A framework for end-to-end approach to Systems Integration

Rashmi Jain*, Anithashree Chandrasekaran and Ozgur Erol

School of Systems and Enterprises,
Stevens Institute of Technology,
Hoboken NJ 07030, USA
Fax: 201-216-5541
E-mail: Rashmi.Jain@stevens.edu
E-mail: Anithashree.Chandrasekaran@stevens.edu
E-mail: Oerol@stevens.edu
*Corresponding author

Abstract: Systems Integration (SI) is an important element of systems engineering which involves the integration of hardware, software, products, services, business processes, and human. The existing standards, models, and guidelines of Systems Engineering and Software Engineering address SI issues partially and usually view SI from a perspective of integrating physical components. They lack a holistic end-to-end approach to SI. Due to the emerging Systems Engineering challenges and the increasing importance of SI, the need for a holistic approach to SI has become critical. A Systems Integration Framework (SIF) was developed that incorporates the relevant aspects of integration from a lifecycle perspective and sets a foundation to an end-to-end approach to SI.

Keywords: SI; systems integration; life cycle; integration readiness; integration maturity; COTS integration; LI; legacy integration; interface; V&V; verification and validation; integrated requirements; enterprise integration; CM; configuration management.

Reference to this paper should be made as follows: Jain, R., Chandrasekaran, A. and Erol, O. (2010) 'A framework for end-to-end approach to Systems Integration', *Int. J. Industrial and Systems Engineering*, Vol. 5, No. 1, pp.79–109.

Biographical notes: Rashmi Jain is an Associate Professor of Systems Engineering at Stevens Institute of Technology. She has over 15 years of experience of working on Information Technology (IT) systems. Prior to joining Stevens she was with Accenture (formerly known as Andersen Consulting). Over the course of her career she has been involved in leading the implementation of large and complex systems engineering and integration projects. Her teaching and research interests include systems integration, systems architecture and design, business process reengineering, and rapid systems engineering. She has authored several papers on these topics.

Anithashree Chandrasekaran is a doctoral candidate in the School of Systems and Enterprise at Stevens Institute of Technology. Her research interests include rapid systems development and its processes, development process reengineering, risk management and modelling, SI, and system design and architecture. She is currently working on developing a dynamic risk assessment

model for rapid system development process. She obtained her BE in Electrical and Electronics Engineering from P.S.G. College of Technology, India. She obtained her MS in Systems Engineering from Stevens Institute of Technology. She is the president of Stevens INCOSE student chapter.

Ozgur Erol is currently pursuing a PhD Degree in Systems Engineering at the School of Systems and Enterprises of Stevens Institute of Technology. She received her BS Degree in Industrial Engineering and MS Degree in Engineering Management from Istanbul Technical University, Turkey, and her MBA Degree from Saint Joseph's University, Philadelphia, PA. She has worked in the area of information technology and organisational reengineering prior to joining the PhD program at Stevens. Her research is in the area of SI with a special concentration in enterprise SI, enterprise architectures and business process management tools and techniques. She is a member of INCOSE and Institute of Industrial Engineers (IIE).

1 Introduction

Companies, today, desire to build systems faster, better, and cheaper. This desire is being driven by competition like never before. Customers are demanding total solution to their needs, and products of best quality at low prices. In turn, organisations are aiming for improvement in productivity of quality components (Alagumurthi et al., 2008). In an environment of fast changing technology and interconnectedness of systems, organisations are being challenged by designing and developing more and more complex systems. Systems Integration (SI) is a universally accepted necessity in the development of complex systems (Grady, 1994). A single company usually does not develop all the components of a system. Some of the components and the subsystems are built in-house and some are supplied by the outside vendors and then all of these subsystems or components have to be integrated into a final system.

Increasingly, systems Engineering practices have started to focus on complex systems. An interesting aspect of complex systems is that though they are built of well-designed, reliable, and efficient subsystems or components, they may not deliver an efficient final system when these components or subsystems are integrated. Complex systems represent a challenge for design and integration. On one hand, the solution space is huge and complex. On the other hand, these systems are increasingly critical for businesses. The cost of their deployment and management amounts to quite a substantial part of corporate budgets.

Software Engineering and computer science communities have attempted to define SI (Nilsson et al., 1990; Land and Crnkovic, 2003). Gulledge (2006) has tried to define the term SI within the scope of information systems research. He summarises his research of academic literature on 'what is integration' and notes that the term's description includes a process, a condition, a system, and an end-state. He focuses on the data integration aspect and categorises integration as 'Big I' and 'Little i'. Where big I indicates all data within an enterprise aligned in a single data-model and stored once, as opposed to 'point-to-point' interfacing of different kinds of technology components for the purposes of sharing information. However, from a systems perspective SI deals with more than just the data, software or the hardware of a computer system. A systems perspective includes all kinds of elements or components that constitute a system. SI is a critical component of

systems Engineering that unifies the product and the process components into a whole. SI ensures that the hardware, software, and human system components will interact to achieve the system purpose and/or satisfy the customer's need (Grady, 1994; Jain, 2007).

2 Partial view of integration

Systems Integration (SI) is a broad and ubiquitous concept, the elements of which are involved in nearly every aspect of the engineering of large systems and systems management (Sage and Lynch, 1998). It is an important element of systems engineering which involves the integration of hardware, software, products, services, business processes, and human (Grady, 1994; Jain, 2007). From a process perspective, SI process creates the links within the systems engineering process from requirements collection to Verification and Validation (V&V) and ultimately to implementation of the system. SI should be implemented from the beginning and throughout the system development rather than being implemented only as a 'down stream' event.

The existing standards, models, and guidelines of systems engineering and software engineering address SI issues partially and usually view SI from a perspective of integrating physical components. These standards and models lack a holistic end-to-end approach to SI. For example, in ISO/IEC-15288 (2008) the definition of SI process activities does not include important issues such as interoperability, interface control and management, Business Process Integration (BPI), and traceability. Due to the emerging systems engineering challenges and the increasing importance of SI, the need for a holistic approach to SI has become critical.

As a part of the ongoing SI research at Stevens Institute of Technology, a Systems Integration Framework (SIF) (Figure 2 discussed in Sections 4 and 5) was developed based on an assessment of existing state-of-art practice, and research on SI processes and models. The SIF incorporates the relevant aspects of integration from a lifecycle perspective and sets a foundation to an end-to-end approach to SI. Our end-to-end approach focuses on how integration issues can be addressed up-front to minimise integration related complexities and challenges later on in the system engineering process.

3 A comprehensive view of Systems Integration

A comprehensive life-cycle based approach would include all the relevant aspects of integration, namely, technology integration, applications and software integration, data and data repository integration, communications network integration, integration and reengineering of business processes, and, last but not the least, integration of the human.

In practice projects experience problems resulting from the decomposition and then integrating system components. As an example, decomposition of functions that cut across multiple functional or physical boundaries (end-to-end) or the decomposition of system-wide attributes and qualities must be done with integration in mind. In a sense, system design and architecture should aim at simplifying integration efforts and building 'integration-friendly' systems. Some of the reasons for failure and sub-optimal

integration of systems are lack of understanding of the scope of SI and its ownership of responsibility, lack of planning for SI early in the systems engineering life-cycle, lack of ownership of interfaces and as a result failure to define and document them, lack of proper CM, and change control process. The counter measure for these problems is to start integration early on in the systems engineering life-cycle and attend to the integration elements throughout.

SI can also be viewed from a layered perspective. This means that any system will have a physical layer, a semantic layer, a functional layer and so on and so forth. Therefore, in order for a system to be fully integrated it will need to address all aspects of integration. One such approach can be summarised in the following four states of SI (Mische, 1998) namely,

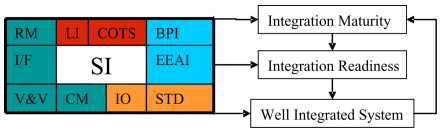
- interconnectivity is the initial and the fundamental state in the SI
- interoperability, the key state of SI, means that all interconnected information system components and equipment should be able to function and interact with each other
- semantic consistency refers to the concern of consistency at data level
- convergent integration involves the amalgamation of components and technology with business processes, people, skills, and knowledge.

When a system is designed, developed, and deployed, SI is needed to put together several subsystems and components which will comprise the final system. The activities of SI yield a functional system that meets a set of requirements. Any gaps left unattended in the entire development process will be revealed during SI. If SI issues are not addressed early on in the development process and throughout the lifecycle of the system development, it will be too late and too costly. SI is a set of complicated activities which if not done right may result in the risk of the final integrated system not performing the intended functionalities. The success of SI depends on several factors that have to be addressed over the different phases of system development lifecycle.

4 Defining a well-integrated system

The Systems Integration Framework (SIF) proposed in Section 5 focuses on improving the probability of success of SI activities by identifying the critical ones and their role in SI. Jain et al. (2008a) discuss the SI process, its constituent sub-processes, and related activities in their paper "A Systems Integration Framework for Process Analysis and Improvement". The authors believe that defining the scope, objectives, and role of all SI processes within the discussed framework is the beginning of a stable SI process that would mature over a period of time (Figure 1). Systems Integration process Maturity (SIM) may be defined as a stable well-documented integration process which supports architecture and design that is simple and 'integration friendly'. It is based on open architecture, scalability, modularity, commonality, re-usability, and current standards, for which clearly defined V&V criteria can be created. SIM is the extent to which repeatable SI process and related activities are defined, managed, and continuously improved using best practices and principles.

Figure 1 Improving Systems Integration readiness (see online version for colours)



SIM indicates the level of stability and sophistication of SI process and related activities. It includes the translation of the integration activities into measurable outcomes. A high maturity level is an indicator of a well-integrated system and is demonstrated in terms of a harmonised, stable, and scalable system.

SIM is also an indication of lower integration risks, the higher the maturity, the lower the risk in integration activities. Lower risk in integration activities means expected and required functionality of the integrated systems are ensured.

The time it would take for a stable and mature SI process to evolve would depend upon several factors such as SI organisation structure, level of SI process (and sub-process) analysis, evaluation, and feedback, metrics of SI activities used for evaluation, adaptability of the SI process to be open to customisation by specific individual development projects.

The SIF lays a foundation for directing SI activities towards the design and architecture of systems that are easy to integrate or in other words 'integration-friendly' or 'integration-ready' (ready to be integrated) systems as shown in Figure 1. The focus is on pre-empting integration issues upfront in the process. Identifying the best practices that will ensure this upfront focus is the challenge that projects need to manage. A mature SI process would lead to a clearer upfront emphasis on integration issues thus resulting in more integration-ready systems. Designing for an 'integration-friendly' system may include practices such as: simple designs, having an open architecture that is based on good standards which are current, relevant, and scalable, and making sure that the system documentation is updated at all times and complete in all aspects. System architecture may be considered as open if it is based on established and current standards which can evolve with technology evolution, and which can address the future scalability of the system. Many forums and working groups have been formed to address and provide guidance or best principles for designing open systems. The prerequisite for designing open systems is establishing interconnections. A universally accepted framework for interconnection is Open System Interconnection (OSI) model. Open system interoperability can be achieved through extrapolating or extending the OSI model. This thought has been explored by researchers in various domains. Open system interoperability architectures or models proposed based on OSI are discussed in OSI based interoperability for networks (Sugarbroad, 1990), Information processing system OSI (ISO/IEC-7498-1, 1994), Open system architecture for interoperability (Jain et al., 2008b; Abuelma'atti et al., 2006).

The SIF describes some of the essential elements of SI activities that may improve SI readiness and lead to the development of 'operationally well-integrated' systems. Over a long run such SI activities may result in a mature SI process. The improvements that would result from SIF are shown in Figure 1. In this figure,[2] the colours indicate the four dimensions of SIF that are discussed in the above SIF description.

Assuming that the above discussed integration and architecture practices are in place they should get reflected in the SI readiness and in the final product – an operationally

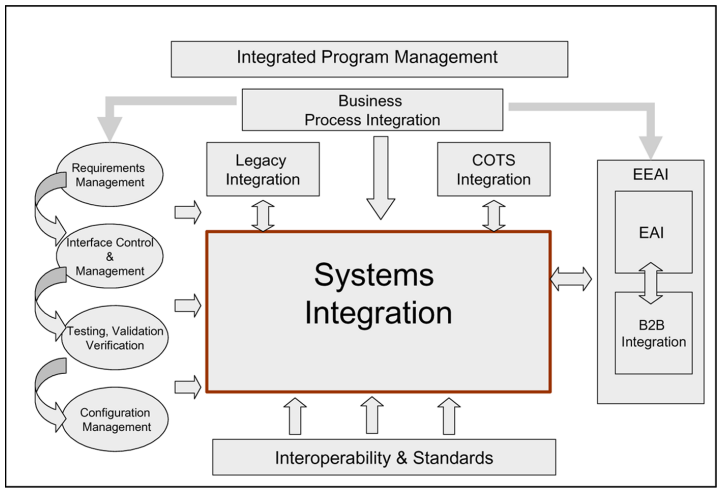
well-integrated system. In the long run the SI processes and related activities should lead to more efficiently integrated and robust systems. Systems that have evolved through a well-though through SI process will be more adaptable, easy and cost-effective to integrate, operationally optimised, predictable for reliability, maintainability, and supportability. So, should it be possible for us to distinguish a 'well-integrated' system from a 'not-so-well-integrated' system? If yes, then what will be the attributes or characteristics of such systems? How will these be demonstrated in the operational behaviours of such systems? How could such operational behaviours indicate critical aspects of SI that one must pay attention to early on from the SIF perspective? Based on our research some critical characteristics of operationally well-integrated systems are demonstrated in their functional and technical compatibility; harmonisation, stability, and scalability; user utility, technology longevity, adaptability, and speed of solution delivery through transparency of technology; harmonised and standardised application systems, data, access paths to data, and graphical user interfaces; and seamless and boundary-less operation.

5 A comprehensive end-to-end SI approach: the proposed framework

The SIF illustrates a comprehensive end-to-end SI approach (Figure 2). This approach is based on the premise that integration occurs throughout the lifecycle of a system. SI is not a one-time activity. Literature on SI falls short of considering the various views that integration takes in the development lifecycle of systems (Sage and Lynch, 1998). Our effort through the SIF is to identify integration embedded in every systems engineering activity. Our current focus of SI processes and activities looks at the following four dimensions of SI interactions as illustrated in the SIF.

- Interoperability is a prerequisite to achieving successful SI. If the sub-systems or systems are not interoperable they cannot be integrated. Standards or other forms of guidelines – international, national or industry-specific provide an opportunity to design systems which can be 'open' and simple to integrate. On the other hand, these guidelines can also constrain the design by requiring following design elements that may lead to more complex SI.
- The other processes of system development lifecycle such as requirements management, interface control and management, testing, V&V, and CM should include and address SI issues.
- Legacy SI and Commercial Off The Shelf (COTS) integration issues are inevitable in integration of most systems. Interoperability and interface management issues of legacy and COTS systems affect the overall success of SI to a significant extent.
- A good understanding of the business processes, and their definition and analysis is essential to defining the scope of SI. BPI creates the basis for how the system functionality flows within a system and interacts with its operational environment. Extended Enterprise Application Integration (EEAI) is the integration of a system within an enterprise and beyond. It is the extension of integration activities to beyond the enterprise domain.

Figure 2 Jain's Systems Integration framework (see online version for colours)



Systems design and architecting involves logical, functional, and physical decomposition of system requirements. Decomposition enables system development activities to be conducted concurrently, efficiently, and in a manageable manner. The complement of decomposition is integration. Every design element that has been decomposed into smaller configuration items will have to be integrated again to obtain the desired system functionality.

The SIF identifies the necessary elements of SI and illustrates the dependencies among them. It describes each of these elements, their constituent activities, tools, techniques, and best practices and factors that may be used for optimising each element for achieving successful SI. These ten elements that constitute the authors' comprehensive scope of SI are discussed below. These elements of the framework are: LI, COTS integration, requirements management, interface management, interoperability and standards, BPI, EEAI, Program Management (PM), testing, V&V, and CM.

5.1 Legacy Integration

Legacy Integration is one of the most important aspects of SI. In a way, we could say that most SI problems arise out of the need to integrate legacy systems. These systems are usually not designed to be 'integration-friendly'. Over a period of time as a result of 'hacked solutions' they evolve into 'spaghetti' architecture. Such systems are very difficult (sometimes even impossible) to be upgraded with enhanced functionality. As a result, the cost of maintaining them progressively keeps increasing with time. Even though the legacy systems are ridden with all these problems, they are mission critical for the organisations and should remain functional at all times (Konstantas, 1996). They fully satisfy system functional requirements and support current business functionality. These are systems that have been thoroughly tested in the actual operational environment. Therefore, it is often difficult to build a business case to retire them. As a result integration of such systems always becomes a huge challenge every time when the technical infrastructure in which they operate is upgraded or enhanced.

The relevant questions that we need to address with regard to legacy SI are: what are the kinds of unique LI issues do we have – their cost and risk assessment, what kind of middlewares, gateways and other solutions one can use for integrating legacy, what are the limitations of LI, and what other kinds of challenges exist for LI.

When do you actually get to know and be able to confirm that legacy has come to a stage where you cannot continue to support it any more? What kind of criteria would you use to evaluate the retirement readiness of legacy systems? How architectural considerations make legacy totally constraining and what can you do and when will you know that it's time to retire a legacy and say that we cannot support it any more. The legacy systems provide a set of design constraints which have to be identified and defined. Some of these limitations of legacy systems are: pollution, embedded knowledge, poor lexicon, coupling, layered architectures, frequency of failures and breakdowns, obsolescence and maintenance cost are discussed in Bianchi et al. (2003).

5.2 COTS based SI

Use of COTS components or products in system development has significant effect on integration, V&V. A COTS item is one 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 modification of the internals (OSD, 2002). Adoption of COTS systems are driven by the time-to-market constraints and the higher level of technology readiness of the COTS products. But the longevity and lack of design flexibility with COTS systems constrain the SI. The average COTS software product undergoes a new release every eight to nine months, with active vendor support for only its latest three releases (Basili and Boehm, 2001). Deciding to go for COTS vs. build it in-house early-on in the system development life-cycle can help in identifying the appropriate interface and performing the adequate V&V (Jain et al., 2008a).

Difficulties and lessons learned from COTS integration in practice are reported in Abts (2000), Abts et al. (2000), Boehm and Abts (1996) and Bansler and Havn (1994). Interfaces with and between COTS products is one of the major challenges of COTS integration. There are no widely agreed upon COTS standards (Voas, 2001) mainly due to marketing strategies aimed at obtaining vendor lock (Morisio and Torchiano, 2002). Selecting the right COTS product from the right supplier has always been a decision that the companies have struggled with. The qualitative vs. quantitative nature of multi-criteria supplier evaluation process is addressed by Noor-E-Alam et al. (2008). They proposed a computing tool which can evaluate the supplier by taking the opinion of expert as a linguistic value in a fuzzy form and incorporating the uncertainty measure. Thus increasing the reliability of expert judgment and making it more informative for decision making. Variability of COTS products and their marketing strategies suggest that there will never be a single unified marketplace of standardised COTS products (Morisio and Torchiano, 2002). Some of the common advantages and disadvantages of COTS integration are listed in Table 1.

Table 1 Advantages and disadvantages of COTS integration

Advantages	*Disadvantages*
Lower costs for initial procurement and logistics support	COTS results in the vendor control over when, where, and how an organisation must reissue hardware or software
Faster deployment (time-to-market)	Vendor may artificially quicken the upgrade and refresh rate negatively impacting the operational readiness
Improved quality and reliability	Incompatibility of rate of change between software and acquisition of human capabilities – especially examples like software to detect man-made noises
Leverage fast-paced commercial technology	Unpredictability from interaction of the unused functions of COTS product with the rest of the system
Reduced development risk	Evolution of COTS products in the wrong direction – driven by the vendor
Support system in place through availability of upgrades and other enhancements	Claims and actual performance does not match
More stable industrial base	Interoperability problems – between features and performance
Decreased reliance on sole providers	Mandate vs. appropriateness to utilise COTS
Improved surge capability	Failure to understand Total Cost of Ownership
Facilitates innovation from small businesses/academia	
Easier to attract new employees	

COTS integration is a high risk activity as COTS components make several assumptions about architectural issues. When these assumptions conflict or do not match, the simplicity of using COTS quickly is replaced with complexity and integration scaffolding (Egyed et al., 2005). However, over the past decade, data and control integration mechanisms and standards have matured significantly. Moreover, COTS products have become end-user programmable, extensible, and interoperable (Egyed et al., 2005). The impact of COTS on the development process depends on the source, customisation, bundle, and role (Morisio and Torchiano, 2002; Egyed et al., 2005; Yakimovich et al., 1999; Abts et al., 2000). Authors Morisio and Torchiano have identified COTS attributes, their possible values, and have characterised COTS based on their attributes and impact on development process (Morisio and Torchiano, 2002). According to Abts et al. (2000), the more granular attributes of COTS that impact SI in particular are correctness, availability/robustness, security, product performance, understandability, ease of use, version compatibility, inter component compatibility, flexibility, installation or upgrade ease, portability, functionality, price, maturity, vendor support, training, and vendor concessions (Abts et al., 2000).

Role of interoperability and interface requirements are very critical for COTS integration decisions. Integrating n number of COTS products involves potentially $n(n-1)/2$ interfaces (Basili and Boehm, 2001). These requirements also result in architectural refinement and decisions. The knowledge of these requirements and architecture at an early stage of development will help in proactive mitigation of the integration, V&V problems. A Simultaneous COTS approach is recommended to

addresses COTS integration issues. In this approach convergent decisions are made considering the following tradeoffs simultaneously (SEI, 2002).

- *Stakeholder needs and business processes.* Requirements (including quality attributes such as performance, security, and reliability), end-user business processes, business drivers, and operational environment.
- *Marketplace.* Available and emerging COTS technology and products, Non-Development Items (NDI), and relevant standards.
- *Architecture and design.* Essential elements of the system and the relationships between them. Elements include structure, behaviour, usage, functionality, performance, resilience, reuse, comprehensibility, economic and technologic constraints and tradeoffs, and aesthetic issues.
- *Programmatics and risk.* The management aspects of the project. These aspects consider the cost, schedule, and risk of building, fielding, and supporting the solution to include the cost, schedule, and risk for changing the necessary business processes.

Yakimovich et al. (1999) identifies five types of COTS integration problems: functional, non-functional, architectural style, architectural conflicts, and interface (Yakimovich et al., 1999). Yakimovich also identifies six integration strategies: tailoring, modification, re-implementation, glueware, architectural changes, and architectural style changes. Strategies for some of these COTS Integration risk assessment and mitigation at an early stage through architecture, performance assessment, and process management are discussed in Yang et al. (2005), Donzelli et al. (2005), Putrycz et al. (2005), Lemahieu et al. (2005) and Warboys et al. (2005).

Critical success attributes of COTS implementation (Grant, 2000) are flexible requirements, incentives to contain sustainment cost, up-front awareness of design considerations, matching COTS functionality with requirements, changing culture driven by volatility of the marketplace and the pace and scope of technology change, and using the right tools. COCOTS model calculates the effort of COTS integration in terms of maintainability, synchronisation, complexity, number of planned upgrades, experience of integration, and the volatility effect (Abts et al., 2000). Kelkar and Gamble (1999), Yakimovich (2001), Payton et al. (1999) and Davis et al. (2002) also present processes that help evaluate ease of integration and its estimated effort before it is done. Proliferating four or more COTS products in a system poses greater integration risks (Garlan et al., 1995). COTS components rarely share the same architectural and interface assumptions (Garlan et al., 1995). If the assumptions differ too much, the integration effort can become too costly, and the project may be cancelled due to budget overrun. So, the objective is to make the integration as effortless as possible. This can only be achieved by choosing components with converging architectural and interface assumptions. Components with converging architectural and interface assumptions are compatible (Yakimovich et al., 1999), while those with diverging architectural and interface assumptions are called incompatible.

5.3 Requirements management

Once a conceptual design for a system is chosen and all operational scenarios (use cases) to understand the context are analysed, the broader category of stakeholder

requirements are then refined and derived to form system requirements or specifications. During the requirements engineering phase of system development it is critical to identify requirements that will impact SI. These requirements termed as integration requirements address the required level of integration and quality. These integration requirements fall under the category of interoperability, interface, qualification or test, operational readiness, integration technology, and compliance and standards. The integration requirements are then used to develop design (architecture) to address the integration issues such as COTS, legacy, interfaces, testability, qualification and compliance.

Integration Requirements Management is one of the most important sub-processes of the SI process. Jain et al. (2008a) provides a more detailed discussion on this process and the six categories of integration requirements. In order to avoid issues and surprises during the implementation it is important to elicit system requirements with SI focus. The delays in product development are mainly attributed to errors and rework that result due to the errors that were introduced in the initial phases but demonstrated during the final phases. The effort and money spent on these errors and fixes are huge (McConnell, 1996). It is a good practice to identify SI stakeholders and gather requirements from them. The integration requirements are reiterated with the stakeholders of SI and signed off so that they can be used as a baseline for the SI process. These requirements have to be documented well and maintained with good Configuration Management (CM) (discussed in detailed in Section 5.9).

The requirements development process and characteristics of good requirements are discussed in IEEE 1233 (1998) and IEEE 830 (1998). The important characteristics of requirements that significantly impact SI are the traceability and testability of requirements. A requirement is verifiable (IEEE 830, 1998) if there exists some finite cost-effective process with which a person or machine can check that the system meets the requirement. In general any ambiguous requirement is not verifiable. Non-verifiable requirements include statements such as ‘works well’, ‘good human interface’, and ‘shall usually happen’. These requirements cannot be verified because it is impossible to define the terms ‘good’, ‘well’, or ‘usually’. Verifiability of requirements directly impacts verifiability and validity of the system and design. It also impacts traceability of requirements.

A requirement is traceable (IEEE 830, 1998) if its origin is clear and if it facilitates its referencing in future development or enhancement documentation. The following two types of traceability are recommended:

- *Backward traceability* (i.e., to previous stages of development): This depends upon each requirement explicitly referencing its source in earlier documents.
- *Forward traceability*: This depends upon each requirement having a unique name or reference number.

Traceability of requirements can help in timely, correct, and complete integration of components and systems. The role of traceability in integration is discussed in Gieszl (1992). The notion of traceability requires a capability to relate the specific requirements through several levels of abstraction to their final implementation and to specific tests that show the satisfaction of those specific requirements. The three important sources of requirements origins are stakeholders, (documented) sources, and process-product

objects (Jarke, 1998). Environmental, organisational, and technical factors influence the implementation of requirements traceability (Ramesh, 1998).

5.4 Interface management

Interfaces are a common failure points on systems (Buede, 2000). System interfaces are critical to the success of SI. Interface is defined as a plane or place at which independent systems or components thereof meet and act or communicate with each other. The interface of a system contains both the logical element and a physical element (or link) that are responsible for carrying items from one component or system to another (Buede, 2000). Interface is the functional and physical characteristics required to exist at a common boundary or connection between systems or items (DOD 4120.24-M, 2000). Simple interfaces result in good design and operational success. Simple interfaces are those interfaces that are simple systems by itself, having manageable contracts, posing low integration risk, and manageable throughout the lifecycle.

The various types of interfaces (Buede, 2000) are: Connectors that facilitate the transmission or exchange of electricity, fluid, force, material, information, etc, Isolators that inhibit such interactions, and Converters that alter the form of the entity being transmitted or exchanged. A more detailed classification of interfaces types is as follows (Sanchez, 2000):

- attachment interfaces: how components attach to each other
- spatial (volumetric) interfaces – the spatial volume allocated to a component
- transfer interfaces that are responsible for what comes in and what leaves the component
- control and communication interfaces are comprised of information exchanges that communicate component state and/or changes
- user interfaces are components that receive ‘requests’ from the user
- environmental interfaces are components that interact with the ambient environment or other components in intended or unintended ways.

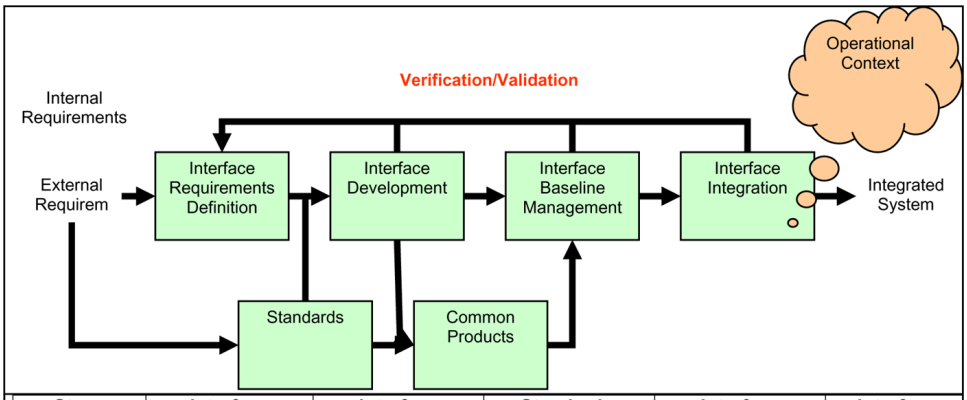
Interface requirements are statements about the interface designs, protocols used, data formats, entity relationships, and processing rules. They define the interfaces and their inputs and outputs. Interface Interactions between elements (Pimmler and Eppinger, 1994) are

- *Spatial*: A spatial-type interaction identifies needs for adjacency or orientation between two elements.
- *Energy*: An energy-type interaction identifies needs for energy transfer between two elements.
- *Information*: An information-type interaction identifies needs for information or signal exchange between two elements.
- *Material*: A material-type interaction identifies needs for materials exchange between two elements.

Interfaces may also be considered from an internal/external perspective. Internal interfaces are those that address elements inside the boundaries established for the system of interest. These interfaces are generally identified and controlled by the contractor responsible for developing the system. External interfaces, on the other hand, are those which involve entity relationships outside the established boundaries of the system (DAU, 2001). Interface requirements must address total system performance, the fidelity of the interface, and any system requirements meant to constrain interface design (Buede, 2000). Typical system performance requirements of concern in designing the interfaces are system throughput and response time. The fidelity of an interface is determined by the integrity of the items, and failure detection and recovery within the interface.

Interface architecture defines the interface specification based on all the system internal and external interface requirements with the support of Interface Definition Language (IDL). Open System Interconnection Model (OSI) is widely used to define interfaces and interconnections at all levels of system abstraction. A baseline interface definition must be agreed upon before the beginning of implementation activities, and the user interface must be defined and maintained as an integral part of the system specification. Interface control and management is an important part of the integration architecture that results in the management of communication, coordination and responsibility across a common boundary between two organisations, phases, or physical entities which are interdependent (Jain, 2007). It ensures that internal and external interfaces are properly identified, integrated, stabilised, and controlled early in order to prevent expensive and time-consuming fixes later. Interface identification, documentation, and assessment are shown in Figure 3.

Figure 3 Interface control and management (see online version for colours)



Stage	Interface Requirements Definition	Interface Development	Standards, Common Products	Interface Baseline Management	Interface Integration
Components	Stakeholders Involvement	Integrated Approach to Design	Standards Selection	Configuration Management	Path Testing
	Management Control	Test Driven Development	Implementation Guidance	Problem Management	Parametric Testing
	Developer Flexibility	Interface Definition Language	Common Products	Change Control	Longevity Testing
	Simplicity		Monitor Conformance	Communication	Scalibility Testing

Source: Jain (2007)

5.5 Interoperability and standards

Standard is a document that establishes engineering and technical requirements for products, processes, procedures, practices, and methods that have been decreed by authority or adopted by consensus (EIA-632, 1999). Standards help ensure that an interface will enable the connection of two components. Each component is required to meet a given standard, and the interface is designed to meet the same standard in order to ensure design and integration success (Buede, 2000). An interface standard is a standard that specifies the physical, functional, and operational relationships between various elements (hardware and software), to permit interchangeability, interconnection, compatibility and/or communications. Interface standards specify the physical, functional, and operational relationships between the various elements (hardware and software), to permit interchangeability, interconnection, compatibility and/or communication. The selection of the appropriate standards for system interfaces should be based on sound market research of available standards. Evaluation and selection of interoperability standards are especially necessary in the application development and integration projects, when there is a need to assess the usefulness of existing models or to find open solutions. Standards have to be evaluated when recommendations are made for a given domain or when their quality is examined. The evaluation of the scope and other aspects of interoperability standards is usually performed against project-specific requirements (Mykkänen and Tuomainen, 2008).

Standards have different levels of formality: formal or proprietary, de jure, and de facto. Proprietary standard is a standard that is exclusively owned by an individual or organisation, the use of which generally would require a license and/or fee. De jure standards are mandated by legal or other authorities. De facto standard is a standard that is widely accepted and used but that lacks formal approval by a recognised standards organisation (FED-STD-1037C, 1996).

Standards can make SI easier and help prevent problems in integration and change (Kuhn, 1990). The benefits attributed to using standards (Buede, 2000) are:

- *Interchangeability*: ability to interchange components with different performance and cost characteristics
- *Interoperability*: the system can now operate with a wider variety of external systems, systems that have also adopted the same conventions
- *Portability*: systems can be moved and operate on other systems
- Reduced cost and risk for equivalent performance
- Increased life cycle is possible when long-lived standards are adopted.

Standards for open systems promote SI friendliness and concurrently they will increase the SIM.

Interoperability can be defined as the ability of the system to ‘play well with others’, both with the systems it was originally designed to work with, and with future systems. It may be desirable in and of itself; also enhances versatility, flexibility and evolvability of systems of systems (McManus and Hastings, 2005). It is the ability of systems, units, or forces to provide data, information, material, and services to and accept the same from other systems, units, or forces, and to use the data, information, material, and services so exchanged to enable them to operate effectively together (DODD-5000.1, 2003).

Interoperability focuses on the exchange of meaningful, context-driven data between autonomous systems. Interoperability can be achieved by designing and building systems against a defined interoperability requirement, and maintaining that interoperability throughout the system changes and upgrades through CM, and testing for interoperability against those requirements. Interoperability Requirements address or help address an operationally recognisable activity or sequence of activities that has a definable starting action, a definable concluding action, and which involves the exchange of data (information or material) between two or more systems (sub-systems or platforms). Such a data exchange may be interactive and may involve the use of more than one transfer medium; however, the information content on all transfer media must be definable. These requirements are related to an operational capability. In most cases few interoperability requirements are identified and interoperability often becomes an issue when system is deployed.

According to Ouksel and Sheth (1999) the four types of interoperability are semantic interoperability, structural interoperability, syntactic interoperability, and system interoperability. Semantic understanding results in operational interoperability of systems. Semantic integration addresses the semantic content of data in different systems. It is more important to be aware of inconsistencies in the semantics rather than to eliminate the differences, which may be difficult, especially in systems from different vendors. Semantic consistency refers to the concern of consistency at data level. Once the information system components and equipment are interconnected and operational, users are able to access systems and manipulate data (create, retrieve, modify, and delete) across various operational environments. For this reason, the implementation of semantic integration is essential to prevent data duplication, redundancy, and instability (Mische, 1998).

Interoperability models have been developed to provide guidance on building architecture for ease of integration and interoperability. These models define different levels of interoperability and attributes of architecture that have to be addressed for each level defined. Some models define various domains of interoperability and their relationships and interaction between them. Some of these interoperability models used today are: Levels of Conceptual Interoperability Model (LCIM), Levels of Information System Interoperability (LISI), NATO Reference Model for Interoperability, Organisational Interoperability Maturity Model, System of Systems Interoperability (SOSI), ECMA/NIST Model, and Defense Information Infrastructure Common Operating Environment (DII COE).

In order to achieve interoperability certain pre-existing conditions are necessary. Some of these conditions are interconnection between systems, compliance to interface and open standards, and ability of systems to provide the required information flow for each operational scenario. (SEI/CMU, 2004) has identified six metrics for interoperability namely coupling, heterogeneity, synchronicity, boundedness, ownership, and usage patterns. System interoperability can also be measured in terms of interface security, data integrity, interface flexibility, interface bandwidth, interface registration and version control, interface management, interface performance, interface documentation, data processing integrity, data presentation, user help, access, monitoring, trouble shooting, upgradeability, reliability, and robustness (Whitt, 2004).

5.6 Two dimensions of enterprise integration: Business Process Integration and Extended Enterprise Application Integration

Traditionally in the systems engineering and technology communities, SI has been described in the context of design and development of systems such as airplanes, factories, command and control systems, communication networks, etc (Rouse, 2005; Sage and Lynch, 1998). Due to the significant shift within the systems engineering community towards considering 'enterprises' as systems (Rouse, 2005; Carlock and Fenton, 2001; Kosanke et al., 1999). Enterprise integration has become an important aspect of SI. An enterprise is a goal-directed complex system of resources – human, information, financial, and physical – and activities, usually of significant operational scope, complication, risk, and duration. Enterprises can range from corporations, to supply chains, to markets, to governments, to economies (Rouse, 2005). Enterprise integration is related to integrating technology, processes, and people to facilitate a greater flow of information and effective decision making across the enterprise with the goal of improving efficiency and competitive advantage. Enterprise integration enables the successful communication between data, applications, processes, people and enterprises and helps an organisation to establish a technology infrastructure that seamlessly links its complex business applications in to a homogenous system so that the processes and the data can be shared across the enterprise, business partners, and customers (Venkatachalam, 2006; Smith et al., 2002; Kosanke et al., 1999; Brosey et al., 2001). The drivers for enterprise integration is to provide timely and accurate exchange of consistent information between business functions to support strategic and tactical business goals in a manner that appears to be seamless (Smith et al., 2002; Venkatachalam, 2006).

Despite its benefits, enterprise integration is a challenge for most of the organisations (Smith et al., 2002). Enterprise integration efforts fail due to a number of technical, organisational and management, and migration planning reasons. The technical issues include: unplanned and stovepipe development of legacy systems, data being specific to the applications and not being designed for sharing, difficulty of adaptation of legacy systems towards new quality-attribute requirements and being affected by needs for interoperability, performance, security, and usability; inadequately defined scope of integration effort, decisions made without performing adequate analyses. The organisational and management issues include the need to obtain and maintain sponsorship and financial commitment at all levels of the organisation for the duration of a potentially lengthy project; the need to break a project up into subprojects so that consistent, intermittent milestones can be measured; and the need to communicate effectively (Smith et al., 2002).

Enterprise integration is defined at different levels of integration including process integration, application integration and data integration (Stohr and Nickerson, 2003; Huang et al., 2003; Lam, 2005). These different levels of integration happen internal or external to the enterprise. In the next two sections BPI and EEAI – we will provide details on these different levels of enterprise integration and their relevance within in SIF. BPI will focus on process level integration while the application and data level integration will be covered in the EEAI.

5.6.1 Business Process Integration

SI does not only concern hardware, software and products but also involves business processes, services and human (Grady, 1994; Jain, 2007). Integration of business systems, or for that matter any system, requires a good understanding of business processes and business process analysis. Systems are designed, developed and engineered to support the business processes within an enterprise (Jain et al., 2008a) and alignment of processes and systems is crucial to better meet the needs of stakeholders for whom the systems are built. This approach sets our foundation of the end-to-end (life cycle) approach to SI. Therefore BPI is an important element of our SIF. An understanding of the concept of a business process and the need to conduct integrated business process analysis is a prerequisite for SI.

BPI refers to integration of processes across multiple systems and environments within or across enterprises (Themistocleous and Corbitt, 2006; Zhu et al., 2004). A business process is a collection of activities that creates an output that is of value to some customer (Hammer and Champy, 1993). Integration of business processes, at enterprise or cross-enterprise levels, helps organisations to increase the quality of services and products by achieving significant improvements (Davenport, 1993). BPI requires integration of data, applications, standards and alignment of processes with IT. The goal of BPI is to create frictionless knowledge and application sharing within the enterprise.

BPI can also be described in four layers of compatibility (Huang et al., 2003), namely, technical, operational, strategic, and political and legal environment compatibility**.** At the technical compatibility layer BPI requires companies to adopt compatible technology, including data formats, communication protocols, network infrastructures, security policies, applications, and computing platforms. For the purposes of operational compatibility, business units need to adopt compatible workflow and business processes to facilitate operational integration. Business units need to develop a common set of goals, cultures, and objectives for achieving strategic compatibility. Finally, at the political and legal environment compatibility layer, business units/partners are required to manage and share cross-border information flow even though they face different legal environments, due to different local laws and industry-specific regulations.

There are three main types of process integration: integration-within-process, cross-functional process-to-process integration, and external process integration. Integration-within-process is achieved by the coordination between individual human and software actors as they perform the work of the organisation, cross-functional process-to-process integration is achieved by the coordination between the efforts of organisational units such as departments and divisions that have different roles to play in the execution of shared processes, and external process integration is achieved by connecting an organisation with its suppliers and customers (Stohr and Nickerson, 2003).

Successful BPI initiatives require modelling and verifying business processes with all stakeholders, automating and eliminating redundant processes, monitoring, measuring and continuously improving the processes (Gold-Bernstein and Ruh, 2004). Business process model helps integrate business processes more effectively and easily and it provides a complete definition, including process scope, process workflow and logic, work breakdown, entities and transformations, business rules, resources and activity times. Process models also help detect improvement opportunities

by finding overlapping or redundant activities, non-value added activities, resource gaps and inefficient processes. Gold-Bernstein and Ruh also suggest that standards play an important role for the success of BPI by providing long-term viability, reuse, and management of business processes. In addition, (Mendoza et al., 2006) suggest a different set of success factors for BPI, including good understanding of organisational structure and determination of its support for the integration of the process, technological infrastructure, effective project leadership, relevant user involvement, high-expertise project team, effective change management strategy and project management. Aubert et al. (2003) suggest four different information quality attributes to evaluate and measure the success of process integration (Aubert et al., 2003). These measures as shown in Table 2 are accessibility, transparency, timeliness, and granularity based on the premise that effective process integration utilises and leverages the information quality.

Table 2 Measures for process integration

Measure	*Attributes of data*
Accessibility	Availability of the data, easy and convenient access to data within the process, and clarity of the data which is easy to manipulate
Transparency	Understandability of data which is passed from one task to another within a process, consistency and completeness of this data, standardisation of data which allows common understanding within the user throughout the process
Timeliness	Currency of the data which is passed from one task to another within a process
Granularity	Level of detail, information passed from one task to another in a process must balance conciseness and completeness, in a completely integrated process, there is enough detail of information for people to perform each activity without overwhelming them with excessive detail

Source: Aubert et al. (2003)

5.6.2 Extended Enterprise Application Integration (EEAI)

In the previous section, we discussed the BPI within the concept of enterprise integration. While BPI mainly concerns the integration of processes, Enterprise Application Integration (EAI) is focused on information and data integration at application level (Raut and Basavaraja, 2003; Hasselbring, 2000; Themistocleous and Corbitt, 2006). EAI was driven by the need to integrate COTS applications that became available and widely used in mid 1990s to support specific business functions. The installation of these products required integration with related legacy systems. EAI originated as the practice of aligning COTS and legacy systems (Cummins, 2002) but it has become one of the most important approaches to solving SI problems within enterprises (Irani et al., 2003). Through solving the integration problem within enterprises, EAI offers some benefits to the enterprises. These benefits are listed by Yang and Lu (2005), Banerjee et al. (2005), Gold-Bernstein and Ruh (2004), Cummins (2002) and Gable (2002).

In our SIF, the term ‘extended enterprise’ represents the concept of viewing the enterprise together with its business partners, suppliers, and customers. Extended enterprise applications integration includes a web of relationships between a company and its employees, managers, partners, customers, suppliers, and markets. EEAI is

concerned with intra and inter enterprise operations and with improving their efficiency and effectiveness (Kosanke et al., 1999). Integration of extended enterprise application is driven by market globalisation, increase in mergers and acquisitions activities, need to comply to regulations, competitive environment which requires increased efficiency and reduced cycle times, cost and effort, and increased customer satisfaction (Gold-Bernstein and Ruh, 2004; Smith et al., 2002; Pinkston, 2001).

Lam (2005) identifies four main categories within extended enterprise integration initiatives: *EAI* – the integration of IT systems within an enterprise, typically to improve business efficiency and to meet needs for real-time information processing; *Web Integration* – the integration of legacy systems with web-based applications, driven by a need to provide customers with a web channel for accessing products, services, or Information; *Business-to-Customer (B2C) integration* – the integration of back-end transactional IT systems, which may be legacy in nature, with web-based front-end applications such as storefronts and personalisation engines to provide B2C solutions; and Business-to-Business (B2B) integration – the integration of IT systems between different organisations to support B2B activities such as integrated supply chain management. (Huang and Irawani, 2005) analysed the role of information sharing in optimising supply-chains. They emphasised that choosing the right partner for selective-information sharing can significantly reduce the manufacturer's cost. Their research focused on how factors such as the manufacturer's capacity and shortfall-holding cost ratio or retailers' market shares and order sizes can affect this partner-selection decision. Each EEAI solution may have different characteristics and utilise different architectures. Within these four categories EAI and B2B are commonly used in enterprise integration solutions. Although they share fundamental architectural principles, each has unique characteristics (Pinkston, 2001).

Despite many benefits of EEAI, many factors affect the success of its implementation. Venkatachalam (2006) groups these factors as people, process, and technology related challenges. People related challenges include change management, proper training, resource allocation, and the ability of the staff to meet what's needed. Process related challenges include systems migration, reengineering, and management. Technology related challenges include the functionality of hardware and software, management of systems and applications, and the integrity and usefulness of data.

A list of critical success factors for the effective implementation of extended enterprise integration can be grouped as business, organisation, technology, and project related (Themistocleous and Irani, 2001; Lam, 2005; Pinkston, 2001; Mendoza et al., 2006). Table 3 lists these factors for the success of EEAI. A similar study by Klein et al. (2007) investigated the benefits of sharing of information between B2B logistics partners, which must be enabled by the integration of disparate information systems across them. Based on data from 91 such dyadic relationships using inter-organisational IT, they found that performance gains accrue when parties share strategic information and customise IT; mutual trust enables IT customisation and strategic-information flows and equitable relationship-specific investments positively impact IT customisation, mutual trust, and performance.

Table 3 Critical success factors for EEAI

Business related factors	• Overall integration strategy
	• Strong business case for EAI
	• Process interoperability with business partners
Organisation related factors	• Top management support
	• Business process change and overcoming resistance to change
	• Good organisational and cultural fit. Handling legacy systems
Technology related factors	• Handling legacy systems
	• Planning technology
	• Common data standards
	• Use of right EAI tools
	• Use of mature technology
Project related factors	• Realistic project plans and schedule, client involvement
	• Communication, consultation and training
	• Required skills and expertise onboard
	• Vendor competence
	• Monitoring and feedback
	• Proper migration approach
	• Adequate testing plans

5.7 Program Management

SI initiatives involving large-scale projects and/or multi-development teams usually require an integrated PM approach. PM is the discipline which helps to manage complex SI programs and coordinate business processes, strategies, several systems, contractors, equipment/product suppliers and in-house staff (Hoffmann, 1992). PM and SI activities should not be considered as separate initiatives; rather PM should be structured such as to be leverage for integration activities. The two functions have important stakes in the success of the system. Therefore, it is important that these two functions do not have competing differences, contradictions, or inconsistencies in defining and realising a successful system.

A PM function structured to support SI aims:

- to provide the foundation for such an appropriate PM structure by defining the roles and responsibilities for the PM team
- to address communication within and across the development teams involved in integrating the system
- to provide a facility for cross-team discussion, and responsibility for overall program (SI) success
- to define the metrics and serve as a guideline for measuring and monitoring effective communication in all teams, at all levels
- to define quality criteria of the program and provide support for assessing quality and suggest improvements.

The Systems Integration Master Plan (SIMP) will be created in consultation with the PM as the resource allocation will be responsibility of the latter. An SIMP would include: system introduction in terms of system description, risk assessment, measures of effectiveness, critical technical parameters (KPP's, TPM's, etc.); an integrated test program summary in terms of a schedule and its management; a description of the development test and evaluation including the subsystem acceptance criteria; a description of the operational test and evaluation including an acceptance criteria against the operational requirements; and a resource summary for entire test and evaluation including equipment, sites, support equipment, and manpower and personnel. Identification and assessment of uncertainties and risks is a critical component of SI planning. Simulation models have been used for this purpose for their cost saving benefits but with their limitations. Wong et al. (2008) successfully developed a tool to measure supply chain performance in the real environment using Data Envelopment Analysis (DEA) supply chain model enhanced with Monte Carlo (random sampling) methodology. This method proves to be a cost saving and efficient way to handle uncertainties and could be used in other relevant fields other than supply chain, to measure efficiency.

Measurement and assessment of SI success is as much important a function as planning and coordinating for PM. Well-defined metrics are required to assess the success of SI initiatives. PM, planning and measurement lead to improved SI Maturity.

5.8 Testing, Verification and Validation

Testing, and Verification and Validation (V&V) are important aspects of SI which qualify the system for the ready use of the user. V&V is a collection of analysis and testing activities across the full life cycle and complements the efforts of other quality-engineering functions (Wallace and Fujii, 1989). Verification establishes the truth of correspondence between a product/system and its specification. The associated activities verify if we are building the right product/system, and that we are testing what we were supposed to test. The purpose of verification is to confirm that the specified design requirements are fulfilled by the system (ISO/IEC-15288, 2008). The results of Verification provide the information required to effect the remedial actions that correct non-conformances in the realised system or the processes that act on it. Validation is the act of establishing the fitness of a product/system for its deployment capability. An example is the comparison of the actual system response of an online transaction to what was originally expected, requested, and finally approved. Validation establishes the fitness of a product/system for its operational mission based on operational or field testing. The purpose of the validation is to provide objective evidence that the services provided by a system when in use comply with stakeholders' requirements (ISO/IEC-15288, 2008). This process performs a comparative assessment and confirms that the stakeholders' requirements are correctly defined. Where variances are identified, these are recorded and guide the corrective actions. Both V&V activities should be traced back to the system requirements. Thus both V&V are back end processes that aid and ensure effective SI. The terms qualification and testing are often used synonymously with V&V.

The quality of V&V directly impacts the overall outcome of SI. This has been strongly felt in the development of embedded software systems. In industrial applications using advanced embedded technologies, real-time software quality has a continuously increasing impact on system quality. A systematic approach using different testing methods to predict the defects and performance is proposed by Amuthakkannan et al. (2008) to improve the quality of real-time software by identifying the defects in software product and improving the development processes. The minimum and optional set of tasks required for V&V and their issues are discussed in Wallace and Fujii (1989) and IEEE-1012 (2004). In the SIPM (Jain et al., 2008a) the prerequisites of the integration V&V phase are addressed in the upstream activities of SI process. These upstream activities provide the required artifacts and requirements for an upfront and effective planning of the V&V phase. Ten such supporting activities of V&V from other phases of SIPM were identified. These are:

- *Integration Requirements Phase*: (Derive Qualification Requirements, Derive Test Cases).
- *Integration Architecture Phase*: (Develop Semantics for Integration, Develop Semantic Specification, Develop Qualification Architecture, Develop Test Sequence, Develop Test Environments Specifications, Test Architecture).
- *Integration Planning Phase*: (Develop System Prototype, Test System Prototype). These ten activities along with the V&V activities aid in achieving better quality of integration V&V.

Qualification requirements address the needs to qualify the system as being designed right, the right system, and an acceptable system (Buede, 2000). Qualification is the process of verifying and validating the system design and then obtaining the stakeholder's acceptance of the design. Qualification is associated with testing, acceptance, V&V. The four elements of the qualification requirements (Buede, 2000) are

- *Observance*: How the estimates (qualification data) for each input/output and system-wide requirements will be obtained, that is, test analysis and simulation, inspection, or demonstration.
- *Verification plan*: How the qualification data will be used to determine that the real systems conforms to the design that is developed.
- *Validation plan*: How the qualification data will be used to determine that the real system complies with the originating requirements.
- *Acceptance plan*: How the qualification data will be used to determine that the real system is acceptable to the stakeholders.

Qualification architecture involves defining and managing the test activities as shown in Figure 4. Test approaches are developed to provide the objectives, schedule, environment requirements, and entry and exit criteria for the test stages. Based on these approaches tests are planned and prepared to identify test conditions, and test cycles, and to define the input data and expected results. It is important to establish the appropriate test environments and to ensure that they are tested prior to the execution. Test cases and sequence for test executions are derived based on the system requirements and the test approaches.

During Verification And Validation (V&V) the test scripts are verified and validated, test environment is developed and the tests are executed in its sequence. These activities are based on the developed qualification plan. There are two types of testing, Functional/Life Cycle Approach and Structural. Figure 5 shows some of these tests, their order in the lifecycle, their dependencies, and how the front end activities are traced to these tests. In functional testing test the functionality of the system and ensure that the user functional requirements and specifications are met. Test conditions are generated to evaluate the correctness of the application. In structural testing we test the structural and physical capability of the system and ensure that the system is structurally and technically sound, can perform the intended tasks, and that the components integration works cohesively. These include backup and recovery testing, contingency testing, job stream testing, operational testing, performance testing, security testing, and stress/volume/scalability testing. The test results are logged and documented. The errors are fixed through rework and change control management. IEEE-1012 (2004) provides detail guidelines on how to perform V&V and what to report at each phase of development for attaining all the benefits of V&V.

Figure 4 Test lifecycle

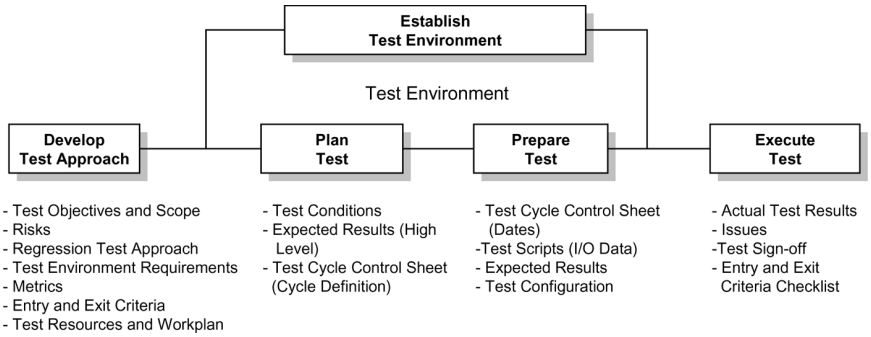
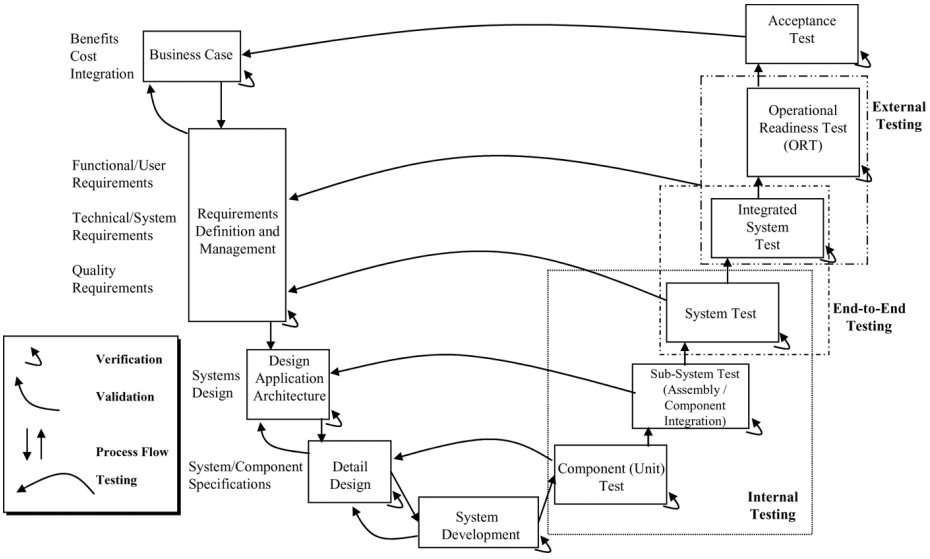


Figure 5 System Verification and Validation



5.9 Configuration Management

Systems need to adapt to the rapid change driven by the global dynamic marketplace, technological evolution, and variety of environments (Fricke and Schulz, 2005). Large and complex systems often face this kind of rapid change and they need new configuration or have to be ported and adopted to different environments (Feiler, 1990). A systematic approach is required in order to keep track of this continuous evolution of a system and its environment. Traceability of changes is critical in controlling and managing their impact on the SI lifecycle. Since change is inevitable for a system, ability to trace and control those changes becomes more important in increasing the effectiveness of SI. CM plays a key role in achieving SI effectiveness and is part of every sub-process of integration (Jain et al., 2008a).

CM is the process of controlling the evolution of families of systems and provides techniques, methods, and procedures to maintain a product history, to uniquely identify and locate each version of a system, and to initiate, evaluate, and control change to the product during development and after release (Feiler, 1990; Stevens et al., 1998; Hass, 2003). In many systems development and system's quality related literature and international standards CM is reported as a key activity in increasing the quality of systems, products and processes. IEEE (IEEE-729-1998) defines the term CM as the process of identifying and defining components in a system, controlling the release and change throughout the life cycle, recording and reporting the status of components and change requests, and verifying the completeness and correctness of system components. The four key elements of CM, as reflected in this definition, are identification, control, accounting, and audit. They are essential to maintaining product and process integrity, providing change management, and providing visibility.

CM is also required for the success of other components of SIF including Requirements Management, Legacy and COTS SI, Interface Management, and Testing, V&V. CM supports the functional and physical configuration of the SI. The primary objective of CM is to ensure effective management of the evolving configuration of a system, both hardware and software, during its lifecycle. Fundamental to this objective is the establishment, control, and maintenance of software and hardware functional, allocated, development, test, and product baselines. Baselines are reference points for maintaining development and control. These baselines, or reference points, are established by review and acceptance of requirements, design, and product specification documents. They may reflect the end of one phase or segment of a phase as well as the beginning of a new phase. The baseline may reflect the standard project milestone event, and the event reviews. As each segment of a phase is approved the software and/or hardware baseline is placed under configuration control (INCOSE, 2006).

Effectiveness of CM activities depends on several factors. One approach measures the CM effectiveness in terms of consistency, stability, and traceability (Stevens et al., 1998). This approach emphasises the importance of consistent and stable information for CM effectiveness in order to increase the traceability of the change requirements throughout the system.

Besides its benefits, CM is a challenging task for systems development. Dart classifies the CM related challenges into five categories as technological, process-oriented, managerial, political, and standardisation issues and suggests a framework for successful CM adoption (Dart, 1991) as shown in Table 4.

Table 4 Framework for successful CM adoption

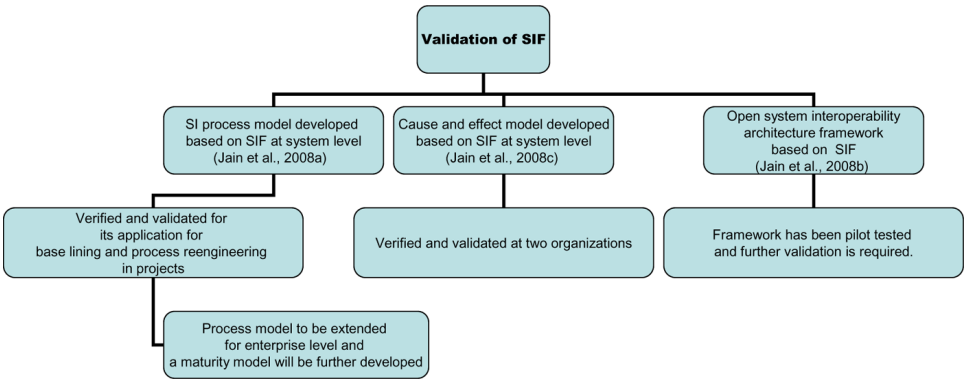
Phase 1	Determine CM status and needs	This phase requires assessment and documentation of the current status and requirements for the Configuration Management. Gathering management support, determining necessary processes, prioritising the CM needs, designating the CM adoption team, and getting involvement of the potential role players in the CM solution are also supportive activities within this phase
Phase 2	Examine state-of-the-art in CM tools	This phase requires identification and evaluation of CM systems
Phase 3	Evaluate candidate CM tools	In this phase, the alternatives are rated and selected based on the evaluation criteria
Phase 4	Write the adoption plan	In this phase, adoption guide and plan for strategy is written, budget is designated, and education and training needs are assessed. A pilot project is selected, in-house standards, policies, and procedures are written, and success criteria and milestones for adoption and for pilot project are defined
Phase 5	Implement a pilot project	In this phase, pilot project is implemented, evaluated and monitored. If required, adoption plan is updated based on pilot project results
Phase 6	Institutionalise CM system	In this phase, policies and procedures are refined at all levels for transitioning to the new CM system, CM system to each group is tailored, training courses, consulting support, user groups are planned. Procedures, policies, processes based on user feedback to improve quality and acceptability are evaluated against success criteria
Phase 7	Prepare to start again	Once the CM system cannot be further tailored and extended, a new adoption process is adopted. In this phase metrics such as timing, scheduling, people numbers are assessed, evaluation criteria, plans, strategies, resistance handling procedures are reviewed

Source: Dart (1991)

6 Framework validation

The framework has been developed over the last several years of research, practice, and teaching by the primary author of this paper. The framework has been the basis of further development of detailed SI activities and processes. These have been used to assess SI process improvement and maturity at three different organisations and close to a dozen different development projects. Several related researches have been conducted by researchers on the different aspects of SI of the framework. Some of these areas include defining a SI process model based on the framework (Jain et al., 2008a), analysing the impact of system architecture on integration complexity (Jain et al., 2008c), and defining open systems interoperability and based on this building an architecture assessment framework. Figure 6 shows the areas of research emerging out the SIF that are currently being pursued and completed. Some of the findings have already been published in peer reviewed journals while others are being currently reviewed for publication.

Figure 6 SIF validation (see online version for colours)



7 Conclusion

SI is an emerging field and the most important part of systems engineering. SI requires a holistic end-to-end approach for purposes of better traceability starting early on. There is a need for a model framework that clearly specifies all aspects of SI and can be used as a baseline to assess SIM. This baseline framework can be used to assess the SI activities over the entire lifecycle in terms of how well they are defined, measured, and managed.

In this paper we defined such a comprehensive approach to SI and introduced a SI framework and discussed how SI maturity, readiness and integration-friendliness measures are built into this framework. The framework identifies four critical development processes for SI, namely integrated requirements engineering, interface control and management, V&V, and CM. The framework also addresses common issues of SI in terms of interoperability, legacy and COTS integration, and enterprise and extended enterprise integration.

The authors have further developed a SI process model by defining the SI processes and activities that would result in system operational effectiveness based on the framework. The proposed framework will provide guidance in identifying the weak areas and gaps and help in allocating efforts and resources appropriately. The authors have also analysed the application of this framework and the process model to assess the effectiveness of SI on projects based on the level of SI activities being performed (Jain et al., 2008a). Future areas of research would involve understanding how these SI activities would help in assessing the integration readiness and maturity of a system by identifying corresponding metrics for each of the activities. Towards this goal the authors are currently working on developing a SI readiness indicator and a model to assess and optimise SI maturity in terms of integration readiness, integration complexity, and operational effectiveness. Such a model will facilitate the projects in assessing the impact of SI activities on system quality and performance and understanding the scope of optimising the SI process. This will go a long way in reducing cost and time for product development and improving competitiveness.

References

Abts, C. (2000) *A Perspective on the Economic Life Span of COTS-based Software Systems: The COTS-LIMO Model*, TR USC-CSE-2000-503, USC Center for Software Engineering, University of Southern California, Los Angeles.

Abts, C., Boehm, B. and Bailey, E. (2000) *COCOTS: A COTS Software Integration Lifecycle Cost Model – Model Overview and Preliminary Data Collection Findings*, TR USC-CSE-2000-501, USC Center for Software Engineering, University of Southern California, Los Angeles.

Abuelma'atti, O., Merabti, M. and Askwith, B. (2006) 'A wireless networked appliances interoperability architecture', *1st International Symposium on Wireless Pervasive Computing*, 16–18 January, Phuket Thailand.

Alagumurthi, N., Palaniradja, K. and Soundararajan, V. (2008) 'Optimisation of process parameters in grinding on different dimensions and perspectives', *Int. J. Industrial and Systems Engineering*, Vol. 3, pp.447–473.

Amuthakkannan, R., Kannan, S.M., Selladurai, V. and Vijayalakshmi, K. (2008) 'Software quality measurement and improvement for real-time systems using quality tools and techniques: a case study', *Int. J. Industrial and Systems Engineering*, Vol. 3, pp.229–256.

Aubert, B.A., Vandenbosch, B. and Mignerat, M. (2003) *Towards the Measurement of Process Integration*, Cirano Scientific Series, Montreal.

Banerjee, N., Chordia, A. and Rajib, P. (2005) 'Seamless enterprise computing using Enterprise Application Integration (EAI)', *Journal of Services Research*, Vol. 5, pp.171–196.

Bansler, J.P. and Havn, E.C. (1994) 'Information systems development with generic systems', *Proceedings of the 2nd European Conference on Information Systems*, Nijenrode University Press, Breukelen, The Netherlands, 30–31 May, pp.707–715.

Basili, V. and Boehm, B. (2001) 'COTS-Based Systems Top 10 List', *IEEE Computer*, Vol. 34, No. 5, pp.91–95

Bianchi, A., Caivano, D., Marengo, V. and Visaggio, G. (2003) 'Iterative reengineering of legacy systems', *IEEE Trans. Softw. Eng.*, Vol. 29, pp.225–241.

Boehm, B. and Abts, C. (1996) 'COTS integration: plug and pray?', *IEEE Computer*, Vol. 32, No. 1, pp.135–138.

Brosey, W.D., Neal, E.R. and Marks, D.F. (2001) 'Grand challenges of Enterprise Integration', *8th IEEE International Conference on Emerging Technologies and Factory Automation*, Antibes-Juan les Pins, France, Vol. 2, 15–18 October, pp.221–227.

Buede, D. (2000) *The Engineering Design of Systems*, John Wiley and Sons, Inc., New York, USA.

Carlock, P.G. and Fenton, R.E. (2001) 'System of Systems (SoS) enterprise systems engineering for information-intensive organizations', *Systems Engineering Journal*, Vol. 4, pp.242–261.

Cummins, F.A. (2002) *Enterprise Integration: An Architecture for Enterprise Application and Systems Integration*, John Wiley and Sons, Inc., New York, USA.

Dart, S.A. (1991) 'Concepts in Configuration Management Systems', *Proceedings of the 3rd International Workshop on Software Configuration Management*, Trondheim, Norway, 12–14 June, pp.1–18.

DAU (2001) *System Engineering Fundamentals*, Defense Acquisition University Press, Jan, Fort Belvoir, USA.

Davenport, T. (1993) *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, Boston.

Davis, L., Gamble, R. and Payton, J. (2002) 'The impact of component architectures on interoperability', *Journal of Systems and Software*, Vol. 61, No. 1, pp.31–45.

DOD 4120.24-M (2000) *Defense Standardization Program Policies and Procedures*, March, Office of the Secretary of Defense, Fort Belvoir, USA.

DODD-5000.1 (2003) *Defense Acquisition System, Department of Defense Directive*, May, Office of the Secretary of Defense, Fort Belvoir, USA.

Donzelli, P., Zelkowitz, M., Basili, V., Allard, D. and Meyer, K.N. (2005) 'Evaluating COTS component dependability in context', *IEEE Software*, Vol. 22, No. 4, pp. 46–53.

Egyed, A., Müller, H. and Perry, D. (2005) 'Integrating COTS into the Development Process', *IEEE Software*, Vol. 22, No. 4, pp.16–18.

EIA-632 (1999) *Standard for Processes for Engineering a System (Upgrade and Revision of EIADS-632)*, Electronics Industry Alliance, GEIA, Arlington, USA.

FED-STD-1037C (1996) *Federal Standard 1037C Glossary of Telecommunication Terms*, FTSC/NCS, Institute of Telecommunication Sciences, Aug, Boulder, USA.

Feiler, P.H. (1990) *Software Configuration Management: Advances in Software Development Environments*, Carnegie-Mellon University Software Engineering Institute, Pittsburgh.

Fricke, E. and Schulz, A.P. (2005) 'Design for Changeability (DfC): principles to enable changes in systems throughout their entire lifecycle', *Systems Engineering Journal*, Vol. 8, pp.342–359.

Gable, J. (2002) 'Enterprise application integration', *Information Management Journal*, Vol. 36, pp.48–52.

Garlan, D., Allen, A. and Ockerbloom, J. (1995) 'Architectural mismatch: why reuse is so hard', *IEEE Software*, Vol. 12, No. 6, pp.17–26.

Gieszl, L.R. (1992) 'Traceability for integration', *Proceedings of the Second International Conference on Systems Integration*, Morristown, USA, 15–18 June, pp.200–228.

Gold-Bernstein, B. and Ruh, W. (2004) *Enterprise Integration: The Essential Guide to Integration Solutions*, Addison-Wesley, Boston.

Grady, J.O. (1994) *Systems Integration, CRC Press*, Lincoln, USA.

Grant, J. (2000) 'Ensuring successful implementation of commercial off-the-shelf products (COTS) in Air Force Systems', *Fort Belvoir*, Defense Systems Management College Press, VA.

Gulledge, T. (2006) 'What is integration?', *Industrial Management + Data Systems*, Vol. 106**,** pp.5–20.

Hammer, M. and Champy, J. (1993) *Reengineering the Corporation: A Manifesto for Business Revolution*, Harper Business, New York.

HASS, A.M.J. (2003) *Configuration Management Principles and Practice*, Pearson Education, Boston.

Hasselbring, W. (2000) 'Information System Integration', *Association for Computing Machinery. Communications of the ACM*, Vol. 43, pp.32–38.

Hoffmann, K.C. (1992) 'Management of enterprise-wide systems integration programs', *Proceedings of the Second International Conference on Systems Integration*, 15–18 June, Morristown, USA, pp.4–13.

Huang, A., Yen, D.C., Chou, D.C. and Xu, Y. (2003) 'Corporate applications integration: challenges, opportunities, and implementation strategies', *Journal of Business and Management*, Vol. 9, pp.137–150.

Huang, B. and Irawani, S.M.R. (2005) 'Production control policies in supply chains with selective-information sharing', *Operations Research*, Vol. 53, pp.662–666.

IEEE 1233 (1998) *IEEE Guide for Developing System Requirements Specifications*, New York, USA.

IEEE 830 (1998) *IEEE Recommended Practice for Software Requirements Specifications*, New York, USA.

IEEE-1012 (2004) *IEEE Standard for Software Verification and Validation*, New York, USA.

IEEE-729-1998 (1998) 'IEEE standard glossary of software', *Engineering Terminology*, *IEEE*.

INCOSE (2006) *Systems Engineering Handbook*, International Council on Systems Engineering (INCOSE), Vol. 3, San Diego, USA.

Irani, Z., Themistocleous, M. and Love, P.E.D. (2003) 'The impact of enterprise application integration on information system lifecycles', *Information and Management*, Vol. 41, pp.177–187.

ISO/IEC-15288 (2008) *Systems Engineering – System Life Cycle Processes*, International Organization for Standardization/International Electrotechnical Commission, Geneva, Switzerland.

ISO/IEC-7498-1 (1994) *Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, Geneva, Switzerland.

Jain, R. (2007) *Lecture Notes for SYS 605 System Integration*, Stevens Institute of Technology, Hoboken, USA.

Jain, R., Chandrasekaran, A. and Erol, O. (2008a) 'A Systems Integration Framework for process analysis and improvement', *SE Journal*, John Wiley (Accepted for Publication – Details not known yet)

Jain, R., Chandrasekaran, A., Elias, G. and Cloutier, R. (2008c) 'Exploring the impact of system architecture and systems requirements on systems integration complexity', *IEEE Systems Journal*, Vol. 2, p.2.

Jain, R., Chandrasekaran, A., Frederick, C. and Albertelli, P. (2008b) *Open System Interoperability: An Architecture Assessment Framework*, Working Paper.

Jarke, M. (1998) 'Requirements tracing', *Commun. ACM*, Vol. 41, pp.32–36.

Kelkar, A. and Gamble, R. (1999) 'Understanding the architectural characteristics behind middleware choices', *1st International Conference in Information Reuse and Integration*, ISCA, Atlanta, 4–6 November, USA.

Klein, R., Rai, A. and Straub, D.W. (2007) 'Competitive and cooperative positioning in supply chain logistics relationships', *Decision Sciences*, Vol. 38**,** pp.611–646.

Konstantas, D. (1996) 'Migration of legacy applications to a CORBA platform: a case study', *Proceedings of the IFIP/IEEE International Conference on Distributed Platforms: Client/Server and Beyond: DCE, CORBA, ODP and Advanced Distributed Applications*, Freiburg, Germany, February 27–March 1, pp.100–112.

Kosanke, K., Vernadat, F. and Zelm, M. (1999) 'CIMOSA: enterprise engineering and integration', *Computers in Industry*, Vol. 40, pp.83–97.

Kuhn, R. (1990) 'On the effective use of software standards in Systems Integration', *Proceedings of 1st International Conference on Systems Integration*, IEEE Computer Society Press, Morristown, USA, pp.455–461.

Lam, W. (2005) 'Investigating success factors in enterprise application integration: a case-driven analysis', *European Journal of Information Systems*, Vol. 14, pp.175–187.

Land, R. and Crnkovic, I. (2003) 'Software Systems Integration and architectural analysis – a case study', *Proceedings of International Conference on Software Maintenance*, Amsterdam, The Netherlands, 22–26 September, pp.338–347.

Lemahieu, W., Snoeck, M., Goethals, F., Backer, M., Haesen, R., Dedene, G. and Vandenbulcke, J. (2005) 'Coordinating COTS applications via a business event layer', *IEEE Software*, Vol. 22, No. 4, pp.28–35.

McConnell, S. (1996) 'Software quality at top speed', *Software Development*, Vol. 4, pp.38–42.

McManus, H. and Hastings, D. (2005) 'A framework for understanding uncertainty and its mitigation and exploitation in complex systems', *EEE Engineering Management Review*, Vol. 34, No. 3, Third Quarter 2006, pp.81–94 (Originally published in 15th INCOSE Symposium, Rochester, NY, 10–15 July, 2005).

Mendoza, E., Mendoza, P. and Grimán, A. (2006) 'Critical success factors for managing systems integration', *Information Systems Management*, Vol. 23, p.56.

Mische, M.A. (1998) 'Defining systems integration', in Mische, M.A. (Ed.): *Reengineering: Systems Integration Success*, CRC Press LLC, Florida, USA, pp.6, 7.

Morisio, M. and Torchiano, M. (2002) *Definition and Classification of Cots: A Proposal*, ICCBSS, Orlando, FL.

Mykkänen, J.A. and Tuomainen, M.P. (2008) 'An evaluation and selection framework for interoperability standards', *Information and Software Technology*, Vol. 50, pp.176–197.

Nilsson, E.G., Nordhagen, E.K. and Oftedal, G. (1990) 'Aspects of systems integration', *Proceedings of the First International Conference on Systems Integration*, Morristown, USA, 23–26 April, pp.434–443.

Noor-E-Alam, M., Hasin, M.A.A., Ullah, A.M.M.S. and Lipi, T.F. (2008) 'Supplier evaluation with GD-based multi criteria decision making', *Int. J. Industrial and Systems Engineering*, Vol. 3, pp.368–381.

OSD (2002) *Commercial Item Acquisition: Considerations and Lessons Learned*, Office of the Secretary of Defense, Fort Belvoir, USA.

Ouksel and Sheth (1999) 'Semantic interoperability in global information systems: a brief introduction to the research area and the special section', *SIGMOD Record*, Vol. 28, No. 1, pp.5–12.

Payton, J., Keshav, R. and Gamble, R.F. (1999) 'System development using the integrating component architecture process', *Proceedings of the FirstWorkshop on Ensuring Successful COTS Development*, Los Angeles, USA, pp.49–51.

Pimmler, T.U. and Eppinger, S.D. (1994) 'Integration analysis of product decompositions', *ASME Conference on Design Theory and Methodology*, Minneapolis, MN.

Pinkston, J. (2001) 'The ins and outs: How EAI differs', *Enterprise Application Integration Journal*, August 2001, pp.48–52.

Putrycz, E., Woodside, M. and Wu, X. (2005) 'Performance techniques for COTS systems', *IEEE Software*, Vol. 22, No. 4, pp.36–44.

Ramesh, B. (1998) 'Factors influencing requirements traceability practice', *Commun. ACM*, Vol. 41, pp.37–44.

Raut, A. and Basavaraja, A. (2003) 'Enterprise Business Process Integration', *TENCON 2003: Conference on Convergent Technologies for Asia-Pacific Region*, Vol. 4, 15–17 October, pp.1549–1553.

Rouse, W.B. (2005) 'Enterprises as systems: essential challenges and approaches to transformation', *Systems Engineering Journal*, Vol. 8, pp.138–150.

Sage, A.P. and Lynch, C.L. (1998) 'Systems Integration and architecting: an overview of principles, practices, and perspectives', *The Journal of Systems Engineering*, Vol. 1, pp.176–227.

Sanchez, R. (2000) 'Modular architectures, knowledge assets, and organizational learning: new management processes for product creation', *Int. J. Technology Management*, Vol. 19, pp.610–629.

SEI (2002) *Evolutionary Process for Integrating COTS-Based Systems: A Overview*, Software Engineering Institute, Carnegie Mellon.

SEI/CMU (2004) *Integration and Interoperability Models for Systems of Systems*, Carnegie Mellon, Software Engineering Institute, Pittsburgh, PA.

Smith, D., O'Brien, L., Kontogiannis, K. and Barbacci, M. (2002) 'Enterprise Integration', *Architect (SEI Interactive News)*, 4Q.

Stevens, R., Brook, P., Jackson, K. and Arnold, S. (1998) *Systems Engineering, Coping with Complexity*, Pearson Education.

Stohr, E.A. and Nickerson, J.V. (Eds.) (2003) *Intra Enterprise Integration: Methods and Direction*, Oxford University Press, New York.

Sugarbroad, I. (1990) 'An OSI-based interoperability architecture for managing hybrid networks', *IEEE Communications Magazine*, Vol. 28, No. 3, March, pp.61–69.

Themistocleous, M. and Corbitt, G. (2006) 'Is Business Process Integration feasible?', *Journal of Enterprise Information Management*, Vol. 19, pp.434–449.

Themistocleous, M. and Irani, Z. (2001) 'Benchmarking the benefits and barriers of application integration', *Benchmarking: An International Journal*, Vol. 8, pp.317–331.

Venkatachalam, A.R. (2006) 'A holistic approach on Enterprise Integration', *Journal of Information Technology Case and Application Research*, Vol. 8, pp.1–6.

Voas, J. (2001) 'Faster, better, cheaper', *IEEE Software*,Vol. 18, No. 3, pp.96, 97.

Wallace, D.R. and Fujii, R.U. (1989) 'Software verification and validation: an overview', *IEEE Software*, Vol. 6, No. 3, pp.10–17.

Warboys, B., Snowdon, B., Greenwood, R.M., Seet, W., Robertson, I., Morrison, R., Balasubramaniam, D., Kirby, G. and Mickan, K. (2005) 'An active-architecture approach to COTS integration', *IEEE Software*, Vol. 22, No. 4, pp.20–27.

Whitt, L. (2004) *The Good, The Bad, and The Ugly of Interoperability Metrics*, Northrop Grumman Mission Systems Presentation, San Diego, United States, 2–6 February.

Wong, W.P., Jaruphongsa, W. and Lee, L.H. (2008) 'Supply chain performance measurement system: a Monte Carlo Dea-based approach', *Int. J. Industrial and Systems Engineering*, Vol. 3, pp.162–188.

Yakimovich, D. (2001) *Dissertation: A Comprehensive Reuse Model for COTS Software Products*, Director: Basili, V., Doctoral Thesis, p.174.

Yakimovich, D., Bieman, J. and Basili, V. (1999) 'Software architecture classification for estimating the cost of cots integration', *Proceedings of the 21st International Conference on Software Engineering*, 16–22 May, Los Angeles, United States, pp.296–302.

Yang, H.M. and Lu, F.V. (2005) 'Integrating inter- and extra-enterprise applications using web services', *Review of Business*, Vol. 26, pp.3–9.

Yang, Y., Bhita, J., Port, D.N. and Boehm, B. (2005) 'Value-Based processes for cots-based applications', *IEEE Software*, Vol. 22, No. 4, pp.54–62.

Zhu, J., Tian, Z., Li, T., Sun, W., Ye, S., Ding, W., Wang, C.C., Wu, G., Weng, L., Huang, S., Liu, B. and Chou, D. (2004) 'Model-driven Business Process Integration and management: a case study with the Bank SinoPac regional service platform', *IBM Journal of Research and Development*, Vol. 48, pp.649–669.

Notes

[1]We thank Professor Angappa Gunasekaran, the editor, and the anonymous reviewers for their constructive and very helpful feedback during the review process. Their comments helped us revise the paper in a very useful manner.

[2]The acronyms used are RM: Requirements Management, I/F: Interface Control and Management, V&V: Verification and Validation, CM: Configuration Management, IO: Interoperability, STD: Standards, EEAI: Extended Enterprise Application Integration, BPI: Business Process Integration, COTS: Commercial Off The Shelf, and LI: Legacy Integration.