



Teaching Model-Based Systems Engineering

Dr Ian Brace

System Design Group Pty Ltd

ian.brace@systemdesigngroup.com.au



Dr Ian Brace is a systems engineer with over twenty five years of experience in military communications and software development. He has worked across the capability life cycle, both in early concept design and the deployment and operation of communication systems. Ian has consulted widely across government and industry in capability analysis and systems engineering.

Ian is the founder of System Design Group Pty Ltd, a consulting company which also delivers training in tactical communications and Model-Based Systems Engineering (MBSE). System Design Group has also developed *Capability Architect*; an MBSE tool particularly designed to support early stages of the design cycle.



1. Engineering Complex Systems

Modern engineering projects are always complicated and often complex. Project teams are often distributed around the globe or across the country and even moderately complex projects have many components. A disciplined design methodology is required that can span teams and scale to suit the size of the project. Systems engineering has emerged as the chosen methodology across Defence, aerospace, public transport, space systems, medical devices and numerous other industry domains.

From a practitioner's perspective systems engineering is an established methodology, codified in ISO 15288 (ISO, 2015) and expanded in the INCOSE Handbook (INCOSE, 2015). It provides a proven framework for the design process that emphasises the need to confidently understand the problem space before trying to build the solution.

A key systems engineering concern is to maintain traceability from the problem statement onto the design. This has the dual purpose of demonstrating that all aspects of the end-user needs are incorporated into the design and conversely that features and functions in the solution are required by the users.

The underpinning mathematical rationale for systems engineering can be developed using set theoretic concepts (Wymore, 1993).

2. Applying Systems Engineering

2.1 Traditional Systems Engineering

Since the 1960s systems engineering has been performed by capturing related design artefacts as documents. Together the document suite captures the problem statement and the design solution. The problem statement has been captured in documents with titles like: *User Needs Statement*, *Operational Concept Document*, *Requirements Specification*, *Function and Performance Specification*. The design solution has been captured in documents with titles like: *System Description Document*, *Design Specification*, *Subsystem Design Description Document*.

In principle systems engineering has a strong emphasis on traceability. However, the collection of engineering documents can often become a challenge to maintaining clear traceability from the problem space to the solution space. Particularly during the requirements development phase, when ideas are still evolving, an engineer might make a change in one document but neglect to trace that change through to all the related documents. Even within a single project document it is common for diagrams to become dated when compared to the accompanying text, or the text to lag an updated diagram. Managing the document suite then consumes significant engineering management effort.

2.2 Moving to Models

Software has transformed all engineering disciplines, bringing improvements in speed and flexibility throughout the design process. Software is routinely used to create engineering drawings, layout circuit boards and solve differential equations. Software can also transform systems engineering by structuring the myriad of data generated by an engineering team and providing engineers with fast and simple access to the entire engineering model. Model-Based Systems Engineering (MBSE) supports the engineers by:

- providing a holistic view of the design that is accessible to the whole engineering team,
- making it simple to trace the effects of design decisions,
- automatically maintaining integrity between diagrams and text,
- focussing engineering effort on design decisions, rather than document management.

An MBSE model contains all the key engineering design information from the initial concept, to the stakeholder needs and the derived system design. By having all the information in one place it is possible to query the model in many different ways and produce specific *views* of the design. These customised *views* can address the needs of particular stakeholders.

Figure 1 illustrates this advantage; by capturing the design data in a single structured model we can show the requirements that specify the problem space; the system behaviour that we have designed as a solution; and the physical structure of the proposed system. Each view is important to a different stakeholder group and, because the views are generated directly from the model, each stakeholder can be confident that the view they see is the most up-to-date record of the design.

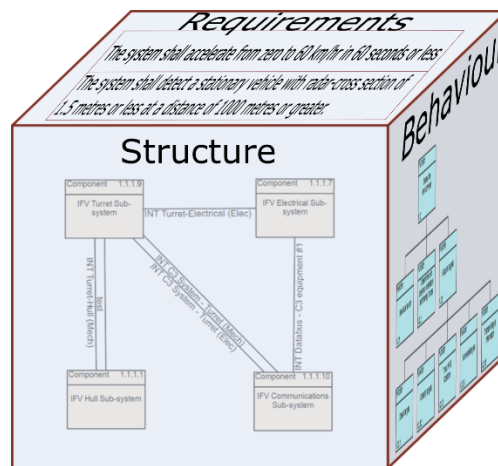


Figure 1 - The MBSE model concept

2.3 It is still Systems Engineering

It is important to realise that using MBSE does not change the fundamental aim of systems engineering. For a project team using MBSE, the project objective is still to deliver a solution with technical integrity within cost and schedule constraints. To determine *what* should be done to achieve that objective, the team should still apply the processes codified in ISO 15288. The key difference on an MBSE team is *how* those processes are implemented: they will be implemented by exploiting the software model, rather than by recording everything in separate documents.

Of course, documents are still important. They are often used for review, particularly by external stakeholders who do not have access to the model. The notion of 'sign off' is still often based against a document that is physically signed. Therefore, MBSE software can usually generate the key engineering documents automatically by exploiting the structured information schema. The documents become just one more *view* of the model data.

3. Teaching MBSE

3.1 Design thinking

Systems engineering is intended to support complex engineering projects. However, the disciplined thinking in systems engineering has been applied across multiple domains and will advantage any design effort. Teaching early engineers to separate the user domain from the system functional domain and the system physical domain will simply enhance their clarity of thought. Enforcing the discipline to determine *what* the system must do before concluding *how* the system will do it is a lesson that cannot be learned too early.

Twenty first century engineers should be taught to use a model-based approach when implementing the systems engineering methodology. Indeed, they almost expect to have software tools available to support their work and recording system requirements in an Excel spreadsheet will not be considered 'MBSE'.

Whilst there are several MBSE tools in use across different industries, they often have a significant training overhead and a high price tag, which make them impractical in a teaching environment. During the learning phase, we want to keep the focus on systems engineering as a methodology, not on tool use. Early engineers need a tool that can be learnt quickly so that they can stay focussed on system thinking.

3.2 Capability Architect

Capability Architect has been developed by System Design Group Pty Ltd to provide an MBSE tool that is accessible to all engineers. *Capability Architect* has been designed to deliver the core MBSE features that engineers expect, but with a low training overhead. Typically, users are competent to use the key tool features after completing a one-hour self-paced tutorial. The product has been developed by engineers, for engineers, to support system thinking and deliver the essential MBSE design functionality.

3.3 Simple User Interface

The *Capability Architect* user interface, shown in Figure 2, displays the three core components of the model.

The Capability Tree allows engineers to navigate across the entire design by selecting the class of interest from the tree. Operational classes include the *Mission* and *Operational Activities*, that describe the use of the system from an end-user's perspective. System classes, such as *Functions* and *System Requirements* capture the required system behaviour and system attributes. Physical classes, such as *Components* and *Links*, describe system solution.

When an element of any class is selected, the Property Pane shows all the attributes of that element, such as the name and description and other customisable attributes.

The Relationships Pane shows how the selected element is related to other elements in other classes across the model. For example, in the Relationships Pane we can see which requirements specify an interface, the external constraint that demanded that interface and the subsystems or components that are connected to that interface.

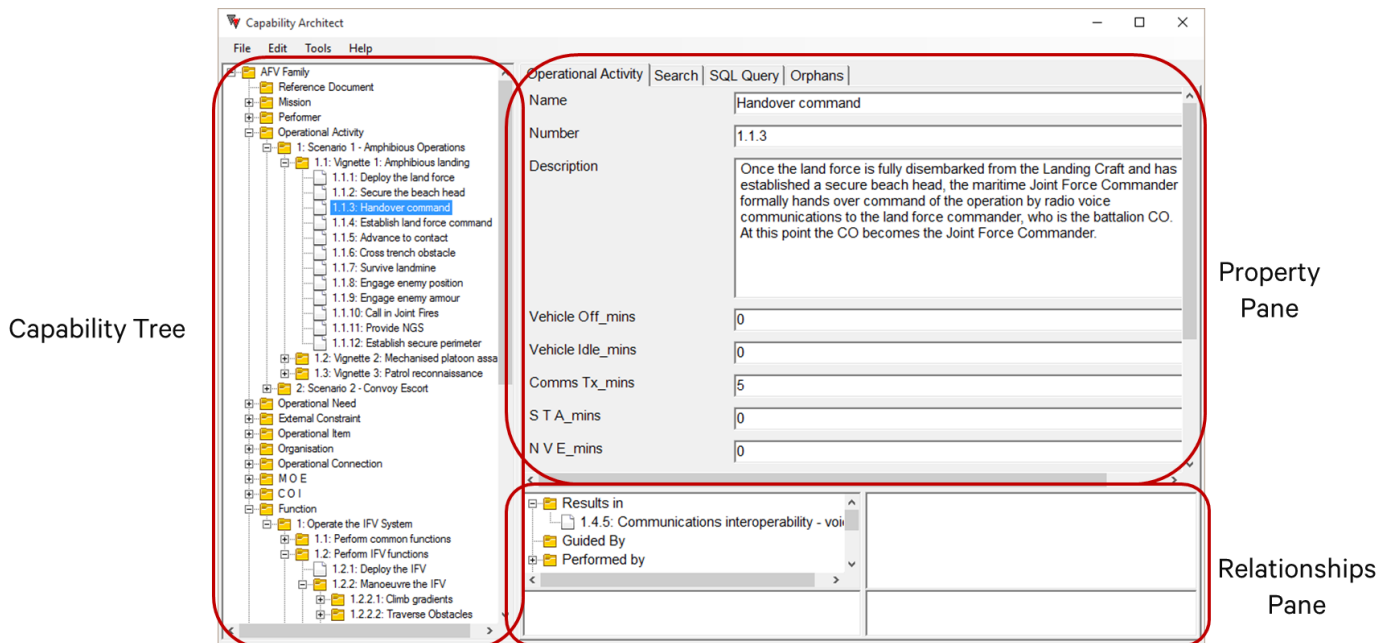


Figure 2 - Capability Architect user interface

3.4 Working visually

A key advantage of using software tools is that software can support a visual development process. *Capability Architect* provides three core diagrams to support design thinking.

3.4.1 Functional Flow Block Diagrams

Functional Flow Block Diagrams (FFBD) are routinely used to develop a visual description of system behaviour. Figure 3 shows an FFBD that captures the end-users' description of how they need to use the system of interest. The FFBD illustrates behaviour and can also illustrate information flows, using data items that are generated by an activity or function. When engineers add a new activity or data item to the diagram, it is also automatically added to the model, ensuring that the relationships captured in the diagrams and the model are always synchronised.

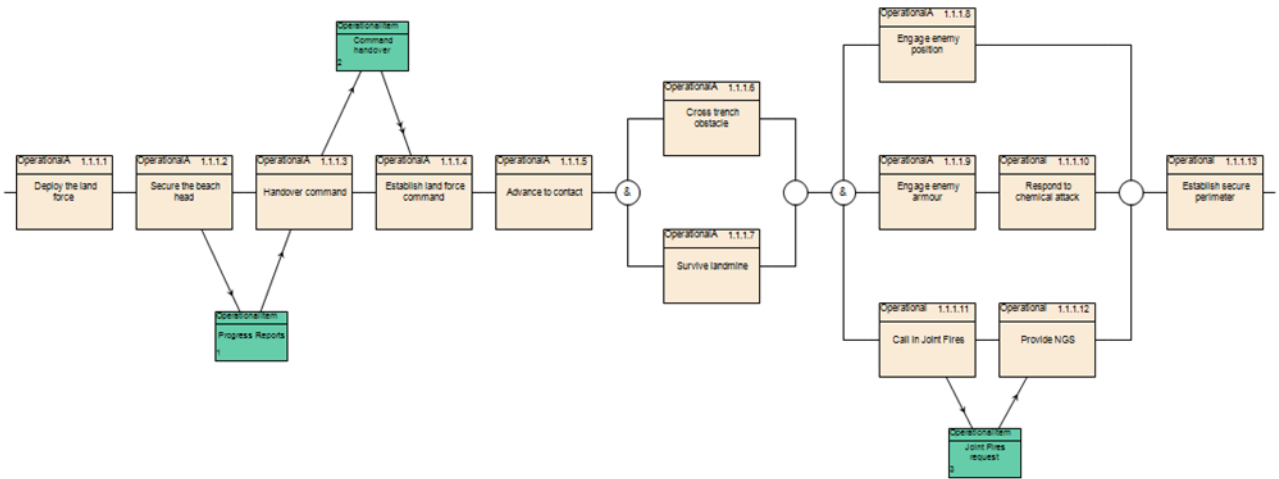


Figure 3 - FFBD of an end-user scenario

3.4.2 Physical Block Diagrams

Physical block diagrams, such as Figure 4, illustrate the key components of the system design and the interfaces between them. Diagrams can be ‘drilled down’ to expose the physical design in greater and greater detail.

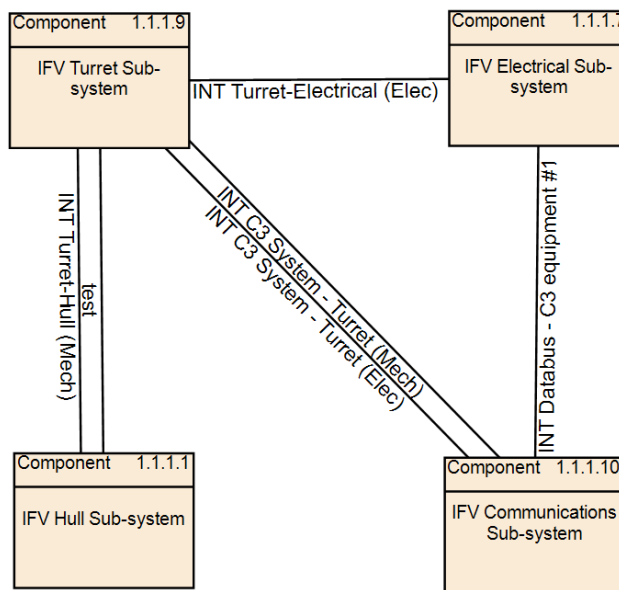


Figure 4 - Physical block diagram

3.4.3 Hierarchy Diagrams

Hierarchy diagrams provide a useful view that can partition *Functions*, *Components*, *Requirements* and other classes into easily understood structures. Hierarchy diagrams can be initiated from any element and the number of levels displayed is configurable.

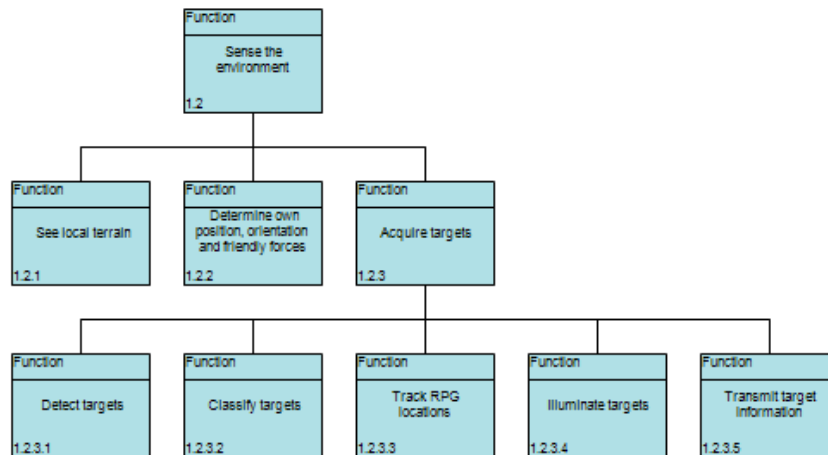


Figure 5 - Hierarchy diagram

3.5 Document generation

MBSE shifts the engineering effort away from document preparation and much more firmly into the design space. However, there is an ongoing need to generate traditional documents and reports, such as specifications, interface documents and system description documents; these documents should be considered as different *views* of the model.

Capability Architect uses a *one-click* document generation system that automatically creates common documents in Microsoft® Word. As shown in Figure 6, the engineering documents are automatically populated with text and diagrams from the model, ensuring that the document reflects the current design state. As engineers develop the design, the key documents can be updated at the click of a button.

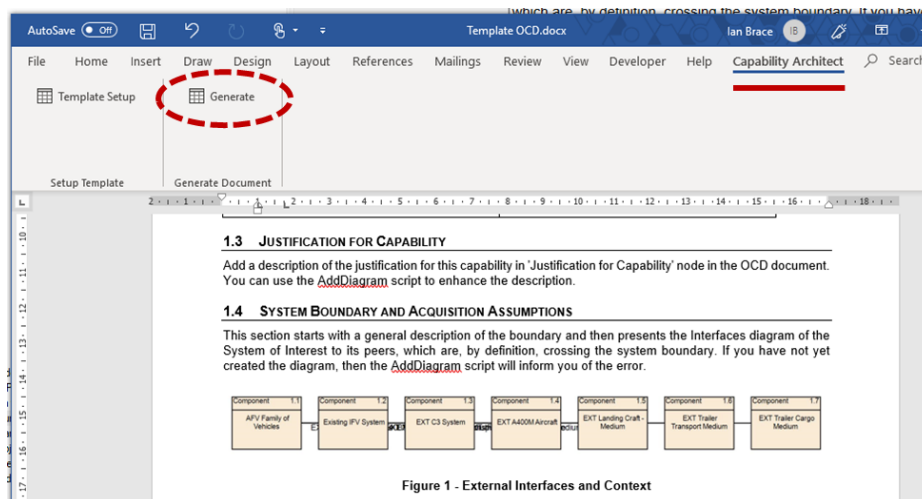


Figure 6 - Example system report

Document generation is fully customisable to support development of new reporting formats. The scripts used to interface with the model and extract the relevant data are written in the ubiquitous Microsoft® Visual Basic for Applications (VBA) which makes it easy to extend the generating functionality. The model can also be accessed using standard Structured Query Language (SQL).

4. Flexible design teams

4.1 Individual projects

For small projects that involve only one or two engineers the *Capability Architect* database can use a simple file-based solution that can reside on a single computer. This provides a great simple and flexible solution.

4.2 Team-based projects

For larger projects that involve a team of engineers, *Capability Architect* can use Microsoft® SQL Server to host the database. SQL Server provides a robust platform that can provide access control measures to manage team permissions and can provide backup facilities to manage business continuity.

When hosted on SQL Server, all engineers can operate in the model simultaneously and *Capability Architect* will propagate each user's changes onto all screens. This ensures that the whole team can see the evolution of the model and collaborate on design development.

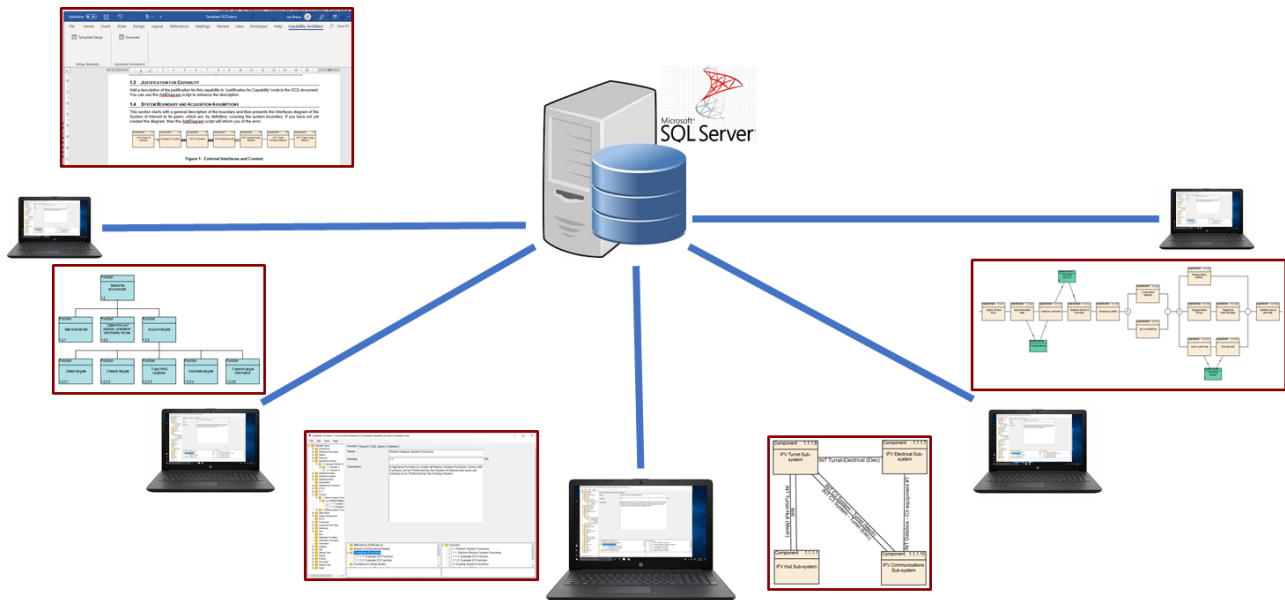


Figure 7 - Team-based Capability Architect

5. Conclusion

MBSE is a natural evolution for systems engineering in the twenty first century. Even moderately complex projects will benefit from a disciplined design methodology that can capture and manage the full range of engineering design artefacts. However, an overly complex MBSE tool can become a distraction to engineers as it shifts the focus from engineering design to tool mastery. The core functions of *Capability Architect* have been designed to be learned quickly in order to minimise that distraction.

Capability Architect provides the key functions necessary to support the systems engineering methodology. Engineers can design in a visual environment that helps them to develop and capture the model of system behaviour. Physical interfaces can be identified, and formal requirements included to specify the interfaces. All this work can be automatically presented in a *one-click* document using the extensible document scripting facilities.

If you would like to know more about *Capability Architect*, or download and test the product, please visit us at

SystemDesignGroup.com.au.

The [Capability Architect Handbook](#) also contains further information, along with a selection of tutorials. You might also like the [Capability Architect video series](#) which demonstrate key features of the product.

References

INCOSE (2015). *Systems Engineering handbook: A Guide for System Life Cycle Processes and Activities, version 3.2.2*, San Diego, CA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC 15288:2015, *Systems and software engineering – System life cycle processes*, International Standards Organisation.

Wymore, A. (1993). *Model-Based Systems Engineering*, Boca Raton, FL: CRC Press.