



Developmental Maturity Models to Reduce Systems Integration Risk

*Written by: Amanda Holt – SYPAQ Engineering Discipline Lead
(aholt@sypaq.com.au)*

Abstract

As communications, information, transport and defence systems become more capable, and inherently more complex, the relative risk to the systems development, integration and support programmes also increases. In response to this increased demand for “systems of systems”, systems integration practices have steadily become more formalised and more specialised in recognition of the improved technical, cost and schedule outcomes that can be achieved by the control and incremental validation of system interfaces throughout the developmental programme.

A common difficulty experienced across major Systems Integration Programmes involves the competition for resources, as often those who are best able to plan the Systems Integration Programme, identify risk, institute mitigations and corrective actions are those responsible for the development of the systems themselves.

Being able to provide progressive, consistent, comparable data to enable these key resources to make assessments of the integration risk, and respond to them, results in improved utilisation of engineering resources.

Introduction

Undertaking a complete Systems Integration Programme to control and validate all system interfaces and functional chains is a significant task, requiring dedicated resources with specialist systems integration expertise coupled with a strong understanding of the systems themselves. Often, these are the same resources supporting the design and development effort. Hence by applying more of their effort to the system integration planning and performance, their ability to support other aspects of the project are compromised.

The benefits of adopting a rigorous, clearly defined Integration Programme as part of the Systems Engineering effort enables efficient use of resources from the requirements definition phase through to formal verification and validation activities. Control of design, configuration and integration data within a clearly mapped system architecture is vital to the successful conduct of the integration programme. Early identification of integration resources, risks and methods enables parallel

development and mitigates against the “plug it in a see” approach that too often results as project resources are stretched. Benefits are often realised by development of interface test tools in parallel with design and development effort as these tools not only reduce risk in the development process, but help to ensure that the interface definitions are consistent, achievable and complete. However development of these tools can be time consuming and expensive, so a means of identifying where to concentrate the limited resources is of value in reducing project risk and project cost.

This paper looks at a method for prioritising the integration effort, based upon a technical risk assessment of the system interfaces, functional chains and functional criticality. This method of prioritisation can be used from the project definition phase through to system acceptance, ensuring that changes in priority are readily assessed and corrective action implemented. Equally, this method can be introduced at any stage within a project to assess the developmental maturity and prioritise integration and test effort.

Developmental Maturity Modelling

Methods of progressively assessing the developmental maturity of systems, subsystems, configuration items, interfaces and support services can be found in many systems engineering discipline processes. This method focuses on addressing one of the greatest risks to current complex system development, the systems integration programme.

This method can be applied from the project definition phase, as a means of prioritising options for subsystem selection to ensure a low risk solution is identified, through the systems engineering lifecycle, by maintaining data that tracks an interface or function element’s maturity over the course of the project using consistent identifiers, but also be identifying additional parameters to help inform key milestones along the systems engineering development cycle.

For instance, at the project onset, the means of assessing the interface maturity would be based mainly upon the status of the systems proposed for integration (COTS, Modified, Developmental), the complexity of the system (High, Medium, Low) and the level of design maturity for the system (Existing, Modified, New). These three parameters will be updated and reassessed throughout the lifecycle of the project. These Developmental Maturity parameters can be broken down to the level of definition required to meet the needs of the project.

At the project onset, the application of these categories will allow for a risk ranking against system elements, interfaces and functions. This will assist in identifying the elements and interfaces that require more robust integration planning, monitoring and control.

For each of the options within the three Developmental Maturity parameters, a relative weighting can be assigned to allow for a quick ranking of all elements within the integrated system, refer Figure 1 below for an example. Once these parameters have been identified, they should remain constant for the remainder of the programme to ensure consistency of risk assessment.

Development Type		Design Complexity		Design Maturity		Risk ID
1	COTS Item	1	Low	1	Existing	1
		2	Medium	1	Existing	1
		3	High	1	Existing	1
2	Non-Developmental Item (Configured)	1	Low	1	Existing	1
		2	Medium	1	Existing	1
		3	High	1	Existing	1
3	Non-Developmental Item (Modified)	1	Low	1	Existing	2
				2	Modified - Functional Changes Known	3
				3	Modified - Functional Changes TBD	5
				4	Modified - Significant	8
		2	Medium	1	Existing	4
				2	Modified - Functional Changes Known	5
				3	Modified - Functional Changes TBD	7
				4	Modified - Significant	12
		3	High	1	Existing	8
				2	Modified - Functional Changes Known	9
				3	Modified - Functional Changes TBD	11
				4	Modified - Significant	16
4	Developmental Item	1	Low	5	New - Functions Defined	8
				6	New - Operational Concept Defined	12
				7	New - R&D	16
		2	Medium	5	New - Functions Defined	12

		3	High	6	New - Operational Concept Defined	16
				7	New - R&D	18
				5	New - Functions Defined	16
				6	New - Operational Concept Defined	18
				7	New - R&D	20

Figure 1: Sample Design Maturity Ranking Table

The example at Figure 1 identifies subcategories relevant to the system for each of the three key parameters. The combination of each of the three parameters is then assigned a relative risk ranking between 1 and 20, with 20 identifying those elements of greatest risk, and hence requiring the prioritise effort.

Similarly, the maturity, complexity and type of interface should also be defined, as shown in Figure 2.

Integration Level		Interface Complexity		Interface Maturity		Developmental Risk ID
1	Stand-Alone					0
						0
						0
2	Light (System Status/Connectivity Data)	1	Low	1	Existing	2
				2	Modified	3
				3	New	5
		2	Medium	1	Existing	4
				2	Modified	5
				3	New	6
		3	High	1	Existing	6
				2	Modified	7
				3	New	8
3	Partial (External Data Dependency)	1	Low	1	Existing	3
				2	Modified	4
				3	New	5
		2	Medium	1	Existing	6
				2	Modified	7
				3	New	8
		3	High	1	Existing	9
				2	Modified	11
				3	New	12
4	High (Critical Timing or Complex Interaction)	1	Low	1	Existing	8
				2	Modified	12
				3	New	16
		2	Medium	1	Existing	12
				2	Modified	16

		3	New	18
		1	Existing	16
	3	2	Modified	18
		3	New	20

Figure 2: Sample Interface Maturity Ranking Table

It is then the combination of design and interface maturity that provides a valid assessment of integration risk.

At the System Requirements Review, and System Functional Review, it would be sufficient to utilise only these parameters in assessing the maturity of each developmental item and interface. As the project progresses through to the Design Phase, additional parameters may be added to demonstrate the level of stability of the design, the level of design certification achieved, the number of outstanding engineering changes required and the like, depending on the nature of the project and the risk areas pertinent to the development.

It is possible to define acceptable levels of Design Maturity for both the system elements and the system interfaces that must be met in order to progress through certain systems engineering review gateways. This aligns the scrutiny of the systems engineering reviews with an objective assessment measure.

Whilst the Design or Interface Type rarely changes as the project commences the Design Phase, the Development Complexity may. As a system design develops, previously understood complexity may be refined to a simpler solution, thus reducing the overall risk of the element. Certainly, as the design develops the Maturity will increase until the design is considered complete, thus reducing the overall risk of the element.

Of course, as system problems and engineering change requests are identified against the baseline design following the Design Review, the Complexity may increase as a functional requirement is better understood from prototype testing or Maturity may decrease as significant changes are identified for introduction, thus increasing the overall risk of the element.

These fluctuations in risk assessment can be a key trigger for corrective action and reprioritised efforts by the project team.

It is as the project progresses from the Design into the Development, Test and Evaluation phases the additional metrics may be applied to support the determination of maturity of the system elements. Common metrics often captured within systems engineering programmes that can provide additional refinement of the risk include:

- Number/Proportion of open Problem Reports
- Number/Proportion of open Engineering Change Proposals
- Number/Proportion of Test Failures
- Number/Proportion of Untested Functions/Interfaces
- Number/Proportion of Requirements Verified

Again, depending on the nature of the project and the risk drivers identified by the engineering team, these parameters can be tailored accordingly.

Clearly, following this process through to the acceptance phase, the Maturity parameters should be reduced to a minimum indicating completion and successful implementation of the design, the integration and test parameters should be reduced to a minimum indicating the successful verification and validation of system elements. Finally, capability based parameters may be introduced to monitor the maturity of system support, documentation and acceptance certification requirements to demonstrate the completion of the system ready for delivery.

The 'trick' with any task like this is that what may be considered high risk for one project may be low risk for another, which is why the framework must be developed for each project to adequately cover the appropriate range of risk drivers. Similarly, one cannot always identify integration risk and design risk as equal, so these have been separated out, which makes a linear definition from COTS to R&D somewhat problematic.

Figure 1 and 2 show the base Maturity Scales, where the actual numbers between 1 and 20 can be adjusted based on project needs (not all numbers are included in the example for instance). This method of assessment obviously has inbuilt assumptions that must be defined relative to the project needs. As an example of these assumptions, 10% modification is the accepted limit for considering something to be non-developmental (however how that 10% is determined can be subject to further arguments - such as number of functions affected, proportion of affected code etc, proportion of interface affected, proportion of requirements changes/added etc.).

It should be noted that where the programme in integrating new or upgraded elements into a legacy system, this process remains valid. However depending on the nature of the legacy system elements, some refinement of the parameters to be applied may be required. For instance, the introduction of a COTS (Legacy) definition at Development Type 1, (shifting COTS to Development Type 2) may be valid to identify the low risk associated with a currently in-service, supported system that is well known to the user and project.

However, if the legacy systems are unstable or otherwise problematic, the introduction of a Legacy (Corrective Action) parameter as Development Type 3 may be valid to identify that there may be significant risk integrating with this known, yet unstable or poorly defined legacy system.

Outcomes

This method reduces the risk to the project by performing iterative, systematic integration and validation activities, allowing early detection of problems and providing a means of tracking the effects of any corrective actions in increasing the maturity of the interfaces and functional elements.

Obviously as resources are dedicated to the planning and monitoring of a given integration element, the technical risk associated with this element will reduce. By

periodic re-assessment of the developmental maturity of interfaces, functions and functional chains, the project can be responsive to emerging risks, and ensure that extraneous effort is not applied to an element that is well within the required risk profile at detriment to other higher risk elements.

Conclusions

For projects that are constrained by cost, schedule and resource limitations, that is most systems engineering programmes, introduction of an objective, developmental maturity model assists the project in prioritising systems integration effort. As the level or risk inherent within the system integration programme can contribute significantly to the project delays, technical problems and increased costs, the ability to assess and direct resources to mitigate the highest integration risks can contribute significantly to a project's ultimate success.

As these models are periodically assessed throughout the systems engineering lifecycle, any variations in the risk assessments can be readily identified with little additional effort outside that required as part of the developmental programme. Additionally, the model, and the data required to populate it can be developed, populated and controlled, in accordance with established guidelines, by systems engineers without the need for specialist systems integration or subsystem expertise. The reports from the model are then able to be assessed and acted upon by the relevant systems integration and subsystem specialists, ensuring the greatest results are achieved from their efforts.