Defence Research and Development Canada | Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

A Collaborative Cellular Framework for Defence Applications

M. Lizotte
DRDC Valcartier

Defence R&D Canada – Valcartier
Technical Note
DRDC Valcartier TN 2004-464
October 2006

Canada

A Collaborative Cellular Framework for Defence Applications

M. Lizotte

DRDC Valcartier

Defence R&D Canada – Valcartier

Technical Note

DRDC Valcartier TN 2004-464

October 2006

A Collaborative Cellular Framework for Defence Applications

Michel Lizotte
Defence R&D Canada
2459 Pie-XI Blvd North,
Val-Bélair, Québec, Canada, G3J 1X5
Michel.Lizotte@drdc-rddc.gc.ca

Abstract

For many years now, DRDC Valcartier has been developing expertise in building military collaborative software. The new concept presented in this paper, the Collaborative Cellular Framework (CCF), is based on the lessons learned from these projects. It introduces the notion of a Collaborative Cellular Network (CCN) allowing an interactive access by any participant to any complex information that all other participants have made available. This framework concept supports application development following this paradigm. It enables sharing of complex objects e.g. a master object along with its satellite objects and relationships.

1. Introduction

Nowadays, collaborative networks of computers are observed everywhere from specialised ones (e.g. banking systems) to very general ones (e.g. internet). One can easily infer that it is advantageous to participants. In such contexts, collaboration means that an entity is allowing other members of the "collaboration agreement" to use some of its resources and/or information. This collaboration trend, particularly in the recent years, only reinforces the pertinence of building such networks for military applications. Major interest into Network-centric Warfare is also an evidence of this trend [6].

In a military situation awareness context, using such networks makes sense particularly if it accelerates commander's understanding of the battlespace. More than ever, commanders must quickly understand situations to take timely decisions. A commander cannot take good and timely decisions without good and timely information. Missing key information known to some units of the Force but not transmitted in time to the commander can turn into a nightmare. Such a situation may arise when a unit do not realise the importance of the information for another unit and does not transmit it or when the time delay between the observation and the information delivery is not adequate. A solution is to provide commander's staff with live links to other units also observing the same battlespace; each unit providing its understanding of the situation.

In addition to this operational architecture deficiency, this paper also addresses issues in architecture of software-intensive systems. Software engineering is a very young discipline in comparison with other kinds of engineering: our era still builds "bridges that collapse". In opposition to bridges, components used for building software have often few years of age. Technologies are evolving so quickly that organisation, including their software development team, can hardly follow the pace. Immaturity of the discipline brings a lot of partial and complete failures of software-intensive system acquisitions. These are generally noticed from product bad quality or unexpected high cost of development and maintenance. In addition, although very promising, software reuse still requires maturing.

In terms of operations, the main target being address by the current paper concerns military collaborative interaction software. In terms of software development, the main targets are reuse of components, scalability to different size of military units and adaptability to technology evolution. The Collaborative Cellular Framework (CCF) proposed in this paper is addressing these issues.

The functional advantage of using such a framework has been demonstrated during a specific experiment in August 2000 [1]. In addition, usage of frameworks is known as reducing development costs and ensuring quality and reliability of common components [5]. Quality and reliability of common components comes from the dedicated architectural effort from experimented personnel in an application domain and are reinforced by repeated uses of the framework. Cost reduction comes from the utilization of common components already functional but also from effort diminution for construction and maintenance of other components i.e. personnel are faster using a standard way of building systems. The component-based framework presented here should not be confused with methodology frameworks such as DoD Architectural Framework (formally called C4ISR AF) or IEEE-12207 targeting architecture descriptions and processes. CCF provides pieces of code directly used to build systems.

The next section of this paper outlines the context within which the framework has been developed. Section 3 provides an overview of the framework while Section 5 and Section 6 provide some technical details. The paper concludes by highlighting the framework features and suggesting future work.

2. Context

The CCF was elaborated during the Land Intelligence and Electronic Warfare Automation (LIEWA) project [1]. The LIEWA project aim was to define and validate a process providing effective battlefield visualisation to LF commanders for understanding an adversary. In order to validate a collaborative process, DRDC Valcartier has built a software system prototype called the All-Source Intelligence Producer (ASIP). Its main objective was to provide validated software system models bringing new technologies and innovative functionality to intelligence users, collaboration being the main focus. The approach was: (1) to elaborate innovative software models; (2) to select emerging technologies; (3) to build a software testbed; (4) to conduct "realistic" military exercises in order to assess the models and measure the impact of the innovation they bring to the user.

As shown on Figure 1, DRDC Valcartier has a relatively long story in collaborative information systems for military intelligence. The ASIP prototype and its rich historical background are the central sources of the CCF concepts being introduced in this report. From 1989 to 1995 an important object-oriented prototyping and trial effort called the Tactical Information Fusion (TIF) project was carried out [2] [3]. This prototype brought a collaborative environment to an intelligence analysis team. Following this, a two-year software architecture effort was conducted [4]. The aim was to elaborate a software architecture based on the lessons learned during the TIF project and expanding the features to support operations other than war (OOTW). ASIP used this target architecture, which includes around 350 conceptual classes. ASIP 1.2 was used for the first LIEWA major experiment known as "1a" and implemented a subset of the target architecture.

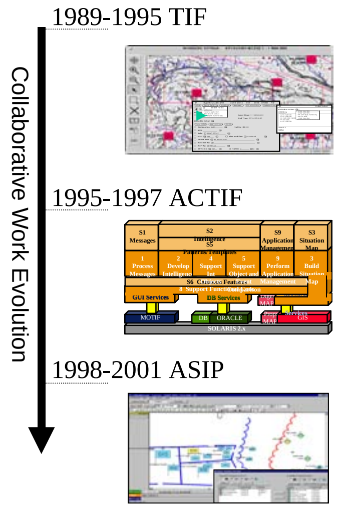


Figure 1. Collaborative Work Experimentation

A lot of know-how emerged from those years but only elements relevant to the CCF are documented in this paper. Here is a high-level view of their contribution:

- Practical experience is the best way to summarize the TIF era. Major demonstrations [2], and military exercises [3] had followed a couple of years of intensive development using emerging technologies. Those major events brought a lot of new functional requirements, clarified some others and, moreover provided knowledge about real bottlenecks and priorities in the field. Although many requirements were related to the intelligence domain many others were relevant for any collaborative military information systems. In addition the development team identified technical deficiencies.
- The ACTIF period can be synthesizes as a thinking phase. An innovative intelligence system concept for an analysis center was elaborated and documented based on results from previous years. ACTIF architecture was validated with military intelligence users through workshops [4].
- The ASIP period has been a validation phase for some new concepts put forward by the ACTIF architecture, that were focussing on collaboration between analysts of an analysis centre. In addition, although not originally thought for it, the collaboration concept was expanded to collaboration between different units within a brigade.

The idea of packaging these collaborative features within an object-oriented framework came during the LIEWA project as a result of Experiment 1a. This experiment was conducted with ASIP 1.2 implementing a subset of the ASIP target architecture as shown in Figure 2. The CCF is the result of extracting elements from the ACTIF and ASIP architectures and re-architecting them to make a coherent set of reusable components and classes independent of the intelligence domain.

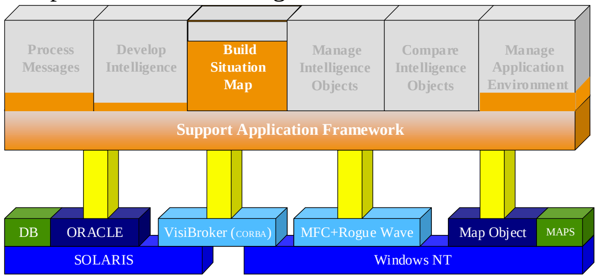


Figure 2. Architecture of ASIP 1.2 Application

3. Framework Overview

The Collaborative Cellular Framework (CCF) primarily targets development of military information systems for interactive collaboration. It introduces the notion of a Collaborative Cellular Network (CCN) and supports application development following this paradigm. To address poor software interoperability with existing systems, a deficiency well recognized by nations involved in military operations, the framework enables existing systems to access the CCN.

In addition to its collaborative flavour, CCF follows current major IT trends such as component-based and N-tiers approaches in order to achieve reusability, scalability and adaptability. The framework includes basic elements that should be part of any information systems supporting military operations. The framework defines a strong set of classes and patterns for implementing military information systems.

3.1. Key Principles

In order to elaborate operational and technical architectures, some key principles were gradually established. They are mainly based on experience of previous years with military collaborative system development and state-of-the-art in IT.

Here are these key principles:

1) A Workspace defined by a Cell, a Role and an Operation or Exercise

The concept of Workspace corresponds to the “virtual office” of a role within a specific Cell working on a specific Operation or Exercise. For instance, a workspace could be defined by the senior analyst role of the 1st intelligence company working on the Exercise Zarpa 2000. A Role does not equal a user since different users, but one at the time, can play the same Role depending on a shift schedule.

2) Live sharing of domain objects

The live sharing of objects enable a client user to see an object owned by others. The owner can make an object private, confidential to the cell or public to any interested user on the network of cells. Once subscribing to such an object, the application of the client user will be noticed of any changes made by the owner to this object. The interfaces provided by a CCF server also enable existing systems, not constructed with the CCF, to interoperate with a CCF server and have the same functionality. For instance, an intelligence legacy standalone system can be enhanced with a bridging module allowing exploitation of a CCF server that manages sharing of intelligence objects.

3) A common environment management subsystem

A CCF application must use an environment management subsystem provided with the framework as an executable component package. Components of this package provide a single user entry point and a single server entry point to any CCF application. In particular, the server-side of this subsystem allows any existing system to link with the cell as long as it has implemented a simple interface to access objects of the server.

4) A set of domain subsystems

A CCF application functionality is split into a set of domains defined by the development team e.g. message, intelligence, symbology, situation map, … A CCF application domain is supported by one or more subsystems built by the application development team, in opposition to the cell configuration subsystem that is provided with the framework. These subsystems are built from a set of classes provided with the framework. This usage of framework classes by application developers is referred as development-time reuse: It requires compilation.

The framework enables “development-time reuse” through component partitioning and layering that allows reuse of the components worked out by the application developers e.g. reuse of Symbology GUI with re-implementation of the persistence.

In addition, the framework approach allows “runtime reuse” of the components produced by the application developers e.g. a set of server for intelligence objects. Such components do not require compilation and would be reuse by another development team.

5) A layered subsystem architecture

Scalability is ensured through the use of N-Tiers architecture. These tiers can be run on a single computer as well as on multiple servers without changing the application. A subsystem is implemented through a set of layers.

6) Object Caching

The framework manages caching of complex objects (master objects along with their subordinate objects and relationships) on the client to increase response time, decreasing network traffic and alleviating the server load.

7) Mobile code

The framework manages mobile code from the server to client applications for specific usage such as validation of object contents. Again it is to speed up the response time, decreasing network traffic and alleviating the server load.

3.2. Operational Perspective: The cells

A CCN is a kind of collaboration agreement providing any participant an immediate access to any complex information (objects and their relationships) that all other participants have made available. A participant being a cell that typically represents an organization with its own specific objectives. Each cell decides what information it makes available.

The latest DRDC Valcartier experiment related to collaborative work was conducted during Exercise LIEWA 1a. This experiment has demonstrated a case where CF can benefit from a collaborative cellular network [1]. The aim was to measure impact of live sharing of tactical situations on the intelligence quality provided by intelligence analysis team. Seventeen military intelligence specialists split in four cells have participated to this exercise. Figure 3 demonstrates a collaborative cellular network (CCN) built for Exercise LIEWA 1a supporting the 20 CMBG intelligence function. The ISTAR CC Cell is looking live at the situation developed by the R22eR, PPCLI and RCR Cells in order to developed a brigade level situation also available live to the these battalion cells. The G2 Cell has a live access to these situations in order to build its estimate of the situation. Such a CCN allows higher level of command to dig in the information i.e. having access instantly to lower level details if they are required to make recommendations. The CCN concept used by the CCF was tested with situation overlay objects during LIEWA 1a but seem applicable to any object of an application domain.

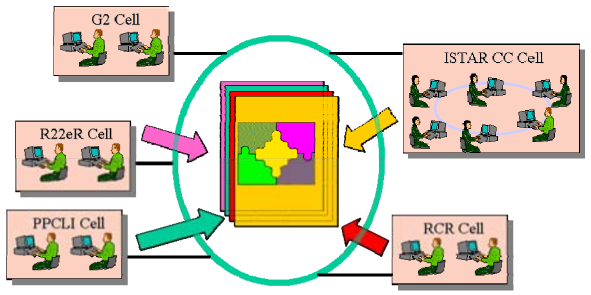


Figure 3.The Collaborative Cellular Network (CCN) used for LIEWA 1a

The LIEWA Experiment 1a was conducted with ASIP 1.2. This version was offering advanced features to draw military situations on an electronic situation map. Intrasystem communication between the ASIP server objects, running on Solaris, and the ASIP clients, running on Windows NT, were achieved through CORBA technology. Among the advanced features, ASIP 1.2 was providing collaborative work features, implemented through the CORBA event service, allowing users to share live military situation overlays. ASIP 1.2 had also a CORBA IDL interface allowing interoperability with non-ASIP clients i.e. other systems.

Figure 4 summarises the conceptual object model of CCF. The concepts introduce in this figure are described below.

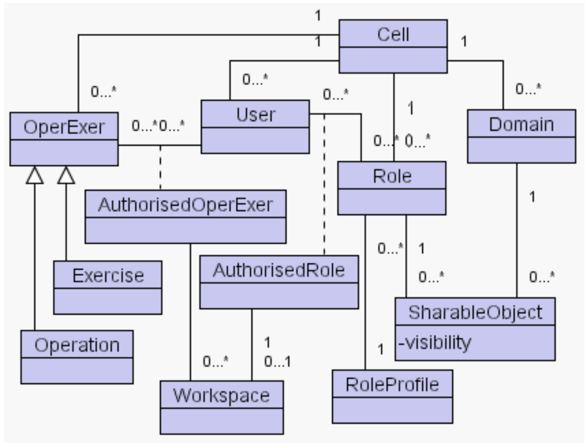


Figure 4. Class Diagram of CCF

Cell

A Cell represents a logical unit of work member of a CCN. A Cell is composed of a set of users collaborating to achieve their duty. Examples of cells are shown in Figure 3.

User Role

Figure 5 introduces the concept of user role. The two bean characters represent users while military pictures represent four user roles. A User Role represents a position being held by a user member of the cell. In opposition to the notion of user, it does not represent a person. Through a profile, a user role identifies the rights and responsibility that a user have e.g. if he is a terrain analyst then he should have the right to modify the map layers. In particular, this concept allows managing shifts into which a user takes over another one at the same position.

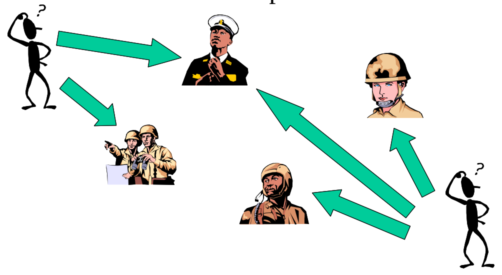


Figure 5. Playing a User Role

Operation and Exercise

The CCF uses standard definitions of Operation and Exercise to manage workspaces. It offers the same functionality and behaviour for both concepts. Figure 6 shows that a cell can simultaneously use many operations and exercises. To have access to an operation or an exercise, a user must have an explicit authorization.

An Operation represents a military action or carrying out of a strategic, tactical, service, training, or administrative military mission. It is also the process of carrying on combat, including movement, supply, attack, defence and manoeuvres needed to gain the objectives of any battle or campaign.

An Exercise represents a military manoeuvre or simulated wartime operation involving planning, preparation, and execution. It is carried out for purpose of training and evaluation.

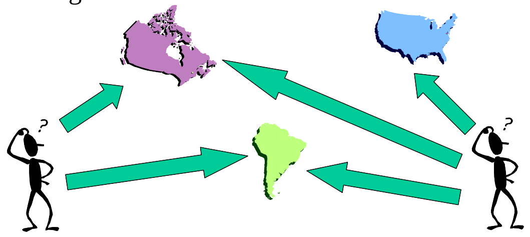


Figure 6. Accessing Multiple Operations

Workspace

In CCF, a workspace is an environment setup identifying a role that a user wants to play and the operation or exercise a user wants to work with, as shown in Figure 7. In other words, a workspace is a user role played over a specific exercise or operation.

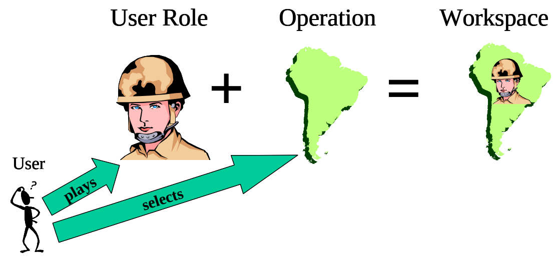


Figure 7. Setting Up a Workspace

Domain

In CCF, a domain is a set of objects about the same subject e.g. messages, situation map, symbology, intelligence, operations, intelligence. Domains are loosely couple while elements within a domain are tightly couple.

Sharable Domain Object

A Sharable Domain Object (SDO) is an entity that can be made available though a publish-subscribe mechanism. A SDO is created and owned by a single user role. For each SDO, the owner can set a visibility property to private, limited or public. A private SDO is only visible to its owner. A limited object is only visible to members of the cell while public objects are visible to all users on the CCN. Typically, a SDO is a complex object composed of a master object (e.g. a military unit), its satellite objects (e.g. sightings) and relationships with other master objects (e.g. subordinate units)

In order to use a limited or public object, a user role subscribes to it and controls its update method. Three methods have been retained. With the live channel method an object update is propagated as soon as the owner performs it. With the second method, the subscriber specifies a frequency at which the object will be refreshed. In the last method the subscriber specifically triggers the refresh.

Which objects should be a SDO and which ones object should not be is not an obvious choice since it depends on the application domain context. During LIEWA 1a, tactical situation overlays were SDOs. Symbols drawn on an overlay were considered as parts of a SDO, satellite objects. It was a functional design choice that the numerous drawings on an overlay would follow the visibility property of the overlay. In other words, the subscription mechanism was not implemented at the symbol level but at the overlay level.

3.3. Technical Perspective: The matrix

As mentioned in introduction, CCF technically aims at reusability (at run-time and development-time), scalability (to different size of military units) and adaptability (to technology evolution and changes). This section presents the CCF approach using the ASIP example introduced in previous Section 2.

Figure 8 shows the target architecture of ASIP. The subsystem *Support Application Framework* encompasses foundations classes offering functionality to develop other subsystem above it. The subsystem *Manage Application Environment* includes functionality required to handle the workspace concepts introduced in the previous section. Other subsystems are implementing domains specific to this intelligence application.

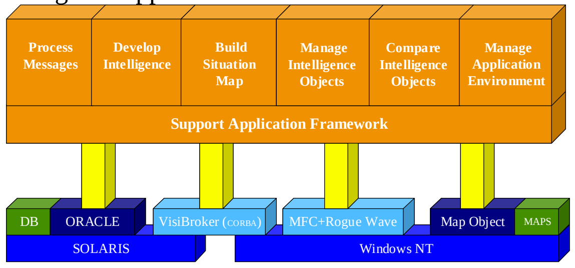


Figure 8. The ASIP target architecture

CCF follows an approach that can be visualised through a matrix. Table 1 shows such a matrix mapping ASIP architecture to the CCF approach. Each row corresponds to a standard CCF layer while each column corresponds to a subsystem of the application. This component strategy is at the core of the CCF technical approach. A row encapsulates a technology while a column encapsulates functionality. The intersection of a layer and a subsystem is a piece, or component, that can be change without impacting the other pieces of the system. CCF enforces standard interfaces between layers and also enforces standard interfaces between subsystems. These relationships are explained below.

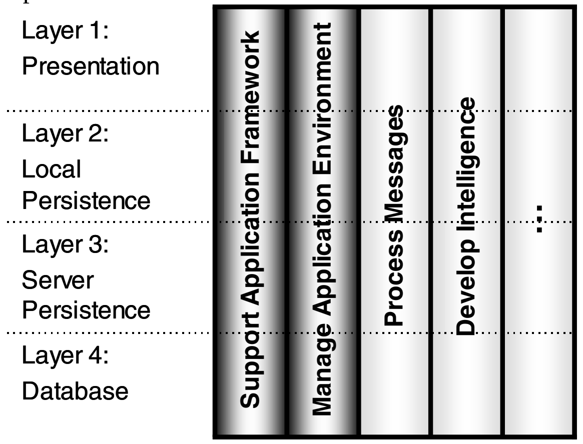

Layer 1: Presentation
Layer 2: Local Persistence
Layer 3: Server Persistence
Layer 4: Database
Support Application Framework
Manage Application Environment
Process Messages
Develop Intelligence
...

Table 1. The matrix strategy

As shown in Figure 9, the Collaborative Cellular Framework (CCF) includes seven levels split in two sets: server-side levels and client-side levels. The arrows in the figure represent dependencies between the levels. The server-side levels support target application (TA) elements running on multiple server computers. They are required if the TA allows multiple concurrent users to share live situations i.e. a collaborative configuration. The framework can also be used for building independent applications i.e. standalone configuration introduced later in this paper.

The server-side levels provide data sharing, persistence and notification services between user' client applications. In addition, any processing services specific to the TA can also be added. The client-side levels support the TA elements running on a single workstation. It provides local data persistence and sharing between local components of the TA.

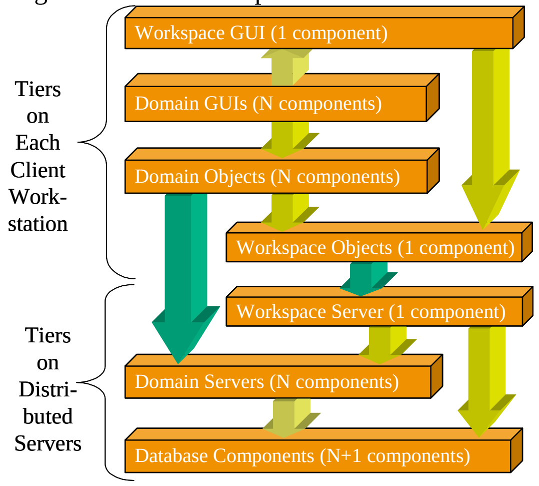

Tiers on Each Client Work-station
Tiers on Distri-buted Servers
Workspace GUI (1 component)
Domain GUIs (N components)
Domain Objects (N components)
Workspace Objects (1 component)
Workspace Server (1 component)
Domain Servers (N components)
Database Components (N+1 components)

Figure 9. Tiers relationships

Within these levels, the framework provides some components, *Manage Application Environment*, and some are to be built, e.g. Process Messages, from classes provided by the framework, *Support Application Framework*. The common components, *Manage Application Environment*, and classes, *Support Application Framework*, are handling the basic framework concepts. The specific components to be built are handling the sharable domain objects and any other parts of the TA not covered by the foundation concepts. Figure 10 emphasises these elements against the tiers introduced in Figure 9.

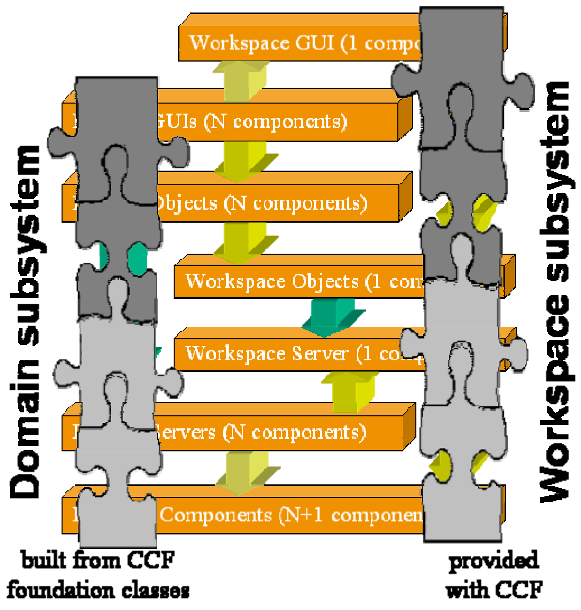


Figure 10. CCF Levels and subsystems

Level 1: The Workspace GUI

The Workspace GUI level has only one common component. It is the first interface presented to the user allowing him to specify a user role and an operation or exercise. This level offers services allowing dynamic completion of the menu bar for integrating different domains within the client application. This level uses Level 4 to store results of interaction with the user.

Level 2: The Domain GUIs

The Domain GUIs level includes the set of specific components providing user interfaces particular to the domains the application is implementing. Each component is about a specific domain e.g. messages, situation map, intelligence, operations, logistics, etc. They allow users to handle a set of domain objects and to access other services offered by the application related to this domain. This level uses Level 1: the menus offered by these specific components are integrated with the basic menus offered by the Workspace GUI. The level also uses Level 3 to store results of domain interaction with the user. Each Domain GUI component uses its corresponding Domain Objects component. These two levels, 2 and 3, are kept separate mainly to allow interface modification without having to alter the domain object itself; users often require modification to interfaces that do not involve any modification to the domain objects.

Level 3: The Domain Objects

The Domain Objects level includes the set of specific components providing temporarily storage and any other services to a client application. This level uses Level 5: each Domain Objects component uses its corresponding Domain Servers component for persistence and collaboration management. This level also uses Level 4: the main objects of a domain are typically in relationships with workspace objects to indicate for which operation they were created, by whom they where modified, etc.

Level 4: The Workspace Objects

The Workspace Objects level has only one common component providing temporarily storage for workspace objects (user role, operations,) to a client application. This level uses Level 5 for persistence and collaboration services.

Level 5: The Workspace Server

The Workspace Server level has only one common component providing persistence and collaboration management for workspace objects (user role, operations, …) to all client applications. This level uses Level 7 for data storage.

Level 6: The Domain Servers

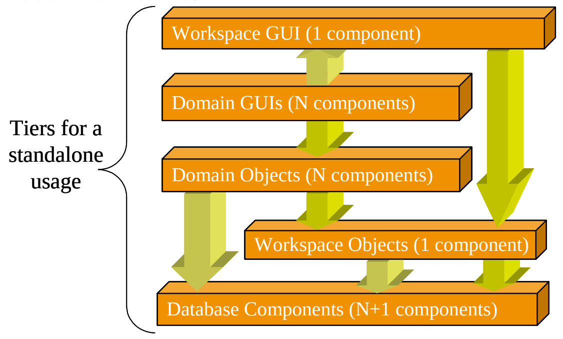
The Domain Server level includes the set of specific components providing persistence, collaboration and any other services to all client applications. Each component provides these services for a specific domain. This level relies in the Level 7 for data storage and integrity.

Level 7: The Database Servers

The Database server includes a set of specific domain components and a common workspace component providing data storage and integrity.

Standalone Configuration

A variant of the CCF can also be used for standalone application in other to benefit from the concepts not specific to a collaborative environment. This variant shown in Figure 11 includes five levels. The CCF four client-side levels are the same. The two server-side levels, 5 and 6, handling the collaborative environment are not present while Level 7 is replaced by an equivalent that do not have to take into account collaboration issues.



Tiers for a standalone usage
Workspace GUI (1 component)
Domain GUIs (N components)
Domain Objects (N components)
Workspace Objects (1 component)
Database Components (N+1 components)

Figure 11. CCF Levels for a Standalone Configuration

4. Using the Workspace subsystem

The subsystem *Manage Application Environment* is provided with CCF. It provides services associated with the key concepts Cell, User, User Roles, Operations and Exercises. It also provides management of the associated privileges and utility services to manage the cell workstation configuration value domains, parameters and colors. Finally, some more specialised services are included such as installation, backup, monitoring, and emergency procedures. In opposition to services provided by the subsystem *Support Application Framework* in Section 5, they are all visible to the user and/or the administrator.

The Cell Server object is the anchor point to access information of a Cell. It is the first object that a client application must reach in order to use computerised services offered by a cell such as user login, specialised servers' start-up and shut down. Through the following operations, the reader will get a very good sense of services offered by this subsystem.

Cell Login
  Login(UserIdentifier)
  Login(UserIdentifier, UserRole, Oper/Exer)
  GetServersList()
  Start the Cell Server
  Stop the Cell Server
  GetCellServerOption(OptionName)
  SetCellServerOption(OptionName, Value)
  GetCellServerMonitoringValue(MonValueName)

Cell Configuration
  GetWorkstationsList()
  GetWorkstation(WorkstationId)
  InsertWorkstation(WorstationId,…)
  UpdateWorkstation(WorstationId,…)
  DeleteWorkstation(WorstationId)

Users
  GetUsersList()
  GetUser(UserId)
  CreateUser(UserId)
  UpdateUser(UserId,…)
  DeleteUser(UserId)

User Roles
  GetUserRolesList()
  CreateUserRole()
  UpdateUserRole()
  DeleteUserRole()

Operations and Exercises
  GetOperationsList()
  GetExerciseList()
  GetOperation(OperationId)
  CreateOperation(OperationId,…)
  GetExercice(ExerciceId)
  CreateExercise(ExerciceId,…)
  UpdateOperation(OperationId,…)
  UpdateExercise(ExerciceId,…)
  DeleteOperation(OperationId)
  DeleteExercise(ExerciceId)

Privileges
  GetPrivilegesList()
  CreatePrivilege()
  UpdatePrivilege()
  DeletePrivilege()
  GetPrivilegeOwners()
  GrantPrivilege()
  RevokePrivilege()

Global Parameters
  GetGlobalParametersList()
  GetGlobalParameter(ParameterName)
  CreateGlobalParameter(ParameterName, Value)
  UpdateGlobalParameter(ParameterName, Value)
  DeleteGlobalParameter(ParameterName)

Standard Codes
  GetCodeTablesList()
  GetChildCode()
  InsertCode()
  UpdateCode()
  DeleteCode()

Colors
  GetColorsList()
  InsertColor()
  UpdateColor()
  DeleteColor()
  MapColor()
  UnmapColor()

5. Developing a specific subsystem

The subsystem *Support Application Framework* is provided with CCF. Qualifying it of subsystem may be misleading since it is mainly composed of abstract classes from which domain subsystem classes (developed by the TA development team) inherit. In other words, it includes the foundation classes of CCF. It provides server-side as well as client-side services to develop an application. The server-side services support development of TAs elements running on multiple server computers. They mainly provide data sharing, persistence and notification services between user' client applications. The client-side services support development TA elements running on a single workstation. It provides local data persistence and sharing between local components of the TA. In addition, this subsystem includes services supporting utility, GUI, graphic, filtering and queries classes that are not further addressed in this paper.

The complexity of these services mainly consists to ensure synchronisation between complex objects (a master object and its satellite objects and relationships with other master objects) and manage their relationships. In addition, they have to meet some quality criteria such as easy of use, performance, scalability, and adaptability. In addition to ASIP 1.2, a CCF prototype was built exploiting Microsoft COM and OMG CORBA to implement most of these services. Through the following lines, the reader will get a good sense of services offered by this subsystem.

CCF Server

This class mainly provides a *Start* and *Terminate* operations in order to ensure safety and security, to initialise data, and to notify the server state for other applications listening on the proper channel.

CCF Client

This class also provides a *Start* and *Terminate* operation but for the client-side. They handle configuration data, splash screens and user logon info and workspace setup (User Role, and Operation or Exercise).

Object Caching and Persistence

This service is provided though the *System Object* class. It implements a distributed data caching mechanism. It implements caches on the client site in order to eliminate redundant access to remote servers and handling event notification and GUI refreshes. It also loads object caches with data from the database.

Event Notification

This service is provided to send and receive signals mainly related to data updates enabling cached object and eventually GUIs to be refreshed. It is closely related to Object Caching.

Data Validation

This service implements a mechanism to check data according to business rules. It enables a client to download software from the appropriate server. This software piece is to be used locally for a preliminary data validation. Typically, once validated, the data is sent to the server to execute a specific transaction. Although the server will perform a complete validation, this mechanism eliminates useless traffic and prevents redundant development of validation routines.

Transactions

This service implements data integrity control. It provides a set of components ensuring integrity of distributed data through a two-phase commit protocol.

6. Conclusion

The CCF marks an important milestone in the DRDC R&D on collaborative military intelligence systems. This framework concept was partially used and tested during a major exercise within the LIEWA project. Other elements, more technical and not tested during this exercise, were implemented and tested within a prototype after the LIEWA exercise.

CCF primarily targets development of military information systems for interactive collaboration. It introduces the notion of a CCN and supports application development following this paradigm.

From an operational perspective, CCF introduces the notion of CCN allowing an interactive access by any participant to any complex information (master objects and their relationships) that all other participants have made available. It is achieved through the following key concepts: Cell, User Role, Operation/Exercise, Workspace, Domain and Sharable Object.

From a technical perspective, CCF follows current major IT trends such as component-based and N-tiers approaches in order to achieve reusability, scalability and adaptability. The CCF approach can be visualised through a matrix where each row corresponds to a standard CCF layer while each column corresponds to a subsystem of the application. A row encapsulates a technology while a column encapsulates functionality. The intersection of a layer and a subsystem is a piece, or component, that can be change without impacting the other pieces of the system.

There are still risks associated with such a framework development. It comes mainly from the usage of emerging technology. The risk associated with this development is more a matter of cost than a matter of feasibility. It is always the case for complex application development with emerging technologies. Developers mastering these technologies are not available and the project has to plan for a learning curve not negligible. Moreover, there is the cost associated with bad decisions because of limited knowledge of technologies. In addition, technology evolution is another important factor to consider: technologies appear but can also disappear. It has happened during the project and it will continue to happen. However, the matrix approach limits the impact.

Although the applicability of the CCN concept has been demonstrated, additional experimentations are required to ensure that CCF benefits overcome its costs.

7. References

[1] D. Blain, *Mesurer et expliquer l’impact du partage en direct des situations militaires du système ASIP sur la qualité du renseignement*,’ Université du Québec à Montréal, (Doctorate thesis), Montreal, 2001.

[2] M. Gauvin, S., Lapointe, and M. Lizotte, *Tactical Information Fusion Prototype: Evaluation Through an Abbreviated Command Post Exercise*, Defence Research Establishment Valcartier, September 1995, UNCLASSIFIED.

[3] M. Gauvin, and G. Thibault, G., *Tactical Information Fusion Prototype: Participation in the ABCA Northern Lights Command Post Exercise*, Defence Research Establishment Valcartier, January 1996, UNCLASSIFIED.

[4] DMR Consulting Group Inc., *ACTIF Architecture, Functional and Internal Standards*, Defence Research Establishment Valcartier, November 1997, UNCLASSIFIED.

[5] D. C. Schmidt and M. E. Fayad. "Lessons Learned Building Reusable OO Frameworks for Distributed Software", *Communications of the ACM* Vol. 40, No. 10, October 1997.

[6] D. S. Alberts, J. J. Garstka, and F. P. Stein, *Network Centric Warfare: Developing and Leveraging Information Superiority*, 2nd Edition (Revised), DoD C4ISR Cooperative Research Program, Washington, D.C., 1999.

Distribution list

Internal distribution

1-	Director General
3-	Document Library
1-	Mr. M. Lizotte (author)
1-	Mr. G. Turcotte, SH/SoS

External distribution

1-	DRDKIM (PDF file)

DOCUMENT CONTROL DATA

1. ORIGINATOR (name and address) Defence R&D Canada Valcartier 2459 Pie-XI Blvd. North Québec, QC G3J 1X8	2. SECURITY CLASSIFICATION (Including special warning terms if applicable) Unclassified	
3. TITLE (Its classification should be indicated by the appropriate abbreviation (S, C, R or U) A collaborative cellular framework for defence applications (U)		
4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.) Lizotte, M.		
5. DATE OF PUBLICATION (month and year) October 2006	6a. NO. OF PAGES 10	6b .NO. OF REFERENCES 6
7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. Give the inclusive dates when a specific reporting period is covered.) Technical note		
8. SPONSORING ACTIVITY (name and address)		
9a. PROJECT OR GRANT NO. (Please specify whether project or grant)	9b. CONTRACT NO.	
10a. ORIGINATOR'S DOCUMENT NUMBER TN 2004-464	10b. OTHER DOCUMENT NOS N/A	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

- [✓] Unlimited distribution
- [ ] Restricted to contractors in approved countries (specify)
- [ ] Restricted to Canadian contractors (with need-to-know)
- [ ] Restricted to Government (with need-to-know)
- [ ] Restricted to Defense departments
- [ ] Others

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

For many years now, DRDC Valcartier has been developing expertise in building military collaborative software. The new concept presented in this paper, the Collaborative Cellular Framework (CCF), is based on the lessons learned from these projects. It introduces the notion of a Collaborative Cellular Network (CCN) allowing an interactive access by any participant to any complex information that all other participants have made available. This framework concept supports application development following this paradigm. It enables sharing of complex objects e.g. a master object along with its satellite objects and relationships.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

object-oriented framework, defence applications, collaborative work, software architecture