

Selecting Distributed Object Technologies in the Presence of Uncertainty: an Experience Report on C4I enterprise modelling

Charles Herring, Zoran Milosevic, and Simon Kaplan[†]
Distributed Systems Technology Centre (DSTC)
School of Information Technology, The University of Queensland[†]
{herring, zoran, kaplan}@dstc.edu.au

Abstract

This paper reports on our experience gained in deriving a reference model for Command, Control, Communication, Computers and Intelligence (C4I) architectures. The reference model addresses a dilemma faced by organisations today: choosing the right distributed object platform for domain-specific applications. In making such a choice, organisations face both technological and market uncertainty arising from the availability of multiple middleware platforms and the lack of mature distributed object solutions for specific domains. We present a general methodology to address both of these uncertainties. This methodology provides a basis for middleware-neutral specification of vertical domain architectures based on Reference Model - Open Distributed Processing (RM-ODP). We investigate the problem of mapping such a specification onto the capabilities of commercial middleware products. To address market uncertainty, our methodology incorporates results from the field of strategic planning embraced by many large enterprises - a scenario planning technique. We have successfully applied this methodology for several large corporations in Australia, including Australian Defence Forces (ADF).

1. Introduction

The availability of multiple middleware technologies in the market and the unpredictability of their long-term viability cause uncertainty for many organisations. This market uncertainty is further exacerbated by the lack of mature distributed object frameworks for domain-specific applications (technological uncertainty). We propose an enterprise modeling based methodology to aid in the selection of distributed object technology. This approach was developed specifically to address both technological and market uncertainty. This methodology is based on two premises.

The first premise reflects the implications of the increasingly global nature of business today on the

underlying computing and communications infrastructure. This globalisation calls for a common understanding and agreement on the fundamental concepts and rules pertinent to a particular vertical industry. These concepts and rules should be standardised. Documenting them in the corresponding Reference Model for that particular domain typically does this. Such a Reference Model enables the production of the domain specific architectures encompassing common business objects and frameworks. In order to focus on the core business issues and enable openness in terms of the use of different middleware solutions, these domain architectures should be specified in a technology-neutral way. Such a specification should be sufficiently general to describe the capabilities of commercial technologies (e.g. CORBA, ActiveX/DCOM, Java-RMI, and DCE). In our experience, Reference Model for Open Distributed Processing (RM-ODP) is a good basis for such a specification. When coupled with sufficiently detailed mappings of the RM-ODP notions onto the concepts of commercial middleware, it can provide a good starting point for construction of domain specific architectures.

The second premise recognises the fact that it is often not possible to predict future technology developments. However, businesses must make timely decisions in order to take competitive advantage of available technologies. To this end, it is beneficial to consider several possible technological scenarios and relate the core business of an organisation to each of these. We find valuable some of the approaches used in strategic planning, in particular the "scenario planning" technique [3].

Section 2 presents a general approach for deriving reference models for vertical domain architectures, based on the RM-ODP framework. In section 3 we show how this approach can be applied in the context of Command, Control, Communication, Computing and Intelligence (C4I) systems. Section 4 provides an example of the use of scenario planning technique applied to distributed

object technology for C4I systems. Section 5 identifies some unresolved problems and research needs.

2. Deriving reference models for domain architectures: a general methodology

In this section we derive the notion of a reference model, a reference architecture, specific architectures and implemented systems. We then present a generic methodology for deriving reference models and the corresponding architectures for vertical domains.

2.1 From a Reference Model to a System

A Reference Model (RM) is a conceptual framework for describing a specific domain of interest - the universe of discourse - and is typically adopted by standard bodies to facilitate the modelling of systems for that particular domain. Reference models embody the fundamental concepts and rules pertinent to the domain of interest and are often used as a starting point for producing a family of specialised standards that address more specialised domains of interest.

In general, when developing a RM it is important to identify the set of fundamental concepts needed to describe the particular universe of discourse (or its vocabulary). These foundation concepts need to be succinctly defined and these definitions collectively form a semantic foundation of a RM.

A set of foundation concepts is one part of a RM. This set facilitates the description of systems, within the domain of interest, by prescribing the concepts to be used. In addition, a RM will typically include some guidelines about how to model a complex system for that domain. These guidelines are embodied within the notion of an architecture. An architecture represents a set of rules defining the structure of the system and the interrelationships between its parts. An important aspect of an architecture is support for abstractions. The use of abstractions enables focusing on the important details and suppressing irrelevant details of a system.

By formulating architectural rules in terms of the foundation concepts of the corresponding RM, one ensures that the systems built by applying these rules will conform to the RM; hence the term Reference Architecture. A reference architecture (RA) is generally a partial specification of generic or abstract components and their relationships. A RA is a starting point for deriving more detailed specifications from which real systems can be built. A more detailed specification is referred to as a specific architecture. Both the Reference Architecture and the foundation concepts constitute a Reference Model (Figure 1).

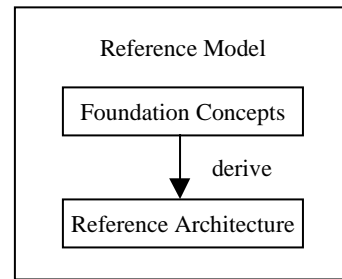


Figure 1 Reference Model Diagram

The process of modelling is an important phase in building large and complex system. It facilitates implementing predictable systems in terms of their behavior and structure. It is believed that modelling according to the concepts and rules of a particular RA ensures building systems that comply with the norms and principles adopted by a particular expert community. This ultimately reduces uncertainty and costs associated with building systems and enables delivering systems that best fit the purpose.

Hence, once a RA is derived, it can be used as a starting point for producing many different architectures; those which further extend the concepts of the RA by focusing on a more specialized domain. These architectures are compliant with the RA owing to the fact that they adopt the guidelines of the RA in question. Finally, once a specific architecture is defined, it can be used as a starting point for implementing concrete systems. Many systems can conform to that specific architecture; for example software systems produced by different vendors. See Figure 2.

Reference models and the corresponding reference architectures can be used in a generic sense - to model any system of interest. In particular, they are useful for those systems that need to comply with a set of norms, prescriptions and guidelines of a particular community. Examples of such normative systems [8] can range from technical systems (e.g. complex machinery, complex civil engineering structures, telecommunication and computer systems) to social systems such as organisations.

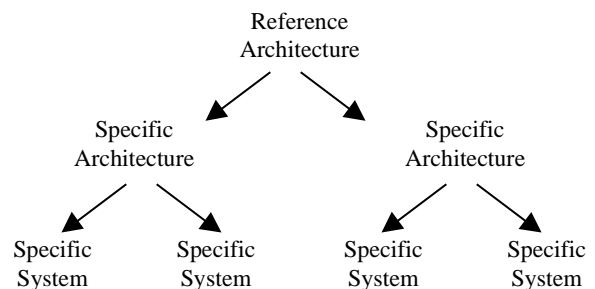


Figure 2 Relationship of Reference Architecture, Specific Architectures and Systems.

It is also worth noting that these generic concepts are applicable to software engineering. In recent years the significance of software architectures has been recognised in many research and industrial initiatives [9].

We note that many complex systems involve the combination of human, organisational and Information Technology (IT) resources. Special care needs to be taken to distinguish between technological and non-technological (i.e. enterprise oriented) aspects of the system. Military systems are a prime example for this. In general, when dealing with vertical industry domains that rely on the use of open distributed systems, we propose the use of the RM-ODP for deriving such models. RM-ODP is described in the next section.

2.2 Reference Model for Open Distributed Processing (RM-ODP)

The RM-ODP is a joint International Organisation for Standardisation (ISO) and the International Telecommunications Union (ITU) standard. It provides a framework for the specification of large scale, heterogeneous distributed systems - this is the domain of RM-ODP. This RM documents key decisions related to open distributed systems, relates components of such systems and sets the agenda for future, more detailed ODP standardisation.[4]

RM-ODP foundation concepts are documented in the RM-ODP Part 2 standard [1]. These concepts are object based and are very general - they can be applied in many different areas. The most basic concepts are object, action and interaction. An object encapsulates its state, which can only be modified by interaction with other objects or by the internal actions of the object. The interaction between objects takes place at interfaces. An object can have one or more interfaces.

A second set of concepts supports the structuring of specifications and introduces the notions of composition and refinement, type and class and of the instantiation of objects or interfaces from templates that describe them [4].

A third set of concepts is introduced to support description of some generic organisational concepts, such as domain, contract and liaison. These concepts can be used to support modelling of relationships between objects.

The RM-ODP Part 2 document also introduces a basic framework for the definition of conformance to the ODP specifications, and for the statements where such conformance applies.

RM-ODP Architecture adopts the notion of viewpoints as an abstraction mechanism. A viewpoint on a system is a form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system. The viewpoints on an ODP system and its environment are:

- *Enterprise viewpoint*: focuses on the purpose, scope and policies for that system.
- *Information viewpoint*: focuses on the semantics of information and information processing.
- *Computational viewpoint*: enables distribution through functional decomposition of the system into objects that interact at interfaces.
- *Engineering viewpoint*: focuses on the mechanisms and functions required to support distributed interaction between objects in the system.
- *Technology viewpoint*: focuses on the choice of technology in that system.

These five viewpoints have been chosen as a necessary and sufficient set to meet the needs of ODP standards [4]. Viewpoints can be applied, at an appropriate level of abstraction, to a complete ODP system, in which case the environment defines the context in which the ODP system operates. Viewpoints can also be applied to individual components of an ODP system, in which case the component's environment will include some abstraction of both the system's environment and other system components.

Viewpoints enable describing a system from a general perspective, including both the technological and non-technological aspects. They enable modelling of a particular system in terms of an architecture of a complete system, where a supporting software architecture is one of its elements. By adopting a viewpoint approach it is possible to provide a clear separation of concerns of different experts involved in modelling of open distributed systems. Each of the viewpoint concepts and structuring rules is described by using the corresponding viewpoint specification language.

Considering that C4I systems need to reflect both organisational and technological concepts, the RM-ODP can be used for deriving a RM for C4I systems (see Section 3).

2.3 A methodology for deriving reference models

The generic principles and concepts presented in the previous two subsections can be used to arrive at a methodology for a RM for a particular vertical industry. We propose a methodology that consists of three phases as follows.

Phase 1: Identify enterprise requirements

The first phase in deriving a RM is understanding the enterprise requirements imposed on the systems for a particular domain. These requirements need to establish the grounds for developing a RM. For example, the enterprise requirements for a telecommunication domain will be quite different than those for C4I systems. We note that the enterprise requirements should include both the organisational and technological requirements. In section 3 we illustrate how a set of enterprise requirements can be

derived for an example of a specific domain, i.e. a C4I domain.

Phase 2: Derive a set of Foundation Concepts

The second phase is the development of foundation concepts for a particular domain (its vocabulary). The experts for that domain should lead this activity. For example, in case of a C4I domain, this should be assigned to defense personnel in charge of Command, Control and Intelligence (C2I) who should work together with the designers of the supporting computer and communications technologies. We argue that many of the RM-ODP foundation concepts can be used as a starting point for deriving foundation concepts for a particular vertical domain. These RM-ODP concepts should be then extended with the domain specific concepts. For example, C4I foundation concepts will include a number of RM-ODP foundation concepts and those notions that are identified in some ongoing C4I initiatives.

Phase 3: Produce a Reference Architecture for the systems of a particular domain

The third phase is the formulation of the RAs for that RM. In general a RA should encompass both the enterprise architecture for that domain, e.g. the specific business objects and frameworks, and the concepts inherent in the existing and future middleware infrastructures.

We think the RM-ODP architecture concepts can be used as a starting point for deriving RAs for many vertical domains because they can address both of these architectural aspects. For example, the use of RM-ODP will enable the

specification of the enterprise architecture of C4I systems as well as a technology independent description of the most fundamental concepts of the underlying information and communication systems. The use of RM-ODP viewpoint specification languages, enriched with the concepts specific to C2I, will ensure clear separation of concerns of different military personnel involved in complex C4I systems. These different abstractions of the same C4I system along with the relationships between the concepts corresponding to different specification languages will enable an improved understanding of the role of humans and the evolving technological capabilities in C4I systems. It will also facilitate building systems in a technology-neutral manner.

3. Application of the methodology to C4I

Defense organisations are undergoing major changes in order to incorporate the capabilities of the emerging technologies to achieve improved jointness at the strategic, tactical and operational levels. These changes incorporate both the organisational and technological aspects and are evident in C4I initiatives around the world. This calls for producing new C4I architectures that will reflect organisational structures of C2I and the related distributed object architectures to support these. This section illustrates how the methodology introduced in the previous section can be used for the specification of C4I architectures. These ideas have been successfully applied to the Australian Defense Forces (ADF) for their C4I systems and the description below is based on our study undertaken for ADF.

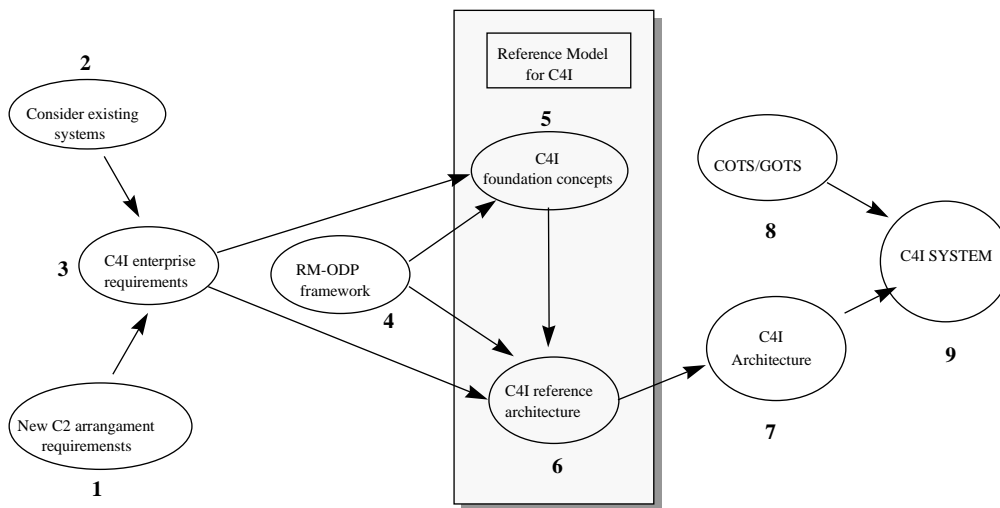


Figure 3 Steps in C4I system development process

3.1 C4I enterprise requirements

At the highest organisational level, modern defense organisations assign a mission critical role to computer and communication technologies for the conduct of their C2I activities - thus the term C4I systems. In addition, it is now required that these technologies be used in an integrated, rather than 'stove-pipe' manner (as was the case in past) to enable joint command and control of different services and units. The adoption of standardised technical solutions (e.g. commercial off the shelf, COTS and government off the shelf, GOTS) across the computer and communication systems of different services is also required to enable more effective joint command, control and operations.

These requirements, along with increasing pressures towards the reduction of life-cycle costs and better manageability of existing C4I systems, demand such a RM for C4I to establish:

- a common terminology of C2I concepts and also the fundamental computer and communication terms used as support for C2I activities (C4I foundation concepts)
- a C4I RA to be used as a starting point for deriving more specific architectures which address particular requirements of the individual services .

Therefore, the new C4I requirements imply a need for a sound reference model for C4I, a C4I-RM (arrows between steps 3 and 5, and 3 and 6 in Figure 3).

In producing a set of enterprise requirements for ADF we have started with the study of several ADF strategic documents that outline the new ADF arrangements (step 1). These documents have given us an insight into the C4I domain and have also elucidated some of important concepts that correspond to the scope of the ODP enterprise viewpoint.

The intention of step 2 was to investigate the applicability of existing ADF and US DOD C4I systems (their legacy systems) and position them with respect to the new C4I initiatives. The analysis of the major US C4I projects was aimed at applying their relevant concepts to the specifics of the new ADF C4I requirements (step 3). These requirements reflect the new ADF arrangements as well as the re-use of existing (legacy) systems currently representing parts of ADF C4I systems. The comprehensive analysis of existing systems has shown that the focus of C4I modelling at present is predominantly on IT architectures. There is little evidence that enterprise level modelling had been done or related to the distributed object technology.

The enterprise requirements identified dictate the definition of a C4I RM, in terms of both the foundation concepts and RA. The full description of the ADF C4I enterprise requirements is beyond the scope of this paper.

3.2 C4I Foundation concepts

In arriving at the C4I foundation concepts we have found a high level of usability of the RM-ODP foundation concepts. These concepts cover both the technological aspects of distributed object systems and most fundamental organisational concepts. In general, the RM-ODP foundation concepts represent a basis for describing a RM of any distributed system or distributed application and can be extended to address the specific domain of interest. When considering C4I systems it appears that RM-ODP foundation concepts will need only minor extensions as the C4I specific concepts can be included in the enterprise viewpoint specification of the C4I reference architecture. Therefore, the C4I foundation concepts accommodate most of the RM-ODP foundation concepts, augmented by some C4I specific notions (step 5).

3.3 C4I Reference Architectures

In this section a C4I reference architecture (C4I-RA) will be derived by using the foundation concepts above and applying the concepts and structuring rules of each of the ODP viewpoint specification languages. The concepts of the RM-ODP and the concepts currently being defined within the ODP Enterprise Language standards [7] will be used, along with the C4I specific concepts such as the definitions of command, control and responsibility. Below, the adopted ODP concepts are *italicized* the first time they are introduced. Figure 4 illustrates the concepts to be discussed in the remainder of this section.

3.3.1 Enterprise viewpoint. An enterprise specification of C4I systems must describe the *role* of C4I as part of a broader ADF organisational structures. In ODP terms, a C4I *enterprise object* can be regarded as part of the ADF *community*. Further, an enterprise specification will need to describe the structure and processes within C4I. This can be specified in terms of *roles* within C4I, *activities* that they perform and policies about *obligations*, *permissions* and *prohibitions*. In producing an enterprise specification it is beneficial to view a C4I system as a combination of C2I subsystem (as this reflects organisational structure of C2I) and the supporting Computer and Communications subsystem. Although both of these systems are visible from the enterprise viewpoint, it is beneficial to separate them. This provides a better understanding of C4I 'enterprise architecture' and the supporting IT and communications system.

A role within a system can be filled by its corresponding enterprise object. It is important to note that enterprise objects can represent human or IT elements. If it is an IT element, an enterprise object can have its information and/or

computation representations. For example, a human or an automated system can play roles such as 'pilot' or a 'decision agent'. In general, the specification of relationships between enterprise and other viewpoint specifications for a particular component of a C4I system needs to establish such correspondence. We also note that an enterprise object can fill multiple roles. For example, the command relationship between military units is well specified. Attached, assigned, direct and general support have precise meanings in terms of the roles and responsibilities between subordinate and superior units.

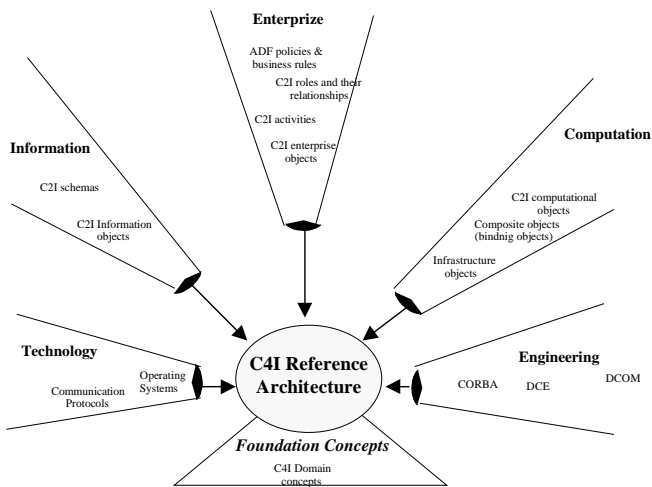


Figure 4 RM-ODP Viewpoints for C4I Domain

The description of organisational aspects of a C4I system should be given in terms of the *policies* that govern:

- interactions between the enterprise objects fulfilling the roles - these policies should cover usual hierarchical military relationships and the relationships between different domains.
- interactions occurring in a federation of different organisational units (i.e. Services, describing the joint activities of Army, Navy and Air Force)
- the assignment of roles to enterprise objects (both human and automata)
- environment contracts of a C4I system (e.g. interactions between this system with industry).

Standardising a set of roles used across different ADF Services can enhance the reusability of C4I components. These will carry the common enterprise semantics of C4I systems. The types for common ADF C4I roles can be stored in an appropriate repository (type repository or meta-object facility). In the case of an enterprise object representing an IT entity, its corresponding information/computational

counterpart needs to be instantiated to implement its behaviour.

In addition to existing concepts from ODP standards, the nature of C4I systems requires the introduction of some C4I-specific concepts. Examples are the concepts of authority, responsibility, command, control and command and control. Most of these concepts are well defined within military domain and where possible these definitions should be included in the C4I enterprise language. Some fundamental concepts are listed as follows (military terms are derived based on the publicly available information):

Authority - the right to determine, adjudicate or otherwise settle issues or dispute; the right to command, control or determine. The notion of authority is closely related to the granting and rescinding of permission.

Responsibility - is the state of delegated authority

Command - the authority that a commander in the military Service lawfully exercises over subordinates by virtue of rank or assignment. Command includes the authority and *responsibility* for effectively using available *resources* and for planning the employment of, organising, directing, coordinating and controlling military forces for the accomplishment of assigned missions.

Commander - a role which can perform command actions.

Control - The authority exercised by a commander over part of the activities of subordinate organisations, or other organisations not normally under his command, which encompasses the responsibility for implementing orders or directives. All or part of this authority may be transferred or delegated.

Command and control - the process of and the means for the exercise of authority and direction by a properly designated commander over assigned forces for the accomplishment of the commander's mission. Command and control capabilities include resources to:

- obtain, report, communicate, process, analyse, synthesise, display and disseminate information to support command planning and decision-making;
- formulate alternative courses of action;
- make decisions; and
- communicate orders to subordinates and monitor the results of actions and the status of forces.

Command Support System - an integrated information storage and retrieval system, together with the necessary personnel and utilities required to support a commander at any level.

Communications (for command and control) - the means by which command and control is transmitted or made known between or within commands.

We note that it is important to specify enterprise aspects of Quality of Service (QoS) in enterprise specification, as this should describe a user's view of what constitutes quality

of a particular service. This description should then be mapped onto the other viewpoint specifications.

The concepts outlined above summarise some initial recommendations for C4I enterprise modelling. For a more detailed enterprise specification, the concepts from other C4I efforts should be collected and combined in order to reflect the new ADF arrangements

3.3.2 Information viewpoint. An information specification defines the semantics of information and the semantics of information processing in an ODP system. Such a specification includes the definition of a) *information objects*, which are derived from the specification of the enterprise objects and b) a set of related schemas which represent relationships between information objects.

An *invariant schema* expresses relationships between information objects that always must be true - for example the relationships between information objects representing geographical points on military maps.

Static schemas express assertions that must be true at a single point in time. For example a description of an operations plan.

Dynamic schemas specify how the information can evolve as the system operates, e.g. many military reports.

We note that some of the US DoD initiatives have produced detailed information specification, in particular those related to C2 Schemas and these will need to be further considered.

3.3.3 Computational viewpoint. A computational specification defines the functional decomposition of an ODP system into the *computational objects* that interact at well-defined *interfaces*. In the context of the C4I domain this means that mission applications should be structured in terms of computational objects and their interfaces.

If an object representing a specific computation function of a command support system is visible within the enterprise specification, i.e. a map object, then a one-to-one correspondence between the enterprise and the computational object can be established.

If a complex interaction between computational objects needs to be supported, a special computational object, a compound *binding* object can be used. This object enables dealing with multi-party interactions as well as managing QoS characteristics of the expected interactions. A compound binding objects is a suitable means of storing standard patterns of computational interactions between objects. We note that the commercial middleware technologies of today cannot provide such a mechanism. In many cases the concept of binding can have its correspondence with the concepts of community specified in the enterprise viewpoint.

It is important to note that the RM-ODP object model is generic and can accommodate any object model, e.g. a CORBA, DCOM or Java-RMI object models. It also provides a number of additional capabilities, such as dealing with streams, QoS requirements, multi-party interactions and different type systems.

Most of the existing C4I architectures and reference models address only the computational aspects of a C4I system. For example, the DARPA Joint Task Force Advanced Technology Demonstration has developed a Reference Architecture (JTF-RA) with servers that represent computational objects [10].

3.3.4 Engineering viewpoint. While the computational specification is concerned with 'when and why' objects interact, the engineering specifications describe 'how' they interact [4]). The engineering representations of computational objects are *basic engineering objects* and primitive computational bindings are visible as *channels* or local bindings (on one node). Channels with many endpoints can also be defined for linking of multiple engineering objects, or engineering objects that provide some ODP functions.

In cases where a channel crosses some technical or organisational boundary, there may be a need for additional checks or transformations to match the requirements on the separate sides of the boundary. These functions are performed by *interceptors*, which form part of the channel. They may need to perform format or protocol conversion, or may provide accounting or access control checks.

We note however that the engineering language concepts should be used in a C4I architecture to the extent that they are needed to describe capabilities of the specific middleware technologies. There should be a mapping between these engineering concepts and the characteristics/mechanisms adopted within specific middleware technologies or operating systems/protocols (to be described in the technology language). This mapping will ensure the link between a technology independent specification and the specific technologies to be used.

In cases where multiple middleware systems are used, the appropriate interceptors should be defined. Examples of such interceptors are bridges, e.g. a CORBA-DCE bridge.

Recent C4I initiatives, i.e. JTF-RA, suggest the use of DCE, CORBA or DCOM engineering solutions as the infrastructure base. The choice of the particular technology to be selected will depend on a number technological and other factors. In the absence of the capability to predict exactly which of these are to prevail, it is recommended to consider all of these technologies by means of the scenarios, as will be discussed in section 4.

3.3.5 Technology viewpoint The technology language provides a link between the set of viewpoint specifications and the real implementation, by listing the standards used to provide the necessary operations in other languages. This language thus provides extra information for implementation and testing by selecting standard solutions for basic component and communication mechanisms.

For C4I systems, this language is important in that it specifies which of the technologies to be used can meet performance requirements and thus the quality of communications and computing services to be used to meet these.

3.3.6 Linking between specification and implementation. When undertaking the modelling of a specific system using the C4I-RA developed here, it will be beneficial to use a particular notation to represent the concepts introduced. We anticipate that the emerging Unified Modelling Language (UML) [11] methodology is a candidate to represent the concepts of different viewpoints.

The C4I RA description derived by applying the RM-ODP specification languages needs to be linked to the characteristics of the commercial distributed object technologies such as CORBA, DCE, DCOM and Java-RMI. (See Figure 5.) One way of establishing such links is by including constraints of these technologies into the corresponding parts of RM-ODP description. Examples of such constraints from the OMA object model are: one interface per computational object, the lack of the notion of computational binding, and the lack of the support for streams. An example from ActiveX/DCOM is the absence of support for multiple inheritance. We plan to investigate a detailed mapping of the RM-ODP languages unto the characteristics of specific middleware technologies.

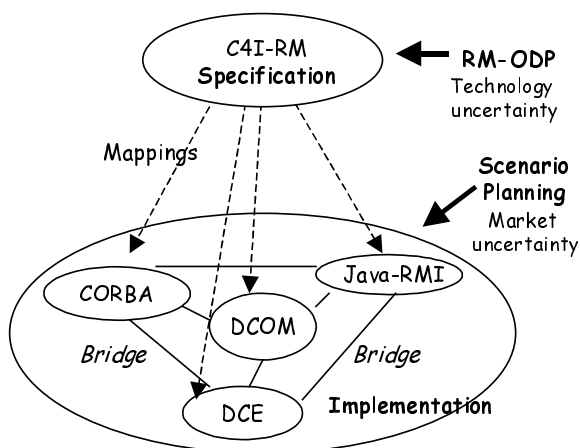


Figure 5 Mapping specifications onto commercial technologies

4. Managing Uncertainty: Software Technology Scenarios

The C4I-RM provides a basis for middleware-neutral specification of C4I systems (Figure 5). It facilitates a stable specification of the enterprise architecture in spite of the unpredictability of current distributed object infrastructure evolution. This approach addresses *technological uncertainty*. The next step toward system development is the mapping of the C4I-RM onto commercial middleware technologies. Currently there exists no easy solution to this problem.

Decision-makers must consider possible future technology scenarios to reflect the influence of market factors. A good understanding of these factors enables the organisation to react to these external trends. In this case study our Australian Defence Force (ADF) clients specifically asked us to address *market uncertainty*. In this section we report on our application of the "scenario planning" approach to develop possible futures of the near-term software industry and their impact on the C4I system.

4.1 Scenario Planning Process

Scenario planning is not a new concept and most large organisations, especially military, have long practiced some variant of it. Scenario planning is based on the insight that it is not necessary to know "The Future" in order to cope with uncertainty. Rather, reasoning from certain knowledge of the present, several possible futures can be postulated. The benefit of these possible futures, or scenarios, is to permit identification of emerging trends, global factors, etc. Thus, decisions about the future can be made in the present with some confidence that uncertainty has been mitigated by identification of invariants within one's control or by identification of leading indicators that could warn of changing fundamentals.

The scenario planning approach used here is based on the work of Peter Schwartz [3]. Schwartz gives eight steps for developing scenarios:

- 1 Identify a decision - Begin with a specific decision or issue.
- 2 Key forces in the local environment - Identify key factors influencing the decision.
- 3 Driving forces - Macro-environmental factors that influence the key forces.
- 4 Ranking - Rank key and driving forces by importance and uncertainty.
- 5 Select Scenarios - The results of ranking determine the dimensions or framework for possible scenarios. Choosing scenarios that capture the important aspects affecting the decision is critical.

6 Play out the scenarios - Return to forces and factors from steps 2 and 3 and ensure they are covered.

7 Implications - Consider the decision or question in view of the different scenarios.

8 Pick leading indicators - Identify indicators that can be monitored over time to verify scenarios.

4.2 Establishing the Scenario Framework

We begin scenario planning (step 1) by asking a definite question critical to decision-making about the implementation of the C4I system: "Should Microsoft (MS) software be used exclusively to build the C4I system."

We then identified key forces in both the local environment (step 2). From our knowledge of the domain of the C4I project, we chose the following major local environmental forces:

- Legacy systems - There are extant C2 systems running on UNIX that contain large amounts of custom and proprietary software.
- Resource constraints - Constraints on money and time have become even more of a concern in recent years.
- Corporate Policy - What are the corporate policies regards computing standards?
- Culture - What biases exist in the organisation towards software development, contracting, and maintenance?
- COTS - How greatly does the organisation depend on COTS for future systems?

Then we looked at the macro-environmental forces (step 3). To determine these global factors we posed the following question: "What is the near-term future of the commercial software market for middleware and the desktop". We felt exploring this question would yield insight into the underlying forces. We think these two aspects, middleware and the desktop environment, are the areas of highest importance and greatest uncertainty in regards to the main study question stated above.

The next step in the scenario planning process (step 4) is ranking the forces affecting the decision. After a detailed analysis we rated both the local and macro-environmental forces in terms of importance and uncertainty. The two forces most highly rated in both uncertainty and importance were (1) the future direction of middleware and (2) the future direction of the desktop. The key question of the future direction of middleware has been whether "open standards" will prevail over proprietary systems, i.e. MS. Although, the future of the desktop has seemed clear cut (MS domination), our subsequent analysis yielded an alternative.

Several potential technology developments could change the face of the desktop. The developments that could lead to a new desktop environment include increased bandwidth, component-oriented software development, the network PC and electronic commerce. These potential

technologies are seen as driving forward the "network PC" concept.

Note that this alternative to MS desktop domination is orthogonal to middleware domination. MS's DCOM could provide the middleware for ubiquitous ActiveX components as long as the system desktop turned out to be as dynamic as we have described.

Given the above considerations, we were confident in choosing the dimensions for our scenario framework to be those of middleware and the desktop (step 5). This framework is shown in **Figure 6**. The labeling of the dimensions is as follows. Closed Desktop implies continuing MS domination. Open Desktop implies the network-PC possibility. Closed Middleware implies a proprietary distribution layer (normally considered to become MS DCOM.) Finally, Open Middleware implies an "open standard" is adopted, e.g., CORBA. Based on these two dimensions we develop five potential scenarios. The scenarios are explored below (step 6 in the process).

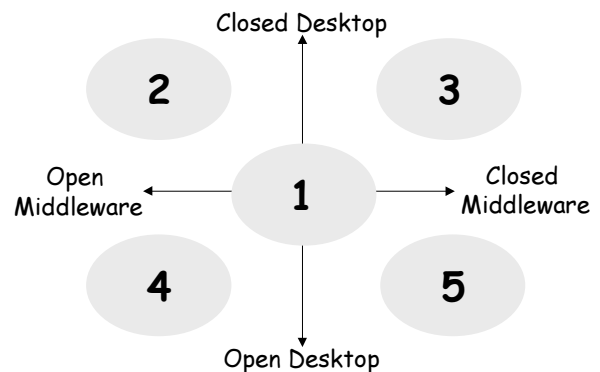


Figure 6. Scenario framework

4.3 Exploring Scenarios

Scenario 1: This represents a continuation of the current situation. There is no clear near-term winner on either the desktop or middleware. MS retains dominance on the PC desktop, but ActiveX and DCOM are not ubiquitous. UNIX systems retain important roles especially in corporate infrastructure with CORBA Orbs providing essential distributed object systems. Netscape, Sun and similar competitors remain viable. The browser metaphors dominate and provide cross-platform functionality.

Scenario 2: This scenario represents a type of extreme "cold war" polarisation with UNIX and CORBA gaining ground in the distributed computing area and MS maintaining the desktop and local computing. Perhaps Windows NT and DCOM cannot scale to WAN levels and CORBA succeeds. However, in the PC desktop arena MS pushes out almost all competition. Proxy wars are fought on the boundaries between DCOM and CORBA (bridging technologies). Middleware (DCE,

CORBA, and Java-RMI) becomes a profitable sideline business.

Scenario 3: Numerical estimates vary, but MS rules the vast majority of desktop computers. It does this by controlling not only the operating system, but most of the applications as well. In this scenario, MS succeeds in achieving monopoly on both the desktop and middleware. The current candidates for middleware are ActiveX and DCOM. In this scenario, DCOM remains relatively closed and beats out CORBA and any other current contenders. Windows NT becomes the ubiquitous networking operating system displacing UNIX. UNIX, CORBA, X-windows, et al, are relegated to niche markets and cult followings.

Scenario 4 : Under this scenario MS's attempt at global domination fails. This can take on several technical manifestations. For example, DCOM fails to scale and CORBA succeeds in the middleware dimension. Alternatively, DCOM becomes an "open standard" and co-exists with, or displaces, CORBA. Also required is an Open Desktop. This could be the "pay-per-view" scheme or some open alternative to Windows is developed.

Scenario 5: The open desktop or Network-PC scheme comes about. There is some degree of choice in the market. MS is constrained to act within this framework. However, MS can own the technology infrastructure. This could be ActiveX and DCOM or a MS proprietary Java-RMI, etc. Perhaps the distinction between UNIX and Windows blurs and they both co-exist.

4.4 Implications

Having developed a range of scenarios, we return to the original question and consider the implications of answering it in the affirmative, i. e., the C4I system is to be built entirely based on MS market offerings. This is step 7 in the process. Here we can "rehearse the future" and seek to understand the implications of the decision. The decision should be looked at in each scenario. Is any particular vulnerability revealed? Does the decision work well for only one scenario?

In **Scenario 1** we have a reasonably good decision. MS retains the desktop and applications, hence our domain-specific component development work is well placed. We have a heterogeneous distributed environment in the sense that non-MS systems are providing the distributed systems infrastructure for wide-area enterprise applications. Java and the like also provide a variety of alternatives for distributed applications. The browser and browser- metaphor take on more importance as a cross-platform solution to heterogeneity.

The **Scenario 2** might be the most "uncomfortable" outcome. While all of our software development work is to

our benefit, we are faced with bridging into the distributed systems layer. This may require us to maintain one platform for the desktop and another type of platform for back-end servers. On the other hand, with the exception of undesirable bridging software, our basic decision is sound.

Obviously, in the **Scenario 3** we have made a good decision. MS will take care of the desktop as well as the distributed application middleware. All we have to do is develop our business specific components. We have no concrete idea as to the cost or magnitude of that effort. MS will certainly not supply those domain-specific components. There are obvious military implications of depending totally on one foreign vendor for such a mission critical system.

Scenario 4. This is the risky one. First, what does it mean to us for MS to loose the desktop? If it means all our domain-specific work is lost, then it is a major setback. What if the open desktop turns out to be "network-centric?" Presumably, this offers us greater variety. What does it mean for MS not to dominate middleware? For example, if we build our system on DCOM and that technology fails. How does this impact us? In short, the effect of this scenario is measured in how much of our MS-based, domain-specific and middleware software would have to be ported or rewritten.

Scenario 5 appears the most favorable for our decision. We have relative homogeneity of operating system and interoperable middleware with no bridging required. The "open desktop" provides a selection-rich marketplace for software components and our domain-specific components are compatible.

4.5 Leading Indicators

Having developed the scenarios, it is important to closely monitor the evolution of actual events. This is required in order to know as soon as possible which, if any, of the scenarios are unfolding. In this final step (Step 8) of the scenario planning process, we look for "leading indicators". In other words, can we identify signposts or trends that aid us in recognizing what scenario is unfolding or where we are in a given scenario.

Thus far we have used the framework to generate scenarios in order to understand possible futures. It would be desirable if this model could also provide some predictive function as well. Now, there exists a third axis or "meta-force" that has not been mentioned in connection with this framework. That force is the drive of both commercial and non-profit (standards bodies) organisations to increase their market share. Can we use this fact to add predictive power to the model?

The approximate placement of several important software products and technologies is shown in Figure 7. It is interesting to note the position browsers occupy in the

framework. This is the current marketplace/battleground, i.e., the "Browser Wars". There is a tug-of-war between Netscape (NS) and MS with various players (Sun, IBM, HP, DEC, Apple, etc.) weakly aligned with either side. The pre-release Internet Explorer 4.0 with its Active Desktop is moving away from the desktop metaphor and in the direction of a browser interface. MS has announced an IE 4.0 for Solaris for example. Any move to expand the "browser" area by a competitor to depend on their technology is a clear gain in market share.

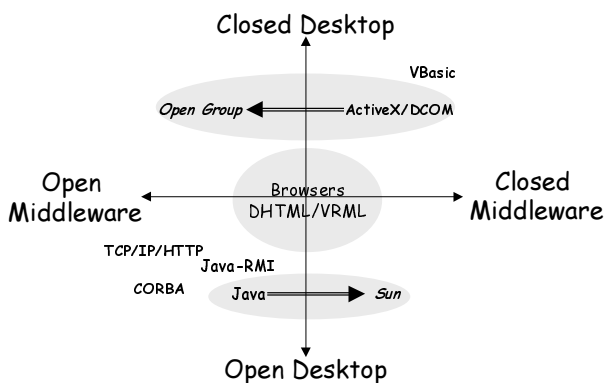


Figure 7. Position of various products within scenario framework

Let us examine two recent developments. First is the move by MS in July 1996 to turn ActiveX (including DCOM) over to the Open Group. (The Open Group is an industry consortium formed from the merger of X/Open and the Open Software Foundation.) This is indicated in Figure 7. The Open Group then formed the Active Group to promote the standardisation and adoption of ActiveX. MS has agreed to provide distribution licenses for specifications, source code reference implementations, validation tests and some trademark rights to the Open Group. On the surface, this appears as simply a move toward "openness" to gain support for an emerging Internet standard. Many see DCOM as the natural successor to DCE. However, it must be driven by the desire to acquire market share by taking over the Open Group and attacking OMG. The move by MS is clearly perceived as a threat by OMG as gauged by their reaction in the press.

Second, the manner in which Sun would like to "standardize" Java. Sun has proposed to the International Standards Organisation (ISO) and the International Electrotechnical Commission (IEC) to become a Publicly Available Submitter (PAS). A PAS acts like a standards body itself, accepting submissions to improve and update a technology and passing them onto the joint technical committee of ISO/IEC for final approval. So far, no for-profit corporation has asked for or been awarded PAS status.

With the exception of Sun's close ally IBM, the entire US computer industry has joined to oppose Sun's scheme.

The above analysis indicates that any move to the left quadrants, toward the Open Middleware area, would tend to increase market share for MS, if indirectly. While, on the other hand, movement into the Closed Middleware area would tend to decrease market share. The point of this exercise is to develop insights into market dynamics for predictive purposes.

4.6 A near-term strategy

From the scenarios developed above, the choice of MS for the desktop environment and for the development of new user applications is clear. The major question left is the choice of development environments. To the extent legacy (Unix) systems are supported, it would appear that Java is the platform independent choice. This would at least permit internet browser access to most new applications. If platform independence is not a concern, the choice of languages should be determined on a project-by-project basis. In this case, the individuals responsible for the development effort should be free to choose the most appropriate language for the task at hand (consistent with corporate policy).

As for middleware, the choice of DCOM seems premature. Adopting DCOM at this point seems too high risk as there is insufficient data available on its performance. This leaves DCE, CORBA, and Java-RMI. We would also delay going with Java-RMI for the same reason given for DCOM. Between DCE and CORBA, we would look to CORBA as it has the most momentum.

5. Conclusions and Future Work

This paper presents our experience in dealing with the uncertainty associated with the adoption of distributed object technology in the C4I enterprise domain. The approach consists of producing a reference model for the C4I domain based on the RM-ODP framework. Such a reference model is a starting point for producing a family of architectures from which concrete C4I systems can be built. This RM helps address the problem of technological uncertainty associated with selecting middleware platforms.

In addition, we adopted the scenario planning technique to address the market uncertainty. This was strongly motivated by the client's desire to answer hard questions about the future of middleware and its impact on the C4I system implementation. The exercise we reported on here was done as an illustrative example for the client. It is not a complete scenario planning process for at least two major reasons. First, it was conducted in a short period of time with limited resources. But, most importantly, the persons affected by its outcome must carry out this process. It is as much a consensus building effort as a technological problem

solving approach. However, since this exercise was completed, developments in desktop and middleware appear to be unfolding according to one of the scenarios. MS is moving into the region of Scenario 5 that we indicated as being most favorable to the initial question.

Our initial experience with the combined application of RM-ODP and scenario planning to the C4I domain has shown some benefits and pointed out areas for further work.

First, it is important to establish a link between RM-ODP specifications and the characteristics of commercial distributed object platforms. This will facilitate implementation of enterprise architectures using specific technologies, such as CORBA, ActiveX/DCOM, Java-RMI and DCE. This requires significant future research.

Second, the availability of the Object Oriented Analysis and Design tools which incorporate RM-ODP concepts would be add value to a software engineering process for open distributed systems. We anticipate that UML [6] will provide some support in this direction.

Finally, "design patterns" [12] have gained much attention lately as a means of improving software design and reuse. We intend to investigate the usefulness of patterns to aid in mapping between high-level specification and implementation.

Acknowledgement

This work supported in part by the Distributed Systems Technologies for C3I program of the Information Technology Division, Defence Science and Technology Organisation (DSTO). The work reported in this paper has also been funded in part by the Cooperative Research Centres Program through the department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

References

1. ITU-T X.902 | ISO/IEC 10746-2: ODP - Reference Model - Part 2, 1995.
2. ISO/IEC IS 10746-3. International Standard 10746-3, ITU-T Recommendation X.903: Open Distributed Processing - Reference Model - Part 3: Architecture, January 1995.
3. P. Schwartz, "The Art of the Long View", Double Day, 1996.
4. P. Linington. RM-ODP: The Architecture. In, ed., K.Raymond, L.Armstrong, Open Distributed Processing; Experiences with distributed environments, Proc. of the 3rd International Conference on Open Distributed Processing, p. 15-33, Chapman & Hall, 1995.
5. M. Uschold, M. King, S. Moralee, Y. Zorgios. The Enterprise Ontology, available at <http://www.aiai.ed.ac.uk/~enterprise/enterprise/ontology-code/enterprise-v0.1/>
6. The Unified Modelling Language - <http://www.rational.com/ot/uml.html>
7. Towards A New ODP Enterprise Language Standard, Milosevic Z. Bearman M, ICODP97, Toronto, May '97\
8. A.J. Jones, M. Sergot, "On the Characterisation of Law and Computer Systems: the normative perspective", in Eds. J. Mayer, R. Wierringa, Deontic Logic in Computer Science: Normative Systems Specifications.
9. M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Discipline, Prentice Hall, 1996.
10. Rick Hayes-Roth and Lee Erman, Joint Task Force Architecture Specification, Teknowledge Federal Systems, May 1994. (<http://jtfweb3.nosc.mil/docs/jtfas-abstract.html>)
11. Viktor Ohnjec, Converging on OOAD Agreement, in Application Development Trends Software Productivity Group, 1997. (<http://www.spgnet.com/ADT/>)
12. E. Gama, et al, Design Patterns, Addison-Wesley, 1995.