Systems and Capability Relation Management in Defence Systems-of-System Context

Pin Chen, Ronnie Gori, Angela Pozgay

Defence Science and Technology Organisation
Department of Defence, Canberra ACT 2600 Australia

Pin Chen	Ronnie Gori
Phone: 0061 2 62566181	0061 2 62566265
Fax: 0061 2 62566180	0061 2 62566180
pin.chen@dsto.defence.gov.au	ronnie.gori@dsto.defence.gov.au

Angela Pozgay
Phone: 0061 2 62566275
Fax: 0061 2 62566180
angela.pozgay@dsto.defence.gov.au

Systems and Capability Relation Management in Defence Systems-of-Systems Context

Pin Chen, Ronnie Gori, Angela Pozgay

Defence Science and Technology Organisation
Department of Defence, Canberra ACT 2600 Australia

Abstract

The increasing complexity of Defence System-of-Systems is a challenge for not only capability and systems planning and management, but also engineering disciplines such as System engineering, Software Engineering and Information Systems. The defence requires an improved ability in context-awareness of relations, impact and dependency between systems and capabilities while consider changes or evolutions in various defence capability and system planning, study and management activities. This paper introduces concepts for and an approach to the defence systems and capability management in conjunction with military scenario management and defence architecture data management.

1. Introduction

"It is not the strongest of the species that survive, nor the most intelligent but the one most responsive to change."

Charles Darwin

Engineering activities in future defence systems and capability development vary, but all result in evolutions that occur in a context of Systems-of-Systems (SoS) where a defence organisation must maintain a sustained, sustainable and controlled SoS evolution as a whole. Change in defence is inevitable. Defence has to embrace change and manage it in order to survive and become more effective and responsive. The concept of System-of-Systems (SoS) has been extensively examined [Maier, 1996; Cook, 2001; Arnold, 2001; Sage, 2001; Kaffenberger 2001, Chen, 2003], and is broadly acknowledged as a challenging issue because of its high inherent complexity.

For a SoS to achieve its desired functionality, its component systems will need to interact in a prescribed and predictable manner. Modelling these interactions, their dependencies, and the impact of changes to any system on the SoS is essential to modelling the SoS. Therefore, an important component of managing development and evolution within a SOS is managing the relations between the component systems. The inherent complexity implied by SoS translates to a similar complexity to managing relations within a SoS. The reality, that individual systems are continuously developed and evolved, only adds to this complexity. Currently, Defence lacks the capacity to manage relations systematically within a SoS during the analysis, planning and development of systems and capabilities.

The responses from capabilities and systems to evolutionary change occurring within a SoS context are determined by the ways that their relations are handled both internally and externally. Relation management is also a critical issue to those commonly used engineering disciplines and approaches, such as Systems Engineering (SE) and architectural approaches (e.g. DoD C4ISR AF). However, it has not been fully explored and addressed in terms of requirements, principles, methods or solutions. Within a SoS context, the relation management of systems and capabilities is an architecture issue that does not belong to individual systems or capabilities, but is instead shared by all involved ones. As a result, it cannot be the responsibility of developers of individual systems and should be addressed at the level of SoS or an organisation [Chen, 2003].

This paper introduces an architecture-model based approach to systems and capability relation management. It firstly classifies concepts and classes related to systems and capability management, and then explicitly defines relations between these concepts and classes. The definitions of concepts, classes and relations form a knowledge schema for establishing a body of knowledge for defence capability and systems management. The approach can let the defence develop an architecture management system that can store and manage knowledge and information of capability, systems and architecture products in context.

2. Background, Challenges and Issues

Constantly changing defence requirements, capabilities and systems are challenging the ability of Defence to handle the complexity and relations of systems and capabilities, and questioning its confidence in managing future planning, development and operation. Defence experience in its efforts to achieve many new defence concepts, such as Networked Centric Warfare (NCW) and Network Enabled Capability (NEC), highlights the great difficulties in understanding, communicating, and representing these concepts. There are further substantial difficulties in migrating these concepts into operational and integrated systems and capabilities. Underpinning many of these difficulties are the difficulties in managing the changing relations between systems and capabilities, particularly within a SoS context. These difficulties and challenges face all main Defence business areas from strategic planning, capability study, acquisition and development to operation. Thus, there is an obvious need for Defence to seek better solutions.

2.1 What are relations between systems and capabilities?

A collection of systems, with an overriding mission or purpose, is generally regarded as comprising a SoS. Various relations exist between these systems. Within a Defence SoS context, systems and capabilities are related in the following possible basic manners:

Structure-related

System A and System B are structure-related if System A has one of the following relationships to System B:

- System A is a component of System B; or
- System A is a basis of System B.

Function-related

System A and System B are function-related if to perform its own functions, System A requires certain functions or services delivered by System B.

Information-related

System A and System B are information-related if there are requirements for information flows or information exchanges between two systems. In other words, there is a connectivity and information reach-ability between them.

Operation-related

System A and System B are operation-related if they are both used in an operation scenario to jointly fulfil a mission.

Generation-related

System A and System B are generation-related if System A will be a replacement of System B.

Due to the features and complexity of systems integration and deployment, in addition to these basic manners, there are other types of relations. These include indirect relations, inheritance relations, dynamic relations, naming of relations and tightness of relations. A system may have either the same relation or different relations to a number of systems.

Relation issues exist throughout the whole life cycle of defence capability management as shown in Figure 1. Different phases, stages or contexts of the life cycle face different relation issues, and deal with different relation contexts in different timeframes.

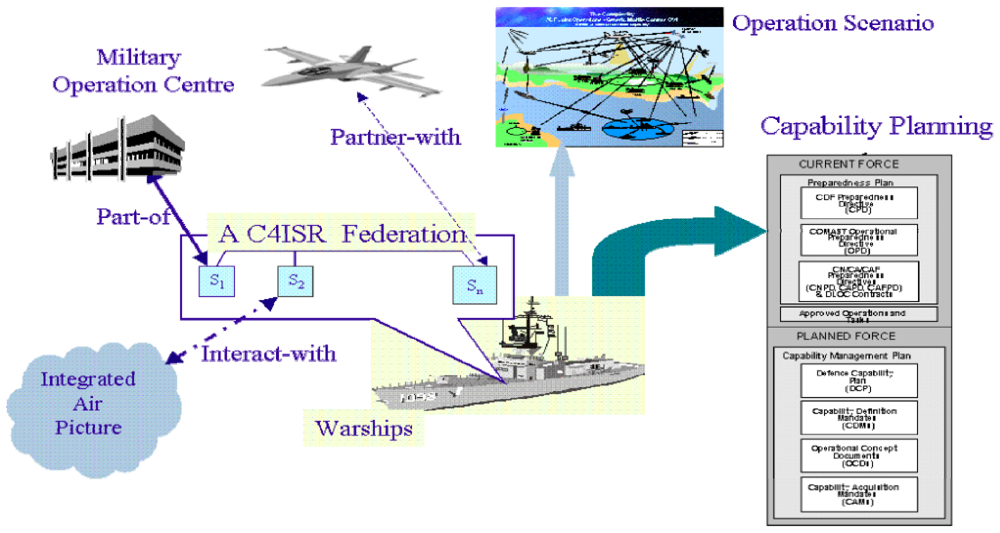
Military Operation Centre
Partner-with
Operation Scenario
Capability Planning
Part-of
A C4ISR Federation
S_1
S_2
S_n
Interact-with
Integrated Air Picture
Warships
CURRENT FORCE
Preparedness Plan
CDF Preparedness Directive (CPD)
COMAST Operational Preparedness Directive (OPD)
CN/CA/CAF Preparedness Directives (CNPD, CAPD, CAFPD) & DLOC Contracts
Approved Operations and Tasks
PLANNED FORCE
Capability Management Plan
Defence Capability Plan (DCP)
Capability Definition Mandates (CDMs)
Operational Concept Documents (OCDs)
Capability Acquisition Mandates (CAMs)

Figure 1. A variety of relations between systems and capabilities

As illustrated in Figure 2, relations between systems and capabilities change when systems or capabilities evolve. A sustained, sustainable, controlled and managed evolution of defence SoS will be difficult to achieve if capability and system relations become unmanageable. Better relation management leads to better responses to changes. To cope with the challenge of successfully managing continually evolving capabilities and systems, Defence needs to develop its ability to manage context, as well as managing relations between systems and capabilities, in terms of their structures, functions, effects and information, but also the impacts generated from their deployment, use, configuration and development plans.

2.2 Challenges and issues

The relations between systems and capabilities are complicated since they are characterised by the following features:

- Managing system and capability relations is an issue concerning the whole life cycle of SoS, including planning, development, management and operation of individual systems and capabilities to the whole SoS;
- The relations can be either conceptualised and represented explicitly or implicitly understood in people' perceptions;
- The relations can be either static and fixed or dynamic;
- The relations are kinds of binding with different degrees of looseness or tightness;
- Architecturally speaking, the relations are determined by the interfaces between systems or the manner by which they are integrated.

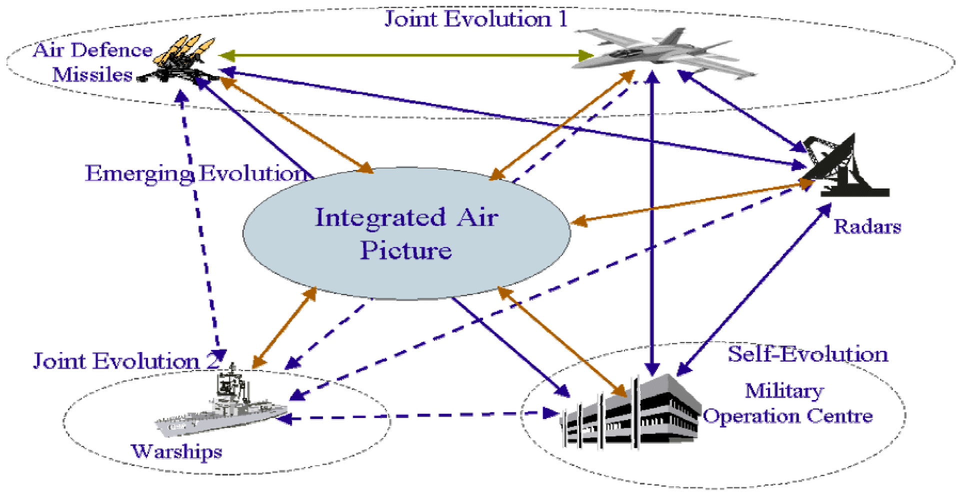

Figure 2. Scenarios of systems and capability evolutions

There are additional questions such as:

- Who should be responsible for definitions and management of the relations; and
- Where and how the relations should be defined and managed.

There are a number of challenges in capability and systems relation management. The first is the variety of contexts where issues of relation management arise and need to be addressed, Relation management is salient to all business areas from strategic planning, capability study, acquisition, systems development, from preparedness to operations, and from individual capabilities or systems and various SoS contexts to the whole force. Secondly, there is a high level complexity in capability and systems relation management since it is not only a technical issue but also a cultural and management issue. The third challenge relates to knowledge management. Knowledge of capability and systems relations currently exist in people's minds and are described and represented differently in various planning and development documents that are kept separately in different sites. This knowledge needs to be captured and maintained in a form that is accessible to other potential users. Finally, there are engineering methodologies or disciplines, such as Systems Engineering, Information Management System, and Software Engineering, which deal with relations of systems from mainly a single system development perspective but not from a viewpoint of SoS management.

The relations between systems and capabilities are important for the whole life cycle of defence capability management. Fundamentally speaking, these relations are critical features and artefacts of the defence capability architecture. They are complicated and dynamic because of the constant evolution of defence capabilities and systems stemming from on-going technological development, changing defence requirements and the increasing complexity of the defence environment. Without specifications and representations of these features and artefacts, Defence will experience great difficulties in evolving its organisation and its capabilities.

The investigation into the issues related to capability and system relation management reveals two important facts. In the current practice, firstly, missing definitions and specifications of these relations indicate that there are holes in the defence capability and defence information environment architecture representation as a whole. Secondly, Defence cannot at present systematically and effectively deal with architecture features and artefacts of its whole capability because of inability in handling relations between capabilities and systems.

In the current practice, these relations are, at best, fully defined by projects that manage acquisitions of individual systems and capabilities. In other words, they are handled mainly at a project level between individual systems and capabilities. They are not considered as fundamental features of the whole defence capability architecture and are not explicitly mandated and shared by other stakeholders.

3. Relation Management Rationales

This exploration of the requirements, challenges and issues regarding capabilities and systems relation management suggests a need to develop a method, drawn from both engineering and management perspectives, that can help Defence systematically manage relations of capabilities and systems, throughout their life cycles, in a context of SoS. Such a method must first help classify, identify and define concepts and their conceptual relations in the context of the planning, development and management of capabilities and systems. The method then also needs to help capture, represent and manage various these relations .

In order to address the requirements, challenges and issues, the relation management for systems and capabilities is aimed to achieve seven main objectives:

- Concept management;
- Conceptual relation management;
- Systems relation management;

- Interface relation management;
- Provision of linkage and traceability of system (and capability?) knowledge;
- Throughout the life cycle of SoS; and
- Being part of the architecture practice;

The high complexity of systems and capabilities' architectures results from its diverse applications in relation to various concepts, such as scenario, capability, platform and system, and their use throughout the whole Defence Capability Systems Life Cycle from planning, development, acquisition, operation, deployment and maintenance. This makes the concepts and relation management be one of important components and fundamental feature of defence architecture data/knowledge management.

These relations should be fully explored and managed at the enterprise level from the following perspectives:

- Architecture relation
- Acquisition relation
- Operation relation

The relation management for capability and systems is established on a basis of two components: *concept and conceptual relation management*, and *object relation management*.

3.1 Concept and conceptual relation management

In the real world, all concepts (such as scenario, capability, platform, system, architecture, project and document) are context-based. The context of a concept is defined by the definition of its attributes and its relation to other concepts. A relation between two concepts is a kind of conceptual linkage that somehow binds the concepts together. Without context specifications, a concept can mean different things to different people Understanding a concept means an understanding of its all definitions of attributes and relations to others.

Any concept may have many "real world" instantiations. Using object-oriented terminology, an object is an instance of a concept. The distinction between two objects that belong to the same concept can also be made according to their contexts. The context of an object is specified by its relations to other objects, and understanding an object requires a full awareness of its attribute values and relations in context.

A relation between two concepts is defined as a characterised linkage with a specific meaning. A relation between objects is an instance of the relation defined between two concepts, or, using object-oriented terminology, two classes). Knowledge of conceptual relations is sometimes documented but often exists only in individual's minds, and is incomplete and inconsistent depending on their understandings. Given a lack of suitable artefacts, methods or mechanisms, it is hard for people to share their knowledge concerning these relations. There has been no method or mechanism to manage them. Fundamentally speaking, concept management in large organisations like Defence, is an issue of knowledge management.

There is, however, an important issue to address when considering concept management. There are often situations where two real world objects, which are quite different things, are mistakenly said to represent the "same" concept. The situation often arises because of a lack of a suitably clear terminology and a general lack of precision in the use of language. Within Defence, confusion often arises when discussions are related to scenarios, capability, systems or architecture.

The approach to this issue is to introduce subclasses under the concept class. This mirrors the situation in the real world, where, for example, a communications system is a type of system, and Link 11 is an instance of a communications system. In other words, a class representing a concept may have a number of subclasses;. Hence a concept can be viewed as a super class for these classes. The motivation to define and manage concepts as class hierarchies is to better capture the semantics within the form of the concept model, through the exploration and definition of the attributes and relations of the subclasses. For the remainder of the paper, a concept is treated as a class in order to avoid possible confusion.

Concept modelling, therefore, needs to distinguish between different concepts (for instance, a system and a project) and where necessary, to define different classes for the same concept (for example, operational architecture view and system architecture view are different classes of the same concept "architecture view") in a concept hierarchy. In other words, the initial stage of concept management requires the establishment of concept taxonomies large concept hierarchies are organised into class packages. There is an analogy with good software engineering practice. The classes within a package should be cohesive, and the classes between packages should only be loosely coupled.

3.2 Relation Management

In order for the relation management for capability and systems to succeed, the relations between concepts/classes must be systematically studied, explored and explicitly defined. The definitions of relations between concepts or classes can be seen as the establishment of a kind of ontology among the taxonomical structures of class definitions.

The class and relation definitions achieved in such a combination of taxonomical structures and ontology linkage lay a foundation for relation management. This approach makes it feasible to specify, maintain and systematically manage the relations between classes, and that the relations between classes can eventually be used correctly and successfully.

A requirement to achieve good relation management is the development of an environment or system that can serve as a basis for: managing concepts and objects in context; conducting analysis, synthesis and evaluation of values and relations of objects; handling complexity; exploring and studying dependency; maintaining traceability; and visualising concepts, objects and their relations and dependencies.

Hence relation management will be achieved through two main activities, that is, conceptual (or class) relation definitions and object context management.

Class relation definition

The relations defined between the concepts of the conceptual models are explicit descriptions and representations of organisational knowledge. They serve as an ontology that underpins the overall conceptual model. Through knowledge engineering of the concept relations, the conceptual model can become a knowledge schema that can support organisation and management of architecture features and artefacts concerning relations between capabilities and systems. Neither the knowledge nor the resulting architecture features and artefacts are currently properly captured and managed.

Through defining and modelling concepts and their relations, the conceptual model becomes a foundation for a sharable body of defence capability architecture knowledge. The model by itself, however, does not directly handle relations between objects although they are defined in it.

Object context management

From a modelling perspective, an object's potential relations are defined according to the relation definitions of the class that the object belongs to. They may be created either when the object is created or subsequent to its creation. From a user's perspective, it must also be possible to modify an object's relations when there is a need.

Object context management requires an environment that is an object store, developed on the basis of the elicited conceptual model, that can enable the desired relation management features. In such an environment, all stored objects require that their full context descriptions, in terms of relations to other objects, are stored as well.

Through specifying the relations between capabilities and systems, these architecture features and artefacts, which are currently missing in the defence capability and defence information environment management as a whole, will be captured and managed.

4. Approaches to the Relation Management

Based on the investigation into the features, requirements, and rationales of the relation management, this paper introduces a concept, called the Defence Architecture Information Model (DAIM), which provides a model-based solution for Defence to establish the relation management for systems and capabilities and enhance its systems engineering, software engineering and architecture practice.

In order to ensure the success in the relation management, DAIM is designed, as expected, to have a joint power of knowledge presentation and organisation that is usually delivered from three separate methods, that is, taxonomy, ontology and meta-data. The approach to DAIM development has been to use an object-oriented modelling language that meets the requirements not only for concept and conceptual relation management. It may also serve as the basis of an object context management system based on the development of an object store which in turn may be based on an object-oriented database.

A class is defined in DAIM because:
1) It represents a concept (e.g. scenario, project, system, and architecture) which there are real world objects associated with, and
2) It has its unique features in terms of attributes, relations and methods or rules that jointly defined its context in DAIM, and
3) A concept or class and its associated objects needs to be managed by DAIM, and
4) A class is considered to be useful in managing systems relation and architecture data.
There are different ways to define and model real world concepts and classes even using the same approach. In order to ensure that DAIM can play the roles discussed earlier, the main objectives of the development team were:

- Firstly, class hierarchies defined in DAIM should cover all information / knowledge entities related to major platforms, systems, scenarios, capabilities and their requirements, projects and architecture resources. In other words, all these concepts will be represented within DAIM such that objects of the relevant classes can then be created to represent real world objects (or entities).
- Through DAIM, secondly, the relations between classes are defined explicitly such that its objects can be linked to other objects defined in the relevant classes.
- Finally, DAIM was developed in the Unified Modelling Language (UML), to serve as a logical data model for the architecture repository development.

Figure 3 shows the semantics, including the attribute and relation definitions, captured in a class definition. DAIM consists of six main class packages represented in UML.

4.1Systems and SoS context

A system can be broadly defined as an integrated set of elements that accomplish a defined objective. People from different disciplines or business interests have different perspectives of what a "system" is. Due to a diversity of use of the term, it is suggested that a system class in the DAIM is an abstraction of a real world system with the following features:

- A clearly defined system boundary;
- Based on a grand design;
- With a system life cycle from planning to retirement/disposal; and
- Has the general features of a system (e.g. lifecycle, component systems, interfaces).

The issues and interests concerned from a system perspective are:

- Internal design of a specific system;
- Interfaces to other systems;
- Development process.

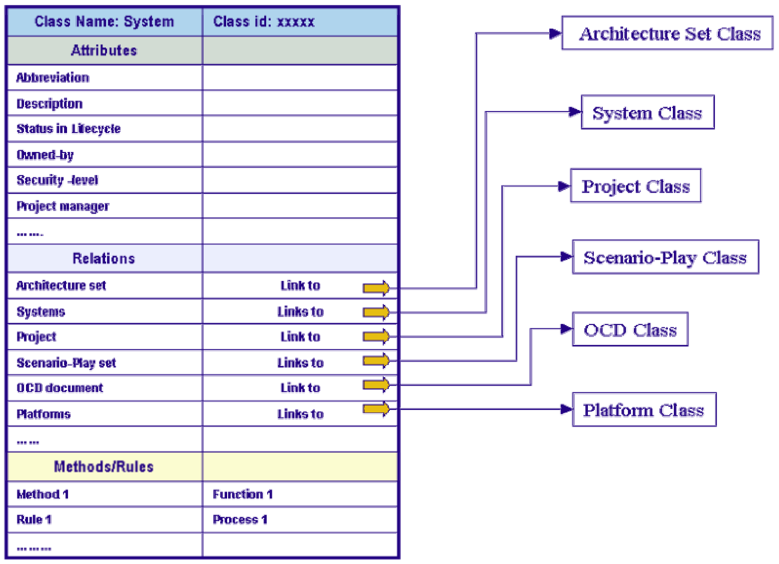

Figure 3. Attributes and relations captured in a system class definition

Because of different features and relations, that is, to model the Defence domain, the System class is taxonomically further defined with a number of subclasses:

- Business system
- Military operation system
- Platform system
- Information-based system
    - Business support system
    - C2 support system
    - Intelligence support system
    - Reconnaissance system
    - Surveillance system
    - Infrastructure system
    - Federation system

There are a variety of relations defined between these system classes. These relation definitions will be used to help specify relations that exist between real world systems when their associated objects are created in the object store. The relations specified between real world system objects will fill a gap in current defence capability architecture descriptions.

The federation system is a site-based federation of all C4ISREW systems, bound to deliver joint functions. Such a site can be either a capability platform, like a warship, or a command centre, where required C4ISR component systems work together. As a communication infrastructure system, each individual C4ISR system , if it is a distributed systems, can be used or integrated into different federation systems on different sites. In other words, a federation system is structured-related to individual C4ISR systems. Or, individual C4ISR systems can be part of a federation system or many different federation systems. Capturing these system relations should enable and support many system and architecture analysis activities.

Also included in the System Package is a SoS Context Class. There are a lot of debates on whether a SoS should be considered as "a system".

In DAIM, we choose to define a SoS context that has the following features. It:
- Has a context of interests;
- Involves a collection of systems, nodes and objects;
- Has no grand design;
- Has no life cycle for the context, but each system has its own life cycle;
- Has a common purpose or function when all involved systems, nodes and objects are working together;
- Has a context evolution resulting from systems evolution;
- May be described in the form of an architecture.

Unlike the systems classes, the issues concerned from a SoS context perspective include:
- Joint effects
- Interoperability levels and analysis
- Information sharing;
- Planning, coordination and management of systems evolution.

There are sub-classes of the SoS Context class. These include:
- C2 information environment;
    - Force domain (e.g. Army, Navy, ..);
    - Operation domain;
    - Region domain;
- Capability information environment;
- Business information environment;

There are strong and important relations between instances (or objects) of *System Class* and *SoS Context Class*. A system instance is:
- defined and studied;
- operated, used and deployed;
- managed and maintained;
- planned, evolved, improved, designed and developed

in the context of one or more objects of SoS context class.

A SoS context object is about:
- a shared understanding or common interests among
- shared requirements for
- a joint effort or function/ capability of;
- a shared environment/situation of;
- an agreement among

a number of systems involved in the context.

A system class object can be associated with a number of objects in different subclasses of the SoS context class.

A SoS context (object) can be considered as a system if:

- all relations among involved systems are defined and stable; and
- all involved systems share a common life cycle; and
- there is an authority of control or ownership; and
- there is a need.

4.2 System classes and relations management

The System and SoS context class hierarchies, together with relation management, should strengthen architecture management for the Defence capability process. It has the potential to capture the supplementary system knowledge regarding SoS architectures required to support Defence's architecture activities and concurrent engineering practice.

The taxonomical structure captured in the System Package can help users to identify system types and their relations to other system objects in the context of capability and system development. The exercise of creating system objects with its context information and managing this information throughout a system's life cycle can significantly improve the awareness of its relations and impact on other components of DAIM.

The DAIM-based repository will have a rich repertoire of systems relations, including *pattern-with, component, based-on, part-of, interoperate-with,* and *used-by.* These relation often have different impacts on evolutions of SoS and require different strategies in development and management. The pattern-with relation, for example, defines a binding or collaborative relation of two systems that is not as strong as the interoperate-with relation, which requires detailed specifications and interface design.

The System Class package is one of the six main class packages included in the DAIM. As a result, the relation management achieved through the DAIM is established across all these class packages. All knowledge relating to the core concepts (scenario, capability, system, architecture and project) can be integrated through relation management and form a body of knowledge on systems and capabilities throughout their life cycles. With the support of such a systematically established body of knowledge, capability planners, analysts and other stakeholders can, as shown in Figure 4, easily and effectively access relevant information and knowledge linking strategic scenarios, operation or exercise scenarios, capability development scenarios, to systems, capabilities or platforms involved. In the current practice, unfortunately, these relations across concepts are not defined, specified and managed. They are also missing components of SoS architecture.

In traditional practices of Systems Engineering and architecture development, systems relations are addressed in the activities of design and development. In the phase of design, systems relations must be fully investigated and defined. The issues related to interfaces and composition then need to be addressed through design and represented in the form of architecture that guides the implementation and evolutions of SoS. The relation management can effectively enhance the applications of Systems Engineering and architecture practice.

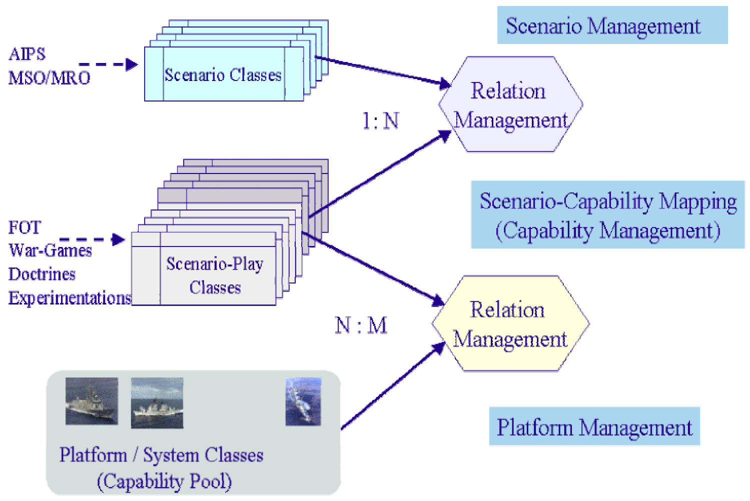

Figure 4. The relation management across concepts

5. Benefits and Applications

The DAIM is a foundation to generate an integrated architecture capability for planning and managing Defence capability and its information environment. Through using the well-developed mechanism for the relation management, DAIM can play a number of important roles in defence capability knowledge engineering and supporting systems and capability management.

With the established knowledge schema that can manage complicated relations among concepts such as operation scenarios, capabilities, systems, and architectures, and their real world objects, DAIM provides the basis for a solution for enterprise-wide architecture knowledge integration and management, that addresses three important issues, that is, defence scenario management, systems and capability relation management, and architecture data management, in an integrated manner.

As discussed earlier, relation management requires an object store as a solution for implementation of the object context management. DAIM can then be used as a data schema for architecture repository development. Based on an object store, it would provide a facility to enable traceability, dependency analysis, synthesis, visualisation, simulation, SoS analysis and experimentation, and capability analysis and development. The repository can provide context descriptions of all DAIM objects and can help generate, understand and synthesize “big pictures“ of various SoS contexts of interest from capability and systems planning and development.

The dependency is a kind of relations that exists between two objects of the same class or different classes. Dependencies can be classified into the following categories:

- Function dependency;
- Structure dependency;
- Information dependency
- Effect dependency;
- Time dependency; and

- Financial dependency.

One of the great challenges in planning, managing and developing SoS is to develop an ability to handle traceability when dealing with “big pictures” of defence capability in various contexts and for different purposes. The establishment of mechanisms for the relation definition and object context management provides a basis and support for highly desirable functions or activities in relation to analysis and visualisation of dependence and traceability. The traceability, as one of main benefits from the relation management, hopefully achieved through the DAIM and its associated object store is characterised as follows:

- Captured in an architecture repository (an object store);
- In a context of the whole defence capability and system architecture space or across concepts, projects and business areas, including:
    - Inter-system
    - Inter-capability
    - Inter-project
    - Inter-architecture
    - Inter-process
    - Inter-business area.

With the ability to effectively handle dependency and traceability in the whole defence information environment (DIE) SoS context through relations defined between concepts (or classes) and specified between objects, DAIM offers an opportunity to develop a knowledge management system and process for Defence capability architecture practice. The benefits to Defence’s many business areas should be substantial, enabling knowledge to be captured, applied at the appropriate time and reused. The consequence should be that Defence should derive substantial additional benefits from its architecture investments.

Through systematic definitions of concepts and relations, in summary, DAIM becomes a conceptual model that describes the entire architecture space of capability and systems It provides:

- A method for concept management (taxonomies and an ontology for scenarios, capabilities, projects, systems and architectures).
- A method also for relation management that covers
    - Scenarios relations;
    - Systems relations;
    - Capability relations;
    - Projects relations;
    - Architecture relations; and
    - Relations across concepts.
- A foundation to enable important architecture activities, such as:
    - Architecture planning;
    - Architecture analysis and evaluation; and
    - Architecture management.
- An environment to deal with issues concerning “big pictures” of defence capability and DIE. It will, in a SoS Context, support :
    - Traceability;
    - Visualisation;
    - Dependency analysis;

- Interoperability analysis.

With the mechanism of concept and relation management in the architecture repository, capability planners, analysts, architects, developers or even war-fighters can query, search and trace various relations and dependencies. As illustrated in Figure 5, all real world objects captured in the repository are searchable with their attributes and relations and can be visualised with its full context in terms relations to other objects in the same class of different classes. If the relation definitions can be further formalised, it should be possible to explore more semantics and rules that can be used to develop automatic reasoning functions related to capability and system relation analysis, synthesis and visualisation.

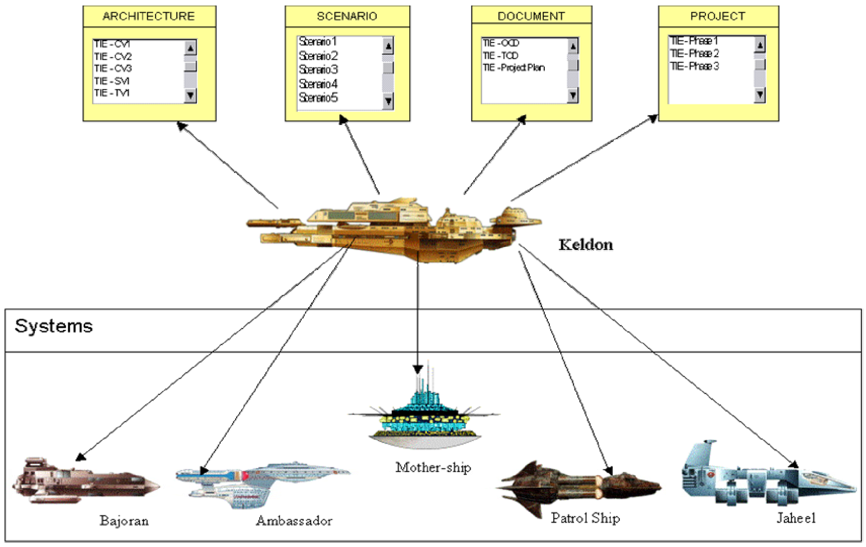

Figure 5. Object context visualisation through DAIM

Another advantage of developing the systems and capability relation management solution in the context of DAIM is that it allows the extension of the relation management coverage to include other related concepts, such as scenarios, projects, architecture and references. Consequently, it also supports the traceability of relations and dependence to be linked with associated scenarios and architecture descriptions. Theses advantages prove the improvement of integrated knowledge management for the defence systems and capability management as a whole.

6. Conclusions

Dynamic changes of defence war-fighting requirements and constant evolutions of technologies require systematic and effective relation management for its systems and capabilities. The approach discussed in this paper presents both concepts and solutions for achieving the relation management that enables context-aware systems and capability planning, development, deployment and management. The systems and capability relation management is introduced as part of the defence architecture data management solution. It adds a new element to the architecture of defence SoS, which is currently missing and difficult to manage. Through systematic and effective capability and systems relation management, a defence organisation can become more confident and capable in the evolution of the organisation and its capability and systems.

Acknowledgement

The authors would like to thank Norman Ferguson and Martin Hamilton of Codarra Advanced Systems, and Michael Isgro and Fung Wong of Swinburne University, for their contributions to both developing DAIM and the DEAL concept demonstrator.

References

S. Arnold and P. Brook, Managing the Wood not the Trees — The Smart Acquisition Approach to Systems of Systems, In Proceedings of the 11th annual Symposium of INCOSE, July 2001.

B. S. Blanchard, System Engineering Management, 2nd ed. John Wiley & Sons, Inc. 1998.

P. G. Carlock and R. E. Fenton, System-of-Systems (SoS) Enterprise Systems Engineering for Information-Intensive Organisations, Sys. Eng 4 (4) 2001, 242-261.

P. Chen. and A. El-Sakka., Context Analysis and Principles Study of Architecture Practice, DSTO Technical Report, DSTO-CT-0151, 2000.

P. Chen and A. Pozgay, Architecture Practice: A Fundamental Discipline for Information Systems, Australasian Conference on Information Systems (ACIS) Dec. 2002. 441-451.

P. Chen and J. Han, Facilitating Systems-of-Systems Evolution with Architecture Support, Proceedings of International Workshop of Principles of Software Evolution (IWPSE), Vienna, Austria, 2001. 130-133.

S. C. Cook On the Acquisition of Systems of Systems, In Proceedings of the 11th annual Symposium of INCOSE, 2001;

H. E. Crisp and P. Chen, Coalition Collaborative Engineering Environment, INCOSE INSIGHT, Vol 5 (3), Oct. 2002. 13-15.

C4ISR Architecture Working Group, Levels of Information Systems Interoperability, US DoD, 1998.

C. Dickerson, Using Architecture Analysis for Mission capability Acquisition, In Proceedings of the International Command and Control Research and Technology Symposium (ICCRT) 2002.

J. O. Grady, System Engineering Planning and Enterprise Identity, CRC Press, 1995.

ISO/IEC 15288: Systems Engineering — System Life Cycle Process, Nov. 2002, Web site at http://www.iso.ch/iso/en/prods-services/ISOstore/store.html

R. Kaffenberger and J. Fischer, Designing Systems of Systems Without Getting Trapped in the Subsystem Maze, In Proceedings of the 11th annual Symposium of INCOSE, July 2001.

J. Leonard, Systems Engineering Fundamentals, Defence Systems Management College Press, December 1999;

H. W. Lawson, A Map of Systems and Systems Engineering, Technical Report of Syntell AB, 2000.

A. H. Levis and L. W. Wagenhals, Developing a Process for C4ISR Architecture Design, Syst Eng 3(4), 2000, pp. 314-321.

M. W. Maier, Architecting Principles for Systems-of-Systems, Proceedings of the 6th Annual Symposium of INCOSE, pp. 567-574, 1996.

A.P. Sage and C. D. Cuppan, On the Systems Engineering and Management of Systems of Systems and Federations of Systems, Information • Knowledge • Systems Management 2 (2001), pp. 1-21.

S. A. Sheard and J. G. Lake, Systems Engineering Standards and Models Compared, Software Productivity Consortium, 1998, Web site at http://www.software.org/pub/externalpapers/

SE Handbook Working Group, Systems Engineering Handbook, Version 2.0, International Council on Systems Engineering (INCOSE), July 2000, Web site at http://www.incose.org.