INTEGRATION STANDARDS FOR CRITICAL SOFTWARE INTENSIVE SYSTEMS

Victoria Stavridou
Department of Computer Science
Queen Mary and Westfield College
University of London
Mile End Road
London E1 4NS
email: victoria@dcs.qmw.ac.uk

Abstract

"Integration" is a term often misused and misunderstood. The existence of integration as an activity in system development is undeniable. Its existence as a phase of the system lifecycle which requires planning and management is, however, frequently ignored. This omission is undesirable, particularly in critical systems where lack of integration planning may affect the system's ability to meet its criticality objectives. Not surprisingly, this confusion is reflected in many standards. It is the purpose of this paper to identify the way various critical systems standards deal with integration and, as a corollary, to suggest possible standardisation activities as well as ways in which organisations relying on software intensive systems for critical functions can approach this problem.

1 Introduction

"Integration" is an often misused term. Typically, it means different things to different communities with predictably confusing results. Integration planning, integration achievement and integration validation are often used interchangeably. This confusion is very harmful; by not understanding what integration is, opportunities for planning and budgeting for it are not used leading to development schedule overruns and ultimately, we believe, to inferior software systems. The size and complexity of typical software intensive systems is such that this problem is likely to be exacerbated in the future. Where the application domain involves critical tasks, the system's ability to meet its criticality objectives may be compromised. It is for these reasons that this article is particularly concerned with integration terminology.

As a first step, we need to distinguish between:

- *What integration is*, that is the conceptual *models* of integration as well as the *kinds* of integration in use.
- *How integration is achieved*, that is, development *strategies*, *mechanisms* enabling integration and integrated system *architectures*.
- *How integration is verified*, that is, *testing* of a developed system to ascertain whether the integration goals have been achieved.

These distinctions, although rather obvious, are ignored surprisingly often in the literature. For instance, B. Beizer routinely equates "integration" with "integration testing" and "integration strategy" [7]. Some useful distinctions are drawn by the European Software Factory team in [23]. Below we define the kinds of integration that are relevant to software intensive systems.

Systems Integration is the practice of combining the functions of a set of subsystems, be it software, hardware or both, to produce a single, unified system that satisfies some need of an organisation (paraphrased from [20]).

This definition is intentionally very general and its components would, in principle, involve not just digital systems but also analog components, power supplies, enclosures, connectors, tools and so on. However, in this article we use the term "systems integration" to denote "digital systems integration". The components of digital systems integration typically are:

- software subsystems (applications, system, custom)
- hardware subsystems (single/multi processor(s))

Software integration is the practice of assembling a set of software components/subsystems to produce a single, unified software system that supports some need of an organisation.

Typical software integration components are applications, reusable modules, custom-made modules, Commercial-Off-the-Shelf systems (COTS) and Government Off-the-Shelf systems (GOTS). *Integration testing* is a very different notion and can be defined as:

Integration Testing is a set of procedures designed to verify whether a given assembly of components (be they systems, software or tools) does indeed satisfy the requirements of an organisation.

2 UK MOD POLICY

In this section we summarise and evaluate current UK Ministry of Defence (MOD) policy on integration issues. We do this by examining individual documents. Our observations are then used to form an overall picture of current MOD policy as articulated in the examined documents. The criterion for including documents in the discussion is that they either *do* or they *should* contain guidance on integration issues.

2.1 Joint MOD/Industry Computing Policy for Military Operational Systems (JPS)

The JPS is a very significant document which provides guidance to MOD project managers on a wide spectrum of issues relating to the specification and procurement of operational computer based systems [19]. JPS is significant because it reflects MOD policy (and industrial support therein) where this has been established. It therefore clearly illustrates consensus within the UK defence and industrial sectors.

Although integration is not explicitly addressed by the JPS, some integration issues are dealt with under the following headings:

- **Project Support Environments and interface standards**. The adoption and development of a set of open interface standards is advocated with added emphasis on Public Tool Interfaces (PTI) and open standards.
- **Architectures**. Whilst MOD has no specific policy on software architectures, it is recognised that rapid development in this area is driven by open systems concepts, standards and products. Disappointingly, JPS contends that it is too early to give firm guidance to projects.
- **COTS software**. JPS recognises the potential advantages of COTS and software reuse and draws attention to the importance of considering the feasibility of integration into existing systems and the incident risks.

Thus MOD is aware of at least some integration issues at a relatively high policy level. On the other hand, the failure to make explicit recommendations on established low level integration standards is a weakness.

2.2 DEFSTAN 05-95/3

DEFSTAN 05-95/3 defines Quality System Requirements for the Design, Development, Supply and Maintenance of Software [8]. It sets out contract specific requirements for managing the software design, development, supply and maintenance processes in order to achieve and assure quality of software. The application of 05-95 is not restricted to any particular form or type of software.

The standard does not make any explicit provision for integration except to say that the contractor's test plan must give consideration to integration testing. However, it does address the practical issue of incorporating existing software into software products delivered by contractors, thereby recognising the need to perform *de facto* integration by *grafting* [11]. The requirement is well defined in that it asks the contractor to ensure that "objective evidence exists that the ability of the software to perform the required functions has been evaluated and validated". The respective guidance is much less well defined however (for example it advises that "documentation requirements for existing software may include functional and interface specifications").

In essence, 05-95 recognises the existence of some integration issues in MOD procurement, but has little to offer in the way of practical or comprehensive guidance.

2.3 Draft DEFSTAN 00-55

Draft DEFSTAN 00-55 [9] sets out the requirements for procedures and technical practices for the development of safety critical software. The requirements of the standard extend to tools and support software used to develop, test, certify and maintain software through all phases of the project lifecycle.

The standard is currently under development and the treatment of integration varies from version to version. At the time of writing, the latest (and not yet published) draft of July 1996, contains three points relevant to integration:

- **Use of previously defined software**. The standard allows the use of previously developed software but requires extensive reverse engineering (the need for which may be partially mitigated by the use of appropriate service history). Guidance in Part 2 of the Standard effectively precludes the use of COTS in safety critical software since it assumes access to COTS sources.
- **Integration** as a phase is identified but no guidance is given on strategies for achieving it.
- **Integration testing** requirements specify that "integration tests shall as a minimum demonstrate the correctness of all interfaces". Guidance on this states that for some systems "it may be possible to omit integration testing as a separate exercise".

In its current form, 00-55 does not seriously attempt to address the problems of software integration. At best, it is concerned with the impact of software reuse on safety but fails to offer direction or guidance on achieving integration in a manner consistent with the safety objectives of the software.

2.4 JAC Paper 1231

Joint Airworthiness Committee (JAC) Paper 1231 [18] is a proposal for sections within DEFSTAN 00-970 [1] covering software and its use in safety critical applications. Two chapters of the proposal are of interest; *software* and the *software lifecycle*.

- **Software**. Paper 1231 is very significant in that it determines other applicable standards as well as laying out its own requirements. In particular it provides the route for selecting either DEFSTAN 00-55 and DEFSTAN 00-56 or RTCA/DO-178B[24] for specific projects. Three requirement areas are relevant to integration:
  - **Interface requirements** contain comprehensive guidance on specifying interface requirements. In particular, it is stated that "the interface requirements specification shall identify all of the relationships that exist between the software configuration item and other software and hardware configuration items comprising the whole system". Although this provision is not directly related to integration, it is so indirectly since a thorough recording of interface specifications is key for successful integration.
  - **Software/Hardware integration testing** is required to verify that the software meets those requirements of the system specification that apply to a software configuration item when operating in its host hardware. No explicit guidance is given on achieving this goal.
  - **Reuse of software** contains guidance on the use of user modifiable software as well as pre-existing software modules but not COTS. The guidance is primarily concerned with the impact of reusing software on safety and does not elaborate on the integration aspects.
- **Software lifecycle** considers concept formulation and feasibility, project definition and full development. Full development is subdivided into a number of activities including:
  - **Software Integration** involves the integration of tested and evaluated software units into software components which should subsequently be tested for compliance with the requirements. Guidance specifies a number of software products (such as test results) which should be evaluated for compliance with the requirements. It is also specified that a test readiness review should be conducted to establish whether the software is ready to start formal acceptance testing.
  - **Software/Hardware Integration**. The purpose of this phase is to demonstrate that the verified flight resident software and the deliverable hardware operate together to satisfy the specified system functional requirements. The section contains explicit procedural and documentation requirements, including an independent review of the source code, the software design documents and the test report for each software configuration item.
  - **System integration and testing** phase involves rig, ground and flight testing of the

complete system as specified by the design authority. On completion of this phase, the design authority holds a final design review, the purpose of which is to confirm that the system as developed satisfies all of the specified requirements. A commitment to production would normally follow.

Paper 1231 stands out from the rest of the MOD related documents, in that it *explicitly* recognises the need for planned integration in system development. It concentrates on procedural and documentation requirements for the integration phases rather than giving guidance on how integration might be achieved. The latter is its chief weakness. Nonetheless, it contains the most explicitly articulated MOD policy on the integration issue, albeit in connection with flight systems.

2.5 Summary

We have found that current articulated MOD policy is weak with respect to integration. The JPS, 05-95 and 00-55 do not contain explicit policy, requirements or guidance. All of them do recognise that there are issues of concern in software reuse and integration testing. JAC paper 1231 is the notable exception in that integration is treated seriously and explicitly in terms of procedures to be followed and documents to be produced. Technical integration issues and guidance thereof, are not included in any of these documents. We will argue that this situation needs to be remedied.

3 US DOD POLICY

In this section we review and discuss the policy of the US Department of Defense (DoD) to integration. After an overview, we concentrate on two key points articulating DoD policy.

It is clearly the case that the DoD, motivated largely by software development costs (a study of nine software developments revealed that 47.3% of the software was delivered but not used successfully, 3% was used after changes, 19.2% was used but extensively reworked or abandoned, 28.8% was paid for but not delivered; only 1.8% of the software was used as delivered [12]), is aggressively pursuing alternative software acquisition processes. Such alternatives often include COTS:

- Hekman [12] notes that in May 1993, the US Navy established a policy that "a selection of a government solution and not a COTS solution constituted disapproving the COTS solution".
- Hanratty [11] states that the DoD has adopted an open systems approach in the acquisition of weapons systems electronics. The governing policy is to increase and institutionalise the use of COTS.
- The Federal Acquisition Streamlining Act recently passed, known as Public Law 103-355, includes amongst its provisions:
  - The use of COTS is required.
  - Requirements must be defined in a way that COTS can be procured.
  - COTS providers can compete in any acquisition.

Thus there is, within DoD, a clear policy move to COTS. COTS require open systems, and open systems in turn require integration standards. DoD is formulating standards with this trend in mind and is, in parallel, pursuing practical implementations of these policies. In what follows we examine both.

3.1 MIL-STD-498

MIL-STD-498 on Software Development and Documentation [22] was issued in December 1994, and replaced all existing DoD software standards (in particular DoD-STD-2167A, DoD-STD-7935A and DoD-STD-1703(NS)). The purpose of the standard is to establish uniform requirements for software development and documentation.

The advocated software development process recognises integration both as an activity and as a phase of testing. The standard distinguishes between software integration and software/hardware integration although they are both approached from a testing angle. The results of the integration activities are recorded in appropriate software development files (SDFs).

MIL-STD-498 also makes a serious attempt to deal with software reuse in that it explicitly specifies that the developer shall "identify and evaluate reusable software products for use in fulfilling the requirements of the contract". The standard also gives detailed guidance on the interpretation required so that reusable software products can be incorporated. However, it does seem to lag behind *de facto* DoD policy on open systems and standards which are essential in order to realise the benefits of software reuse (in the guise perhaps of COTS).

MIL-STD-498 is due to be replaced shortly by ANSI Joint Standard 016 (also known as IEEE 1498). The conclusions in this section will still apply.

3.2 DII COE

The Defense Information Infrastructure (DII) Common Operating Environment (COE) [10] represents a pioneering realisation of DoD policy on software reuse, open systems and integration. The DII COE arose out of the need for a near term replacement of the World-Wide Military Command and Control System (WWMCCS).

During this replacement exercise, a simple observation was made about command and control systems: certain functions (mapping, track management, communication interfaces etc) are so fundamental that they are required for virtually every command and control system. Yet these functions are built over and over again in incompatible ways even when the requirements are the same. If these common functions could be extracted, implemented as a set of extensible low level building blocks, and made readily available to systems designers, development schedules could be accelerated and substantial savings could be achieved through software reuse. Moreover, interoperability would be significantly improved because common software is used across systems for common functions.

The DII COE concept is best described as an *architecture* that is fully compliant with the DoD Technical Architecture for Information Management (TAFIM), an *approach* for building interoperable systems, a collection of *reusable software* components, a *software infrastructure* for supporting mission area applications and a set of *guidelines* and *standards*. The guidelines and standards specify how to reuse existing software and how to built new software so that integration is seamless and, to a large extent, automated.

The COE is a "plug and play" open architecture designed round a client/server model. Functionality is added to or removed from the target system in small units called *segments*. In this respect, the COE represents a radical departure from traditional development programmes. It emphasises incremental development and fielding, realising a "build a little – test a little – field a lot" philosophy.

The DII COE is currently used in two systems: the Global Command and Control System (GCCS) which is already fielded at a number of operational sites (and was in fact used during the 1994 Haiti crisis), and the Global Combat Support System (GCSS) which is presently under development. It is anticipated that the COE will also be used in Electronic Commerce/Electronic Data Interchange (EC/EDI), transportation, base support, personnel, health affairs and finance.

Since the COE is an open architecture, it is not tied to a specific hardware platform. It emphasises industry standard software components, including a POSIX-compliant operating system and windowing standards such as NT, X Windows and Motif. Currently the COTS versions listed in table 1 are the approved COE configurations, with Windows NT, Windows 95 and CDE all in the pipeline.

COTS	**HP 700 Series**	**SPARC Series**
Operating System	HP-UX 9.0.1	Solaris 2.3
X Windows	X11R5	X11R5
Motif	1.2.2	1.2.2
NewsPrint	N/A	2.5
Applix	3.2.8	3.2.8
Sybase	10.0.2a	10.0.2a
Oracle	7.1.3	7.1.3
Netscape	1.1	1.1
Mosaic	2.4	2.4

Table 1: DII COE Supported Configurations

The DII COE embodies some interesting ideas in the area of automated integration [21] which originated from the Joint Maritime Command Information System (JMCIS) programme, a predecessor of DII COE. The basic elements used in integration are *applications*, *segments* and *variants*. Automated integration is then logically broken into a number of distinct processes as shown in figure 1.

Automated integration as practised in the DII COE has two distinct advantages:

- **Efficiency**. Integrating by hand a large number of components developed by a variety of developers working from asynchronous incremental schedules can cause integration to rapidly become a choke point.

- **Reproducibility**. Automating a process ensures that it will be followed in a repeatable fashion. In this case, the software tools used by the integration facility are made available to the developers so that they can use the same tools in the same environment as will be used once their product is submitted. If it works for them, it will work when integrated in the system.

System architects and software developers retain considerable freedom in building the system, but runtime environmental conflicts are identified and resolved through automated tools that enforce COE

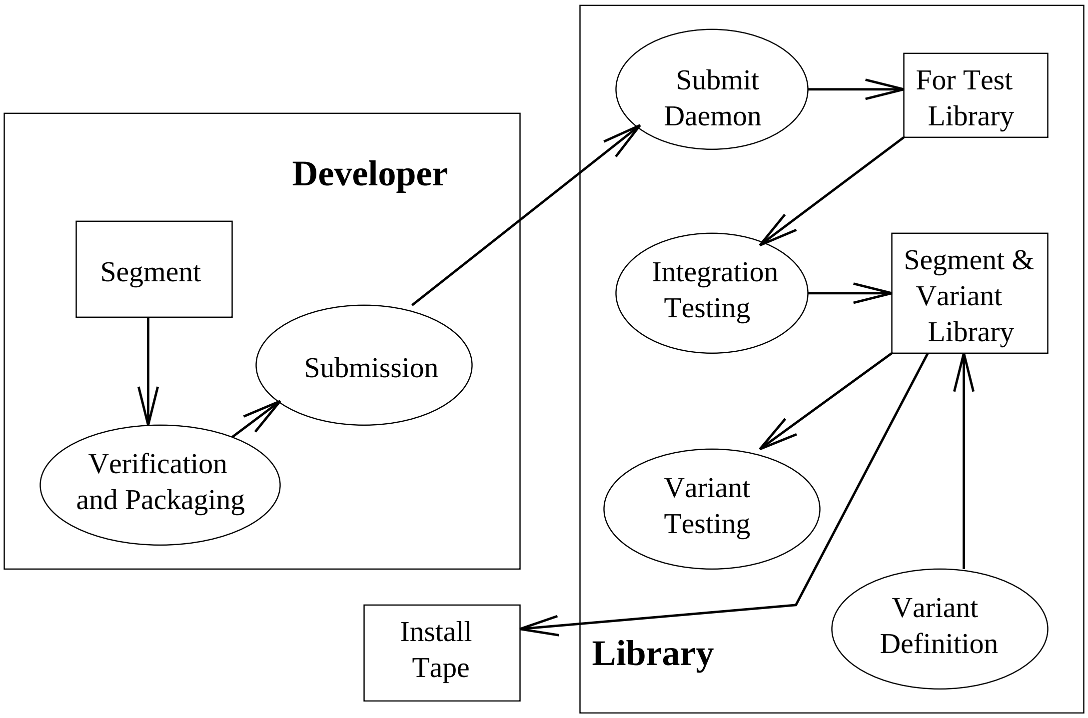

Figure 1: Automated Integration Flow

principles. An important side effect of this is that traditional integration tasks become the responsibility of the developer. Developers are required to integrate and test their software within the COE prior to delivering it to the government. This simplifies integration because it is performed by those who understand the software design best, it reduces costs because integration is performed earlier and at a lower level in the process, and it allows the government to concentrate on validation instead of integration.

Although the DII COE by no means addresses all integration issues, it is a good example of what can be achieved using existing low level standards and COTS assuming an appropriate level of ambition.

4 RTCA/EUROCAE STANDARDS

RTCA Inc (previously Requirements and Technical Concepts for Aviation, Inc) is an association of governmental and industrial organisations in the US. Its stated objective is the procurement of sound technical solutions to problems involving the application of electronics and telecommunications to aeronautical operations. Together with its sister European body, the European Organisation for Civil Aviation Equipment (EUROCAE), RTCA defines the regulatory and technical framework for the majority of the world's civil as well as military airborne systems. Current guidance and policy is articulated in two significant documents; RTCA/DO-178B and ARP 4754. We examine these in turn.

4.1 RTCA/DO-178B

RTCA/DO-178B [24] is concerned with software considerations in airborne systems and equipment certification. The purpose of the document is to provide guidelines for the production of software for airborne systems and equipment that performs its intended function with a level of confidence in safety that complies with airworthiness requirements.

The document explicitly recognises the existence of integration as a phase in the software lifecycle. It provides a rather low level and vague definition of what the integration process is ("the target computer and the source code and object code from the software coding process are used with the linking and loading data in the integration process to develop the integrated airborne system or equipment") and identifies the process objective and activities. In particular, it states that the integration process consists of *software integration* and *hardware/software integration*. This separation is used in the document in order to define the software testing process which is split along similar lines. It goes to some length in giving guidance on the derivation of suitable test cases from the requirements in order to satisfy the objectives of hardware/software integration testing and software integration testing.

The testing point of view is maintained in further sections of the document involving software reuse and COTS. Integration issues are addressed by requiring extra or repeat testing.

In summary, RTCA/DO-178B does recognise the existence of integration as a separate phase of the life-cycle. However, because of the vague definition of what integration actually is and the limited point of view afforded by testing, the document does not provide substantive guidance on how to plan, achieve and document integration.

4.2 ARP 4754

Aerospace Recommended Practice (ARP) 4754 addresses certification considerations for highly integrated or complex aircraft systems [3]. The document is being developed at the FAA's request in order to define the nature and scope of system level information for demonstrating regulatory compliance of appropriate avionics systems.. The need for the ARP arose during the revision of RTCA/DO-178 which produced RTCA/DO-178B. EUROCAE intends to publish the document as a EUROCAE standard following publication by the Society of Automotive Engineers (SAE).

The ARP briefly addresses integration in terms of the proposed generic approach to the development of aircraft systems. It distinguishes between *hardware/software integration* and *system integration*. The distinction between these is not obvious from the text. The former is meant to produce a "physical item under configuration control together with development assurance data". The latter is defined as the task of "ensuring all the aircraft systems operate correctly individually and together as installed on the aircraft". This is regarded as the means of demonstrating satisfaction of "intersystem requirements", a term without technical meaning. The emphasis of the whole treatment is again towards the discovery of deficiencies in the system rather than on guidance in achieving an integrated system in the first place.

5 NATO STANDARDS

NATO policy on software development is articulated in Allied Quality Assurance Publication (AQAP) 150 on Quality Assurance Requirements for Software Development [4] and its companion guidance document AQAP-159 [5]. AQAP-150 is currently being revised as is AQAP-159. AQAP-150 specifies the "project oriented requirements to manage the quality of the software development process". It applies to software development, maintenance and reuse.

The document does not provide any practical guidance on software development and, hence, integration. Instead, it concentrates on the process at a meta-level and the provided guidance suggests considering the waterfall and spiral models amongst others.

Use of COTS is addressed in terms of their impact on the process. The integration characteristics of COTS are not addressed.

A step in the right direction is the requirement that "consideration shall be given to ... integration ... testing" which marks a departure from the first edition of the AQAP. No further guidance is given.

In summary, AQAPs 150 and 159 do not address integration in any meaningful way.

6 ISO/IEC STANDARDS

ISO (the International Organization for Standardization) and IEC (International Electrotechnical Commission) form the main body of worldwide standardisation. We examine a number of standards below.

6.1 ISO/IEC 12207

ISO/IEC 12207 is an information technology standard on software life cycle processes [17] which was published in 1995. Its objective is to establish a common framework for software life cycle processes, with well-defined terminology that can be referenced by the software industry. It contains processes, activities and tools that are relevant to software procurement.

Integration is identified as part of the development process alongside requirements analysis, design, coding, testing and installation (para 5.3). In fact, the standard distinguishes between *software integration* and *system integration*.

Software integration is reasonably well treated here. The standard requires that an *integration plan* detailing test requirements, procedures, data, responsibilities and schedule, be developed and documented. A set of criteria are defined for evaluating the plan and the other integration activities. These include traceability, consistency, coverage, conformance and feasibility.

System integration is similarly treated, although no integration plan is required and traceability and consistency are no longer included in the list of evaluating criteria.

Verification of the goals of integration is briefly addressed as a separate verification task.

In summary, although ISO/IEC 12207 does not give any guidance as to how integration might be achieved or what its implications are for project management, it does at least recognise integration as a separate phase of development worthy of planning and documenting.

6.2 IEC 1508

IEC 1508 addresses the functional safety of safety related systems [16] in seven parts including requirements for electrical/electronic/programmable electronic systems (E/E/PES) and software. *E/E/PES integration* is addressed in Part 2 of the draft standard on E/E/PES requirements and *PE integration* is addressed in Part 3 on software requirements.

E/E/PES integration involves "merging of the software onto the target hardware" as well as "the integration of modules, sensors and actuators to form the E/E/PES". The integration process is defined in the *E/E/PES Design Documentation*. Integration testing is done according to the *E/E/PES Integration Test Specification* and its results are recorded in the *E/E/PES Integration Test Report*. A number of integration techniques and measures and their relevance to various safety integrity levels are identified: functional testing, project management, documentation, black box testing and "proven by use".

Programmable electronics (PE) integration involves combining the software with the programmable electronics "to ensure their compatibility and to meet the requirements of the intended safety integrity level". The central piece of documentation in this activity is the PESITS (Programmable Electronics and Software Integration Test Specification) which is meant to record the testing strategy employed. The *PESITS Test Report* records the results of testing. A similar document structure is used for dealing with software integration. Although a "Software/Hardware Integration Plan" is mentioned, no definition of this is given, nor are there are further references to it. Functional, black box and performance testing are suggested as appropriate methods for achieving PE integration.

IEC 1508 clearly makes an effort to deal with integration. It does so primarily from a testing/safety integrity point of view and not particularly well. This is not surprising given the size of the draft standard which must make maintaining its consistency (and probably its use) unmanageable. For instance, a software/hardware integration plan is mentioned but not defined anywhere; there are cyclical references; software integration is defined as "the division of the software into manageable modules, integrating the modules into a complete system, and testing that all modules interact correctly to perform their intended function and not perform unintended functions" which would appear to equate integration with software design, development and testing. This treatment of integration is not satisfactory.

7 NUCLEAR INDUSTRY STANDARDS

In this section we review the integration policies and procedures of the nuclear power sector as articulated in three key standards.

7.1 IEC 880

IEC 880 on Software for Computers in the Safety Systems of Nuclear Power Stations [14] is a very significant standard. It effectively regulates the highly reliable software required for computers which are used in the systems of nuclear power plants for safety functions, including safety actuation systems, safety system support features and the protection systems.

IEC 880 explicitly recognises integration as a development phase, albeit one restricted to bringing together software and hardware components. It requires the production of a *system integration plan* covering aspects such as configuration control, system integration, error resolution and integration verification testing. Significantly it calls for the early preparation of the system integration plan to allow for any specific integration requirements to be included in the design of the hardware and the software. Guidance is given on configuration control, error resolution and integration testing.

Although IEC 880 does not offer guidance on strategies for achieving integration, it does deal with integration in more comprehensive manner than most other standards. Its emphasis on the need to plan and document integration activities as part of the design of the system is particularly wise.

7.2 IEEE Std 7-4.3.2-1993

IEEE 7-4.3.2 [15] addresses the use of computer systems in safety systems of nuclear power stations. The standard serves to amplify criteria in IEEE Std 603-1991 [6] for computers used as components of safety systems. These criteria taken together establish the minimum functional and design requirements for such computers in US nuclear power plants.

Integration is addressed from the V&V point of view. Integration *requirements* and integration *testing* are identified as steps of the V&V for development activities. Integration requirements V&V is defined as a review of documentation for integration of computer hardware and software. This might include an integration plan as well as test procedures and associated acceptance criteria demonstrating the adequacy of computer hardware and software interfaces.

V&V of integration requirements is followed by V&V of *computer integration* which is defined by the standard as a "general phase intended to apply to all aspects of combining lower-level components to create a higher-level functional unit". Although no guidance is given on how to achieve integration, the standard advocates integration testing in particular with relation to abnormal conditions and events.

7.3 CE-1001-STD

CE-1001-STD Rev. 1 [2] articulates the Canadian position. It has been jointly prepared by Ontario Hydro and AECL CANDU to specify the requirements for the engineering of safety critical software used in real-time protection systems in nuclear generating stations.

Integration testing is identified as a phase of verification. The standard defines the inputs and the outputs of the process (integration test procedures and report) and gives guidance on determining suitable test cases. Integration is, therefore, treated from the V&V viewpoint.

8 DISCUSSION AND THE WAY FORWARD

We summarise the treatment of integration in the examined standards in table 2. From this, a number of observations can be made:

- The majority of standards address integration testing and adopt a V&V perspective of integration. Some require specific integration testing documentation.
- Many standards distinguish between the phases of integration.
- Many standards consider software reuse and COTS although virtually none does so from the integration point of view.
- Very few standards consider open systems or other integration-friendly architectures.
- Virtually no standards consider the achievement of integration.
- No standards address integration project risks.

Despite this rather bleak picture, efforts such as the DoD DII COE exemplify what can be achieved with current technology.

STANDARDS		Integration phases	Integration documents	Integration strategies	Integration testing	V&V perspective	Open systems	Low level integration standards	Component interface requirements	COTS S/W reuse
MOD	JPS						o	o		o
	DS 05-95/3				o					o
	Draft DS 00-55				o	o				o
	JAC 1231	•	•		o				•	•
DoD	MS 498	o	•		•	o				•
	DII COE	o		o	o		•	•	•	•
NATO	AQAP-150/9				o					o
ISO/IEC	ISO/IEC 12207	•	•		o					
	IEC 1508	•	•		•	o				
Aviation	RTCA/DO-178B	•			•	o				o
	ARP 4754				o					o
Nuclear power	IEEE 7-4.3.2	o	o		•	o				
	CE-1001	o	o		•	o				
	IEC 880	o	•	o	•					

•	provides guidance on item
o	mentions item
blank	does not mention item

Table 2: Integration in Standards

8.1 Policy

We feel that the state of the art as well as the practice of integration is now sufficiently well developed for organisations to formulate relevant policy. Such policy should **recognise integration as a separate phase of system development** with design, implementation and validation aspects. In addition, it is desirable that organisations clarify their policy on a number of related issues:

- **Open systems** to recognise their significance in facilitating integration, standardisation and cost effectiveness.
- **COTS** to facilitate their use in appropriate circumstances in order to foster competition and cost effectiveness.
- **Project risk management** for computer based systems acquisition in order to mitigate contractor failures to deliver essential systems, in recognition of the fact that procurement risk *always* belongs to the procurer rather than the supplier.

8.2 Standardisation

Given the current state of the art and practice of integration we believe that standardisation is appropriate for the following issues:

- **A family of low level integration standards**. The NIST APP(Application Portability Profile) [20] could be a reasonable starting point. Standardising on such a family of standards (*de facto*) can improve application portability and thus ease the integration of software developed by different organisations and (possibly) based on a variety of hardware platforms. The DII COE is an example of how much can be achieved with current technology.
- **A computer based systems acquisition risk management standard**. It is felt that such risk management programmes are sufficiently well developed to be of immediate utility to organisations. Such a standard should address procurer/developer issues as well as software risk evaluation insofar as this is relevant in assessing contractor bids. The SEI Team Risk Management (TRM) method [13] and their Software Risk Evaluation (SRE) approach [25]are reasonable starting points for a standardisation activity.

8.3 Guidance

While it is felt that standardisation is not appropriate for many integration issues at present, it is possible to provide technical guidance to project managers in order to facilitate bid selection and evaluation. Such guidance would be expected to cover the following aspects as a minimum:

- Definition of integration and its role in the system lifecycle.
- Integration activities and steps.
- Documentation of integration activities.
- Integration architectures.
- Integration mechanisms.
- Example applications of the guidance.

8.4 Future actions

We have identified a need for two future actions.

- It is felt that it would be appropriate to investigate the possibility of identifying an organisation's core software requirements over one or more class of systems. If such a core can be identified, it would be feasible to initiate a practical integration effort across relevant projects along the lines of the DoD DII COE work in order to assess the possible impact of that technology on costs and schedules.
- There is a need to identify current integration practice within the organisation. This need goes further than identifying intent as expressed in policy documents; it relates to the practice of integration at the coal face by managers today. The findings should be used to tailor policy, standardisation and guidance actions.

Acknowledgements

Parts of this paper have previously appeared in an unpublished, internal MOD report entitled *A Study of Integration in Software Intensive Systems*, March 1996. This work would not have been possible without the inspiration and support of Kevin Geary. Robin Bloomfield kindly supplied a copy of IEC 880.

References

[1] *Draft Defence Standard 00-970, Issue 1, Design Requirements for Service Aircraft.* Ministry of Defence, Directorate of Standardisation, Kentigern House, 65 Brown Street, Glasgow G2 8EX, December 1983.

[2] *CE-1001-STD Rev.1, Standard for Software Engineering of Safety Critical Software.* Atomic Energy Authority of Canada Ltd and Ontario Hydro, November 1994.

[3] *Draft ARP 4754, Certification Considerations for Highly-Integrated or Complex Aircraft Systems.* Systems Integration Requirements Task Group, Society of Automotive Engineers, September 1995.

[4] *NATO Allied Quality Assurance Publication 150 on Software Quality Assurance Requirements for Software Development, Draft C.* 2nd edition, August 1995.

[5] *NATO Allied Quality Assurance Publication 159 on Guidance for the Use of AQAP-150.* 1st edition, April 1994.

[6] *ANSI/IEEE Std 603-1991, Standard Criteria for Safety Systems for Nuclear Power Generating Stations.* 1991.

[7] B. Beizer. *Software System Testing and Quality Assurance.* Van Nostrand Reinhold, 1984.

[8] *Defence Standard 05-95/Issue 3, Quality System Requirements for the Design, Development, Supply and Maintenance of Software.* Ministry of Defence, Directorate of Standardisation, Kentigern House, 65 Brown Street, Glasgow G2 8EX, June 1995.

[9] *Draft Defence Standard 00-55 on the Procurement of Safety Critical Software in Defence Equipment.* Ministry of Defence, Directorate of Standardisation, Kentigern House, 65 Brown Street, Glasgow G2 8EX, August 1995.

[10] *Defense Information Infrastructure (DII) Common Operating Environment (COE) Integration and Runtime Specification (I & RTS), Preliminary, Version 2.0.* Joint Interoperability and Engineering Organisation, Defense Information Systems Agency (DISA), October 1995.

[11] J. Hanratty. Panel presentation on Open Systems, Standards and COTS & Integrating Large Systems. In A.W. Brown, D.J. Carney, and M.D. McFalls, editors, *Proceedings of the SEI/MCC Symposium on the Use of COTS in Systems Integration*, SEI Special Report CMU/SEI-95-SR-007, June 1995.

[12] J.G. Hekman. Panel presentation on Department of Defense Needs in COTS and Systems Integration. In A.W. Brown, D.J. Carney, and M.D. McFalls, editors, *Procs of the SEI/MCC Symposium on the Use of COTS in Systems Integration*, SEI Special Report CMU/SEI-95-SR-007, June 1995.

[13] R.P. Higuera, D.P. Gluch, A.J. Dorofee, R.L. Murphy, J.A. Walker, and R.C. Williams. An Introduction to Team Risk Management (Version 1.0). Technical Report CMU/SEI-94-SR-1, Software Engineering Institute, May 1994.

[14] *Software for Computers in the Safety Systems of Nuclear Power Stations.* Number Publication 880. International Electrotechnical Commission, Bureau Central de la Commission Electrotechnique Internationale, 3 rue de Varembe, Geneva, Switzerland, 1986.

[15] *IEEE Standard for Digital Computers in Safety Systems of Nuclear Power Generating Stations, IEEE Std 7-4.3.2-1993.* IEEE Power Engineering Society, February 1994.

[16] *Draft IEC 1508: Functional Safety: Safety Related Systems.* July 1995.

[17] *ISO/IEC International Standard 12207, Information Technology - Software life cycle processes, Reference No. ISO/IEC 12207: 1995(E).* First edition, 1995.

[18] *Joint Airworthiness Committee (JAC) Paper 1231 - Software, Issue 4.* September 1995.

[19] *Joint MOD/Industry Computing Policy for Military Operational Systems.* CS E2, 3rd Revision. Computing Policy Consultative Committee MOD/Industry Working Group on Computing Policy, DRA Malvern, March 1995.

[20] D.R. Kuhn. On the Effective Use of Software Standards in Systems Integration. In P.A. Ng, C.V. Ramamoorthy, L.C. Seifert, and R.T. Yeh, editors, *Proceedings of 1st International Conference on Systems Integration*, pages 455–461. IEEE Computer Society Press, 1990.

[21] D.S. Lange. Automated Integration for the Joint Maritime Command Information System. In *Procs of 2nd IEEE International Software Engineering Standards Symposium, Montreal*, pages 26–34, August 1995.

[22] *Military Standard MIL-STD-498, Software Development and Documentation, AMSC No. N7009.* Department of Defense, USA, December 1994.

[23] E.G. Nilsson, E.K. Nordhagen, and G. Oftedal. Aspects of Systems Integration. In P.A. Ng, C.V. Ramamoorthy, L.C. Seifert, and R.T. Yeh, editors, *Proceedings of 1st International Conference on Systems Integration*, pages 434–443. IEEE Computer Society Press, 1990.

[24] *RTCA/DO-178B, Software Considerations in Airborne Systems and Equipment Certification.* RTCA, Inc, 1140 Connecticut Ave, NW, Suite 1020, Washington DC, 20036, USA, December 1992.

[25] F.J. Sisti and S. Joseph. Software Risk Evaluation Method Version 1.0. Technical Report CMU/SEI-94-TR-19, Software Engineering Institute, December 1994.