An object-oriented organic architecture for next generation intelligent reconfigurable mobile networks.

George Ribeiro-Justo[1]
Tereska Karran[2]

[1]Cap Gemini Ernst & Young UK
[2]Cavendish School of Computer Science, University of Westminster

An Object-oriented Organic Architecture for Next Generation Intelligent Reconfigurable Mobile Networks

George R. Ribeiro-Justo
Cap Gemini Ernst & Young UK[1]
George.Justo@capgemini.co.uk

Tereska Karran
University of Westminster
karrant@wmin.ac.uk

Abstract

Next generation mobile networks have great potential in providing personalised and efficient quality of service by using re-configurable platforms. The foundation is the concept of software radio where both the mobile terminal and the serving network can be re-configurable. This approach becomes more effective when combined with historic-based prediction strategies that enable the system to learn about application behaviour and predict its resource consumption. We extend that concept by proposing the use of an object-oriented intelligent decision making architecture, which supports general and large-scale applications. The proposed architecture applies the principles of business intelligence and data warehousing, together with the concept of organic viable systems. The architecture is applied to the CAST (Configurable radio with Advanced Software Technology) platform.

1. Introduction

Adapting the application behaviour to resource availability and user goal has become a widespread strategy in mobile computing [17]. The idea of changing the quality of service (or as some authors call it the 'fidelity' of the application [17]) can be very effective in adapting the application resource consumption to the resource availability. This is generally achieved by using history-based predictions of the application behaviour and resource consumption. Most of these approaches, however, are application specific and only work for small applications. We propose a general solution, which can be applied to large applications. We use the concept of business intelligence as the foundation for our solution.

Business intelligence (BI) is a new discipline that aims at supporting decision-makers in large enterprises. It provides tools and techniques that consolidate and transform corporate data into critical information for decision-making. BI tools are designed specifically to access, analyse and report on diverse and disperse historical data. BI, therefore, owes most of its advantages to the quality of data it uses, and makes data warehousing the heart of any good BI solution.

BI systems are usually complex distributed systems that consolidate historical data from different data sources, using different formats and performing complex analytical processing on the historical data. Such complex systems require solid reference architectures in order to simplify their development and management and to provide the best decision-making support. To solve this problem, we have proposed a new distributed object-oriented reference architecture called CODA (Complex Organic Distributed Architecture)

CODA represents a new generation of decision-making system that includes a means of monitoring and controlling objectives to allow the enterprise to evolve dynamically with a certain degree of autonomy. It applies a theoretical foundation for modelling evolutionary enterprises provided by (organic) principles, defined by the Viable System Model [2]. CODA refines and adapts Beer's Viable Systems Model with reference to distributed decision supporting systems. Although originally conceived for defining management principles, the model is further adapted to the control of data flow in a complex decision support system. The result is a flexible, organic and adaptable BI architecture.

Many of the concepts of BI are not new, but have evolved and been refined based on experience gained from early host-based corporate information systems, and more recently, from data warehousing systems. CODA is a distributed intelligent information system, which combines data warehousing, data mining and agent technology.

[1] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied of Cap Gemini Ernst & Young.

0-7695-1300-X/01/$10.00 © 2001 IEEE

CODA separates the intelligent information system into five functionally distinct layers; each supported by a data warehouse component and an intelligent component. Each data warehouse component is separated by filter components, which restructure data into formats suitable for the information processing functions to be performed. This approach is based on the way that organic systems manage complex and adaptive behaviour. The advantages of CODA are that it is intuitively easy to manage, and supports complex evolution by using data intelligently. At the same time, CODA minimises information flow between the system components and provides sufficient component autonomy.

This paper demonstrates the use of CODA as a solution for generating and using historical data to support resource management decisions in reconfigurable mobile networks. The paper presents the main components of the architectures using UML.

2. The Complex Organic Distributed Architecture

An organic model is capable of providing a layered, filtered architecture for managing information in a distributed context. CODA achieves this by refining and adapting Beer's Viable Systems Model [2] with reference to distributed information systems.

2.1. CODA Principles

The Viable Systems Model (VSM) is based on the way a biological organism, such as the human nervous system, processes data in terms of objectives. Incoming data is levelled according to the type of activity performed and filtered so that only the relevant information is presented when decisions are made.

According to the cybernetic model of any viable system, there are five 'necessary and sufficient' subsystems involved in any viable organism and organisation [2]. To be viable a system should therefore be organised according to those levels. We define these in CODA, focusing mainly on data complexity and business functions, as follows:

1. **Operations**: This layer deals with simple linear data, which usually corresponds to typical transaction processing and business operations. The operational data warehouse usually links together data from databases in several locations.
2. **Monitor operations**: In this layer, the data is often dimensional and aggregated. For instance, data is organised by time or group. This layer is responsible for monitoring business operations.
3. **Monitor the monitors**: This layer deals with multidimensional data and provides capability for analysing trend behaviour. At this level, business operations are monitored in terms of external trends.
4. **Control**: This level should be able to "learn" about simple emergent behaviour, trends and forecast and be able to run predictions and simulations automatically.
5. **Command:** This is the highest level, which should be able to deal with any variety not treated by the lower layers [26]. This means to recognise new threats and opportunities.

Figure 1 presents a diagrammatic view of CODA with its principal components and their relationships. The details of the components are described in the following sections. The reader should refer to [12, 19] for more details about CODA.

2.2. CODA Structure

In CODA, information processes (mapped to architectural components) are distributed and leveled. Each component can be upgraded and implemented incrementally. In a fully implemented organic structure, components are semi-independent. They can manage themselves using Critical Success Factors (CSFs) but can also refer to components above. Each component has a part in fulfilling enterprise objectives. The combined interactions of the business processes demonstrate the emergent behavior of the organization. Each component should also have enough information to make the right decisions. Only a subset of the information available to the whole organisation is relevant to each component. The general structure of CODA is illustrated in Figure 1.

CODA corresponds to a typical object-oriented (OO) layered architecture [20, 21], where each layer denotes a level of the VSM. Data sources, which can be data marts, data warehouses and legacy processing, must be attached to the right layer. The criteria to allocate a particular element to a certain layer (level) in the architecture are defined according to their processing type as in the VSM, as described above. Observe that other data warehouse architectures [22] may suggest the separation of data in two levels according to the summarization process; that is, lightly summarized data is a data that is distilled from a low level of detail whilst highly summarized data is compact data. Unlike CODA, however, this leveling does not take into account the user of the data.

A main objective of CODA is to provide an OO distributed BI management architecture. CORBA usually provides the application message layer in CODA. The components of the architecture provide CORBA Application Programming Interfaces (APIs), which allow universal and transparent access. CORBA is a key

technology for interoperable object-oriented distributed systems [23, 24] thereby other components can be easily integrated into CODA. Since CODA is an OO architecture, most components in the architecture correspond to a class or object, or are wrapped to provide an OO interface. For example, a business task is modelled as a method, if it is a simple task, or as a class, if it is a complex task. Interactions between components are usually carried out via the CORBA object request broker [23].

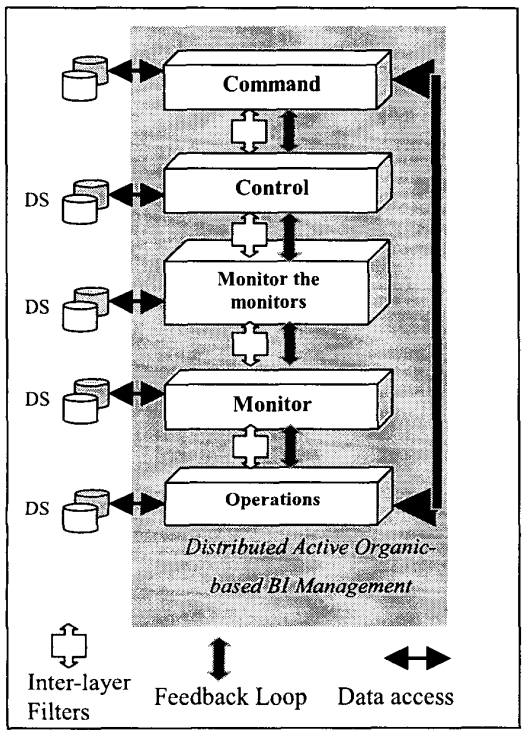

Figure 1. Overview of CODA

Interaction between the layers is achieved by two key elements of CODA, namely, the filters and the feedback loop. As previously stated, each layer is separated by filters, which ensure that only the necessary information reaches it. The feedback loop allows components in a layer to only report unusual failure or success in meeting their CSF to the above layers. This means that tasks in the above layer(s) must take action. The feedback loop corresponds to an *event-based, implicit invocation style* [21], where one layer generates *alerting events*, related to failures or successes, and the above layer registers the task(s) to be invoked.

Each layer is structured in a similar way, providing filtering, feedback and security capabilities, as illustrated in Figure 2. Two types of components usually provide data access in CODA: wrappers and filters. Object wrappers are widely used to integrate legacy or non-CORBA compliant applications. The filters are responsible for providing clean and secure data.

3. Next Generation Reconfigurable Mobile Networks

We use the typical components of a 3G network [3, 16] as the basis for the elements of our model. A mobile equipment (ME), or mobile station (MS), contains a SIM card that uniquely identifies the equipment. One or more MSs can connect to a base station transceiver (BST). Several BSTs together are controlled by a base station controller (BSC). The BTS and BSC together form the base station subsystem. The BST is referred to as the base station (BS). The combined traffic of the MSs in their respective cells is routed through the mobile switch centre (MSC). In addition, connections originating from or terminating in the fixed network are handled by a dedicated gateway MSC. A location area (LA) consists of several cell groups. Each cell group is assigned to a BSC. Each administrative (AR) is made of at least one LA. Each AR is assigned to an MSC.

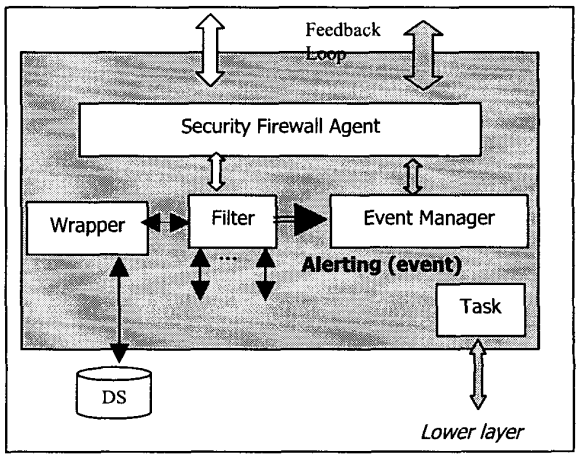

Figure 2. CODA typical layer

Several databases are available for call control and network management: the home location register (HLR), the visited locations register (VLR), the authentication centre (AUC) and the equipment identify register (EIR). The HLR stores permanent data, such as the user profile, and temporary information, such as the user's current location. The HRL is always first queried to determine the user's current location. The VLR is responsible for a group of location areas and stores data about the users who are currently in its area of responsibility. This

includes part of the permanent user data that has been transmitted from the HRL to the VLR for fast access. The AUC generates and stores security-related data such as authentication and encryption keys. More details can be seen in [3,16].

The EC Framework V CAST project [14] is investigating re-configurability in existing and future mobile networks. The CAST model illustrated in this paper is a simplified version of the generic model described above. The main principle of the CAST platform is re-configurability, therefore in order to provide the best service, re-configuration will be used as much as possible.

The CAST project does not establish a strategy for selection of the types of services that will be provided by the system. We assume that these will be derived from the services defined by a typical 3G platform. In addition, it is not an objective of CAST to define novel types of services. Our model should be general enough to deal with general classes of services. We assume, however, that services can be categorised by the amount of resources they require. For instance, voice service requires fewer resources than coloured video-on-demand service. We will focus on re-configurable resources in our model. The idea is to show how quality of service can be improved by managing and reconfiguring the resources available.

The overall UML CAST model is presented in Figure 3. We opted for a simple but general model, which can give us flexibility for extension as the project progresses. Observe that although it is important to include conceptual components of the serving mobile network, the focus is on re-configurability of the whole network in relation to MS and BS.

The aim is to develop a model that contains sufficient information for effective viable operation. In addition, it must keep information to a minimum so that fast responses are possible, which is important in CAST because of the real-time nature of the system.

4. An Organic Architecture for Next Generation Reconfigurable Mobile Networks

4.1. The Main Components of the Architecture

The main components of the proposed architecture are illustrated in Figure 4. We apply the *data access object (DAO) pattern* to model the CODA tasks. Each CODA data source (data warehouse or otherwise) is accessed and controlled by DAOs.

4.2. The Operations Level

The operations level (see Figure 5) includes all data required for effective operation, in this case the operation of the MS (handset) and BST. The data stored in each MS is necessarily limited. However, the BSC stores details of each task activation managed by the BST and MS requesting a service. These details are archived within a specified period. We have selected a time period of three hours, or when 10 service requests have been handled, or after any reconfiguration request, whichever is sooner. This is where up to 1 MB of memory may be needed to record a task activation. We allow for 1MB of data store in the handset to record up to six hours of service requests or two task reconfigurations, since it may not be possible to download the information if the system is busy. Service requests have priority, and are executed before archive requests so a BSC may have up to double the amount of data at very busy times.

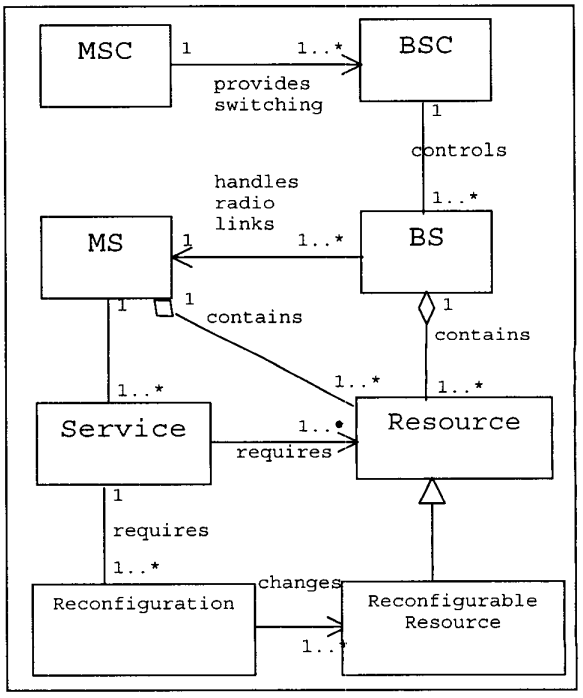

Figure 3. CAST conceptual model in UML

When an MS makes a service request to a BST, the service request is dealt with using the minimum data for an effective execution of the service. However, the CAST system needs to store detailed information about service requests if it is to improve overall system performance. The data collected must include time location and group variables. Therefore, a request for more detailed information from the BSC may follow the execution of the

service request if performance issues prevented a download of all required information at execution time. The request for further information can be normally made when both MS and BST systems are idle.

4.3. Monitoring Level

The BSC (see Figure 6) is able to make some decisions on operational processes by aggregating the operations and resource requests in time bands, speed of execution bandwidth available and bandwidth used etc. It has a small data warehouse. The BST and the MS data stores are archived periodically. In the case of the BSC we have assumed that the size of the data stored on each resource request is up to 1 MB. The data is archived to the MSC after three hours, after 300 service requests or after a global reconfiguration request whichever is sooner.

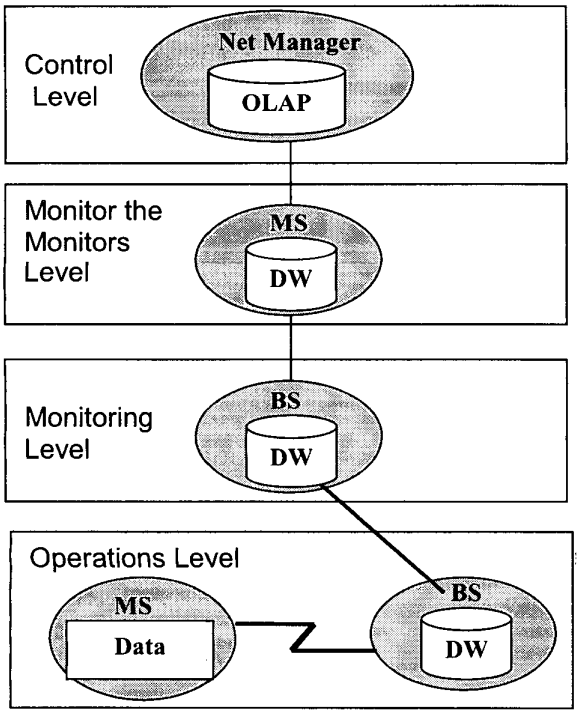

Figure 4. CODA for reconfigurable mobile networks

The BSC is able to handle most reconfiguration requests from the handset and most service requests. However, if CSF ranges for performance are not met (speed, bandwidth and location), then the BSC operations monitor will seek advice from the levels above on which course of action to take.

Data about all service requests (both executed and failed) is collected and stored in the BSC and archived periodically. This is not a large data store, and the collected data is archived and downloaded regularly (after 300 resource requests or every three hours whichever is sooner).

4.4. The Monitor the Monitors Level

At the higher CODA levels the collected data is stored, dimensionalised and analysed. The MSC and Coda Net Manager Levels manage the regularly archived information so that the base station and handset service can be optimised and reconfigured. They also make predictions about new services and service loads, and fine-tune the operations as needed via global reconfiguration. They are able to provide some operational advice if performance success factor ranges are not met.

The MSC (see Figure 7) performs the 'monitor the monitors' function in the CAST intelligent system. This requires a larger warehouse capacity than that of the BSC. It should be large enough to manage a multidimensional database composed of data from 1 - 10 BSCs. It should have sufficient capacity to store one week of data together with a comparison archive of data from one similar week from the same time last year. For example, the same week or one when operating circumstances were similar (for example when a test match series was on). Space and complexity considerations mean that this level should not perform forecasting and trend analysis. If we apply the CODA architecture then forecasting and trend analysis should be performed by the Controlling levels, in our model this is the CODA net manager.

The MSC monitors the BSCs at all times, receiving regular status reports that all is well in those BSCs it is monitoring. The MSC checks all BSC operations for seven days in three-hour time bands using aggregations. Each new day the seventh most distant day is archived and data is sent to the CODA Net Controller. The archived data is organised by time location and group. Based on simple analysis, the MSC is able to make predictions about service needs and reconfigurations. This will be used to optimise the performance of the base stations. Incoming data is compared with the previous hour and previous time band for consistency.

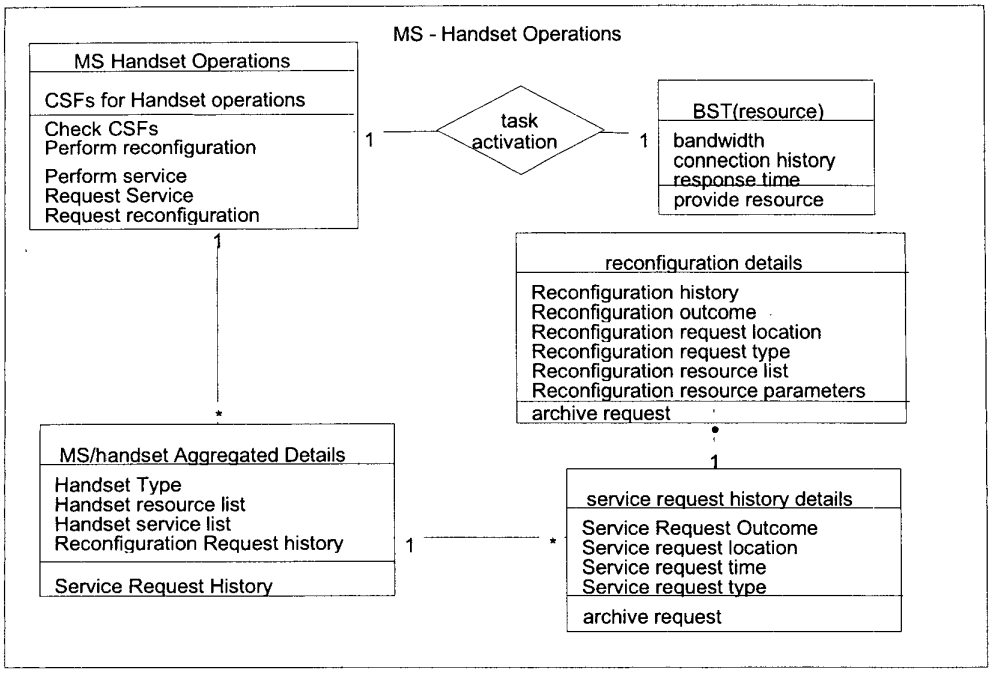

Figure 5. The operations level

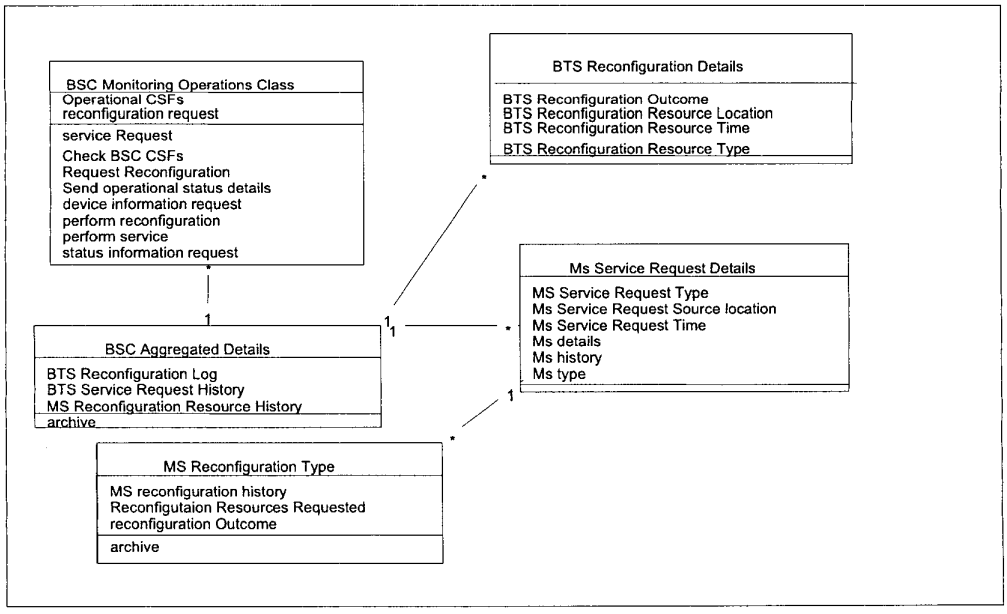

Figure 6. The monitoring level

It is also checked against data for the previous day and a similar day from the last year (this is downloaded from the Coda Net as the MSC data warehouse is limited in scope). The MSC Software contains tasks, which check for rises and falls in the usage. Tasks also check performance status within predefined ranges set by MSC CSFs.

Using OLAP aggregations and slice, dice and pivots operations [22], the MSC is able to optimise the performance of the system using average service-request-rates, and totals from past performance to optimise the current performance of the BTS and MS systems under its control.

The MSC monitors current performance against past performance and then against predicted performance. The data is managed using a grid, which matches expected with actual data. The expected data and actual data are among the CSFs for monitoring performance. If the actual performance falls outside expected tolerances then the CODA Net manager is invoked for immediate assistance.

4.5. The Control Level

The CODA Net manager (Figure 8) is the top level CODA layer implemented on the demonstrator. It performs the controlling function described by CODA. The CODA Net Manager has the same data as the MSC but at a vastly increased capacity. It monitors several MSCs (1 - 10 in the demonstrator) and archives data yearly. This large warehouse has full OLAP database functionality using, time group, location, forecast and version data.

The history data in the warehouse dates 12 months back and three months forward from the current operations. This vast amount of stored data is used to forecast and assess trends and make decisions about performance. The CODA Net manager can reset CSFs in the levels below and interrupt processing if user information (drawn from the user databases) requests it.
The CODA Net manager is able to fine-tune the operations of the mobile system by managing and reconfiguring resources to meet predicted loads. It is able to make some predictions about new services based on analysis of trends, although this would be performed more effectively by a further intelligence layer, which can make more complex trend deductions.

Thus, CODA is able to present a potential solution for the problems posed in managing large networks using intelligent reconfiguration controllers. The CODA system has the advantage of not requiring major reconfiguration of existing hardware, as it is able to act on stored data, which is regularly archived.

4.6. Interfaces to External Components

The best way of defining the interfaces in an architecture is through the concept of service. A service defines a contract between components at an abstract level. The interface of a component can be defined by the services it requires from and provides to other components. The interfaces between CODA components have been specified in the previous sections. Now we have to establish what services the CAST CODA components will provide to and require from components outside the architecture. The overall service architecture is described in Figure 9 where an arrow pointing out denotes a service provided and similarly an arrow pointing in denotes a service required by the component.

Since the reconfiguration takes place at both the MS and BS levels, this is where CAST CODA obtains information about resources status in order to support the application service. This service is referred to as *Resource Services*. The *Resource Services* must provide information to CAST CODA and does not expect any information from CAST CODA. In general, CAST CODA will only request information about the latest application service and the resources used to support that service. CAST CODA may also request information about the availability and status of resources on the MS or BS. The resource services could be implemented by the resource controller or the network management system. This decision, however, does not affect CAST CODA.

The *Reconfiguration Management Services* interact with CAST CODA in two ways. Firstly, it will provide information about the request for an application (reconfiguration) service by the BS or MS. Afterwards, it should also provide information about the result of the request, namely, whether the reconfiguration was successful or not. If the reconfiguration fails, information about the cause of the failure should also be provided. CAST CODA can then request from the resources services the details of the resources status.

CAST CODA will provide intelligent decision-making service to the *Reconfiguration Management Services* at various levels. The reconfiguration management services gives the detail of the application request and CAST CODA, based on its intelligence, will respond whether that service should be provided or not. CODA can also provide details about the decision.

Again, the *Reconfiguration Management Services* should provide feedback to CODA as to whether its recommendation has been accepted or not. This will enable CODA to learn.

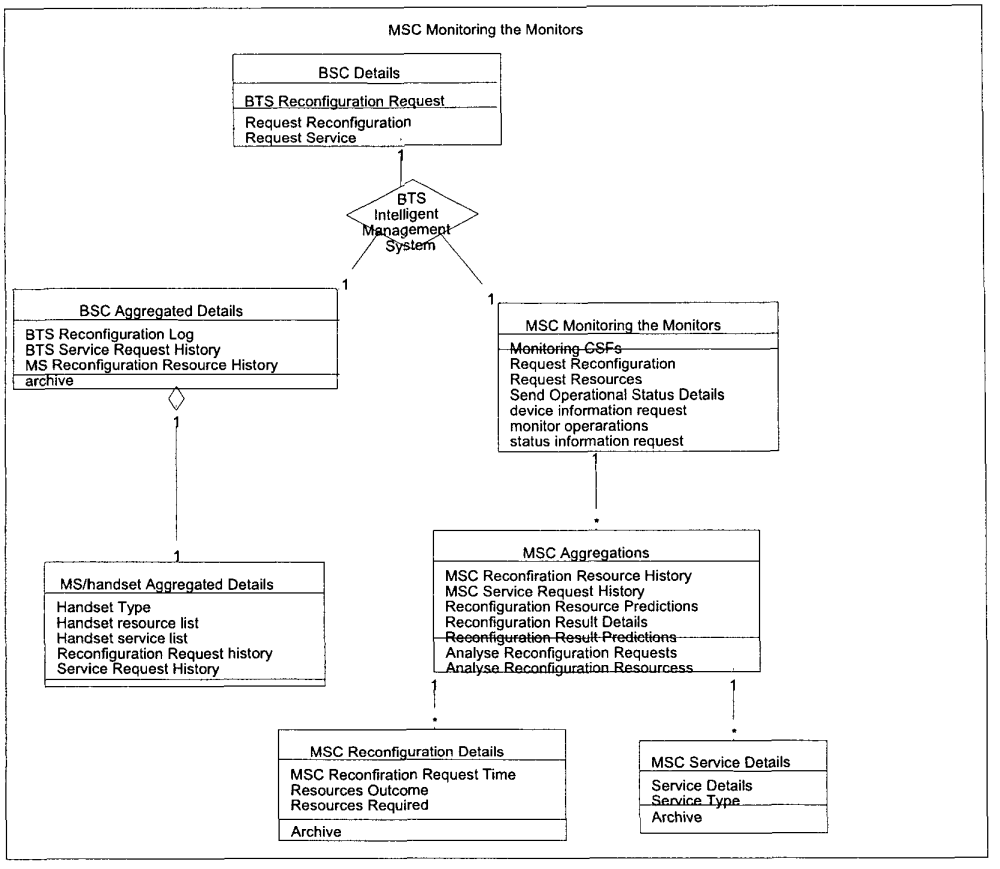

Figure 7. The monitor the monitors level

Since CODA is proactive and it may also autonomously provide advice if it finds that the system requires reconfiguration in order to be viable. This works as a service request to the reconfiguration management service. This type of service should contain a priority, which indicates whether or nor the reconfiguration management services can ignore CODA's request.

Finally, it is also necessary to define the *connectors* between CODA and the external services. In Figure 9, the connectors are denoted by the arrows. A connector is an architectural element that mediates the interaction among components. It is essential in an architecture to specify how the components interact, especially external components. Since CAST is based on distributed object technology, more specifically using Java, it is expected that the components will use Java based distributed object connectors. Possible candidates are Java RMI (Remote Method Invocation) and CORBA. We do not anticipate the use of any other type of connector. In both cases, the services will be defined using an interface definition language. Each service will therefore correspond to object-oriented methods or operations. The main operations of CAST CODA have been defined in the UML models presented in section 4.

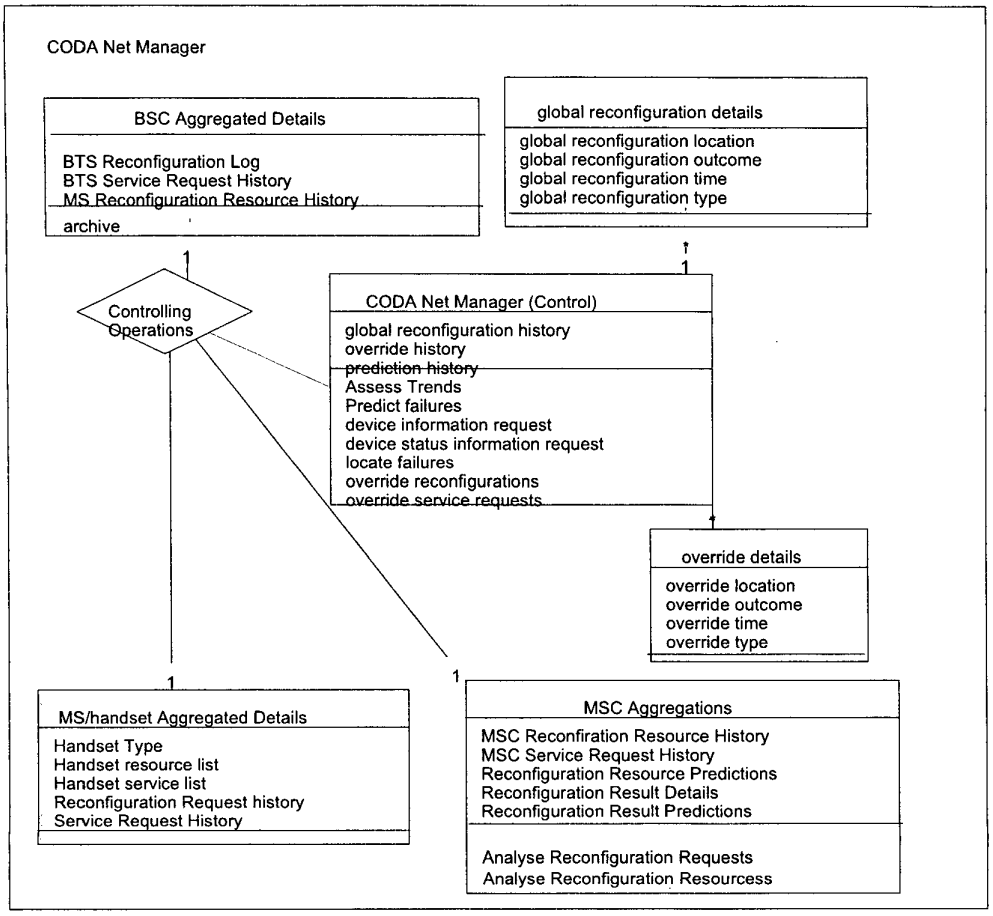

Figure 8. The control level

5. Conclusions and Future Work

The main aim of this paper was to present a distributed object-oriented architecture to support intelligent decisions within next generation mobile networks. Most history-based prediction approaches for mobile application adaptation use simple log techniques and algorithmic complexity analysis. These approaches are application-specific and can only be applied to small applications. The paper has presented a general solution based on CODA that can be applied to large-scale applications.

The proposed architecture is an extension of CODA, an organic BI architecture. The paper has shown how CODA was easily adapted to this type of application. The result has been a flexible and evolving architecture, which employs the VSM principles to provide intelligent decisions for the re-configurable platform. Although the proposed solution has focused on CAST, it can be easily applied to other re-configurable platforms. This emphasizes our claim that CODA is sufficiently flexible to be applied to a large number of applications.

The key challenge in implementing the CAST CODA architecture is performance. Because of its real-time nature, reconfigurable mobile network demands fast responses from the decision support system. We believe that the simplifications applied to the architecture can provide reasonable performance. We have greatly reduced the amount of data being transferred between the CODA layers as well as the window between transfers. We will also make use of compression and caching techniques to reduce performance overheads.

6. Acknowledgement

This document is based on the work carried in the Eu-sponsored collaborative research project CAST (hhtp://www.cast5.freeserve.co.uk). Nevertheless, only the authors are responsible for the views expressed here.

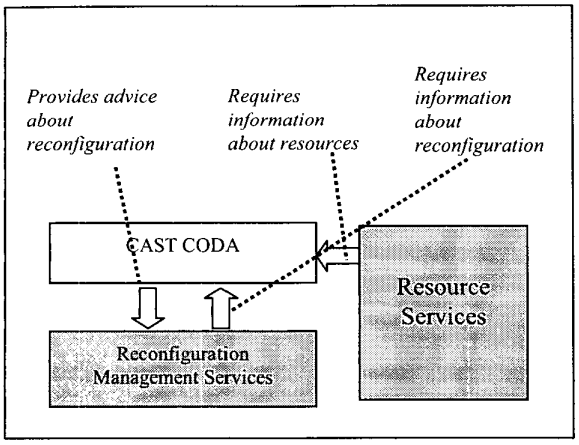

Figure 9. Interface to external systems

7. References

[1] Ashby W.R.. *Introduction to Cybernetics*, Chapman Hall, London, 1965.

[2] Beer, S. The viable system model: Its provenance, development, methodology and pathology. *Journal of Operational Research Society*, 35(1):7-25, 1984.

[3] Bettstetter, C., H. Vögel and J. Eberspächer. GSM Phase 2+ General packet radio Service GPRS: Architecture, Protocol, and Air Interface. *IEEE Communications Surveys*, vol. 2, no. 3, 1999.

[4] Booch, G., I. Jacobson, and J. Rumbaugh. *Th Unified Modeling Language Users Guide.* Addison-Wesley, 1998.

[5] Buzydlowski, J.W., Song, I. Y, and Hassel, L. A Framework for Object-Oriented On-Line Analytic Processing, *DOLAP'98: ACM First International Workshop on Data Warehousing and OLAP*, Washington, USA, 1998.

[6] Eriksson. H., and M. Penker. *Business Modeling with UML: Business Patterns at Work.* Wiley 2000.

[7] Hofmeister, C., R. Nord and D. Soni. *Applied Software Architecture*, Addison-Wesley, 2000.

[8] Espejo, R., Schuhmsnn, W. and Schwaninger, M. Organizational *Transformation and Learning, Cybernetic Approach to Management.* Wiley, 1996.

[9] Garlan, D. and J. P. Sousa. Documenting Software Architectrures: Recommendations for Industrial Practice, *Technical Report CMU-CS-00-169*, School of Computer Science, Carnegie Mellon University, USA, 2000.

[10] Hunt, J. *Java For Practitioners*, Springer, 1999.

[11] Karran, T., G. Ribeiro Justo and J. Zemerly: An Organic Architecture for Component Evolution in Decision Systems, 4th World Conference on Systemics, Cybernetics and Informatics (SCI 2000), Orlando, USA, July, IIIS, 2000.

[12] Karran, T., G. R. Ribeiro Justo and J. Zemely: An Organic Knowledge Information Management Architecture, International Conference on Intelligent Systems and Control, Santa Barbara, California, USA, 1999.

[13] Kruchten, P. *The Rational Unified Process: An Introduction*, Addison-Wesley, 1998.

[14] Madani, K., B. Bosch, B. Honary, G. Ribeiro-Justo, et al. Configurable radio with Advances Software technology (CAST) – Initial Concepts, ISR Mobile Communications Summit 2000, pp. 139-144.

[15] Megaache, S., T. Karran and G. R. Ribeiro Justo: A Role-based Security Architecture for Business Intelligence, TOOLS USA 2000: 34th International Conference on Distributed Objects, Santa Barbara, California July 30 - August 3, IEEE Press, 2000.

[16] Morawek, R. and H. Öczelik. UMTS: basic Network Architecture, http://www.unet.univie.ac.at, 200.

[17] Narayanan, D., J. Flinn and M. Satyanarayana. Using History to Improve Mobile Application Adaptation, *3rd IEEE Workshop on Mobile Computing System and Applications*, December 2000.

[18] Nyanchama M. and S. Osborne. The Role Graph Model and Conflict of Interest. *ACM Transactions on Information Security Systems*, Vol. 12(1):3-33, 1999.

[19] Ribeiro Justo, G. R., T. Karran and J. Zemely, An Organic Architecture for Distributed Knowledge Management. In Industrial Knowledge Management – A Micro Level Approach, R. Roy Ed., Springer, 2000. ISBN 1-85233-339-1.

[20] Ribeiro Justo, G. R. and P. Cunha. An Architectural Application Framework for Evolving Distribute Systems. *Journal of Systems Architecture: Special Issue on New Trends in Programming and Execution Model for Parallel Architectures, Heterogeneous Distributed Systems and Mobile Computing*, Vol. 45:1375-1384, 1999.

[21] Shaw, M. and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline,* Prentice Hall, 1996.

[22] Singh H. S. *Data Warehousing: Concepts, Technologies, Implementation and Management*, PTR, 1998.

[23] Umar, A. *Object Oriented Client/Server Internet Environments.* Prentice Hall, 1997.

[24] Zahavi, R. *Enterprise Application Integration with CORBA: Component and Web-Based Solutions*, OMG Press, 2000.

[25] Zuidweg, H., M. Campolargo, J. Delagado and A. Mullery Eds. Intelligence in Services and Networks, 6th International Conference on Intelligence and Services in Networks, LNCS 1597, Springer, 1999.

[26] Waelchi, F. The VSM and Ashby's Law as Illuminants of historical Management Thought, *In The Viable Systems Model: Interpretations and Applications of Stafford Beer's VSM* Eds. R. Espejo and R. Harnden, 1996.

[27] White, C. J. The IBM Business Intelligence Software Solution, DataBase Associates, Version 4, May 2000.