On Identifying a Methodology for the Integration of Commercial-Off-The-Shelf Products into Research Software Systems

Stephane Collignon[1,2]
Stephen Cook[2]
[1]DSTO Edinburgh, PO Box 1500, Edinburgh SA 5111, Australia
[2]SEEC, University of South Australia, Mawson Lakes SA 5095, Australia
Email: stephane.collignon@dsto.defence.gov.au

ABSTRACT

For the purposes of this paper we consider project-specific research software to be software that is produced in a research laboratory for specific purposes to support what in Defence terms are known as capability systems. It is different in character to the production software that is embedded in capability systems in that it is produced in a laboratory environment for specific purposes such as experimentation, concept exploration, or risk reduction. It normally has a limited lifetime, is maintained by the development team, and is built using Commercial Off The Shelf (COTS) components to the greatest possible extent. The paper opens by introducing the current Australian Defence Capability Development model that shows six lifecycle phases. We use this to define three types of research software that map onto three of the phases: concept development software, system demonstration software, and midlife upgrade support software. For each of these types, we define the software quality attributes that are most important and use these to identify the types of methodologies most appropriate to the development of each type of research software.

Keywords: COTS, defence, methodology, research software

Introduction

The Defence Capability Lifecycle Management Guide (2001) introduced the two-pass approval process into the Australian Defence Organisation. It is currently being refined as a consequence of the Australian Government accepting the recommendation of the Kinnaird Report (2003) and is shown at the highest level of abstraction in Figure 1 below. The first phase seeks to establish the Defence need for new or enhanced capabilities and the subsequent two phases define the capability system to be acquired.

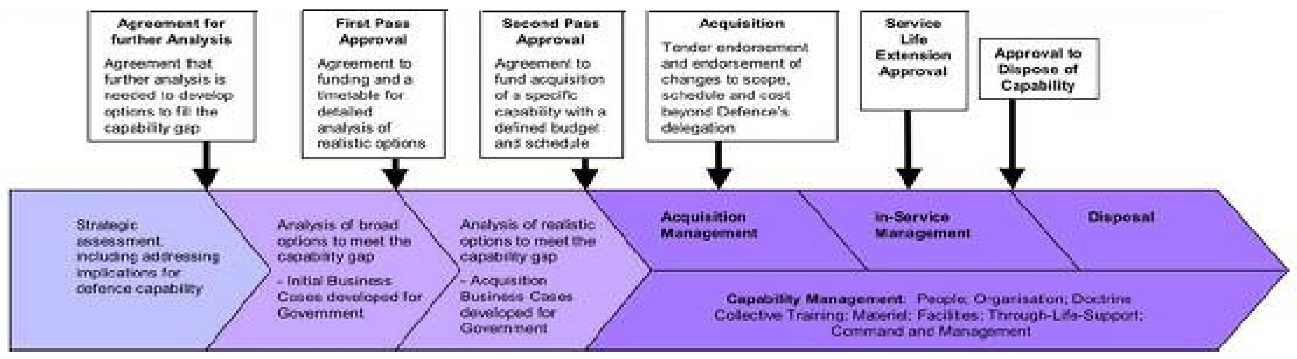
Agreement for further Analysis
Agreement that further analysis is needed to develop options to fill the capability gap
First Pass Approval
Agreement to funding and a timetable for detailed analysis of realistic options
Second Pass Approval
Agreement to fund acquisition of a specific capability with a defined budget and schedule
Acquisition
Tender endorsement and endorsement of changes to scope, schedule and cost beyond Defence's delegation
Service Life Extension Approval
Approval to Dispose of Capability
Strategic assessment, including addressing implications for defence capability
Analysis of broad options to meet the capability gap
– Initial Business Cases developed for Government
Analysis of realistic options to meet the capability gap
– Acquisition Business Cases developed for Government
Acquisition Management
In-Service Management
Disposal
Capability Management: People; Organisation; Doctrine; Collective Training; Materiel; Facilities; Through-Life-Support; Command and Management

Figure 1: The Australian Defence Capability Development Process (Kinnaird, 2003).

The paper is organised as follows. Firstly, the paper will categorise the research activities that are conducted in support of capability development and where each is most prevalent. Secondly, the paper will define the likely properties of each research software category to help guide the selection of software components. Thirdly, the paper will describe a meta-methodology that can help guide the selection of an appropriate methodology to conduct the development of research software systems.

Software and Research Environment

The research sector, like the industrial sector uses Commercial-Off-The-Shelf (COTS) products. According to Brownsword et al (2000) the term COTS refers to a product that is:

- Sold, leased, or licensed to the general public;
- Offered by a vendor trying to profit from it;
- Supported and evolved by the vendor, who retains the intellectual property rights;
- Available in multiple, identical copies; and
- Used without source code modification.

It is useful to note that COTS software used in research can be divided into two broad categories:

- Commercially-oriented software such as operating systems and application packages that are widely used in industry; and
- Research-dedicated software, such as Matlab, Simulink (2004), or Labview (2004).

This paper uses the notion of "context" (Smith, 2004). Context can be viewed as a combination of quality attributes in the context of a particular system development acquisition.

We have identified three quite different contexts in which COTS software can be applied in the research environment:

- Type I: Research exploration that seeks to either examine the potential of new technology or elicit user requirements from a military need;

- Type II: The construction of system concept demonstrators; and
- Type III: The investigation of operational upgrades to fielded (operational) systems and the integration of legacy systems into systems of systems.

Table 1 characterises the three different contexts.

	TYPE I: USE OF COTS IN RESEARCH EXPLORATION PHASE	**TYPE II: BUILDING A SYSTEM DEMONSTRATOR FROM COTS COMPONENTS**	**TYPE III: OPERATIONAL UPGRADES USING COTS COMPONENTS**
Desirable COTS Software Properties	• Support elicitation of the research requirements • Controllable (programmable by user) • Monitorable • Properties generally associated with research-dedicated software (Matlab/Labview type) • Rapid prototyping	• Supports rapid prototyping • Supports integration investigation with other system components and sibling systems • Contemporary technology with existing support base	• Must support the integration of the research software with the fielded system. • Supports the integration of legacy fielded systems • Contemporary technology with existing support base that interoperates with legacy technology
Number of Software Iterations to Complete Research Stage	Generally multiple, as dictated by the researcher modus operandi. Stops when final requirements are met	Development cycle normally defined in the contract: single or sometimes double when the system requirements are gradually refined during the system acquisition process.	Very likely to employ multiple cycles possibly in an evolutionary way.
Nature of Knowledge Base Used	• Defined by the scientists' knowledge and experience • Derived from existing architectures, tools, platforms and prototypes that are available in the current research environment	• Uses a target architecture • Commercial tools • Research results from exploration phase • Low-level programming techniques such as adding technical glue code.	• Enterprise architecture and DODAF • Metalanguages such as UML • Low-level programming techniques, such as adding technical glue code between software modules.
Research Output:	• Proven concept • Results and advice • Prototype system component • Information to commence next phase	• Useful demonstrator to influence ADF capability development decision-making • COTS-based system component	• Useful system upgrade that provides the information required for an acquisition project • System-of-Systems (SoS) capability for a major deployment or exercises

Table 1: Three contexts in which COTS software can be employed to support research activities.

These three types are used in all capability lifecycle phases but certain types tend to predominate as shown in Table 2 below.

The Australian Capability Systems Lifecycle Phase	**Research Activity Stages**
Capability Need Definition	Type I
Concept Exploration and Option Refinement (First Pass)	Type 1 and some Type II
Project Definition and Risk Reduction (Second Pass)	Type II and some Type I
Acquisition	Type II and some Type I
In Service and Upgrade	Type III and some of Type II

Table 2: Relationship between capability systems lifecycle phases and likely research software types.

The three types of research software have common requirements:

- The software must be able to produce the information required for project decision-making;
- The results produced by the software must be credible; and
- The internal state of the software is likely to need to be more observable and controllable than in production systems.

This paper will now proceed to describe a software engineering meta-methodology that can be used to create research systems that possess the properties listed above. The process will be illustrated for a Type I context.

Multi-methodology Selection

There is a large range of methodologies available for development of information systems. Rather than select a single methodology, Avison and Fitzgerald (1995, 2003) use social theory to categorise a problem domain to help identify appropriate methodologies from the set available. Cropley et al (2003) extended this approach by identifying the problem context and its position in the product lifecycle, to further assist in the selection of methodologies that are particularly well suited to the problem context. From these, Cropley selects a small set of methodologies that can be used in parallel, the synthesis of which, can provide a profound insight into the problem. The process for selecting software development methodologies is shown in Figure 2 and each process is described below. The heart of the process is a modified multi-variate value analysis that chooses the function that maximizes the objective function function $v(\mathbf{x})$:

$$v(\mathbf{x}) = \sum_{i=1}^{n} w_i v_i(x_i)$$

Where x_i is the numeric value of the attribute i, $v_i(x_i)$ is a user-defined value function that translates x_i into a value and where w_i is the weighting value for attribute i, (Buede, 2000).

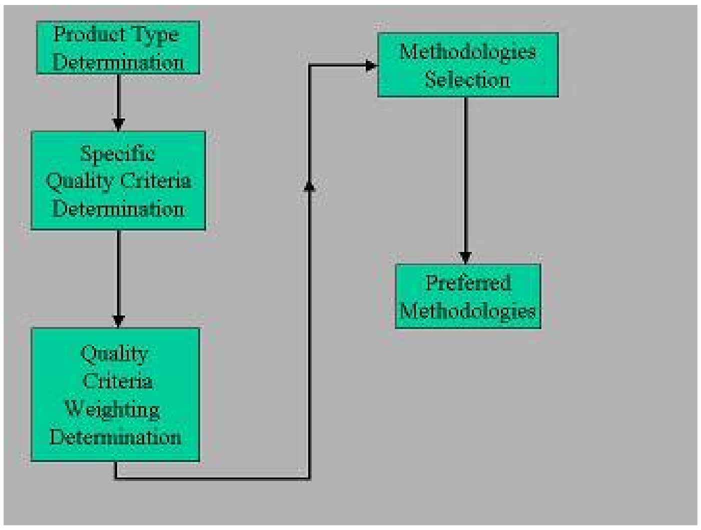
Product Type Determination
Specific Quality Criteria Determination
Quality Criteria Weighting Determination
Methodologies Selection
Preferred Methodologies

Figure 2: Proposed methodology selection plan

Determination of product type

Table A1, in Appendix A1 shows the mapping between software developments phases and generic systems engineering phases (Blanchard and Fabrycky, 1998). This mapping shows that the complete set of methodology phases is needed when the output expected from research is an integrated capability or a system-of-systems. When the output is less integrated, for example when the output is advice or a concept demonstrator, the number of methodology phases required decreases.

Development Context / **Nature of Output Product**	**I RESEARCH EXPLORATION PHASE**	**II SYSTEM DEMONSTRATOR**	**III OPERATIONAL UPGRADES**
Advice	▨		
Model	▨		
Prototype	▨	▨	
Independent System component		▨	
Single system that supports experimentation		▨	▨
Integrated System Component in System-of-Systems			▨
Modified Existing System Component of a System-of-systems			▨

Table 3: Research outputs sorted by research context.

It should be noted that in this generic approach, not all of the phases are sequential, as described in the Waterfall model. In a research environment, these phases may be overlapping and be iterative. Also when the research environment is a cooperative effort between industry and defence, the research phases may be disjointed or even parallel, depending on the resources allocated for these phases.

The first step in determining appropriate methodologies is to identify the output of the research program. The output can be identified from the research contract deliverables, in conjunction with the shaded areas in Table 3.

Specific quality criteria determination

This second step starts with the determination of the generic methodology phases necessary for the realisation of the product from the shaded areas in Table 4.

Research Output / **Phase Methodology**	Advice	Proven Concept or Model	Prototype	Single system that supports experimentation	Integrated System Component in System-of-Systems	Modified Existing System Component of a System-of-systems
Strategy (S)	▨	▨	▨	▨	▨	▨
Feasibility (F)	▨	▨	▨	▨	▨	▨
Analysis (A)	▨	▨	▨	▨	▨	▨
Logical Design (LD)		▨	▨	▨	▨	▨
Physical Design (PD)		▨	▨	▨	▨	▨
Programming (P)		▨	▨	▨	▨	▨
Testing & Evaluation (T&E)		▨	▨	▨	▨	▨
Implementation (I)			▨	▨	▨	▨
Maintenance (M)					▨	▨

Table 4: Mapping between research outputs and generic methodology phases.

Next, having determined the methodology phases necessary for the realisation of the research output, we need to select the quality criteria related to the methodology phases identified above.
Table 5 (adapted from Cropley et al, 2003) shows the quality criteria attributes for each of the methodology phases: (S = strategy, F = feasibility, A = analysis, LD = logical design, PD = physical design, P = programming, T = testing[1] , I = implementation, E = evaluation $_{(*)}$, M = maintenance).

[1] Cropley et al (2003) merged both test and evaluation phases in the mapping of the system engineering activities and phases (Table A1). As these two phases have different quality criteria, this paper will keep the original distinction from Avison and Fitzgerald, 2003 for methodology selection purposes. It should also be noted that Cropley et al (2003) modified Avison and Fitzgerald's (2003) original list of criteria by replacing "portability" with "scalability" and "ease of learning" with "usability" and we have elected to retain the new criteria.

Methodologies Phases Quality Criteria	S	F	A	LD	PD	P	T	I	E	M
Acceptability			▨	▨					▨	
Availability	▨	▨	▨	▨	▨		▨		▨	▨
Cohesiveness	▨	▨	▨				▨			
Compatibility	▨	▨	▨	▨	▨		▨		▨	
Documentation	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Economy			▨		▨	▨	▨	▨	▨	▨
Effectiveness	▨	▨	▨				▨		▨	
Efficiency					▨	▨	▨			
Fast development			▨	▨	▨	▨	▨		▨	
Flexibility			▨	▨	▨	▨	▨		▨	
Functionality				▨					▨	
Implementability	▨	▨	▨	▨	▨	▨	▨		▨	
Low coupling	▨	▨	▨	▨	▨					
Maintainability				▨	▨					
Reliability		▨	▨	▨	▨					
Robustness				▨	▨					
Scalability					▨					
Security		▨	▨	▨						
Simplicity	▨	▨	▨	▨	▨	▨	▨		▨	
Testability	▨	▨	▨	▨	▨	▨	▨		▨	
Timeliness			▨	▨	▨	▨		▨		▨
Usability		▨	▨				▨		▨	
Visibility		▨	▨	▨	▨	▨	▨	▨	▨	

Table 5: Matching methodologies phases and quality criteria.

By jointly considering Tables 4 and 5 it is possible to extract the relevant quality attributes for each of the research contexts. For example, if the product were to be advice, the list of quality criteria will be shown in Table 6. Note that Table 6 is a subset of the total list of quality criteria of Table 5.

Methodologies Phases Quality Criteria	S	F	A
Acceptability			▨
Availability	▨	▨	▨
Cohesiveness	▨	▨	▨
Compatibility	▨	▨	▨
Documentation	▨	▨	▨
Economy			▨
Effectiveness	▨	▨	▨
Fast development			▨
Flexibility			▨
Implementability	▨	▨	▨
Low coupling	▨	▨	▨
Reliability		▨	▨
Security		▨	▨
Simplicity	▨	▨	▨
Testability	▨	▨	▨
Timeliness			▨
Usability		▨	▨
Visibility		▨	▨

Table 6: Quality criteria necessary for the realisation of a Type 1 product (advice).

Quality criteria weighting determination

Table 7 gives the definition of each quality criterion and an importance ranking that we have assigned to it. The rank value was determined by the principal author on the basis of the relative importance of each criterion to the success of a Type I software development.

Software Specific Quality Attribute	Rank	Criterion Description
Acceptability	2	Whether the researcher finds the research software system usable and whether it fulfils their information needs.
Availability	3	Whether the research software system is accessible when required.
Cohesiveness	1	Whether the research software system allows efficient interaction between the researcher area specific knowledge it contains and other the information systems linked to the production of the research outputs.
Compatibility	7	Whether the research software system fits with other systems and parts of the organisation.
Documentation	9	Whether the research software system is sufficiently documented to allow communication between the researchers and sponsors.
Economy	17	Whether the research software system already exists or, if purchased can be re-used in future projects.
Effectiveness	4	Whether the research software system performs and operates well to support the production of the required research output.
Fast development	16	Whether the use of research software system can be created quickly.
Flexibility	15	Whether the research software system is easy to learn, use and adapt to the research process.
Implementability	5	Whether the implementation of the research software system is feasible in technical, social, economic and organizational senses.
Low coupling	18	Whether research software system can be used without affecting the other interacting subsystems.
Reliability	8	Whether the research software system dependent error rate is minimized and outputs are consistent and correct.
Security	11	Whether the research software system is robust against misuse
Simplicity	10	Whether research software system causes little ambiguities or complexities.
Testability	6	Whether the research software system used can be tested thoroughly to avoid failure while supporting the research effort.
Timeliness	14	Whether the research software system operates in a sustained and identical manner in every step of the research.
Usability	13	Whether the knowledge required for use of research software system is easy to acquire.
Visibility	12	Whether it is possible for research software system user to trace the causes of every result obtained with the software.

Table 7: Specific quality attributes for context – Type 1 products.

The rank order of Table 7 is between 1 (highest) and 18 (lowest), for each of the 18 software quality criterion relevant to the research software output product.

Table 8 is an example of weighting determination table for the example Type 1 research software using the rank-order centroid procedure, (Buede, 2000), where the weights w_i can be derived as follows:

$$w_i = \left(\frac{1}{n}\right)\sum_{j=i}^{n}\left(\frac{1}{r_j}\right)$$

$$\begin{aligned} w_1 &= (1 + 1/2 + 1/3 + \ldots + 1/n)/n \\ w_2 &= (0 + 1/2 + 1/3 + \ldots + 1/n)/n \\ w_3 &= (0 + 0 + 1/3 + \ldots + 1/n)/n \\ &\vdots \\ w_n &= (0 + 0 + 0 + \ldots + 1/n)/n. \end{aligned}$$

While one could argue the about the ranking order chosen above, the table illustrates the trade-off weights derived.

Quality Criteria	***R_i***	***w_i***
Cohesiveness	1	0.194173
Acceptability	2	0.138617
Availability	3	0.110839
Effectiveness	4	0.092321
Implementability	5	0.078432
Testability	6	0.067321
Compatibility	7	0.058062
Reliability	8	0.050125
Documentation	9	0.043181
Simplicity	10	0.037008
Security	11	0.031452
Visibility	12	0.026402
Usability	13	0.021772
Timeliness	14	0.017499
Flexibility	15	0.01353
Fast development	16	0.009827
Economy	17	0.006354
Low coupling	18	0.003086

Table 8: Distribution of the software quality attributes for a Type I product.

The next step is to rank the relative importance of the lifecycle phases appropriate to the problem context. For our example, these are the strategy, feasibility, and analysis phases and the results are given in Table 9.

Methodologies Phases / Quality Criteria	Strategy	Feasibility	Analysis
Acceptability			0.14
Availability	0.11	0.11	0.11
Cohesiveness	0.19	0.19	0.19
Compatibility	0.058	0.058	0.06
Documentation	0.043	0.043	0.04
Economy			0.0063
Effectiveness	0.092	0.092	0.092
Fast development			0.0098
Flexibility			0.014
Implementability	0.078	0.078	0.078
Low coupling	0.0031	0.0031	0.0031
Reliability		0.050	0.050
Security		0.031	0.031
Simplicity	0.037	0.037	0.037
Testability	0.067	0.067	0.067
Timeliness			0.018
Usability		0.022	0.022
Visibility		0.026	0.026
Sum	**0.68**	**0.81**	**1.00**

Table 9: Weight distribution for lifecycle phases relevant to Type I products.

In the following steps, the sum of each column will be used to assign weights in calculating the score for each methodology.

Methodology selection

Having determined the weights corresponding to each quality attributes relevant to Type I products, we shall next evaluate the methodology candidates proposed by Avison & Fitzgerald (2003) and listed in Appendices A and B.

Table 10 adapted from Avison & Fitzgerald (2003: p567) gives the relevance of each methodology to each lifecycle phase where:

- 0 Methodology does not cover the stage.
- 1 Areas mentioned but not associated to any process or rule.
- 2 Methodology addresses the area but not in depth or detail.
- 3 Methodology covers the area with techniques, methods and support.

Methodologies / **Phases**	**Methodologies and their Focus Areas**												
	Process			**Blended**			**Object-oriented**		**People**	**Organisational**			**RDM**
	STRADIS	YSM	JSD	SSADM	MERISE	IEM	OOA	RUP	ETHICS	SSM	PI	ISAC	DSDM
Strategy	0	1	0	1	2	3	0	1	0	3	3	0	2
Feasibility	3	3	0	3	3	2	0	2	2	2	2	2	3
Analysis	3	3	2	3	3	3	3	3	3	3	3	3	3
Logical Design	3	3	3	3	3	3	3	3	3	0	0	3	3
Physical Design	3	3	3	3	3	3	2	3	2	0	0	2	3
Programming	2	0	3	0	0	2	0	2	0	0	0	0	2
Testing	2	0	2	0	0	3	0	3	0	0	0	0	2
Implementation	1	0	1	2	2	2	0	3	2	0	1	0	2
Evaluation	1	0	0	1	0	2	0	2	2	0	0	0	0
Maintenance	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 10: Coverage of the candidate methodologies by software development phases.

In Table 11 we finally complete the multi-attribute value analysis. By summing the product of the phase coverage score by the weight for each phase from Table 9

Phases	**Methodologies and their Focus Areas**												
	Process			**Blended**			**Object-oriented**		**People**	**Organisational**			**RDM**
	STRADIS	**YSM**	**JSD**	**SSADM**	**Merise**	**IE**	**OOA**	**RUP**	**ETHICS**	**SSM**	**PI**	**ISAC**	**DSDM**
Strategy	0*0.68	1*0.68	0*0.68	1*0.68	2*0.68	3*0.68	0*0.68	1*0.68	0*0.68	3*0.68	3*0.68	0*0.68	2*0.68
Feasibility	3*0.81	3*0.81	0*0.81	3*0.81	3*0.81	2*0.81	0*0.81	2*0.81	2*0.81	2*0.81	2*0.81	2*0.81	3*0.81
Analysis	3*1	3*1	2*1	3*1	3*1	3*1	3*1	3*1	3*1	3*1	3*1	3*1	3*1
Sum	5.43	6.11	2	6.11	**6.79**	**6.66**	3	5.3	4.62	**6.66**	**6.66**	4.62	**6.79**

Table 11: Methodology scoring according to the focus area.

Preferred methodologies

Table 12 shows the preferred methodology candidates for a Type I product:

Position	**Methodology**	**Type**
1st position	Merise	Blended
	Dynamic Systems Development Methods	Rapid Development Methodology
2nd position	Information Engineering	Blended
	Soft System Methodology	Organisational
	Process Innovation	Organisational

Table 12: Selected methodologies selection for a Type I product.

The selected methodology candidates have close scores despite the fact they come from different methodology groups. A preferred solution can be built with the selected methodology candidates on the basis of their respective complementarities and their suitability to the research context.

For instance, possible options could be:

1. Merise or PI, both with a very large ‘soft’ focus approach; and
2. DSDM or SSADM, with a ‘soft’ focus approach, applied to rapid development.

Summary

This paper proposed a methodology using a context-based differentiation of the research outputs. This methodology is based on a Waterfall-like conceptual organisational separation of the research phases, as shown in figure 3.

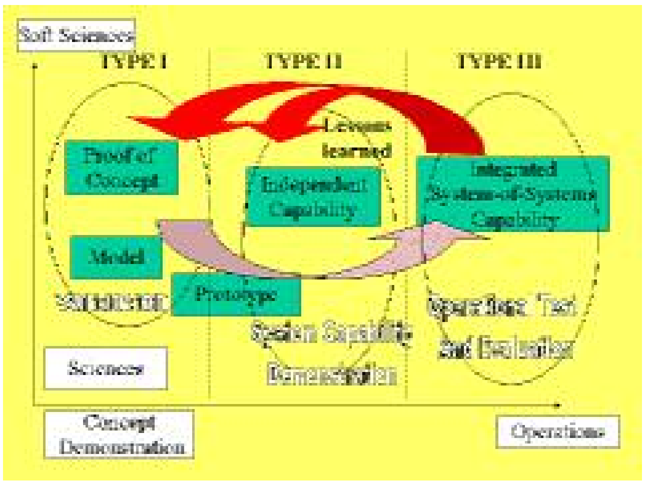

Figure 3: The three research contexts.

Conclusion

This paper has described how to select software development methodologies suited to the integration of COTS products into larger software systems being constructed to meet research needs. The following questions will be addressed in future work:

- “How to design an interactive software tool using this proposed methodology to deal with the multiple contexts of a research environment?” and
- “How can research knowledge be captured, stored and reused efficiently in the multiple contexts of a research environment?”

Further work is needed to complement this approach to:

- Extend the coverage of this selection process to research outputs developed for contexts not considered in this paper; and
- To include in the selection process knowledge development process model (Hanakawa et al. 1998) to support and facilitate the transition and evolution of a product from one context to another.

Appendix A Methodology Phases Equivalence

For reference, Table A1 extracted from Cropley (2003) lists the Avision and Fitzgerald phases and their conventional systems engineering equivalent.

Methodology Phases	Summary of Characteristics	SE Process and Lifecycle Equivalent
Strategy (S)	Organisational context, system purpose and planning.	SE process input to conceptual and preliminary design
Feasibility (F)	Economic, social and technical evaluation of target system.	SE process input to conceptual and preliminary design
Analysis (A)	User requirements, other?	Requirements analysis
Logical design (LD)	Functional architecture	Requirements analysis and functional analysis
Physical design (PD)	Physical architecture	Functional allocation
Programming (P)	Physical system development	Synthesis
Testing & Evaluation (T&E)	Planning and process of T&E	System T&E
Implementation (I)	Planning and implementation of technical, social, organisational aspects of the system.	Development, production, construction
Maintenance (M)	Specific tasks and planning for maintenance.	Use and support

Table A1: Methodology phases mapped to conventional SE activities and phases (Cropley, 2003).

Appendix B

Avison and Fitzgerald (2003) have gathered a wide range of methodologies under different themes, as showed in Table 12. To ease the understanding of this table, this paper presents brief definitions of each of the methodologies.

B.1 Process-oriented Theme

STRADIS stands for *Structured Analysis and Design of Information Systems*. Structured design is concerned with the selection and organisation of modules and interfaces that would solve a pre-defined problem. In essence the methodology uses a wide variety of techniques which are found in other methods and concentrates, as do others, on functional decomposition and the use of data flow diagrams. It is concerned mainly with systems analysis, to a lesser extent with systems design and hardly at all with implementation.

YSM (Yourdon Systems Method) YSM is similar to STRADIS in its use of functional decomposition, however a middle-out approach is adopted and slightly more emphasis is placed on the importance of data structures. YSM is based on functional decomposition, i.e. the breaking down of a complex problem into manageable units in a disciplined way. The development of a methodology, like Yourdon, stemmed from the perceived benefits of software engineering. In its environment model, YSM displays a systems view of understanding the system. This view is not seen as the main objective of YSM, but as a tool in the analysis stages. It has a clear objective to develop a computerised information system. It is designed specifically for real time systems. YSM is supported by case tools, which aids in the understanding of real world processes and in communicating the knowledge acquired.

JSD (Jackson System Development) is a strong semi-formal method: a way of analysing a situation into a set of independent but connected dynamic objects with a high degree of information hiding. JSD is unusual in that it was the first method to focus on the design and simulation of a real system. Additional control and reporting functions are added to the central simulation. It is suitable for MIS and Real Time systems.

B.2 Blended Methodologies

SSADM (Structured Systems Analysis and Design Method, a set of standards developed in the early 1980s for systems analysis and application design widely used for government computing projects in the United Kingdom. SSADM uses a combination of text and diagrams throughout the whole life cycle, from the initial design idea to the actual physical design of the application.

MERISE (Methode d'Etude et de Realisation Informatique pour les Systemes d'Enteprise) is a method for the study and implementation of business information systems. The initial purpose of MERISE was to develop an information system design methodology which could be used by both private firms and civil services to produce data processing applications which use databases in a real-time environment, and which will be more reliable. It became a dynamic modelling method, which models the behavioural aspects of an information system during the analysis and design phases of information systems. The MERISE method is based on separation of the data and the treatments to be carried out in several conceptual and physical models. The separation of the data and the treatments ensures longevity to the model. Indeed, the fitting of the data does not have to be often altered, while the data treatment is frequently reviewed.

IEM (Information Engineering Methodology) is a rigorous architectural approach to planning, analysing, designing, and implementing applications within an enterprise. Within the IEM, the enterprise carefully and thoroughly analyses its information requirements before beginning to build the applications that will support these requirements. The IEM is a flexible methodology that can be used in different environments and for different purposes. For any given business situation, the IEM defines an appropriate development path. Rapid Application Development is itself one such path, to be used for rapid development of stand-alone systems, incorporating CASE tools and rapid development techniques.

B.3 Object-Oriented Theme

OOA (Object-Oriented Analysis) strives to understand and model, in terms of object-oriented concepts (objects and classes), a particular problem within a problem domain (from its requirements, domain and environment) from a user-oriented or domain expert's perspective and with an emphasis on modeling the real-world (the system and its context/ user-environment). The product, or resultant model, of OOA specifies a complete system and a complete set of requirements and external interface of the system to be built, often obtained from a domain model.

RUP (Rational Unified Process) is a software design methodology created by the Rational Software Corporation. It describes how to effectively deploy software using commercially proven techniques, and is a heavyweight process (also described as a Thick methodology), and hence particularly applicable to larger software development teams working on large projects. The RUP defines the following guidelines and templates for team members to follow during a product's lifecycle:

1. Develop Software Iteratively;
2. Manage Requirements;
3. Use Component Based Architecture;
4. Visually model software;
5. Verify software quality and
6. Control changes to software.

B.4 Rapid Development Theme

DSDM (Dynamics System Development Method) is primarily based on continuous user involvement in an iterative (prototype-based) development process that is responsive to changing business requirements but still sufficiently defined for use with a formal quality management system if required.

B.5 People-Oriented Theme

ETHICS (Effective Technical and Human Implementation of Computer-based Systems), devised by Enid Mumford of the Manchester Business School is a methodology based on a participative approach to information systems development and people-oriented.

Ethics requires a design approach that covers technology and the organizational context in which the technology is placed. This implies the total design of departments, functions or areas using new technology including roles, relationships, activities, and jobs.

B.6 Organisational-Oriented Theme

SSM (Soft System methodology). The two approaches to system development are: (1) Hard Systems approach and (2) Soft Systems approach. Hard systems approach is based on systems engineering and systems analysis. The people are treated as passive observers of the system development process. However, this approach is not suitable in organizational environment that involves political, social, or human activities. Development of such systems require an active involvement of every stakeholder. The approach that encompasses all the stakeholders of the system is the soft system approach. SSM is a methodology that adopts such 'soft system approach'. SSM provides an effective and efficient way to carry out a **Systems Analysis** of processes in which technological processes and human activities are interdependent. It is used when the objectives of the system are hard to define, decision-taking is uncertain, measures of performance are at best qualitative and human behaviour is irrational. SSM proposes some alternative solutions and selects the feasible one.

PI (Process Innovation) is an approach to implement Business Process Re-engineering, with five recommended stages:

- Development of the business vision and process objectives;
- Identification of the processes to be redesigned (through the use of IT);
- Understanding and measurement of the existing process;
- Identification of the information technology levers (those IT capabilities that can fundamentally influence process redesign) and
- Design and construction of a new process.

ISAC (Information systems work and analysis of change) is a methodology that helps developing information systems by emphasising cooperation between users, developers and sponsors. The main techniques applied in ISAC are analysis and design. The methodology consists of five phases that are broken down in several sub-steps:

- Change analysis;
- Activity study;
- Information analysis;
- Data system design and
- Equipment adaptation.

References

Avison, D E and Fitzgerald, D (1995), *Information Systems Development: Methodologies, Techniques and Tools*, McGraw Hill Book Company Europe. Second edition. 1995.

Avison, D E and Fitzgerald, D (2003), *Information Systems Development: Methodologies, Techniques and Tools*, McGraw Hill Book Company Europe. Third edition. 2003.

Blanchard B S & Fabrycky W J (1998), *Systems Engineering and Analysis*, 3rd Ed., Prentice Hall, 1998.

Brownsword, L., Oberndorf, T., Sledge, C (2000), *Developing New Processes for COTS Based Systems,* IEEE Software July/August 2000.

Buede D., (2000), *The Engineering Design of Systems, Wylie*, New York 2000.

Cropley D, Yue Y, Cook, S (2003), *On Identifying a Methodology for Land C2 Architecture Development*, Land Operations Division, DSTO, Edinburgh, 2003.

Hanakawa, N, Morisaki, S, Matsumoto K, (1998*), A Learning Curve Based Simulation for Software Development*, Proceedings of the 20th International Conference on Software Engineering, Kyoto, Japan 1998.

http://www.mathworks.com/ The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098, USA Phone: 508-647-7000.

http://defweb.cbr.defence.gov.au/home/documents/departmental/manuals/cslcm.htm, (2002) Capability Systems Life Cycle Management Manual, Department of Defence, 2002.

ISO/IEC 15288 (2002), System engineering – System life cycle processes, ISO/IEC 2002**.**

Kinnaird, M. (2003), Defence Procurement Review, 15 August 2003.

Smith, J. (2004), *An alternative to TRLs for COTS Software-Intensive Systems*, Carnegie Mellon, Software Engineering Institute, Pittsburgh, PA 15213-3890, 2004.

SSC San Diego Systems Engineering Process Office (2003), *Overview of 5000-Series Acquisition Directives*, http://akss.dau.mil, 2003.

US DoD*, Interim Defence Acquisition Guidebook (2002)*. October 30, 2002.

www.ni.com/**labview**/ National Instruments Corporation11500 N Mopac Expwy Austin, TX, USA Phone: 78759-3504.