into account the key goals and requirements for that component.
- Call for tenders. Tendering [32, 33] is a procedure that is mandatory in some contexts (e.g., for public administrations when the system has a high budget). Although somewhat non-agile, since it breaks the development into two clearly distinguished parts (before and after tendering resolution), from the selection point of view tendering reveals to be an unexpected source of agility. Making the initial bid public implies receiving lots of feedback from suppliers that compete for that bid, pointing out new needs or even better, highlighting problems that are in the initial call for proposals. Furthermore, the way suppliers apply for the bid is an additional point of useful information to be considered when selecting. A variation of tendering is the use of questionnaires as a complement to gather information on products and suppliers [34].

A final comment is that some of these practices improve agility in general but may collide a bit with particular principles. One could wonder whether the general agile stream of COTS selection may coordinate with secondary, more stable streams such as repository maintenance.

4.2. Research agenda

- Define a maturity model for COTS selection processes. This model would allow organizations for which selection processes are a deal progressing towards a degree of excellence. This idea has been explored in [35]. Of course, this model should be agile-oriented, itself, and therefore its key areas adapted to this context.
- Propose new business models. Currently there are profit and non-profit organizations and companies that act as intermediaries, offer huge catalogues of products, gather COTS descriptions, etc. [36, 37]. The business models around can determine new practices that are currently undermined.
- Design a new COTS-based development method based in agile principles, highly customizable to particular types of organizations.

5. Conclusions

In this paper, we have analyzed current COTS selection methods under the perspective of the agile principles. We have identified what characteristics of these methods influence either positively or negatively which principles, and we have identified some practices that could eventually improve the methods from the agile point of view, as well as set a research agenda for 3 particular important issues. Most of our observation and practices align with some of the lessons identified in several reports [6, 13, 26, 27] which can be considered as a preliminary validation of our work, of course pending of a real validation planning which is part of our future work.

As far as we know, there is not much work done concerning COTS-based selection and agility. In fact, we just are aware of [38] in the context of the whole implantation process of ERP systems, more focused on project management and implementation than in selection, which is natural due to the coarse granularity of ERP systems. The paper is conducted from a practical point of view, more than from literature research as done in our proposal, identifying agile practices and heuristics that apply in the ERP context, although some of them are applicable in general to COTS components.

Another stream of related research is the adaptation of existing development process to embrace COTS-based systems. In [39], RUP is analysed from the COTS perspective, and we may found some similarities to COTS selection methods, such as the definition of specific roles and the iteration planning.

About future work, besides validation (see above), as mentioned in the introduction, we aim at replicating the analysis for the integration and evolution phases and next to put together the results for driving conclusions on the agility of the whole cycle of COTS-based software development.

6. References

[1] Martin, Robert C. *Agile Development: Principles, Patterns and Process*, Prentice Hall, 2002.

[2] Cockburn, A., Highsmith, J. “Agile Software Development: The People Factor”. *IEEE Computer*, December 2002.

[3] Beck, K. *Extreme Programming Explained: Embrace Change*. Addison Wesley, 1999.

[4] M. Beedle, K. Schwaber. *Agile Software Development with SCRUM*. Prentice Hall, 2001.

[5] Beck, K., *et al. Manifesto for Agile Software Development* http://www.agilemanifesto.org, 2001.

[6] FAA SERC. “Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software System”. October 2, 2000.

[7] Yu, E. “Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering”. *Proceedings of the 3rd IEEE ISRE*, 1997.

[8] Yu, E., Mylopoulos, J., Lesperance, Y. “AI Models for Business Process Reengineering”. *IEEE Expert*, August 1996, pp. 16-23.

[9] Lamsweerde, A. “Goal-Oriented Requirements Engineering: A Guided Tour”. *Proceedings of the 5th IEEE ISRE*, 2001.

[10] Larman, C., Basili, V. “Iterative and Incremental Development: A Brief History”. *IEEE Computer*, November 2004.

[11] Larman, C. *Applying UML and Patterns* (3rd edition). Prentice Hall, 2005.

[12] Lauesen, S. “COTS Tenders and Integration Requirements”. *Proceedings of the 12th IEEE RE*, 2004.

[13] Brownsword, L., Place, P. “Lessons Learned Applying Commercial Off-the-Shelf Products”. Report CMU/SEI-99-TN-015, June 2000.

[14] Kunda, D., Brooks, L. “Applying Social-Technical Approach for COTS Selection”. *Proceedings of the fourth UKAIS Conference*, University of York, April 1999.

[15] Curtis, B., Krasner, H., Iscole, N. “A field study of the software design process for large system” *Communication of the ACM*, 31(11):1268-1286, November 1988.

[16] Egyed, A., Kwan, J., Madachy, R. "Developing Multimedia Applications with the WinWin Spiral Model". In *Proceedings ESEC/FSE 97*, November 1997

[17] Saaty, T.L. “How to make a decision: The analytic hierarchy process”. *European Journal of Operations Research*, no. 48, pp. 9 - 26, 1990.

[18] COCOTS at http://sunset.usc.edu/research/COCOTS/, last accessed March 2005.

[19] Maiden, N., Kim, H., Ncube, C. “Rethinking Process Guidance for Selecting Software Components”. *Proceedings of 1st ICCBSS*, LNCS 2255, 2002.

[20] Kontio, J. “A Case Study in Applying a Systematic Method for COTS Selection”. In *Proceedings 18th Intl’ ICSE*, 1996.

[21] Chung, L., Cooper, K., Courtney, S. “COTS-Aware Requirements Engineering and Software Architecting”. *Proceedings of the SERP* 2004.

[22] Dorda, C., Dean, C., Morris, E., Oberndorf, P. “A Process for COTS Software Product Evaluation.”. *Proceedings of 1st ICCBSS*, LNCS 2255, 2002.

[23] Alves, C., Castro, J. "CRE: A Systematic Method for COTS Selection". *XV Brazilian Symposium on Software Engineering*, Rio de Janeiro, Brazil, October 2001 CRE

[24] Kunda, D. “STACE: Social Technical Approach to COTS Software Evaluation”. In *Component-Based Softwa-re Quality - Methods and Techniques*, LNCS 2693, 2003.

[25] Morris, A. “COTS Score: An Acceptance Methodology for COTS Software”. *Proceedings of the 19th DASC*, Philadelphia, PA., October 2000.

[26] M. Morisio, C.B. Reaman, V.R. Basili, A.T. Parra, S.E. Kraft, S.E. Condon. “COTS-based software development: processes and open issues”. *Journal of Systems and Software* 61 (2002): 189-199.

[27] Maiden, N., Ncube, C. “Acquiring COTS Software Selection Requirements.” *IEEE Software* 15(2), 1998.

[28] Ncube, C., Maiden, N. "PORE: Procurement Oriented Requirements Engineering Method for a Component-Based System Engineering Development Paradigm.” In *Proceedings of the 2nd CBSE*, 1999.

[29] Kontio, J. “OTSO: A Systematic Process for Reusable Software Component Selection”. *University of Maryland Technical Report CS-TR-3478, College Park, MD*, 1995.

[30] Chung, L., Cooper, K. “Matching, Ranking, and Selecting Components: A COTS-Aware Requirements Engineering and Software Architecting Approach”. *Proceedings 1st MPEC Workshop*, 2004.

[31] Torchiano, M., Morisio, M. ”Overlooked Aspects of COTS-Based Development”. *IEEE Software* 21(2), 2004.

[32] Lauesen, L. “Experiences from a tender process”. *Proceedings of REFSQ’04*, Riga.

[33] Krystkowiak, M., Bucciarelli, B., Dubois, E. “COTS Selection for SMEs: a report on a case study and on a supporting tool”. *Proceedings of the 1st RECOTS Workshop*, September 2003.

[34] Ncube, C., Maiden, N. ”Selecting COTS Anti-Virus Software for an International Bank: Some Lessons Learned”. *Proceedings 1st MPEC Workshop*, 2004.

[35] Olson, T. “Using CMMI/SS to Manage COTS&MOTS Software”. *Proceedings 2nd Annual CMMI Technology Conference and User Group*, 2002.

[36] Mielnik, J.-C., Lang, B., Laurière, S., Schlosser, J.-G., Bouthors, V. “eCots Platform: An Inter-Industrial Initiative for COTS-Related Information Sharing”. *Proceedings of 2nd ICCBSS*, LNCS 2580, 2003.

[37] ComponentSource, http://www.componentsource.com, last accessed Feb. 2005.

[38] Alleman, G.B. “Agile Project Management Methods for ERP: How to Apply Agile Processes to Complex COTS Projects and Live to Tell About It”. In *XP/Agile Universe* LNCS 2418, 2002.

[39] Chan, R. “Adopting RUP in a COTS-Implementation Project”. *The Rational Edge*, May 2003.