Top-down Meets Bottom-up: Experiences in Integrating Existing Components in Transport Systems

Duncan Dowling duncan.dowling@praxis-his.com
David Jackson david.jackson@praxis-his.com
Praxis High Integrity System Limited
www.praxis-his.com

Abstract

Discussions of system engineering methodologies often assume that the engineers designing a system have substantial freedom of choice in determining how the system will be built – requirements are driven down from system goals to subsystem and equipment specifications.

When a project aims to upgrade an existing system, or when interoperability requirements constrain the choice of components, subsystems or interfaces, this degree of freedom is not available to the system engineering team and places constraints on specifications.

Based on our experiences of large transportation system projects in Europe and Asia, we discuss some of the challenges that these constraints pose for system engineering methods and management processes. We discuss some approaches to the problems of requirements management and the definition of system engineering work packages, and highlight some principles that we have found useful in meeting these challenges.

Background – System Development Paradigms

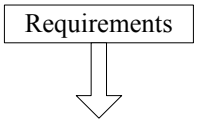
As befits the products of a scientific activity, when describing and comparing the results of a system engineering project, we tend to present our work as a rational progression from the initial objectives – the system requirements – to the result - the system delivered. We relate features of the system back to motivating requirements, and trace satisfaction of those requirements "down" into the implementation.

This mode of development constitutes what we will refer to as "top down" description. It adequately represents the system engineering process on those occasions when the engineering team are given a brief and a relatively clean sheet in order find a good solution to the problem.

More often, however, a system is constrained as much by the solutions available as by the problem that it is intended to solve. Development consists, in part, of taking known components – often commercial-off-the-shelf (COTS) items – and combining them in a way which moves some way towards the requirements without completely satisfying them.

This mode of development is "bottom up", and is often the most pragmatic approach to take when requirements are flexible while implementation constraints (including cost) may not be.

Top-down development – Searching for optimum solutions



This style of development underlies many of the standard system development lifecycle models. The simplest views, including the common "waterfall" and "V" models (Figure 1) are characterised by a sequential set of activities that are carried out once only and the output of one is the input to the next.

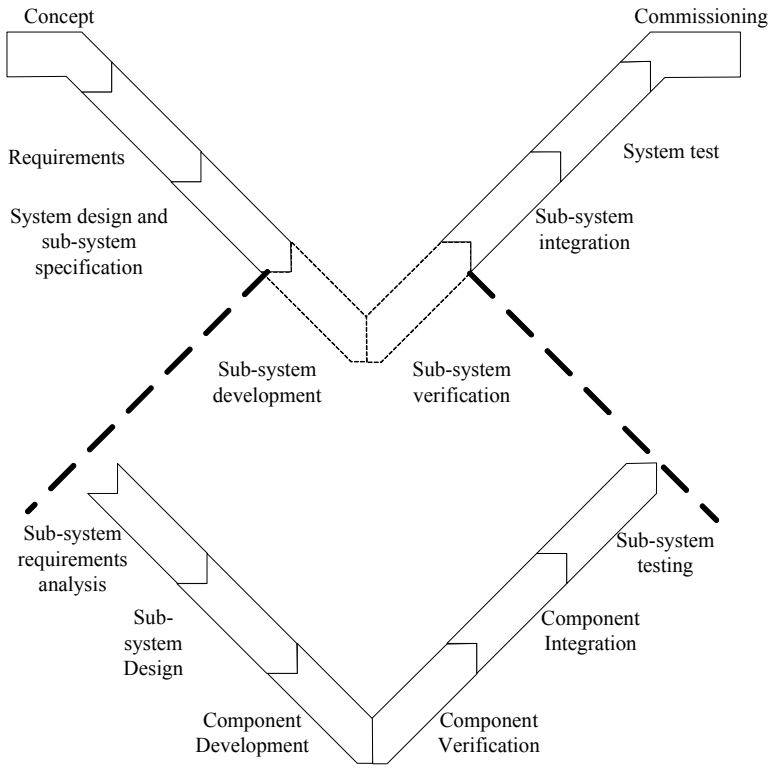
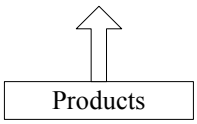


Figure 1 – "V" Model Lifecycle for Systems and Subsystems

The drawbacks of these simple models are widely appreciated; one of the most serious is the difficulty of responding to changes in a system's requirements or environment. This can be managed by explicitly introducing iterative development into the model, as, for example, in the spiral model (Boehm). Each iteration providing the system developers with the opportunity to revise the design based on current circumstances, taking account of changing requirements or the need to address particular design decisions or risks.

Bottom-up development – Making best use of existing technology



Despite the analytical attractions of the top-down models, pragmatic concerns dictate that many systems will be built up from components whose maturity is greater than the system. Few systems are built without precedents, and often only the existence of relatively cheap subsystems or components (whose development costs can be amortised over a number of deployments) makes a particular deployment feasible.

In such a development, it is clear that the final behaviour of the system will represent a compromise between the available components and the user requirements. The inevitability of this compromise, coupled with the existence of elements of a system in a relatively complete form and the attractions of new technology, present a substantial temptation to perform "development by product selection" at the expense of exploration of the integration at the system level. The system engineering process becomes a process of product integration followed, where necessary for regulatory or political purposes, by a retrospective documentation of the correspondence between the integrated system and the user requirements (Figure 2).

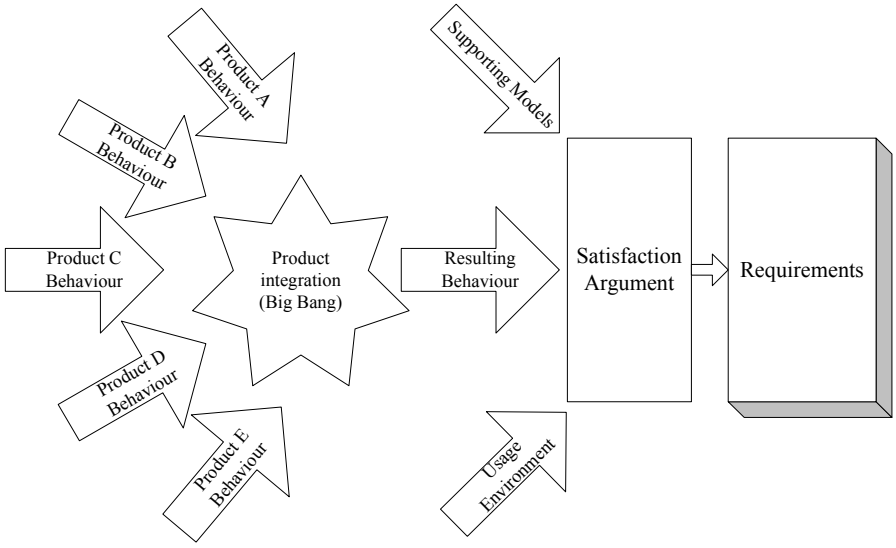


Figure 2 – Bottom-up Product Integration

There is also a risk that focussing on existing products may distort the requirements definition process – a requirements set assembled with a specific set of product capabilities in mind may lack the coherence and completeness desirable for a successful system development.

The major risks of such an approach derive from this late attention to the compatibility of the products with the user requirements, and each other. The project faces risks in:

- Combining products that may not have been developed to operate together, leading to difficulties in maintaining their individual performance when integrated.

- Establishing compliance with user needs, particularly when these concern emergent properties that cannot be directly established by individual components, but arise essentially at the level of an entire system. (Non-functional properties such as reliability and performance are often in this category.)
- Demonstrating compliance with requirements, either to the customer, or possibly to regulatory or certification bodies.

It is the difficulties of balancing cost-effective system-level engineering with the drive to use existing components with minimal changes, cost and delay that this paper seeks to address.

Comparison.

In summary, both development strategies are appropriate to particular development scenarios, and both have advantages and risks. Their characteristics are summarised in Table 1.

Approach	**Application Characteristics**	**Advantages**	**Disadvantages**
Top Down	Requirements driven	More likely to deliver a working solution. Well-understood	Slow, which means that the delivered working solution may no longer be the right system.
Bottom Up	Implementation driven	Appears cheap Products available immediately	Getting extra functionality may be difficult. Integration risk. Certification difficulties. Unpredictable non-functional properties. General COTS issues such as long term support, version control, etc

Table 1 – Both Development Approaches have Advantages and Disadvantages

Case Study One – Working within an existing design

Context

Alstom Transport had already successfully delivered a high-speed train line in South Korea and managed to follow this up by winning another contract to be lead integrator for the E&M (Electrical and Mechanical) equipment, as well as for supplying signalling equipment. The new line was to have 10 stations on a route providing commuter and high-speed services linking Seoul and the domestic and international airports.

The project had been won as a result of approval of a detailed proposal that had subsequently been used to form the basis of the technical contract.

The E&M system consisted of significant subsystems including Signalling (SIG), Rolling Stock (RST), Power Supply, SCADA, Communications and Overhead Catenary System (OCS).

Activities

One of the authors (Dowling) joined the project for 6 months and worked with the systems engineering team, with the objective of producing documentation to mitigate the risk that the integration of the subsystems would not provide the required functionality. These included:

- development of a system architecture;
- identification of the functional requirements at the E&M level
- identification of the subsystems involved in delivering the functions
- definition of how subsystems achieve the functions requiring more than one E&M subsystem

Of most relevance to this paper was the last activity. By working closely with members of the system engineering team and the subsystem suppliers, scenario/activity diagrams were produced to model the interactions between subsystems necessary to provide the required E&M system functionality.

For the purpose of illustration, Figures 3 and 4 give activity diagrams for two functions.

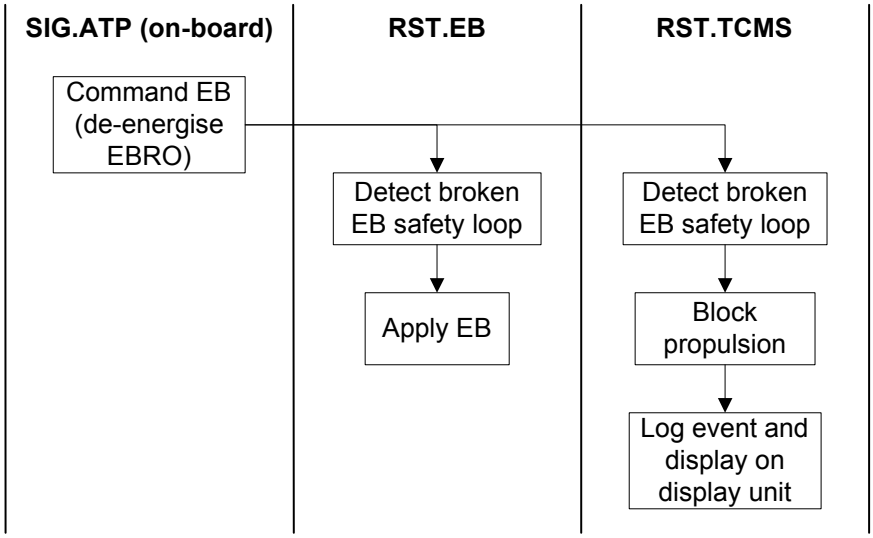


Figure 3 – Activity Diagram for Function: Emergency Brake (EB) triggered by ATP

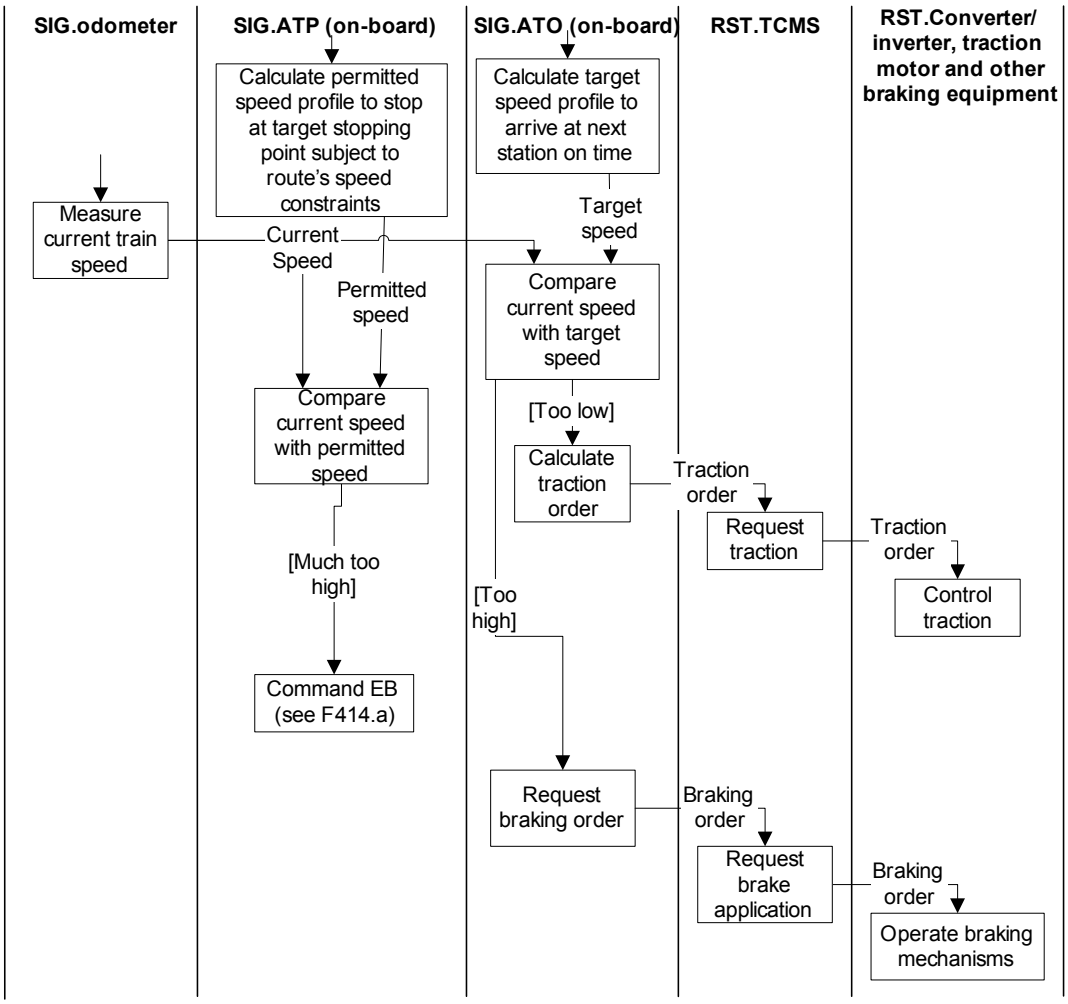


Figure 4 – Activity Diagram for Function: Accelerate and decelerate the train on the main line. (This function has a number of modes. This diagram shows that representing automatic speed supervision.)

As well as achieving their main objective these diagrams also:

- helped communication;
- provided engineers with a better understanding of the system's operation;
- derived functional requirements for subsystems; and
- identified interfaces between subsystems and in some cases within subsystems.

Case Study Two – Engineering Interoperability

The Challenge of Interoperability

One common reason for using existing elements in a system design is the need to maintain interoperability with other systems, whether pre-existing or projected. Interoperability essentially introduces pre-existing interfaces into our candidate designs. These interfaces may or may not be associated with pre-existing components or subsystems that support them, but in either case present a 'bottom-up' constraint on our design.

Such interface constraints arise in many applications, of which the following are only a few examples:

- In railway control infrastructure, where interoperability is mandated to allow many rolling-stock types to use the railway;
- In IT network services, where interoperability is required to minimize infrastructure costs; and
- In communications system design, where interoperability is needed to allow different product families to be combined in ad-hoc networks throughout the life of the product.

In some cases, interoperability constraints may be relatively easy to meet, as the common elements may be flexible enough to allow use by a range of possible system designs. Thus, for example, TCP/IP network protocols and components can be used by many different applications and architectures. Where interoperability is needed at the level of application functions, however, pre-existing interfaces impose greater constraints. Thus compliance with the SNMP management protocol will influence the design of a network management system, and compliance with the European Rail Traffic Management System, ERTMS, places substantial constraints on the development and deployment of a train control and protection system. We will consider the latter example in more detail.

Context

ERTMS (UNIFE) is a technology designed to increase the interoperability of European high-speed rail services by establishing common standards for the control and protection of rolling-stock operating on suitably equipped lines. At the core of

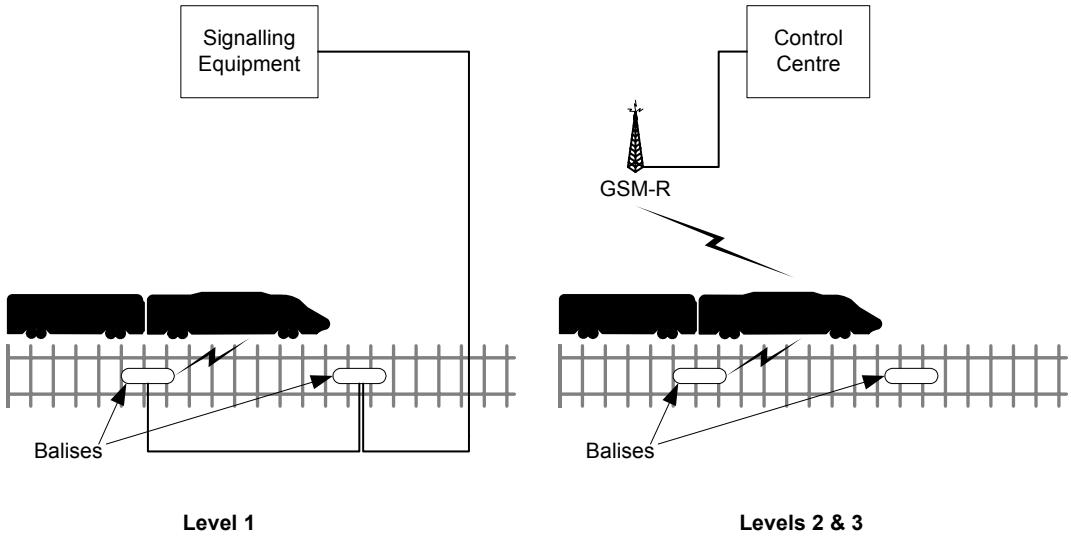


Figure 5 – ERTMS Communication Mechanisms

ERTMS is the concept of cab signalling, where instructions are given to a train driver not by signals placed at the line-side, but by an on-board display that indicates a *movement authority* describing the speed and distance which the train is permitted to travel.

The key functions are monitoring of train location, and communication of movement authorities to each train, based on the current state of the railway and timetable information. The ERTMS standard defines several levels of implementation. Depending on the level, communication of signalling information is either by track-mounted *balises* that pass messages to pick-up loops on passing trains or by a dedicated cellular radio network, GSM-R (Figure 5). Train location can be detected either by conventional signalling equipment, such as track-circuits and axle counters, or by explicit communication with train-borne equipment that maintains a record of its current location by combining odometry and information received from the balises.

Detailed operating procedures governing the way in which this information is used may vary according to local requirements, and elements of the system may be procured from different manufacturers, but the interfaces between train and infrastructure must be standardised to maintain interoperability. ERTMS thus defines several key interfaces, including the radio link between train and control centre and the messages passed between balises and trains. Compliance with these interface specifications is clearly not sufficient to guarantee compatibility between railway systems (as many physical and procedural elements are also involved), but such compliance is clearly a necessary condition for interoperability.

Development Approaches

Development of an ERTMS compliant railway control system is subject to a number of major constraints:

- The architecture and interfaces must be compliant with the ERTMS standards;
- The operating procedures required must be compliant, to an extent to be agreed, with the expectations of the particular railway operators and authorities; and
- The resulting system must meet the relevant safety requirements and gain any necessary regulatory approvals.

The stringency of these constraints will vary according to circumstances – a wider variation from previous practice, for example, is likely to be allowed in the case of a newly built line than is likely to be acceptable where the new scheme must work alongside existing infrastructure.

Development of such a system involves a combination of techniques (Figure 6). At some stages, especially in the early phases, user requirements and constraints (including interoperability and safety requirements) must be formalised and apportioned to elements in a candidate system architecture. In others, properties of available candidate components must be combined to determine if they meet the system-level requirements, and if not, where the inadequacies lie.

Where inconsistencies are identified, development can progress by a combination of activities:

- Definition of a modification or configuration change to a candidate component in order to meet the user requirement
- Re-negotiation of user requirements to bring them within the capabilities of the candidate design

- Replacement of elements of the candidate design by alternative or bespoke components (where permissible).

Typically several iterations of such reviews and re-negotiation will typically be required. Management of risk during this process requires careful prioritisation of issues.

Assuming that the requirements are reasonable, and the implementation choices are realistic, these contrasting activities should allow us to derive a system implementation which can be characterised by three elements: a (revised) set of user requirements; a set of specifications for (modified) components, and the design and analysis information which links these first two elements together – following the terminology of the REVEAL® technique we refer to this information as a *satisfaction argument* (Hammond et al.). Depending on the process by which the design has been derived, this last element may not be formally documented, although it is clearly good system engineering practice that it should be maintained to support verification and modification.

The form in which component specifications are derived is worthy of some discussion. It is clear that as we are referring to adaptations or deployments of existing components, there is no strict necessity for such specifications to be complete in the sense that would be required for a project-specific development. (Of course, in any project of substantial size there are likely to be some bespoke elements for which

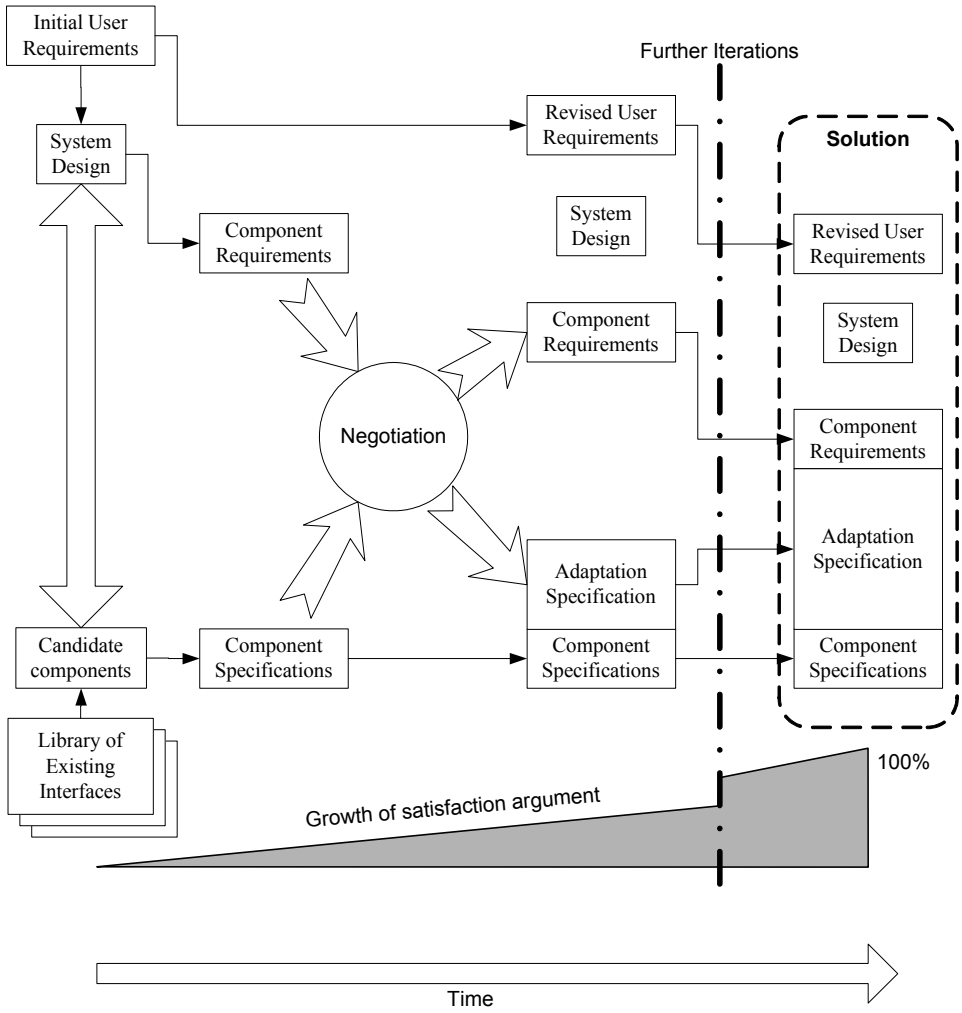


Figure 6 – Development lifecycle using standard interface components

this is not the case, but these can be handled by conventional practices outside the scope of this paper.) Nevertheless, in some circumstances it may be worthwhile developing and maintaining a complete specification for a new variant of a product – this may be the case with safety-related equipment, for example, where it is necessary to document the complete behaviour of a component as part of a certification argument.

An alternative approach that may prove less costly is to maintain specifications for a specific application as a set of changes to the basic product specification. This has the advantage that the scope of the adaptation specification can be restricted to those areas that it has proved necessary to adapt for the application in question. There are potential disadvantages, however, in the need to maintain consistency between a core product specification and a potentially varied set of adaptation specifications. Such a divided specification also complicates the process of defining verification activities, as there is no longer a single source that can be used for verification or test definition.

Benefits and Issues

The approach outlined here has a number of key characteristics:

- Existing components can be used to inform the development of requirements and system designs from an early stage, potentially minimising the adaptations required.
- The derived specifications for adapted components need not be complete, reducing the work required in their construction and maintenance.
- An explicit satisfaction argument between user requirements and specifications for adapted products can be developed as work progresses, facilitating value engineering.
- By deriving the requirements, component specifications, and design description by a parallel iterative process, it is possible to prioritise the resolution of high-impact risks, and thus manage the overall project risk effectively.

There remain a number of issues that need to be resolved with care in the context of any specific project:

- Functional adaptations to existing products may well be difficult and potentially high-risk. If operational constraints are not assessed against likely solutions relatively early in the development, the chosen solution may prove difficult.
- Developing a compromise between capabilities of existing components and user requirements is likely to be an iterative process, therefore careful management of risks and priorities is required to achieve efficient progress.
- Management of specifications structured as modifications to a core product is potentially complex, particularly if a number of modifications are being managed, or if the core product itself is evolving.

It should also be remembered that use of off-the-shelf interfaces and components, while often a necessary condition for interoperability, is not sufficient if high level operating environments and principles are incompatible.

Lessons Learned – General Approaches

Our experiences lead us to believe that there is little benefit in attempting to adopt a dogmatic approach to defining a project development life cycle: we must attempt to find an optimum balance between top-down (requirements-driven) and bottom-up

(implementation driven) working. The balance will be different for each project, depending on specific details of the environment, implementation constraints, and political needs. This section attempts to identify general factors that influence the choice of approach, and to highlight some of elements of the development process that we believe are crucial to the success of projects of this nature.

Factors to be considered in choosing a development process

Perhaps the major factor is the necessity for, and the acceptable degree of, *compromise* between ideal requirements and practical implementation. For any non-trivial system, it will almost certainly be impossible to find a set of existing products that completely meet the needs of all stakeholders. This raises the questions of where and when compromises will be made.

At the simplest level, we must consider the extent to which we choose to adapt the requirements to suit the available products or vice versa (Figure 7). In practice, the resolution of this issue is unlikely to be the same across the whole range of system behaviour – we will seek to identify areas in which compromise is expected to favour strict compliance with the requirements and also areas in which acceptance of existing products will be more important. Agreement and documentation of principles governing these decisions early in the project will help to facilitate resolution of specific issues later.

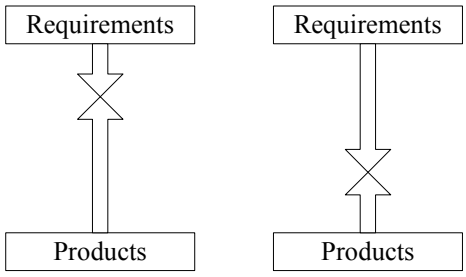


Figure 7 – Compromises between requirements and implementation

The timing of such decisions has an important impact on the costs and timescales of a project irrespective of the technical conclusions reached. If substantial effort is expended on user requirements and system specification before the scope and extent of compromise is considered, much early work may be rendered nugatory when the compromises are actually made.

Whatever development approach is taken, it is of course crucial that a system is *adequate* in that it meets the needs of its users. Compromises may be made, but the agreed requirements and the behaviour provided by the products or subsystems should match, as opposed to the mismatch represented in Figure 8. Of course, this risk is not confined to systems using existing products, but may arise in any system development if the subsystem is incorrectly specified, the subsystem doesn't meet its specification or the requirements have changed thereby invalidating the specification. The use of existing products increases the risk, however, as the degree of control which a single project can exercise over product specifications (either in their original development or in their continued adaptation) is much more limited.

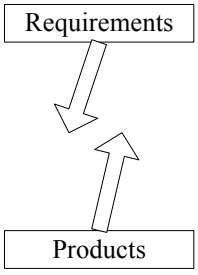


Figure 8 – Inadequate match between requirements and products

Key to reaching a solution is continuous monitoring and evaluation of technical risks through requirements analysis (are we over-specifying what we want?), product evaluation (might we choose an inferior product?), system modelling (will the

systems integrate and co-operate in their environment?). Failure to manage the risks of existing components operating in new environments can be extremely costly (See (ESA), for example.)

Underlying these technical factors are the organisational and political relationships between the parties involved in the development. Key factors will include:

- the relationships between the end customer, the lead integrator and the equipment suppliers;
- the flexibility of the end customer to accept something different from that which they originally specified; and
- the flexibility of the suppliers to customise their products, and the associated cost.

Key Process Elements

Below is an outline process derived from our experience that takes the system development from concept to development. Key is the identification and resolution of the gaps between requirements and the functionality provided by the subsystems.

Identify concept e.g. a train line between A and Z providing services stopping at B, C and D and taking 45 minutes. It should be capable of carrying 10000 people per hour and a minimum waiting time of 10 minutes.

Identify the system's environment / context – identifies and defines the entities in the environment of the system, including interfaces between themselves and the system itself (i.e. external interfaces)

Identify functional requirements – at use case level e.g. station controller broadcasts a public address message to passengers on platform at that station

Produce high-level design. Identifies the main subsystems and the top-level interfaces between them and the system's environment

Identify subsystem requirements and interfaces they need to support. This is reasonably straightforward for requirements satisfied by single subsystems, in other cases, it is necessary to derive requirements for those subsystems. For functional requirements, this can be done by modelling the 'use case' in a scenario. For non-functional requirements different types of analysis are needed, such as reliability modelling, performance modelling, response time analysis, fault tree analysis etc

Identify suitable products. This involves researching available products that might be able to satisfy the requirements (function, reliability, safety, interfaces).

Address gaps. Gaps in this context[1] mean that the integration of the subsystems is not expected to satisfy the requirements. Options for addressing the gap include:

- using a different product;
- determining if the unsatisfied requirements can be satisfied in a different way (i.e. by revising the different scenario) which makes use of the functionality that the subsystem is believed to provide;
- consideration of the impact of the gap on the system, which may lead to

[1] In other contexts it might refer to incompleteness of the requirements set. This is a separate problem and outside the scope of this paper.

redefinition of the requirement;
- modifying the product so it has the required functionality

Non-functional requirements are an important consideration when selecting a product, or a supplier. These include: price, delivery, safety, security, reliability, performance, maintainability, usability, long term support, power supply requirements, size, interoperability etc. They should not be addressed too late in the programme otherwise you might be already over-committed, making a change impossible.

Conclusions

The issues relating to the use of existing products (COTS, MOTS etc) are more extensive than those discussed in this paper and a decision to use them should include consideration of the many other issues already mentioned.

Important issues raised in this paper as a result of our experiences include the need to:

- Ensure a clear distinction between requirements and component specifications.
- Control the alignment of requirements and candidate system capabilities to ensure that all no gaps arise between capabilities and expectations of the eventual solution.
- Balance the cost and value of changes in system requirements against changes in existing components.
- Manage the risk of undesirable emergent properties to minimize the chance that issues will arise late in the project.
- Maintain visibility or control of specifications of (possibly modified) products that may be changing in parallel with the system development.
- Develop relationships with suppliers and customers to ensure that dependencies are met effectively.

Some important principles and techniques to address these issues are:

- Structured information management. It is important to be able to distinguish between types of information and the relative importance of different elements. These factors determine the nature and scope of possible changes.
- Information dissemination. All parties must have clear and unambiguous visibility of key information.
- Requirements traceability and satisfaction arguments. To provide confidence that requirements are achieved, and to aid the impact analysis essential to effective value engineering.
- Change management. To monitor and control the specifications of potential modifications to a product whose baseline may itself be evolving, controlled management and communication of changes is essential.
- Risk management. To ensure that key issues are resolved early, activities such as prototyping and modelling should be used to predict possible conflicts.

References

Boehm, B.W., A spiral model of software development and enhancement, *Computer*, 21(5):61-72, 1988.

ESA (European Space Agency), Ariane 5 flight 501 Inquiry Board Report, Press

Release No. 33-1996, 1996.
Hammond, J.A.R., Rawlings, R.M., and Hall, J.A., Will it work?, *Proceedings of RE'01, 5th IEEE International Symposium on Requirements Engineering*, August 2001. See also http://www.revealmethod.com)
UNIFE (Union of the European Railway Industries), Welcome to the World of ERTMS, http://www.ertms.com.

Acknowledgements

The authors would like to thank Alstom Transport for allowing us to publish the material used in Case Study One. The discussion in Case Study Two is based on experiences in several projects, covering a number of years – the authors would like to acknowledge the inspiration provided by all those who have worked with them on large system integration and development projects over the last five years. We are also grateful to Rosamund Rawlings for her helpful comments on a draft of this paper.

Biography

Duncan Dowling. Duncan is an experienced systems engineer and is a Chartered Engineer with the IMechE and the IEE. Duncan has over 10 years experience on large and small projects in the Rail, Aerospace, Defence, Telecoms and Manufacturing sectors, in the UK and overseas. He is experienced in all stages of the development of high integrity and complex systems, particularly in the areas of systems engineering and safety engineering.

His broad background includes mechanical, electrical, software and production engineering, and he has spent several years of working and travelling overseas, including 6 months recently in South Korea for Alstom Transport.

David Jackson. David has over fifteen years' experience in high-quality software engineering, technical consultancy and project management, with a particular focus on the definition and development of high-integrity systems. His recent activities include:

- Function manager responsible for the subsystem design teams of a major railway control system (ERTMS) development.
- Acting as technical authority for the prototyping and feasibility stages of the development of a protection system for a novel urban transport system.
- Business development in the rail and telecommunications industries.
- Acting as technical team leader on a substantial software development project being carried out to very demanding timescales for a new client in the telecommunications industry.