Improving Alarm Interoperability with External Systems for a Wearable Command Unit Using Service-Oriented Architecture

IVAN GLAUSER

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2007-04

Improving Alarm Interoperability with External Systems for a Wearable Command Unit Using Service-Oriented Architecture

Ivan Glauser
glauser@kth.se
Royal Institute of Technology
Stockholm, Sweden

A Master of Science thesis project performed at
Saab Security Systems, Järfälla

Examiner: Gerald Q. Maguire Jr., ICT/KTH
Industry Advisor: Fredrik Öström, Saab Security Systems

This version was last updated 15 February 2007

Department of Communication Systems (CoS),
Royal Institute of Technology (KTH), Stockholm, Sweden.

Abstract

This thesis investigates different aspects of implementing a Service-Oriented Architecture (SOA) for an alarm and crisis management system called Wearable Command Unit (WCU) developed by Saab Security Systems.

The WCU system must be able to integrate easily with external systems in order to move into new markets and survive as a product. The focus of this report is a general solution for communicating alarm information from external systems to the WCU. A given requirement was that the solution must be based on SOA. Therefore, the concept of SOA is investigated and its applicability is considered for the WCU architecture.

A design proposal based on a combination of open information and communication technologies was made to show how WCU may use SOA to receive alarm information from external systems. The design proposal was evaluated by a load test as well as comparing its conformance to SOA. The load test showed that the proposed solution can process incoming messages at a rate of 2 ms per message when client and server are run on the same machine. The result of the comparison showed that the WCU can, with small modifications, apply a SOA.

While this thesis has only investigated the use of SOA in the context of alarm information, there is a clear trend toward integrating information for diverse systems to enable users to have better quality information. Providing first responders with the information that they need, when and where they need it can enable them to save lives, save property, and reduce the risk to the public of incidents.

An important result from this thesis is the observation that a system that needs to integrate with many distinct systems can be better prepared if made SOA conformant. This requires the system to have an interface towards other systems based on platform independent protocols. Systems such as the WCU, which are based on Windows Communication Foundation (WCF), can easily add such an interface by configuring WCF in an appropriate way.

Key Words: SOA, .NET, WCF, Web Services, alarm system, XML.

Sammanfattning

Detta examensarbete undersöker olika möjligheter att implementera *Service-Oriented Architecture* (SOA) för ett larm- och krishanteringssystem kallat *Wearable Command Unit* (WCU) utvecklat av Saab Security Systems.

För att kunna nå nya marknader och utvecklas som produkt, är det viktigt att WCU-systemet på ett enkelt sätt kan integreras med externa system. Detta examensarbete fokuserar på att ta fram en generell lösning för att kommunicera larminformation från externa system till WCU. Ett förbestämt krav var att lösningen måste vara baserad på SOA. Begreppet SOA undersöks och dess tillämpningsbarhet för WCU undersöks.

Ett designförslag baserat på en kombination av öppna informations- och kommunikations-teknologier gjordes för att visa hur WCU kan använda SOA för att ta emot larminformation från externa system. Designförslaget utvärderades genom ett belastningstest, samt genom att jämföra dess konformitet med SOA. Belastningstestet visade att designförslaget kan processa inkommande larm i en hastighet av 2 ms per meddelande när klienten och servern körs på samma maskin. Resultatet av jämförelsen visade att WCU kan, med små modifieringar, implementera en SOA.

Detta examensarbete har endast undersökt användandet av SOA vad gäller larminformation, men det finns även en klar tendens mot att integrera annan information ifrån olika system för att på så sätt ge användare av systemet kvalitativ information. Genom att ge framskjutna enheter lämplig information, när och där de behöver det, kan de bli bättre förberedda på att rädda liv och egendom, och samtidigt minska olycksrisken för allmänheten.

Ett viktigt resultat från detta examensarbete är iakttagelsen att ett system som behöver integreras med många andra olika system kan bli bättre förberett genom att göra det SOA-baserat. För att ett system ska vara SOA-baserat krävs att det har ett gränssnitt baserat på plattformsoberoende protokoll mot andra system. System som WCU, som är baserade på *Windows Communication Foundation* (WCF), kan med lätthet lägga till ett sådant gränssnitt genom lämplig konfigurering av WCF.

Acknowledgements

I would first of all like to thank my examiner and supervisor professor Gerald Q. Maguire Jr. for his valuable guidance, encouragement and assistance. I would also like to thank the WCU design team at Saab Security Systems for providing me with material and information about the WCU system. Special thanks go out to Fredrik Öström for giving me valuable guidance in my work and a continuous flow of feedback, and David Andersen for giving me feedback and comments on the report. I would also like to thank my opponent Mikael Corp for reviewing my work and providing me with valuable comments. Finally I would like to thank my friend Patrick Carroll for proofreading parts of the report.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

ISP	Internet Service Provider
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security
VPN	Virtual Private Network
WCF	Windows Communication Foundation
WCU	Wearable Command Unit

Part I – Introduction & Method

1 Introduction

This thesis project was done at Saab Security Systems, Järfälla, as a part of a Master of Science degree in Information and Communication Technology.

This thesis project utilizes a combination of information and communication technologies to solve the problem of *integrating* information from *external* sources for presentation to fixed and mobile users.

1.1 Background Information

The Wearable Command Unit (WCU) is a system developed by Saab Security Systems for communication and distribution of information. This system distributes information to enable common situation awareness and includes built-in, customizable alarm and case management as well as reporting tools. Mobile and stationary users receive position and status updates which are displayed as a dynamic map. Each user can easily initiate verbal and textual communication with other users [18]. There is also support for Automatic Vehicle Location (AVL) and video. The system is sold to various customers with different needs, and the number of customers is constantly growing.

As the WCU system has grown, several problems with the core architecture have emerged. Difficulties integrating with other systems, as well as difficulties to extend it with new functionality have proven hard to overcome. Because of this, work with a second generation[1] of the system has already begun. The new version aims to improve the core architecture in order to make the system more stable as well as more adaptable to future requirements.

Since the system is still under development several questions in the design process remain unsolved. One such question is how to enable the system to better integrate with other systems; are there any standard solutions that can be used?

Different types of information may need to be interchanged depending on what type of system the WCU is to integrate with. Examples of such information are video, telemedicine

[1] For the rest of this paper the term WCU refers to the second generation of WCU (version 2.0) unless stated otherwise

(medical status and medical advice), sensor data (for detection, identification, and tracking), positions (personnel, vehicles, and assets), unit status, and maps [22].

This thesis project has focused on alarm information and how WCU may exchange such information with external systems. Alarm information refers to the description of alerts and warnings in the public safety domain. This also includes alerts and warnings automatically generated by sensors. The proposed design described in this thesis may, with small modifications also be used for exchanging other types of information. However, how this is to be done is not presented in this paper as it is outside the scope of this work.

1.2 Problem Specification

A property desired in the new version of the WCU is that it should be based on Service-Oriented Architecture (SOA). This need emerged on one hand for marketing reasons, but also because it would improve the extensibility of the system by facilitating integration with other systems. The WCU is based on Microsoft's Windows Communication Foundation (WCF), which itself implements the concept of services. However, the definition of SOA is vague (see chapter 3), thus an investigation was needed in order to determine whether or how the WCU actually relates to SOA. As previously stated in section 1.1, the focus of this investigation was primarily how to communicate alarm information.

In the earlier version of the WCU (version 1.2) there is a solution for communicating alarm information between the WCU and an external system (see section 5.1). However, this earlier system implemented a problem specific solution that proved to be difficult to integrate with other systems. In addition, we expect that the need for a more general solution for WCU 2.0 will emerge as the system expands into new markets. The new solution should be based on SOA in order to conform to the requirements explained in the previous paragraph.

Another question is how the different systems may achieve a common understanding of the alarm information sent between them? How should data be represented in order for both parties to understand each other properly? These questions are of great importance in order to avoid misinterpretation of the information, as well as avoiding the need for problem specific solutions in the future.

1.3 Target Group

This thesis is aimed at those involved in SOA adoption, such as companies, vendors, consultants, and researchers. Those involved in public safety systems may also find the content of this thesis valuable. The final results should guide Saab Security Systems and other companies and organizations in further investigation of how communication of alarm information could be implemented using a SOA. However, the main focus is supporting Saab Security Systems in their continuing development of the WCU system.

1.4 Thesis Outline

Table 1:1 presents an overview of this thesis. Part I includes an introduction (this chapter) and a description of the method that was used during the work of this thesis. Part II investigates the concept of SOA and WCU, describes related work, and how the WCU alarm functionality is related to SOA. Part III presents a proposed design, including a description of the architecture and the software used. In Part IV the evaluation of the proposed design is presented. Part V presents conclusions and suggestions of future work.

Table 1:1 – Thesis outline

Part I: Introduction & Method

Chapter 1 gives an introduction and presents the background to this thesis.

Chapter 2 describes the method that was used during the work of this thesis.

Part II: Initial Survey

Chapter 3 presents the concept and the metrics used for evaluation of SOA.

Chapter 4 gives an overview of the WCU system.

Chapter 5 presents related work and systems.

Chapter 6 describes how the WCU alarm functionality is related to SOA.

Part III: Proposed Design

Chapter 7 gives an architectural overview of the proposed design.

Chapter 8 describes how WCF was configured for the proposed design.

Chapter 9 describes alarm data models.

Part IV: Evaluation

Chapter 10 presents the evaluation of the proposed design.

Part V: Conclusions & Future Work

Chapter 11 concludes and summarizes this report.

Chapter 12 suggests future work.

References & Appendices

1.5 Thesis Structure

Each arrow in Figure 1:1 represents a connection between chapters. Chapter 1 and chapter 2 give an overview over the problem area and the method used in this thesis. The information presented in chapters 3 and 4 is essential for chapter 5. The evaluation made in chapter 6 is based on all the previous chapters. Chapter 7 gives an architectural overview of the proposed design and should be read before chapter 8 and chapter 9. Part III is essential in order to fully understand the evaluation in chapter 10. Finally, the conclusions presented in chapter 11 are recommended to read before chapter 12, where suggestions for future work are presented.

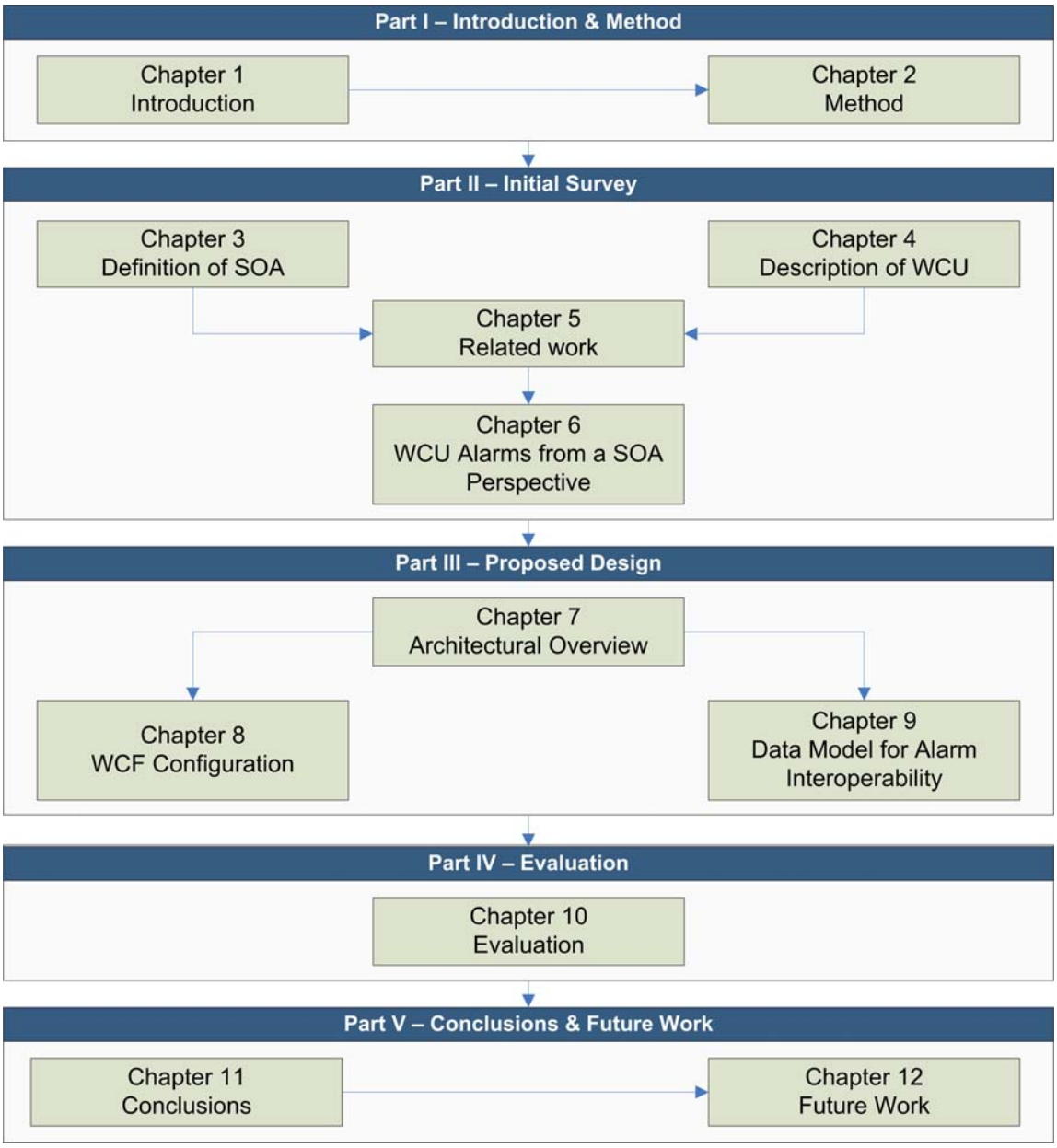

Figure 1:1 – Report structure

2 Method

This thesis project was structured in five phases as follows.

2.1 Initial Phase

The main focus of the initial phase was to gain a deeper insight into the concept SOA, as well as to understand the WCU's architecture.

The concept SOA was investigated and the metrics that were used for the comparison with WCU were defined. This step required extensive study of literature as well as an examination of existing definitions of SOA.

The second part of this phase consisted of studying and describing the WCU system. Since this thesis main focus is on the alarm functionality of the WCU this is described in greater detail (see section 4.3 and section 4.4). Another important part was to understand the communication taking place inside the WCU system. This helped to determine if the proposed design should concentrate on the central WCU Server or upon the WCU clients. One external system, the SOS InfoServer has previously been integrated with the WCU. How this connection was made was also studied in this phase.

2.2 Related Work Phase

The related work phase concentrated on identifying similar existing systems. Both alarm systems, systems in the public safety domain, and general SOA-based systems were investigated. Also general ideas and documentation about how to use alarm services in public safety were studied. This included identification of alarm service vendors and service consumers, as well as how alarms are handled in general.

2.3 Comparison Phase

This phase consisted of making a comparison between the WCU system and the definition of SOA made in the initial phase. A comparison was made considering the internal communication of the WCU as well as with the existing solution for connecting the WCU to the external CoordCom system [17]. Suggestions of how to improve the conformance to the SOA definition were made based upon these two comparisons.

2.4 Design Process Phase

The design process phase consisted of two steps:

The first step was to study the software to be used for implementation of the design proposal. This included Microsoft's Windows Communication Foundation (WCF) as well as the Microsoft BizTalk Server. The analysis of Microsoft's BizTalk Server has been moved to

Appendix D. This is after a decision was made by the author of this thesis to exclude it from the proposed design. This decision was based upon the result of the analysis.

Another component of this step included identifying existing standard protocols and specifications concerning exchanging public safety information between different systems. Of particular interest were standards for describing alarms in public safety.

The second step consisted of creating a design proposal for how the WCU can improve its interoperability with other systems, in particular presenting a solution for making the integration with other alarm systems less problem specific in the future. A prototype of the proposed design was then implemented using the software studied in the first step of this phase.

2.5 Evaluation Phase

The evaluation phase consisted of a conformance comparison between the proposed design and the SOA definition given in the initial phase. Additional aspects in the proposed design; such as scalability, security, and performance was also evaluated. The performance test was mainly to get a rough estimation of how fast alarm messages could be received and processed. The result showed that the proposed design will not constitute a bottleneck in the system. However, a more accurate research with real life alarm statistics should be done before deciding for sure that the design will hold for the desired performance constraints.

Part II – Initial Survey

3 Definition of SOA

This chapter gives an overview of SOA and the key concepts associated with it. Although SOA can be applied to a variety of architectural concepts and domains, this thesis will only consider SOA from a software perspective.

3.1 Introduction to SOA

Service-Oriented Architecture, commonly abbreviated SOA, is a concept that has gained extensive attention in software development communities in recent years. It has been applied, and claimed to be applied, in a broad area of applications and domains. This has lead to a variety of definitions.

To give a generalized definition of SOA, one could say that it is a paradigm or framework for organizing and exchanging information using services. In contrast to a classical architecture that focuses on classes, data, and process models the center of attention is services, their interfaces, and the interaction between these services.

The problem of integrating divergent distributed systems is not new. Both DCOM[2] and CORBA[3] are two of the first technologies tackling this problem. However, although they also wanted to achieve platform independence, they targeted application **objects** rather than **services** and **processes** as SOA does. In an object-oriented paradigm, data is tightly bounded to its processing while in a service-oriented paradigm data and its processing is separated.

Because SOA has received so much attention recently it has become even more difficult to explain or define. Because no general definition exists, the diversity of areas in which it is applicable has grown. Recently an attempt to stop this process of an escalating number of diverse definitions was made by the OASIS consortium. They have created a SOA Reference Model that seeks to unify SOA by introducing a common semantics into an abstract framework for implementing SOA. Important parts and conclusions from this model are presented in section 3.3.

The reminder of this chapter is as follows. Section 3.2 describes three existing SOA definitions. Section 3.3 describes the OASIS SOA Reference Model. Section 3.4 focuses on the key concepts of SOA. Section 3.5 describes when to use SOA. Section 3.6 discusses the

[2] Distributed Component Object Model, introduced by Microsoft in 1996
[3] Common Object Request Broker Architecture, introduced by OMG in 1990

benefits of SOA, and in Section 3.7 the SOA metrics that will be used in this thesis are defined and presented.

3.2 Existing SOA Definitions

The number of SOA definitions is nearly as many as its implementations. A range of participants from large consortiums to enterprises and individuals have made their contribution. Most definitions have much in common, but use different words or concepts for what could be seen as the same underlying content. Among the more established are the definitions given by the standardization organizations of OMG, W3C, and OpenGroup that are presented in this section[4]. An interpretation of the content and meaning is made for each definition.

3.2.1 Object Management Group[5] (OMG) SOA Definition

The definition by OMG [36] in Figure 3:1 states that a SOA must contain *providers* and *consumers* of *services*. The first bullet says that participants in the SOA system must *not depend on each other*, e.g. a failure in one part should not affect other parts of the system. There should also be no or little technological dependency between participants. The contracts set in the system have to be followed in order to participate, and to allow for "*a variety of technologies to be used*", standard protocols have to be used in the communication interfaces between the participants.

"Service Oriented Architecture is an architectural style for a community of providers and consumers of services to achieve mutual value, that:

- *Allows participants in the communities to work together with minimal co-dependence or technology dependence*
- *Specifies the contracts to which organizations, people and technologies must adhere in order to participate in the community*
- *Provides for business value and business processes to be realized by the community*
- *Allows for a variety of technologies to be used to facilitate interactions within the community"*

Figure 3:1 – Object Management Group SOA definition [36]

3.2.2 World Wide Web Consortium[6] (W3C) SOA Definition

The W3C definition [37] in Figure 3:2 states that a SOA must contain *services*, e.g. programs, processes, etc. that are described in "*terms of what it does*". The internal implementation of services should be hidden from external users, and the interaction with services is provided

[4] For more definitions of SOA see Appendix A

[5] An international, open membership, not-for-profit computer industry consortium

[6] An international consortium with over 500 members: leading industries, research and development institutes, standardization organizations, and governments. W3C is working to develop protocols and guidelines that ensure long-term growth for the Web.

in terms of *message exchange*. The semantics of a service and a *description* of all information necessary to interact with a service must be described in a way *interpretable by machines*. To be independent of underlying platform and technology, messages and interfaces between participating parts must be based on *standardized format* (protocols).

"A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties:

- *Logical view: The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.*
- *Message orientation: The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.*
- *Description orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.*
- *Granularity: Services tend to use a small number of operations with relatively large and complex messages.*
- *Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.*
- *Platform neutral: Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint."*

Figure 3:2 – World Wide Web Consortium SOA definition [37]

3.2.3 OpenGroup[7] SOA Definition

The OpenGroup's SOA definition [38] in Figure 3:3 states that a service must be *self-contained*, e.g. it must constitute "a complete and independent unit in and of itself" [45]. How a service is implemented should be *hidden from the outside*. Only what is necessary to interact with it should be available to its users. The *interface* of a service, the *policies* and *rules* of interacting with it, and what the *outcome of using it* will be, must be *described* to other parts. To

[7] A vendor- and technology-neutral consortium, whose vision is to "enable access to integrated information within and between enterprises based on open standards and global interoperability"

be independent of underlying infrastructure and implement "location transparency", *open standard protocols in the interfaces* must be used.

"Service-Oriented Architecture (SOA) is an architectural style that supports service orientation.

Service orientation is a way of thinking in terms of services and service-based development and the outcomes of services.

A service:
Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit; provide weather data, consolidate drilling reports)

- *Is self-contained*
- *May be composed of other services*
- *Is a "black box" to consumers of the service*

An architectural style is the combination of distinctive features in which architecture is performed or expressed.

The SOA architectural style has the following distinctive features:

- *It is based on the design of the services – which mirror real-world business activities – comprising the enterprise (or inter-enterprise) business processes.*
- *Service representation utilizes business descriptions to provide context (i.e., business process, goal, rule, policy, service interface, and service component) and implements services using service orchestration.*
- *It places unique requirements on the infrastructure – it is recommended that implementations use open standards to realize interoperability and location transparency.*
- *Implementations are environment-specific – they are constrained or enabled by context and must be described within that context.*
- *It requires strong governance of service representation and implementation.*
- *It requires a "Litmus Test", which determines a "good service"."*

Figure 3:3 – Open Group SOA definition [38]

3.3 The OASIS SOA Reference Model

The Organization for the Advancement of Structured Information Standards[8] (OASIS) consortium has created a SOA Reference Model [1] that strives to preserve a common understanding of SOA. It is an attempt to unify important concepts of existing SOA

[8] A non-profit, international consortium founded in 1993 that drives the development, convergence, and adoption of e-business standards

implementations and guide architects in implementing SOA. At the time of writing this is an official OASIS Committee Specification, one level below a full OASIS standard.

The idea behind the SOA Reference Model is to use it as an abstract framework to build concrete SOA implementations. The model also contains Conformance Guidelines to help architects and others to decide whether their work is conformant.

3.3.1 Conformance Guidelines

Figure 3:4 shows the conformance guidelines as presented in [1]:

"Any design for a system that adopts the SOA approach will

- *Have entries that can be identified as services as defined by this Reference Model*[9]
- *Be able to identify how visibility is established between service providers and consumers*
- *Be able to identify how interaction is mediated*
- *Be able to identify the effect of using services is understood*
- *Have descriptions associated with services*
- *Be able to identify the execution context required to support interaction*
- *Be able to identify how policies are handled and how contracts may be modeled and enforced."*

Figure 3:4 – OASIS SOA conformance guidelines [1]

Each one of the statements in Figure 3:4 are explained below:

The first statement; *"have entries that can be identified as services as defined by this Reference Model",* indicates that a SOA approach will have services. In [1] a service is described as "the means by which the needs of a consumer are brought together with the capabilities of a provider".

The second statement indicates that a SOA approach will *"be able to identify how visibility is established between service providers and consumers"*. According to [1] visibility is satisfied when participating parts can interact with each other. For interaction to be possible, both parties have to be reachable, be aware of each other, and be willing to interact.

The third statement indicates that a SOA approach will *"be able to identify how interaction is mediated"*. How interaction is mediated requires that information of how to interact with a service can be found by any participant in the system. This requires that information about how to connect to and use a service must be described in a way interpretable by its consumers.

The effect of using a service has to be described in a way that can be found and interpreted by its consumers. This is the content of the fourth statement: *"be able to identify the effect of using services is understood"*.

[9] See [1] Section 3.1

The next statement indicates that a SOA approach will *"have descriptions associated with services"*. Service description is necessary in order for participants to understand what a service is and what will be the result of invoking it. The description must be described with semantics interpretable by all participants in the system according to [1].

The next statement in Figure 3:4 is *"Be able to identify the execution context required to support interaction"*. The execution context is described in [1] as an agreement of protocols, semantics, policies and other conditions that describe how a service can and may be used by what participants. It can be a temporary connection or a well-defined coordination that can be reused. Different instances of the same service are distinguished by their execution context. The execution context also involves the interpretation of data, e.g. a particular string has a particular meaning in a service interaction in a particular execution context.

The last statement indicates that a SOA approach will *"Be able to identify how policies are handled and how contracts may be modeled and enforced"*. Policies are described in [1] as the constraints or conditions on the use, deployment or description of a service as defined by any participant. This includes aspects such as security, privacy, manageability, Quality of Service and so on. A contract on the other hand is defined as the agreement by two or more parties, not necessary arrived at by a mechanism that is a part of an SOA. Both policies and contracts may or may not be presented in a form that permits automated interpretation.

3.4 Key Concepts of SOA

The following key concepts are the result of an analysis of selected definitions as well as a variety of publications and other sources. One important source was the OASIS SOA Reference Model [1]. The terms and designations used could be exchanged with others as long as their significance is preserved. The concepts presented here should be seen in relation to the work presented in this thesis and **not** as a universal description of what SOA must or should contain.

3.4.1 Essential Components of a SOA

Every SOA requires the existence of a service, a service provider, a service consumer, a service registry, and a service description. How these components are described or implemented vary, but the main idea is that services are implementations of functions, procedures, or similar and that providers and consumers interact with each other by using these implementations. The service description is needed in order for participants to dynamically find and bind to services that have been published in the service registry.

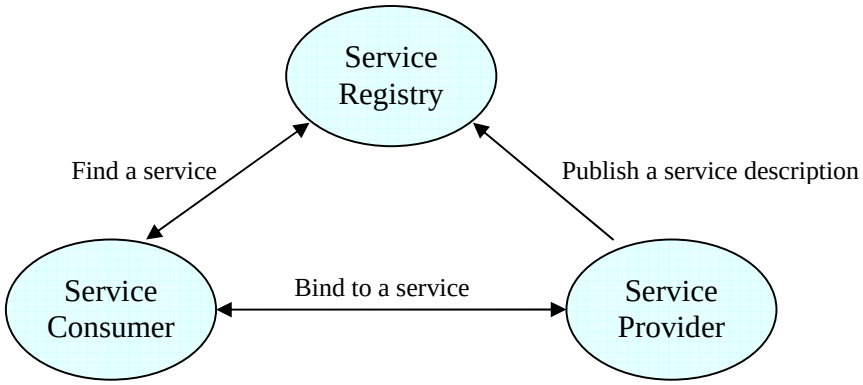

Figure 3:5 – Conceptual model of Service-Oriented Architecture

A service provider publishes (publish) a service to the service registry. A service consumer sends a request (find) and searches the service registry for a service it needs by providing a keyword or a service name. Finally, the service consumer receives the location of the service and connects (binds) to the service provider. [2], [13], [23].

Services
A service is an abstract concept that can be defined in a numerous ways [3], [4]: one general definition applicable to this thesis is "a unit of work to be performed on behalf of a participating party." Functions, procedures, and applications can all be seen as services or a set of services [23].

Service provider and service consumer
A service provider can be seen as an autonomous participant in the system that implements a service and makes it available. A service consumer can be seen as a participant in the system by consuming a service. Whether a participant has the role of a provider or a consumer depends only on the actions performed by the participant at the moment of interaction [12].

Service registry (broker)
A service registry is needed for participants to know where to look for services and service descriptions. The registry works as a "matchmaker between service requestor and service provider" [23]. It stores an association between a name and a description, and the address and location details for a named service [5].

Service description (policies and contracts)
A service description is needed in order to regulate the access and security of services as well as under what conditions interaction with a service can be done. It is essential since it describes what a service will do[10], what data must be passed to the service, and what data is to be expected in return. Policies and contracts [1] are also a part of the service description. The service description should be described in a generally available way and be accessible to all parties.

[10] E.g. what the result will be of invoking the service (as the internal implementation of the service should be hidden from outside)

3.4.2 Essential Properties of a SOA

The key issue in the success of SOA is its independence from the underlying network architecture and its ability to integrate other systems. All service interfaces, and hence implicitly the services, must be universally available to all participants in the system[11]. By agreeing upon a common message and data model, services can interconnect independently of the underlying technology and internal service implementation. Services should be self contained, that is they should always provide the same functionality independent of other services [6]. This allows for a stable and flexible system where a component failure does not necessarily mean that another parts of the system fail.

3.4.3 What Makes SOA Different From Traditional Software?

In [7], three fundamental differences that SOA possesses in contrast to traditional software are described: "Standard-based Interoperability", "Dynamic Composition via Discovery", and "Dynamic Governance and Orchestration". Standard-based Interoperability refers to the use of standard protocols in the interfaces and communication. Dynamic Composition via Discovery refers to the fact that composition and discovery of services can be carried out at runtime. Dynamic Governance and Orchestration refers to the possibility of dynamic scheduling of the execution of services by the use of policies and coordination controllers.

Traditional systems do not have self-describing functionality as they lack the service description provided in SOA. The only description in traditional systems is normally information contained as comments in the code, and these are generally not machine-interpretable.

3.5 When to Use SOA

It should be much easier for a service requestor to understand the contract of a service than to implement the service itself. This is an important aspect since the service requestor does not have to care about the implementation of the service [8].

Service implementation within applications is often not as beneficial as in and between larger systems (e.g. enterprises) [13]. This comes about naturally since the problems to be solved in small scale systems and applications usually can be solved in a more cost effective way without using services. That is, the strength of SOA comes in **application** integration.

3.6 Benefits of Using SOA

Generally there are two main benefits of implementing a SOA. They are based on the fact that SOA uses open standard protocols for exchanging data, and that processing and data are separated. This leads to a system which is highly dynamic in several aspects.

Most stand-alone systems have a problem integrating with other systems. Specialized code has to be written to transform and manipulate data in order to transfer the data between

[11] As long as this does not contradict any policy in the system

applications. The use of standard protocols for information exchange and data makes integration less troublesome.

There is no need to rewrite entire applications to implement SOA. An existing application can be used as a service by providing a SOA-based interface on top of it. SOA-based systems are also highly extensible, that is, allow for new services, changes to services, or new versions of services during runtime (without interfering with the rest of the system). This is achieved by using policies and versioning to distinguish execution context.

Once a service has been described and published, it can be accessed and interacted with by all participants in the system that adhere to the policy associated with the service.

SOA can take the advantage of data flow computation and context-aware [9] systems to allow for automation of tasks and interactions between participants and services. Thus services can be dynamically composed based upon the user's current context and services need only be invoked when there is data for them to process.

The possibility of switching between redundant services at runtime makes the system more adaptable and stable. Rules of how to compare services [6] are beneficial for this. High reliability and load balancing can also be achieved by using redundant services and, or service registries [10], [11].

3.7 SOA Metrics Defined

The definition of SOA metrics in Figure 3:6 is based upon the material presented in this chapter.

A SOA must have self-contained services that adhere to a service description describing what it does, how to interact with it, and what the outcome of using it will be. This information must be presented in a machine-interpretable format that is generally accessible to all potential consumers of the service.

The internal implementation of a service is hidden from external users, and all interaction with it must be done via its interface using message exchange. The interfaces must be based on standard protocols so that they are independent of the specific underlying technology.

Figure 3:6 – SOA metrics

4 Description of the Wearable Command Unit

This chapter gives a general overview of the Saab Security System's Wearable Command Unit: what it is, what its architectural looks like, its main components, how the internal alarm communication operates, and some example use cases. This thesis will only consider version 2.0 of the WCU. This version of the system is currently under development at Saab Security Systems.

4.1 What is the Wearable Command Unit?

Saab Security System's Wearable Command Unit (WCU) is a system for communication and information distribution between mobile and stationary nodes [14]. Its main purpose is for use in crisis management to establish "situation awareness" among its users. By situation awareness it is meant that all of the users should be able to have a common understanding of what the situation is, where users and objects in their environment are, what the status of these users and objects is, etc. The core components of the WCU system (showed in Figure 4:1) are the WCU Server, the Command & Control Client, the Field Client, and the Smartphone Client.

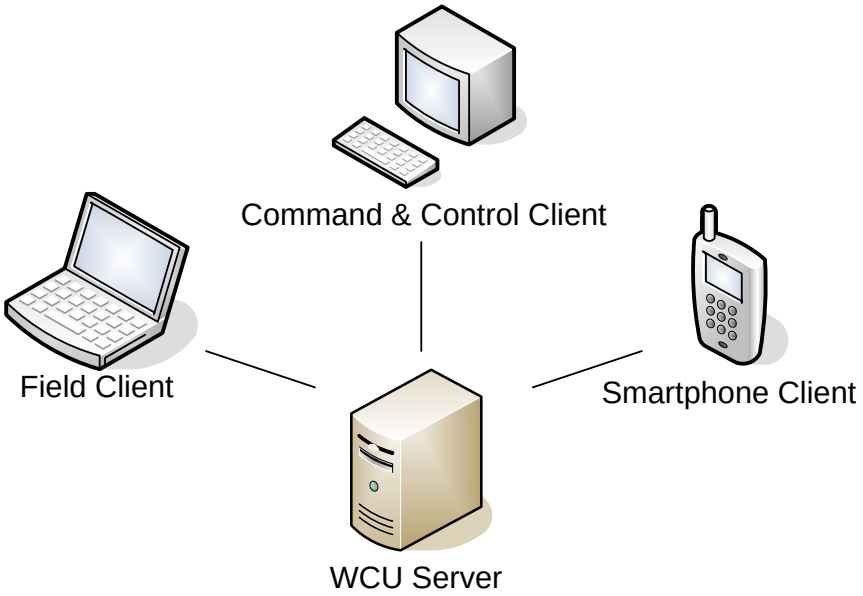

Figure 4:1 – WCU main components

The WCU is designed to be used in crisis management on both a local level as well as by authorities on an international level. One of the main markets for the WCU is in public safety. By providing rapid information distribution, alarm management, and other support, the involved parties and authorities can get a quick overview of the situation. Other target markets are surveillance of restricted areas and buildings; for example at airports, mines, and harbors.

4.1.1 WCU Server

The WCU Server has the role of distributing messages between participating clients. This is controlled by subscription rules as well as a set of roles associated with each client. The WCU Server also handles authentication in the system. This means that no client can connect without first authenticating and registering with the server. The WCU Server is accessed and managed via a web based interface.

Version 2.0 of the server, which at the moment is evolving at Saab Security Systems, will not be a pure centralized message broker. This is since subscription rules and other information is kept locally on the server instead of retrieving this information from a database. This is due to performance reasons and time constraints in the development process. In the planned version 2.1 of the system this is to be changed, making it a true message broker.

4.1.2 WCU Clients

WCU clients are built using a plug-in architecture. That is, most of the functionality such as maps, video interaction, and logging is derived from plug-ins. This makes it easy to customize the product for specific needs among different customers.

4.1.2.1 Command & Control Client

The purpose of the Command & Control Client is to manage and control resources, units, and alarms in the system. This type of client generally runs on a stationary machine. This client displays a map indicating incoming alarms and events, as well as related information (positions, messages, photos, video, etc.).

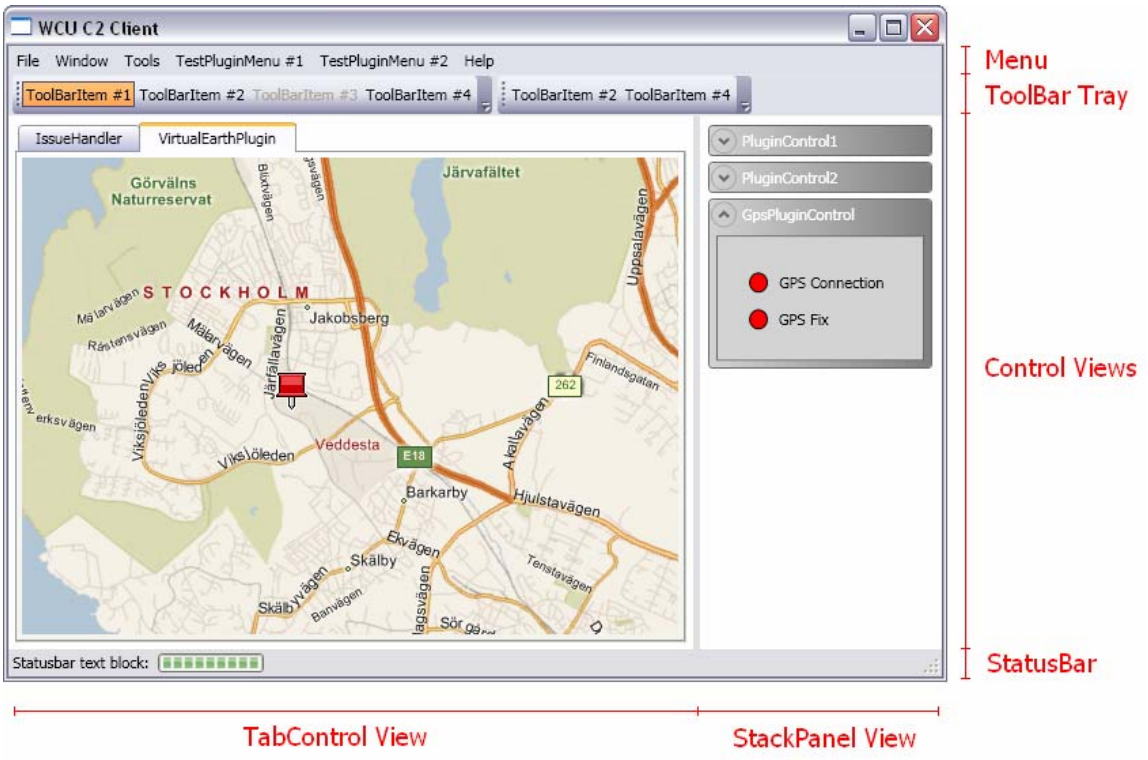

Figure 4:2 – WCU Command & Control Client GUI prototype

Figure 4:2 shows an early prototype of the WCU Command & Control Client graphical user interface (GUI). At the top is a menu for accessing basic functionality, as well as plug-in specific menus. A ToolBar Tray, providing tools and functionality for the currently loaded plug-ins can be found below the menu. A TabControl View is found to the left below the ToolBar Tray. This is where plug-ins have the option to display their main user interface. The selected tab view (VirtualEarthPlugin) in the figure shows a map of a suburb of Stockholm. The red marker indicates an active alarm. The StackPanel View to the right lists status values for each of the currently loaded plug-ins.

4.1.2.2 Field Client

The WCU Field Client is normally run on a mobile Tablet PC or laptop equipped with a GPS receiver. This type of client is mainly used in vehicles or out in the field by a commander or similar manager. The Field Client receives alarms and missions from the WCU Server depending on its role and subscription status (as set in the Command & Control Client). New alarms may also be generated by the Field Client. In this case the alarm is sent to the WCU Server which redistributes it to the other relevant clients. The map in the Field Client GUI displays the exact location of all resources associated with the same alarm. All annotations that are drawn on the map are immediately distributed to all participating parties.

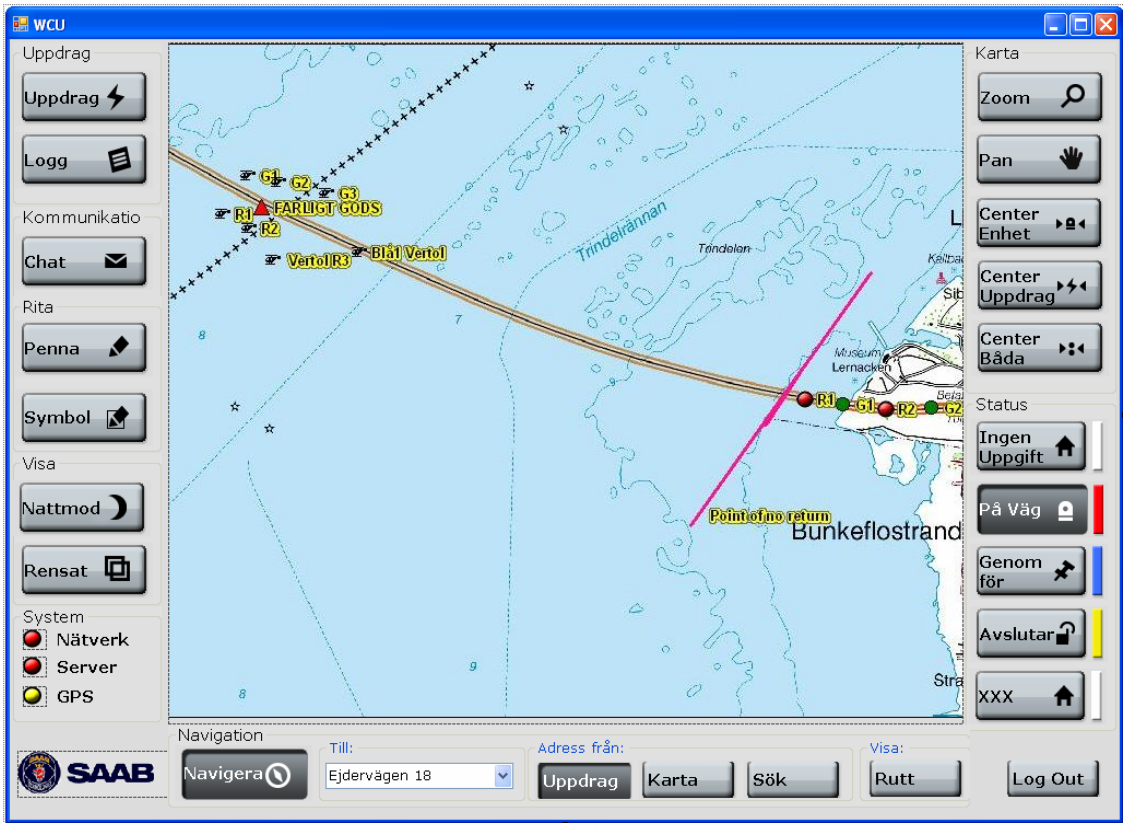

Figure 4:3 – WCU Field Client GUI prototype

Figure 4:3 shows an early prototype of the WCU Field Client GUI. All buttons and text in the figure are in Swedish. The center of attention is the map in the middle displaying alarms,

available units (which may be a guard, fire truck, police officer …), different type of sensors, and other location related information. To the left of the map are buttons for displaying mission and log information, for starting a chat, annotating the map, and for adapting the GUI for night usage by changing its colors. In the bottom left hand corner are also indicators of whether the client is currently connected to the network, to the server, and if GPS is activated. The right side of the figure displays buttons for navigating the map as well as setting this client's status. At the bottom are buttons for navigating to, or searching for a specific address, displaying a map, showing the best route to a location, and logging out.

The map view shown displays the area near the Öresund bridge. The red triangle indicates the location of dangerous materials (farlig gods) that are being transported across the bridge. This transport is being over seen by seven helicopter based units. Each of these units is labeled on the map. An annotation (marked "Point of no Return") which starts two or three piers out from the shore (near Lernacken on the Swedish coast) indicates that the goods transport must move past this area and once past cannot return.

4.1.2.3 Smartphone Client

The Smartphone Client can be seen as a thin Field Client. It receives and reports back alarms in the same way that the Field Client does. Its role in the system as well as its subscription rules are also managed by the Command & Control Client. It has only the most essential functionality due to the limited display of the handset. This includes GPS positioning, message exchange, and alarm report generation. The Smartphone Client is used mainly by personnel on foot.

4.2 Architecture Overview

All of the code for the WCU is written in C# on top of the Microsoft .NET Framework 3.0 using Visual Studio 2005. The Command & Control Client and the Field Client are running Microsoft's Windows XP operating system with the service package 2 (SP2) fixes. The Smartphone Client runs Microsoft's Windows CE, and the WCU Server is running Microsoft's Windows Server 2003.

Figure 4:4 gives an overview of the connectivity in the WCU system. The mobile Field Client and the Smartphone Client are connected through GPRS or 3G packet data services to an internet service provider (ISP). The Command & Control Client may also be connected through GPRS or 3G, but is normally connected to the WCU Server via a fixed line. This internet service provider may place a Network Address Translation (NAT) between the local mobile network and the Internet. In addition there may be another NAT between the ISP and the wireless network provider.

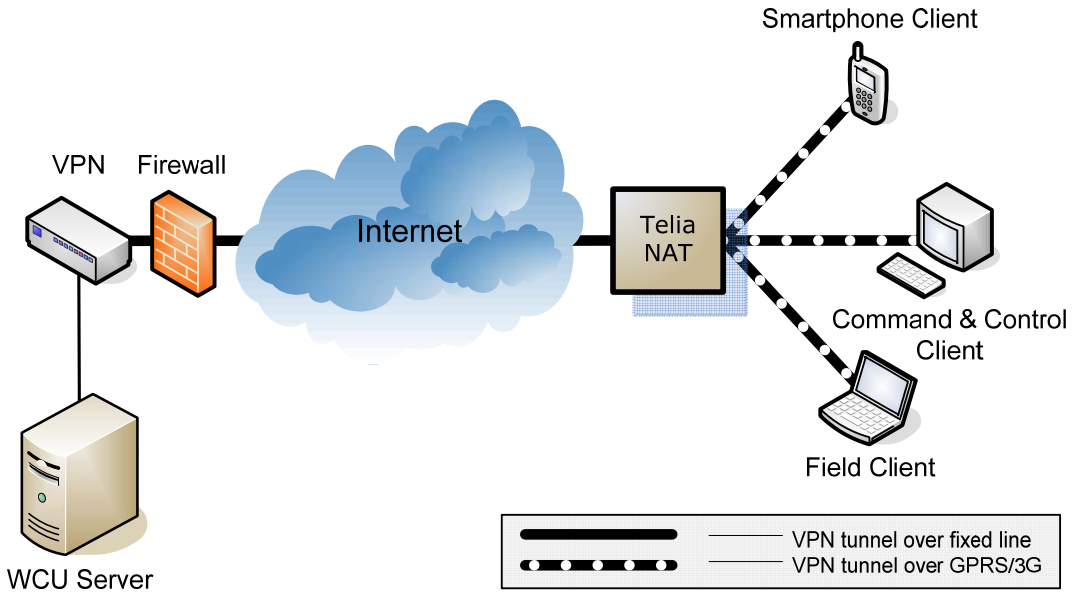

Figure 4:4 – WCU connectivity overview

In the earlier version of WCU (version 1.2) a VPN server is used for providing security and tunneling between the WCU clients and the WCU Server as they communicate over the public network. Since problems have been encountered with the current solution – sessions are lost as mobile clients move out of range – the WCU 2.0 design team is considering alternative VPN solutions that will support session consistency and network roaming. Outside of the VPN is a firewall located; providing security between the intranet and Internet.

The system is centralized, which means that all communication between WCU clients has to go via the WCU Server. For example, if a client wants to send an image-file to another client it must first send it to the WCU Server which will forward it to the intended recipient. A centralized solution was chosen for several reasons: the design team wanted to log all events in the system in a *central log*, and it was also much more *straightforward* and less *time consuming* to build a centralized solution than to build a complex distributed solution. Since the system is much about providing "common situation awareness", a majority of the information is *consistent* between clients. Keeping this information in a centralized location facilitates the work of maintaining it up-to-date among participants. Downsides of using a centralized solution are among others *performance*, *cost*, and *reliability* issues.

4.3 Alarms

WCU uses a centralized solution with a WCU Server that handles and distributes alarms to subscribing clients. All alarms in WCU are represented as Indication objects[12]. An Indication object has properties such as alarm type (such as fire and security) and priority. The Indication class inherits the WcuMapObject class, which in turn inherits the atomic WcuObject class. Both the WcuMapObject class and the WcuObject classes are abstract

[12] See Appendix C and section C.3

classes. The WcuMapObject class has location based properties such as coordinates, geometry, symbols and location. The atomic WcuObject has properties such as begin time, end time, name, id, and text. There is also a special designed class called ExternalIndication that is aimed at describing external systems. This class inherits the Indication class and has properties such as ExternalSystem and ExternalSystemId. An overview of the alarm representation in WCU can be seen in Figure 4:5.

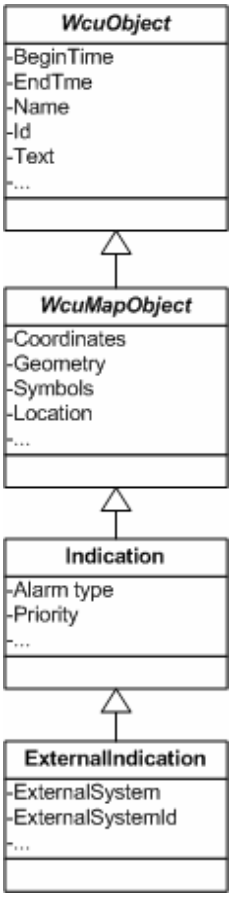

Figure 4:5 – WCU alarm class diagram

There are also other classes that can be used for describing parts of an alarm in the WCU. One such class is the WcuFile class, whose purpose is to describe different types of files that can be a part of the system. These classes also inherit the atomic WcuObject abstract class.

Depending on what role a WCU client is associated with as it authenticates with the system, it is given a number of predefined subscriptions and capabilities. When the WCU Server receives an alarm it makes a selection from the clients that have the appropriate subscriptions and redistributes the alarm message to those clients. The selection of client(s) depends on the content of the alarm as well as other system state information, such as the status of each client.

4.4 Internal Alarm Communication

WCU internal communication is defined as the communication taking place between the WCU Server and WCU clients in the system.

The Microsoft Windows Communication Foundation (WCF) is used for all communication taking place within the WCU network. WCF is based on the concept of services and support several types of bindings (wire-level agreements)[13]. More about bindings in WCF can be found in section 8.1. There is also support for constructing custom bindings. Table 4:1 lists the services involved in the minimum required communication taking place when a client sends a new Indication (alarm) to the WCU Server. All services except for the "authentication" service use the netMsmqBinding, a queued and asynchronous binding that uses the Microsoft Message Queuing (MSMQ) protocol.

Table 4:1 – Services published at WCU Server

Service Name	Description	Binding	Host
Authentication service	Used by clients to login to or logout from the WCU Server	netTcpBinding	Server
Subscription service	Used by clients to subscribe to new and updated objects and events	netMsmqBinding	Server
Notification service	Used by clients to inform the WCU Server of all its known objects	netMsmqBinding	Server
Publishing service	Used by clients to publish new or updated objects	netMsmqBinding	Server
WcuObject publishing service	Used by server for sending back WcuObject information	netMsmqBinding	Client

Among the services published by the WCU Server are: an "authentication service" for authenticating clients connecting to and disconnecting from the system, a "subscribing service" that clients make use of to get a notification every time the server receives a new or modified object, a "notification service" that clients use to inform the WCU Server of all its known objects, and a "publishing service" for publishing new objects to the server. The reason the client host the "WcuObject publishing" service is that the MSMQ protocol does not support duplex communication. A client uses this service to receive WcuObject information from the server.

A client that connects with the server must begin with calling the "authentication" service and set up a channel to all other services required in order to initiate communication. This includes sending information about all its known objects to the WCU Server through the "notification" service. After this has been done, the client may use the "publishing" service to publish new objects to the server. This service implements a method that accepts different types of WCU objects (for example an Indication). As the method receives a new WCU object, the server forwards a copy of the object to every client that is registered as a subscriber to this type of object. This is done by the WCU Server by calling the "WcuObject publishing" service on the WCU clients.

[13] A complete list of supported bindings can be found in Appendix B

4.5 Use Cases

4.5.1 Alarm Report in a Surveillance System (Internal Alarm)

1. A security guard on duty discovers a broken card reader.
2. The event is reported by the guard via a WCU client. This involves creating and sending an alarm to the WCU Server.
3. The system automatically selects what action to carry out depending on the report and other information. If the system can not decide what to do it will inform an operator stationed at a Command & Control Client.
4. One or more WCU clients can be associated with an alarm event.
5. The alarm report is sent together with other information to the WCU clients associated with this alarm.
6. When one of these clients reports that the cause for the alarm has been addressed (in this case the reader is repaired or replaced), then the alarm is dismissed and reported back to the Command & Control Client.

4.5.2 SOS Emergency Alarm (External Alarm)

1. An emergency alarm about an accident is received by the SOS Infoserver.
2. The WCU Server receives the alarm via CoordCom, the WCU Mail Server, and the WCU Client as showed in Figure 5:1.
3. An XML-file[14] attached to the email contains all information about the alarm, such as the location of the accident as well as which resources needed to be dispatched. The WCU Server uses this information to dispatch the relevant resources.
4. All involved WCU clients can communicate with each other as well as with the central Command & Control Client as long as the alarm is active. Interaction may include exchanging messages, video, sound, and other media. Supporting information such as relevant laws, medical advice, etc, may be retrieved directly from databases or via the WCU Server[15].
5. When the alarm is dismissed, a cancellation of the alarm is sent to all participating parties from SOS Alarm and a case log is created and saved by the WCU Server.

4.5.3 Incoming CAP Alarm (External Alarm) – Extended Functionality[16]

1. A sensor connected to an external system is registering a suspected movement within an alert zone.
2. The external system generates a Common Alert Protocol (CAP) [21] alert message and sends this message to a WCU client. The message contains the address of a video source associated with the alert zone.
3. The WCU client translates the CAP alert message to WCU-format and sends it to the WCU Server which in turn informs the appropriate WCU clients.
4. The WCU clients can then decide what action to carry out. When the alarm has been terminated the WCU client informs the WCU Server.

[14] See section C.4
[15] How this type of functionality is to be implemented depend on the core WCU 2.0 architecture which at the moment of writing still is under construction
[16] See section 9.4

5 Related Work

Section 5.1 describes how the earlier version of WCU (version 1.2) was connected to an external system. The remaining part of this chapter (section 5.2 and 5.3) describes solutions for how SOA could have been and may be used for implementing alarm interoperability in public safety and surveillance systems.

5.1 Interoperability Solution in WCU 1.2

The WCU Server in WCU version 1.2 accepts only incoming alarms represented as WCU-objects. To receive alarms in other formats from external systems, a specially designed WCU client is used as a translating bridge between the two systems. The only external system that WCU 1.2 has been integrated with is Ericsson's CoordCom [17], which is a system for Call Taking and Dispatching with decision support used by Swedish SOS [17]. To distribute information about alarms to the public, CoordCom uses a SOS InfoServer based on FTP.

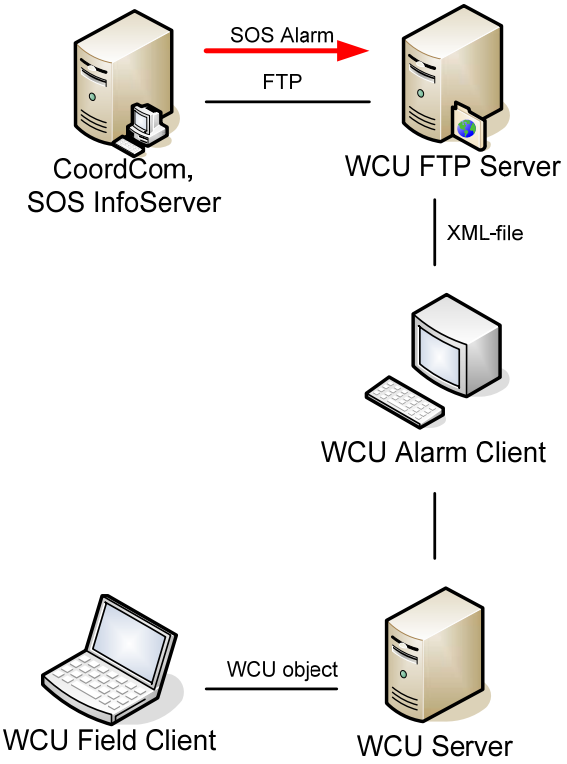

Figure 5:1 – WCU SOS alarm interface

Figure 5:1 shows the chain of events involved in the dispatch of a new SOS alarm. For each new alarm that is created in CoordCom, the InfoServer uploads an XML-file[17] to a designated directory at the WCU FTP Server. The XML-file contains information about the dispatched alarm, such as dispatched vehicles, severity of the alarm, etc. As soon as a new

[17] See section C.4

file is copied to the directory at the WCU FTP Server an event is triggered, making the WCU Alarm Client load and parse the file. From the information in the XML-file the WCU Alarm Client creates a new WCU-formatted alarm[18] and sends this to the WCU Server. The WCU Server in turn distributes the alarm to the relevant Field Clients.

5.2 Related Systems

Organizations, companies, and authorities have started to see the potential of integrating public safety systems and surveillance systems. This has lead to a number of emerging solutions on the market. A selection of these systems is named below.

5.2.1 Motorola Public Safety and Fire Service Solutions

Motorola provides a suite of solutions within the public safety and fire service domain. Among these is the Motorola Computer Aided Dispatch (CAD) [43] software built on Microsoft's Windows 2003 OS. It is a system for automatically dispatching alarms to one or more public safety agencies via a wide area network. It uses ESRI's ArcGIS software to map alarms to geospatial data [24]. It includes capabilities such as incident priorities, resource management, system status, management plans, automated routing, and location information including hazard data, mapping, and automatic vehicle location.

5.2.2 LogMate Alarm Management System

The LogMate Alarm Management System (AMS) [40] is a software application for archiving plant alarms and event information. It is built on the Microsoft's .NET platform and uses a Microsoft SQL Server database. The system consists of two other components in addition to the database: a "Capture" component that is responsible for collecting alarm and events and parsing this information into the database, and a Web service interface – built on Microsoft's Internet Information Services (IIS) – that provides clients with both local and remote access to the database [25]. There is also a customizable filter for defining how the information that is collected by the "Capture" component is parsed into the fields in the database.

5.2.3 Sun Ridge Systems Integrated Public Safety Software

Sun Ridge Systems [42] provides a collection of integrable systems within the public safety domain. This includes a computer aided dispatch system for managing incidents events and units, a Law Enforcement Records Management System for access and tracking police data records, a Mobile Computer Field System with touch-screen, status reporting, information retrieval, and loging, and a Pocket PC for access to person and vehicular information, reports, mug shots, and incident information. There is also support for automatic vehicle location where the location of dispatched vehicles is displayed on a map. All movements on the map are recorded which makes it possible to replay and analyze real-time scenarios.

5.2.4 Tiburon Public Safety Solutions

Tiburon provides a set of software solutions within the public safety domain [44]. The suite of software include a computer aided dispatch solution for law enforcement and fire fighting with mapping, automatic vehicle location, and incident management, and a field automation

[18] See section C.3

system for officers and fire personnel in field, that includes reporting tools, emergency notification, unit and incident status, and file transfer.

5.3 Alarm Handling

How to present alarms to operators as well as to recognize and discard false or irrelevant alarms is of great important for life-critical alarm systems in public safety [28]. An example of when irrelevant alarms can cause problem is if a fire starts in a building and several sensors start sending a large amount of alarm messages containing more or less the same information. In these cases it is important to filter the duplicate alarm information and to present only what is relevant (in this particular situation) to the operator receiving the alarm [34].

Another important issue is to have well trained operators that are prepared to handle all type of incoming alarms. That is, all alarms or pattern of alarms should have a pre-defined response [34]. The operators must also understand all the information and terminology presented on-screen [28].

All alarms should be saved and logged in a database available to the operators of the system [42]. In this way operators can decide what action to carry out depending on how terminated alarms where previously handled.

It is very important to set intelligent priorities on both alarms and the access of resources in order to avoid conflicts [22]. These priorities should not be fixed but should be adapted based upon a specific situation. The priority of incoming alarms depending on severity is an example of this dynamic adaptation.

A number of characteristics are presented in [34] as it comes to alarms: relevance (they should have an operational value), uniqueness (not duplicated), they should not be obsolete, prioritized (to inform the operator of which alarm that is of highest importance), easy to understand, identifying the occurred problem, and focusing on the most important issues.

5.3.1 Human-Machine Interface

How alarms are presented to a human user becomes increasingly important as the number of alarms and the rate of alarms increase. By prioritizing alarms, the operator can lower the number of alarms that they need to deal with at one time.

In [40] users are presented with a customizable data grid containing triggered alarms and events. Colors, filters, and rules for sorting the rows can be adapted to the specific needs of a particular situation. In [34] attributes of a good alarm list message are presented. These are among others: usage of known nomenclature, consistent abbreviations, consistent hierarchical message structure, and a clear identification of the occurred condition.

5.3.2 False Alarms

In 2005, 98 percent of the automatic fire alarms and burglary alarms received by the Swedish SOS Alarm Center where false alarms [39]. This clearly indicates how important it is to include intelligent false alarm filtering in these kinds of systems. The earlier a false alarm can

be stopped the lower the cost of handling it. However, all alarms that are not false alarms must reach their intended recipient. It is important to note that filtering out alarms which **should** have propagated can lead to even more devastating costs.

By combining alarm and event data with statistics, malfunctioning sensors can be identified and unnecessary alarms can be reduced. Collected statistics is also good for analyzing trends in the system. Examples of this kind of functionality can be found in [40].

6 WCU Alarms from a SOA Perspective

This chapter describes how the WCU alarm communication is related to SOA as it was defined in Chapter 3. The SOA definition in Figure 3:6 is compared to how the WCÚ internal alarm communication is implemented (as described in section 4.4) and how the WCU external alarm interface (described in section 5.1) is implemented.

6.1 SOA Definition Revisited

From the SOA definition in Figure 3:6 one can conclude the following:

Table 6:1 – SOA definition revisited

1. A SOA must:
    a. Contain one or more services.

2. A service must:
    a. Be self-contained.
    b. Hide its internal implementation from external users.
    c. Adhere to a service description.
    d. Interact through its interface using message exchange.

3. The service description must:
    a. Describe what the service does.
    b. Describe how to interact with the service.
    c. Describe what the outcome of the service will be.
    d. Be presented in a machine-interpretable format.
    e. Be accessible to all potential consumers of the service.

4. The service interface must:
    a. Be based on standard protocols to allow for independence from the underlying technology.

The remaining part of this chapter will investigate how the statements in Table 6:1 relate to the WCU alarm communication.

6.2 The WCU Internal Alarm Communication

As described in section 4.4 alarms in the WCU system are represented as WCU objects that are sent between the WCU Server and WCU clients. To accomplish this communication, a number of services are published. All but the "authentication" service are implemented in WCF using the NetMsmqBinding.

The services published at the WCU server have to be executed in a predefined sequence in order to authenticate and initiate communication with the system. This does not however imply that they are dependent on any other parts for their particular functionality; hence they are self-contained. They also hide their internal implementation from external users since they can only be interacted with through their well-defined interface. WCF uses message exchange (SOAP messages) to accomplish this interaction. All potential consumers that have network access to the services and adhere to the rules stated in the service contract may access the services.

In WCF every service is associated with an endpoint containing an address of the service location, a contract describing what the service does and what the outcome of using it will be, and a binding which defines how to interact with the service. This information is presented in a machine-interpretable way, where the format depends upon the binding that has been associated with the service.

The last statement in Table 6:1 says that "a services interface must be based on standard protocols to allow for independency of underlying technology". Since the NetMsmqBinding is used for communication, clients that want to interact with such a service are enforced to use this protocol. The SOAP messages in the NetMsmqBinding are optimized to use a binary format only readable by WCF. This makes the internal communication in the WCU work well since all components are built using WCF. However, problems will arise when it comes to integration with external systems running on other platforms.

MSMQ is only available on Microsoft platforms. Programming interfaces that support MSMQ are .NET Framework, C/C++ library, and a COM library. Although third-party solutions exist; no MSMQ interface is included in other programming languages as a standard. This makes MSMQ unsuitable for cross-platform interaction.

Part III of this thesis presents a design proposal for a SOA-based solution to how WCU may solve interoperability issues with external systems.

Part III - Proposed Design

The previous chapter concluded that WCU 2.0 does not comply with SOA due to the fact that it uses optimized messages in binary format for internal communication. To make WCU conform to the SOA definition given in section 3.7, it will have to implement services that have a platform independent interface.

Another design issue is whether the existing services provided by WCU should change their bindings to ones that do not optimize the messages using a binary format, or if WCU should provide other services, specific for external systems? Something that strongly speaks for providing services specific for external systems is that the existing services in WCU are very tight connected to the system; with predefined workflows, authentication and subscription rules, etc. Such aspects should not be of concern to an external party whose only purpose is to send in information (alarms) to the WCU system. Another reason is that by allowing an external system to use the internal WCU services, that system will be given the same permissions and abilities in the system as any other WCU client. That is, there will be a loss of control of information entering and leaving the system.

The idea of the WCU design team is to make the WCU Server consistent, and instead have as much of the logic as possible in the clients. In this way, changes to the system can easily be made without exchanging or modifying the server. Different customers sending in alarm information to a service on the WCU Server might require different workflow schemes. By placing the logic outside of the server (in a WCU client), these workflows can be adapted and changed without any modification of the server implementation.

Since WCU-formatted alarms are tightly connected to the WCU system, a standardized format, or data model for describing and exchange alarm information with other systems is to prefer. This would also allow the WCU system, as well as other systems to be exchanged and modified without having to concern about interoperability issues regarding the alarm data model.

The following chapters are as follows: Chapter 7 describes the architectural overview of the proposed design, Chapter 8 describes the software that was used in the design, more specific how WCF was configured, and Chapter 9 explains how the data model interoperability problem is addressed in the proposed design.

As a part of this thesis project was also a survey made to decide whether to use Microsoft BizTalk Server for implementing a SOA based interface. The result from this study as well as the reasons for not selecting it as an alternative solution is given in Appendix D.

7 Architectural Overview

As described in section 5.1 the WCU version 1.2 used a client as a gateway to communicate with other systems. Such an approach is preferred for WCU 2.0 as well due to the reasons given in section 6.2. This makes a WCU client, instead of the WCU Server responsible for interoperability with external systems. The client may independently from the rest of the system implement its own interfaces and logic, while still maintaining an unmodified connection with the WCU Server, in the same way other WCU clients do.

As previously described in section 4.1.2 the WCU clients are based on a plug-in architecture. This facilitates building an interface to enable communication with external systems. The alarm plug-in, whose implementation is described in more detail in Chapter 8, will in this thesis be referred to as the "WS Alarm plug-in".

7.1 Architectural Overview

Figure 7:1 gives an architectural overview of the design proposal. A WCU client running the WS Alarm plug-in works as a gateway for external system. The plug-in has an interface which is built to accept non-WCU-formatted alarms. The WCU client is preferably connected to the WCU Server through a LAN as showed in Figure 7:1, but the option of connecting though GPRS/3G via the VPN tunnel is also possible[19].

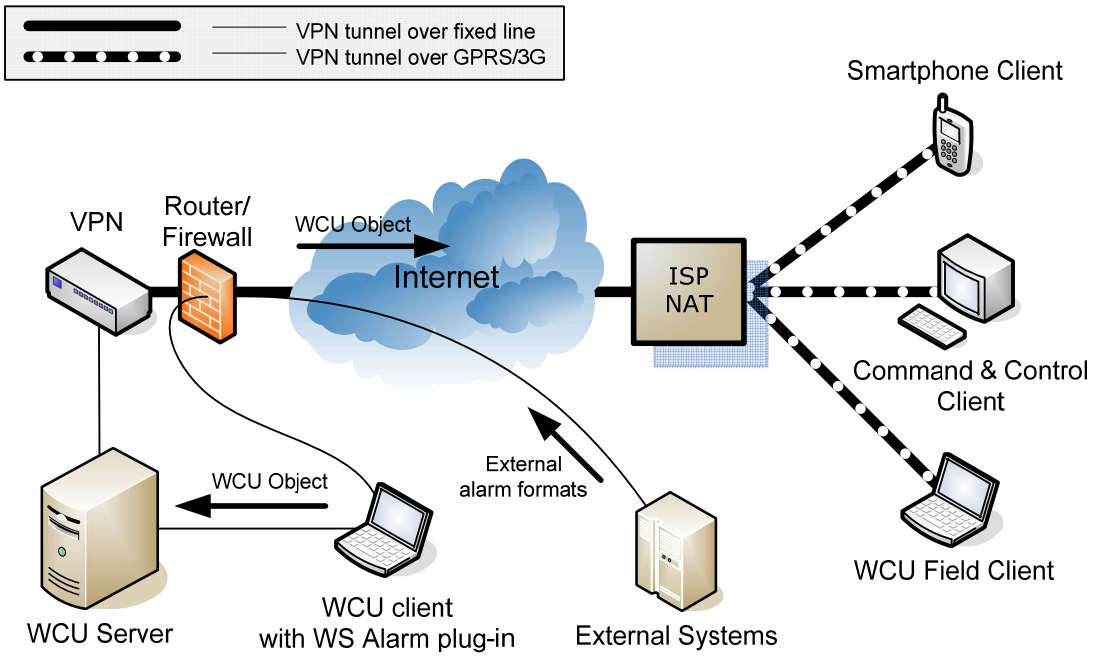

Figure 7:1 – Architectural overview of proposed design

[19] Explained in section 4.2

The problems with having the WCU client run the WS Alarm plug-in on a GPRS/3G connection are several: first of all, all traffic would have to go via the VPN tunnel. This would mean that the external system would have to send the alarm messages via the VPN tunnel through an ISP Network Address Translation (NAT) to the WCU client. The WCU client would then have to translate the message to a WCU object interpretable by the WCU and send this back to the WCU Server through the VPN tunnel.

The other option is to connect the WCU client running the WS Alarm plug-in directly to the WCU Server via LAN. This avoids the delay associated with GPRS/3G and greatly reduces network traffic. No port forwarding is needed either since the external system can connect directly to the WCU client's public IP. This option is preferable and should be used when possible.

8 WCF Configuration

WCU 2.0 is built on the Microsoft .NET 3.0 platform and all communication is based on the Windows Communication Foundation (WCF) as described in section 4.4. WCF has support for building platform independent communication solutions. This is achieved by selecting the appropriate service bindings as well as other parameters. A proposal of how WCF should be configured for the WS Alarm plug-in is described below.

8.1 WCF Bindings

One of the requirements of the service which is to accepting incoming alarm messages from external systems is that it must use platform independent protocols on both transport and message level. It should also implement some level of security since the external system may connect to the service through a public network.

The goal of the Windows Communication Foundation (WCF) is to put all Microsoft distributed technologies in a single package. It supports communication with among others: .NET Remoting, ASP.NET Web Services (ASMX), and WS-* standards. WCF provides nine system-provided bindings[20], and also has the ability to create custom bindings. To be able to support interoperability between different platforms other than Microsoft, WCF provides the BasicHttpBinding and the WSHttpBinding, both operating using SOAP over HTTP. WSDualHttpBinding and WSFederationHttpBinding also operate using SOAP over HTTP. However, they provide additional functionality (duplex contracts and sessions, and federated security) that will not be required in the proposed design solution.

The binding BasicHttpBinding represents the "Basic Profile 1.1" provided by the Web Services Interoperability Organization (WS-I) [50]. It covers the core Web services standards[21] and includes support for messaging, description, discovery, and security of Web services. The "Basic Profile 1.1" is designed for services with a direct connection with its clients and where SSL is relied upon for security [26]. Security in this binding is disabled by default, but it can be configured for either message level or transport layer security.

The binding WSHttpBinding represents the WS-* standards and extends the BasicHttpBinding by adding functionality such as reliable sessions, transactions, transport security, and SOAP security. The WS-* standards are the result of the effort of large organizations and consortiums including OASIS and W3C. This binding has security on the message level by default, but transport level security may also be enabled. Message level security is achieved by [49] the use of Web Services Security SOAP Message Security (WS-Security) [30], WS-Trust [31], WS-SecureConversation [32], and WS-SecurityPolicy [36].

WCF, as mentioned earlier also provides the option to create custom bindings. However, the binding BasicHttpBinding fulfills all of the requirements presented in the beginning of this subsection.

[20] See Table B:1, Appendix B
[21] See Table B:2, Appendix B

8.2 Web Service Discovery in WCF

There exist different mechanisms for Web service discovery. These range from sending the service description as a file, to using the more sophisticated and dynamic Universal Description, Discovery, and Integration (UDDI) [47] registry. The tradeoffs are between functionality and simplicity. The binding BasicHttpBinding in WCF supports both a WSDL repository and UDDI registries.

A WSDL repository primarily uses HTTP GET to retrieve the service description published on a Web site. It is a static solution that provides little flexibility, as the service consumer generally has to know the location of the published service in advance.

A UDDI registry is on the other hand a much more sophisticated method of publishing and discovering web service descriptions. There exist two types of UDDI registries: public and private. Private registries are more common and are mostly used in intranets. Compared to a WSDL repository, the UDDI registries add extra functionality, such as subscription to changes in the registry, security, replication, and implementation of policies. To use UDDI with WCF, Microsoft's Windows Server 2003 is required for hosting the registry.

The proposed design will be based on a service receiving alarms in a non-WCU format. Although there are several advantages to using UDDI, there will be little or no need for such a solution in the proposed design. This is because only one service will be published, and the external solutions will know in advance the location of that service. SOA is used in this solution mainly to tackle the integration problems. Thus a static WSDL repository is sufficient. The option to build a solution based on UDDI is discussed in section 12.

8.3 Web Service Description in WCF

WCF implements the Web Services Description Language (WSDL) [46] which is the de-facto standard for describing Web services. It contains information about a service such as its address, its binding, a description of its operations, and what data is to be communicated with the service. WSDL descriptions for methods (in a published service) can be generated automatically in WCF, and fetched by others using a simple HTTP/GET request. This is achieved by setting the httpGetEnabled attribute equal to "true" in the service binding.

8.4 Web Service Security in WCF

The binding BasicHttpBinding in WCF is targeted at the WS-I Basic Profile [48] and includes support for: no security, HTTPS security [29], SOAP security [30], or HTTPS security with SOAP credentials.

Transport level security (HTTPS) does only support hop-to-hop security and is dependent on the transport protocol (HTTP). It supports streaming and is generally better as it comes to performance. Message security provides end-to-end security over SOAP intermediaries and is dependent on the WS-Security specification.

As described in section 7.1 may the WS Alarm plug-in communicate over an insecure public network with an external system. This implies that the exchanged information must be protected in some way from others than the intended recipient. Implementing WS-Security would introduce a dependency for all communicating parties and make the system more dependent on a specific technology. Implementing transport security on the other hand might not be enough if SOAP intermediaries will be used in communication.

Since the WS Alarm plug-in will communicate over a public internet it may not be know of the route that the received alarm message has taken. If SOAP intermediaries are used or not used should not be of concern to the WCU system. This implies that message (SOAP) security should be used in the proposed solution. In WCF this is achieved by setting *SecurityMode* element to *Message*, and setting the *MessageCredentialType* property to appropriate value. The proposed design will use the *UserName* value since this will cover the security requirements that the system have today.

8.5 Hosting of Web Services in WCF

There are four different ways of hosting a Web service in WCF via: a Managed Application, a Managed Windows Service, Internet Information Services (IIS), or in Windows Process Activation Service (WAS).

The Managed Application is the most flexible of the four, since no additional software or infrastructure is needed. The service and its endpoint are embedded directly inside the application code. The option to host a service as a Managed Windows Services gives the control of the services' lifetime to the operating system. For example the service can be configured to run automatically as the system boots up. To host a service in IIS requires that IIS is installed and configured properly. IIS only has support for HTTP protocols. WAS is a generalization of IIS that works with protocols other then HTTP. This solution requires installation and configuration of WAS activation components.

Since the service will run from a plug-in in a WCU client, it will be hosted in a managed application. This avoids the need to run IIS or WAS on the WCU client.

9 Data Model for Alarm Interoperability

Although different systems can integrate and communicate using SOA, they may still have difficulties understanding each other due to the need to interpret data more precisely. For this purpose open standards have been developed around exchanging and interpreting alarm information.

9.1 Global Justice Extensible Markup Language

The Global Justice Extensible Markup Language (XML) Data Model (Global JXDM) [41] was developed by the Global ISWG[22]'s XML Structure Task Force (XSTF). It is a data reference model for exchanging information within the justice and public safety communities. It consists of a data model (JXDM), a data dictionary (JXDD), and an XML schema [19], [41]. Global JXDM is a large[23] specification with detailed elements about accident types, incident types, insurance information, etc. The specification is targeted at use in the U.S. and includes many U.S.-, and Canada-specific elements.

Because Global JXDM describes an extensive amount of information compared to the WCU-format, and since it is mainly targeted at the American market, it is not a very appropriate candidate for the WCU system. A subset of the specification could be used (for example the alert subset in the model) but a more appropriate specification would be to prefer if found.

9.2 Emergency Data Exchange Language

The Emergency Data Exchange Language (EDXL) [20] is a standard developed by OASIS for routing emergency messages to recipients. It acts as a container and may carry any kind of payload (for example one or more VEDS or CAP messages, see sections 9.3 and 9.4 below). Among the supported capabilities are distribution type, geography, incident, sender ID, and recipient ID [19].

EDXL is used as envelop for routing other emergency messages, and not for describing alarm information. Thus, if it is to be used in the proposed solution another specification would be needed in conjunction with EDXL. This option was considered, but it turned out that sufficient information (message sender ID, sent time) for determining the destination of alarm messages was already provided by the specification chosen for the proposed solution (see section 9.4 below). WCU support for EDXL messages is further discussed in section 12 about future work.

[22] Global Justice Information Sharing Initiative's (Global) Infrastructure and Standards Working Group
[23] The .xsd file alone is 2,69 MB

9.3 Vehicular Emergency Data Set

The Vehicular Emergency Data Set (VEDS) [27] is a standard proposed by the ComCARE Alliance[24] CAN Data Set Working Group for describing emergency information for vehicular emergencies as XML. There are currently discussions about harmonizing VEDS data elements with Global JXDM, as well as whether to incorporate VEDS into the EDXL process or not [27].

9.4 Common Alerting Protocol

The Common Alerting Protocol (CAP) [21] is a standard developed by OASIS for exchanging alert information. It is based on XML and supports capabilities such as geo-positions, multiple languages, updates, cancellations, facility for audio and video, and digital encryption. Among the organizations and companies that currently are using CAP are the U.S. Department of Homeland Security and the U.S. National Weather Service[25].

A minor flaw of the CAP protocol is that it does not use pure data-centric XML, something that would have been preferred. On the other hand does it provides a standard for describing alarm information and ensure information consistency between interacting parties.

The CAP protocol promises to be able to convert "to and from all kind of sensor and alerting technologies" [21]. If the external system can not describe its alarm information in CAP format it will also be difficult to describe it in WCU format. This makes CAP a good candidate for exchanging alarm information with other systems.

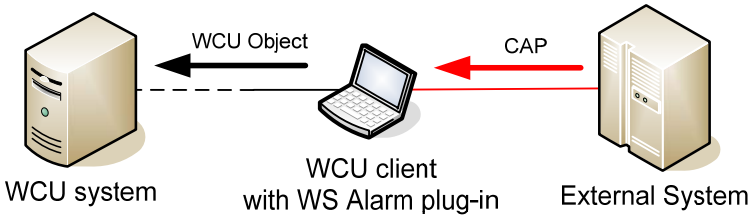

Figure 9:1 – CAP interface in WCU

Figure 9:1 show an external system sending an alarm described in CAP-format to the WS Alarm plug-in running on a WCU client. As the plug-in receives a CAP message it translates it to a WCU Object, and sends it to the WCU client framework. The framework takes care of forwarding the WCU Object to the WCU system.

The CAP specification was created with the intention to be compatible with variety of systems. Thus many of the elements will never be used during the CAP to WCU translation. However, a number of elements in the CAP specification are required. A table of a proposed translation between CAP and WCU Object can be found in Appendix E.

[24] A US advocacy coalition of 100 member organizations dedicated to advancing emergency response
[25] Other CAP users can be found at: http://www.incident.com/cookbook/index.php/Who_Is_Using_CAP

Part IV – Evaluation

10 Evaluation

The main goal of this thesis was to provide solutions for two main problems:

The first was to make a definition of SOA as well as to compare this definition to WCU. The evaluation of this step is described in section 10.1.

The second problem was to propose a general solution of how WCU may exchange alarm information with other systems. This solution had to conform to the definition of SOA given in this thesis. The evaluation of this, second problem is described in section 10.2.

10.1 Evaluation of the Solution to the First Problem

The SOA metrics given in 3.7 are based on common elements extracted from several SOA definitions [36], [37], [38] made by a number of large standardization organizations. The metrics [1] created by OASIS, as well as other papers and publications were important resources to develop this set of metrics.

The comparison that was made can be seen as a comparison between the SOA definition in section 3.7 and a specific configuration of WCF. The outcome of this comparison is very much dependent on how SOA was defined in the previous stage. However, it was shown that the solution proposed was not conformant as it was too platform dependent.

10.2 Evaluation of the Solution to the Second Problem

10.2.1 Load Test

To estimate the performance of the proposed design, a CAP alarm generator client was built. This client connected to the service published by the WS Alarm plug-in and sent an alarm in CAP format to it. The clock was started just before the first message was sent from the alarm generator client, and stopped immediately after the last message had been received and translated to WCU format by the WS Alarm plug-in. The messages were sent from the alarm generator client using a for-loop. Time was measured using System.DateTime.Now.Ticks in .NET. This clock has an approximate resolution of 10 milliseconds [25]. Both the alarm generator client and the WCU client running the WS Alarm plug-in were executed as Windows executables locally on the same machine. The reason two separated machines were not used was due to time constraints in the evaluation phase. The test machine had an Intel Pentium processor running at 2.8 GHz and 1GB of RAM, and the operating system was Microsoft Windows XP Professional with SP2 installed.

A complete list with the values of the measurements can be found in Appendix F. The number of messages (iterations in the for-loop) was varied between 1000 messages and 1 message. The first measured value was removed due to the initialization process done the first time a connection occurs to a service.

Table 10:1 – Processing time per message in milliseconds

Number of Messages	1000	600	400	200	100	50	20	1
Processing time per message (ms)	**2.7**	**2.5**	**2.7**	**2.8**	**3.1**	**3.8**	**6.3**	**62.6**

Table 10:1 shows the median time divided by the number of messages that was sent in each category. As can be seen is there little delay even when 1000 messages were sent. To test whether the time was constrained by the sender or the receiver of the alarm message an additional test performed, measuring the time it took to send 1000 messages (1000 iterations of the for-loop). The result of the measurement showed that this time was exactly the same as the time it took from that the first message was sent until the last message was received and processed by the WS alarm plug-in. This indicates that the for-loop at the sender-side was the bottleneck in the performed test. To verify this, another test was carried out where two alarm generator clients where started, each one sending 1000 messages each at approximately the same time.

Table 10:2 – Processing time in seconds for sending 1000 messages in parallel

Test Iteration	Alarm Generator Client 1	Alarm Generator Client 2	WS Alarm plug-in	
			1000 msg.	1000 msg.
1	2.942	3.083	1.888	1.809
2	3.304	3.099	1.793	1.730
3	3.319	3.083	1.778	1.699
4	3.256	3.068	1.778	1.683
5	3.225	3.005	1.872	1.746
Median value	**3.256**	**3.083**	**1.793**	**1.730**

Table 10:2 shows the result for the test when two alarm generator clients sent 1000 messages each at approximately the same time to the WS Alarm plug-in. The median time it took for the WS Alarm plug-in to receive 1000 messages was reduced from the earlier 2.7 seconds (see Table 10:1) to approximately 1.8 seconds. This confirms that the bottleneck in the earlier performance evaluation was the sender (alarm generator client). The time it took to send alarms from the alarm generator client also increased. This is because the sender now needs to wait for the service at the WS Alarm plug-in to be available in order to send a new message.

The evaluation test described above was done mainly to estimate whether the proposed solution may introduce a bottleneck to the system or not. The systems that today are known to be candidates for integration with WCU have a maximum alarm generation of around 1000 messages during a period of 24 hours. Future demands may however require the WCU to receive and handle a much larger number of alarms than today. A more precise performance measurement would be required in such a case.

10.2.2 CAP

It was proposed that a translation be made between WCU format and CAP, as well as between CAP and other alarm formats in general. As mentioned in section 9.4 the CAP protocol is able to convert "to and from all kinds of sensor and alerting technologies" [21]. This was not evaluated in this thesis, but assumed to be true. Whether alarm information is conformant to CAP is something that has to be checked for each specific system. This thesis concentrated on making a translation between CAP and WCU format. All elements in the CAP protocol could not be matched to elements in the WCU format; however all mandatory elements translated successfully. There are also some questions about the translation that remain unsolved as the WCU architecture is still evolving.

One downside of using CAP is that it requires all other systems to use this specification. The idea is however that this should be easier than understanding and using of the WCU-format. The WCU-format is also very tight connected to the internal WCU system, something that makes it less appropriate for use as a data model for exchanging alarm information with other systems.

10.2.3 Short comings of the Proposed Solution

The proposed solution does not address how alarm messages and information is sent back to the external system. The reason that this was not included in the proposed solution is that there was no urgent need for such functionality. However, this could be done in several ways: for example the external system may publish one or more services that the WCU system can call. Another option is to have the WS Alarm plug-in publish a subscription service that the external system listens to. In this way the external system will receive all updates and other information that is needed. How this problem is to be solved depends much on what information should be exposed for external systems as well as if this information should be public for all participants, or private for a specific system.

If only **one** client running the WS Alarm plug-in is connected to an external system and **one** server is used, there will be two single points of failure; compared to one single point of failure if there had been no client and the same service would have been published by the server instead. If the client fails, no alarms may be received from the other system.

The proposed solution does not include support for communication technologies other than Web services. Although Web services can be implemented on most platforms, there might be some systems that are unable to use this technology. Another issue that has not been investigated in greater detail is how WCF's implementation of Web services conforms to how other vendors have implemented Web Services. Different tools for implementing Web services may use arbitrary data types, and encode and structure messages in different ways [35]. Thus in the future there is a need for interoperability testing.

10.2.4 Advantages of the Proposed Solution

Increased stability and load balancing may be achieved in the system by having several parallel clients running the WS Alarm plug-in. Load balancing by having external systems randomly alternate to which client it should send an alarm to, and increased stability by resending an alarm to another client if the first one fails for some reason (i.e. can not be reached or returns a error message).

Other partners can easily understand the data model for alarms and how to communicate with WCU since a standard format (CAP) is used. The description of this standard format is generally available in open documents.

It is easy to extend the plug-in with more services for accepting alarms in formats other then CAP. And since the service is published at a client and not at the server, updates and new functionalities can be added without disrupting the main system. All that the external system will have to do is to temporarily change which client it should connect to.

The solution is SOA-based and built upon SOAP over HTTP. This is a technology that is standardized and available for almost all common platforms. Additionally, a standardized data model for describing alarms (CAP) is used, - making it easy for other systems to understand how to communicate with WCU.

The solution is independent of the WCU Server. This allows the WCU Server to change without affecting the interface to external systems.

Part V – Conclusions & Future Work

11 Conclusions

The proposed design presented in this thesis solves the problem of integrating different alarm systems that might wish to send alarm information to the WCU. The solution is general in that any other system may send alarm information to WCU, as long as this alarm information is sent as a CAP message and the external system supports the Web service standards WSDL and SOAP. The solution conforms to the SOA definition presented in section 3.7.

WCF supports the concept SOA, but requires proper configuration to make it platform-independent and conformant to the SOA definition given in this thesis. A suggestion of how this can be made has been presented in the proposed solution.

By introducing a standard alarm protocol (CAP) between the WCU system and external systems, the work of translating the alarm information was made easier. Since the WCU-format is tightly connected to the WCU system, and not a stand alone alarm representation, it would be difficult to use the CAP format internally in WCU. The idea is also that by using CAP as an intermediate alarm representation, both the internal WCU-format as well as external alarm formats may change their internal alarm data representation independently from each other.

The proposed design presents a solution to how the WCU system may integrate smoothly with other systems. This makes the product well prepared for an expansion on the markets, as more and more systems will get connected to it. The solution presented in this thesis may also be used as a foundation for constructing solutions with other types of information that is to be exchanged.

The services in WCU use MSMQ for communicating internally in the system. The reason this protocol was chosen was mainly to get reliability, message queuing, and asynchronous communication. However, MSMQ as well a set of complicated initialization rules, makes it difficult for systems running on other platforms to integrate with the WCU. One solution to this problem could be to exchange the bindings in the WCU services to ones that support platform independent communication. The load test performed as a part of this thesis showed that such services can reach high performance demands. Another option of how to make it easier for other systems to integrate with the WCU is to maintain the internal

architecture and provide an external interface to the system. An example of such a solution was presented as a part of this thesis project.

The WCU system is based on a centralized architecture. A centralized system uses a single point to where all participants connect to. Explicitly, all communication is dependent on the centralized server. The idea behind Service-Oriented Architecture is to make it easier for different parts of a system to exchange information with each other independently of the other parts of the system. That is, by using a centralized architecture the WCU will be unable to utilize some of the benefits that a SOA may provide. However, a SOA could be implemented beneficial in such a system too by adding functionality (services) so that clients also may connect to each other or to other parts of the system.

12 Future Work

The solution presented in this thesis uses a static WSDL repository. This is sufficient for the given problem. However, a UDDI registry has the benefit of providing a more flexible and dynamic solution. Load-balancing and business logic are examples of functionality a UDDI registry may provide. Another reason for implementing a UDDI registry is if the connection should be dynamic and the external systems or clients using the provided services are not known in advance.

The load test presented in section 10.2.1 showed that the proposed design fulfills the performance requirements of WCU. However, these requirements might increase as the system is connected to other systems with the possibility of generating a large number of alarms. This possibility should be investigated by putting the WS Alarm plug-in under a much higher load of alarms.

The proposed design does not address false alarms or the priority of alarms. Such functionality could prove to be extremely beneficial if implemented correctly. However, how this should be done is left for further research.

The proposed solution makes external systems dependent on the CAP protocol in order to communicate with the WCU. However, this dependency can easily be broken by introducing services that accept alarm messages in other formats. Examples of other alarm related specifications are EDXL (presented in section 9.2) and VEDS (presented in section 9.3). EDXL is used for routing alarm messages to appropriate recipients and may carry with it one or more alarm messages in its body. Support for such a specification may come in handy when several systems are communicating and the intended recipient is not known in advance. VEDS is a specification for describing vehicular emergency information. Support for this protocol could be beneficial if the WCU system is to be integrated with external systems where vehicular related emergencies are common.

One demand that most likely will emerge is for other systems to subscribe to WCU alarms. The presented solution does only handle incoming alarms in CAP-format. To send back alarms, and to allow others to subscribe to CAP alarms, the solution needs to be extended. This can be done in several ways. One solution is to have the external system publishing a service to which the WCU connects. The external system could send the information about the location of a service description in a CAP message to the WCU. In this way the WCU may locate the service description, establish a connection and send back a CAP message. Another way is to have the WCU (WS Alarm plug-in) providing a subscription service to which other systems may subscribe. Which alternative that is to prefer or how this is to be done is left for further investigation.

References

Publications

[1] M. MacKenzie, K. Laskey, F. McCabe, P. Brown, and R. Metz, "Reference Model for Service Oriented Architecture 1.0 – Committee Specification 1", OASIS, 19 July 2006, Available at HTTP: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, [Accessed 5 September 2006]

[2] Yih-Cheng Lee, Chi-Ming Ma, and Shih-Chien Chou, "A Service-Oriented Architecture for design and development of middleware", In Proc. 12th Asia-Pacific Software Engineering Conference (APSEC '05), December 2005, page 217-221, doi 10.1109/APSEC.2005.16

[3] R. Perrey, M. Lycett, "Service-Oriented Architecture", in Proc. Applications and the Internet Workshops, 27-31 Jan. 2003, pp. 116-119.

[4] S. Jones, "Toward an acceptable definition of service", IEEE Software, Volume 22, Issue 3, May-June 2005, pp. 87-93, doi 10.1109/MS.2005.80

[5] S. Vinoski, "Service Discovery 101", Internet Computing, IEEE. Volume 7, Issue 1, Jan.-Feb. 2003. pp. 69-71, doi 10.1109/MIC.2003.1167342

[6] Do Van Thanh and I. Jorstad, "A Service-Oriented Architecture framework for mobile services", in Proc. Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ E-Learning on Telecommunications Workshop, AICT/SAPIR/ELETE 2005, 17-20 July 2005, pp. 65-70, doi 10.1109/AICT.2005.14

[7] W.T. Tsai, Chun Fan, Yinong Chen, R. Paul, and Jen-Yao Chung, "Architecture classification for SOA-based applications". Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 24-26 April 2006, pp. 295-302, doi 10.1109/ISORC.2006.18

[8] H. Zhu, "Building reusable components with service-oriented architectures", IRI - 2005 IEEE International Conference on Information Reuse and Integration, 15-17 Aug. 2005, pp. 96-101. doi 10.1109/IRI-05.2005.1506456

[9] D.R. de Almeida, C. de Souza Baptista, E.R. da Silva, C.E.C. Campelo, H.F. de Figueiredo, Y.A. Lacerda, "A context-aware system based on service-oriented architecture", 20th International Conference on Advanced Information Networking

and Applications, Volume 1, 18-20 April 2006, pp. 205-120, doi 10.1109/AINA.2006.16

[10] M. Jiang, A. Willey, ”Service-Oriented Architecture for Deploying and Integrating Enterprise Applications”, 5th Working IEEE/IFIP Conference on Software Architecture, 06-10 Nov. 2005, pp. 272-273, doi 10.1109/WICSA.2005.60

[11] A. Uyar, W. Wu, H. Bulut, G. Fox, ”Service-oriented architecture for a scalable videoconferencing system”, International Conference on Pervasive Services, 11-14 July 2005, pp. 445-448, doi 10.1109/PERSER.2005.1506564

[12] “Real-World SOA: Definition, Implementation and Use of SOA with CentraSite™”, White Paper, Fujitsu Computer Systems Corporation and Software AG, 11 July 2006, Available at HTTP: http://www.soaworks.com/pdf/White_Paper_CentraSite.pdf, [Accessed 7 September 2006]

[13] M.P. Papazoglou, ”Service-oriented computing: concepts, characteristics and directions”, in Proc. of the Fourth International Conference on Web Information Systems Engineering, 10-12 Dec. 2003, pp. 3-12, doi 10.1109/WISE.2003.1254461

[14] J. Kessler, “Produktspecifikation WCU 2.0”, L/SC-06:0006, [Company confidential document], Saab Security Systems, 11 September 2006

[15] D. Chappell, “Understanding BizTalk Server 2006”, Microsoft Corporation, August 2005, Available at HTTP: http://download.microsoft.com/documents/australia/windowsserversystem/biztalk2006/Understanding_BTS06.pdf, [Accessed 20 November 2006]

[16] “Security in BizTalk Server 2004”, White Paper, Microsoft, December 2003, Available at HTTP: http://www.microsoft.com/technet/prodtechnol/biztalk/2004/whitepapers/security.mspx, [Accessed 20 November 2006]

[17] “CoordCom™ Public Safety Communication Center”, Brochure, Ericsson Microwave Systems AB, Available at HTTP: http://www.sos112.info/ericsson/CoordCom%20PS-05%20Broschure%20lo.pdf, [Accessed 20 November 2006]

[18] J. Kessler, “White Paper WCU 2.0”, [Company confidential document], Saab Security Systems, 29 October 2006

[19] "A Framework for Justice Information Sharing: Service-Oriented Architecture (SOA)", The Global Infrastructure/Standards Working Group, 9 December 2004, Available at HTTP: http://it.ojp.gov/documents/20041209_SOA_Report.pdf, [Accessed 20 November 2006]

[20] "Emergency Data Exchange Language (EDXL) Distribution Element, v 1.0", OASIS Standard EDXL-DE v1.0, OASIS, 1 May 2006, Available at HTTP: docs.oasis-

open.org/emergency/edxl-de/v1.0/EDXL-DE_Spec_v1.0.pdf, [Accessed 20 November 2006]

[21] "Common Alerting Protocol, v 1.1", OASIS Standard CAP-V1.1, OASIS, October 2005, Available at HTTP: http://www.oasis-open.org/committees/download.php/14759/emergency-CAPv1.1.pdf, [Accessed 20 November 2006]

[22] "C4ISR for Network-Oriented Defense", White Paper, Ericsson, March 2006, doi: 284 23-3064 Uen Rev A, Available at HTTP: http://www.ericsson.com/technology/whitepapers/3064_C4isr_b.pdf, [Accessed 20 November 2006]

[23] S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama, "Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI", 2 ed., Sams, June 28, 2004, ISBN: 0672326418

[24] "Computer Aided Dispatch Software", Fact Sheet, Motorola, DJS-MCAD-1, 2004, Available at HTTP: http://www.motorola.com/governmentandenterprise/contentdir/en_US/Files/ProductInformation/Motorola_CAD_final.pdf, [Accessed 20 November 2006]

[25] "LogMateAMS", Brochure, TiPSIncorporated, Available at HTTP: http://www.tipsweb.com/downloads/LogMateAMS_Brochure.pdf, [Accessed 16 November 2006]

[26] C. McMurtry, M. Mercuri, and N. Watling, "Microsoft Windows Communication Foundation: Hands-On", SAMS Publishing, May 26, 2006. ISBN 0-672-32877-1

[27] ComCARE Alliance, "Vehicular Emergency Data Set (VEDS) version 2.0", Recommendation, March 2004, Available at HTTP: http://www.comcare.org/uploads/VEDS_2.0.pdf, [Accessed 19 December 2006]

[28] Health and Safety Executive (HSE), "Better Alarm Handling", HSE information sheet, Available at HTTP: http://www.hse.gov.uk/pubns/chis6.pdf, [Accessed 19 December 2006]

[29] Transport Layer Security Working Group – IETF, "The SSL Protocol Version 3.0", Internet-Draft, Available at HTTP: http://wp.netscape.com/eng/ssl3/draft302.txt, [Accessed 21 December 2006]

[30] OASIS, Web Services Security: SOAP Message Security 1.1, Standard Specification, 1 February 2006, Available at HTTP: http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf, [Accessed 21 December 2006]

[31] S. Anderson, Et al, Web Services Trust Language (WS-Trust), February 2005, Available at HTTP: ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, [Accessed 21 December 2006]

[32] S. Anderson, Et al, Web Services Secure Conversation Language (WS-SecureConversation), February 2005, Availabel at HTTP: ftp://www6.software.ibm.com/software/developer/library/ws-secureconversation.pdf, [Accessed 21 December 2006]

[33] H. Lockhart, Et al, Web Services Federation Language (WS-Federation) Version 1.1, December 2006, Available at HTTP: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf, [Accessed 21 December 2006]

[34] Railway Industry Advisory Committee (RIAC) Human Factors Working Group, "Alarm Handling", information sheet, Available at HTTP: http://www.rail-reg.gov.uk/upload/pdf/hf-alarm-handling.pdf, [Accessed 16 January 2007]

[35] V. Paulsson, Web Service Interoperability Issues – From a .NET perspective, Master of Science Thesis, IMIT/TSLAB, November 2005, Available at HTTP: http://web.it.kth.se/~jm/theses/paulsson.pdf, [Accessed 25 January 2007]

Web-pages

[36] Open Management Group SOA definition, [online], Available at HTTP: http://colab.cim3.net/cgi-bin/wiki.pl?OMGSoaGlossary, [Accessed 21 November 2006]

[37] World Wide Web Consortium SOA definition, [online], Available at HTTP: http://colab.cim3.net/cgi-bin/wiki.pl?WwwCSoaGlossary, [Accessed 21 November 2006]

[38] OpenGroup SOA definition, [online], Available at HTTP: http://www.opengroup.org/projects/soa/doc.tpl?gdid=10632, [Accessed 21 November 2006]

[39] Sveriges Radio, [online], Available at HTTP: http://www.sr.se/cgi-bin/stockholm/nyheter/artikel.asp?artikel=740441, [Accessed 19 December 2006]

[40] TiPS Incorporated, [online], Available at HTTP: http://www.tipsweb.com/products/logmate/, [Accessed 19 December 2006]

[41] Global Justice XML Data Model 3.0.3, [online], Available at HTTP: http://it.ojp.gov/jxdm/3.0.3/index.html, [Accessed 19 December 2006]

[42] Sun Ridge Systems Inc. Integrated Public Safety Software , [online], Available at HTTP: http://www.sunridgesystems.com/, [Accessed 19 December 2006]

[43] Motorola Computer Aided Dispatch (CAD) System, [online], Available at HTTP: http://www.motorola.com/governmentandenterprise/northamerica/en-

us/public/functions/browsesolution/Browsesolution.aspx?navigationpath=id_803i/id_1423i/id_1292i, [Accessed 19 December 2006]

[44] Tiburon Inc., [online], Available at HTTP: http://www.tiburoninc.com/, [Accessed 20 December 2006]

[45] WordNet, Princeton University, [online], Available at HTTP: http://wordnet.princeton.edu/perl/webwn?s=self-contained, [Accessed 20 December 2006]

[46] Web Service Description Language (WSDL), [online], Available at HTTP: http://www.w3.org/TR/wsdl, [Accessed 21 December 2006]

[47] Universal Description, Discovery, and Integration (UDDI), [online], Available at HTTP: http://www.uddi.org/, [Accessed 21 December 2006]

[48] Basic Profile Version 1.1, [online], Available at HTTP: http://www.ws-i.org/Profiles/BasicProfile-1.1.html, [Accessed 21 December 2006]

[49] Security Overview, MSDN, Microsoft Windows SDK, [online], Available at HTTP: http://msdn2.microsoft.com/en-us/library/ms735093.aspx, [Accessed 21 December 2006]

[50] Web Service Interoperability Organization (WS-I), [online], Available at HTTP: http://www.ws-i.org/, [Accessed 21 December 2006]

[51] DateTime.Now property, MSDN, Microsoft, [online], Available at HTTP: http://msdn2.microsoft.com/en-us/library/system.datetime.now.aspx, [Accessed 26 January 2007]

Appendices

Appendix A – SOA Definitions

OASIS[26]

Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.

In general, entities (people and organizations) create capabilities to solve or support a solution for the problems they face in the course of their business. It is natural to think of one person's needs being met by capabilities offered by someone else; or, in the world of distributed computing, one computer agent's requirements being met by a computer agent belonging to a different owner. There is not necessarily a one-to-one correlation between needs and capabilities; the granularity of needs and capabilities vary from fundamental to complex, and any given need may require the combining of numerous capabilities while any single capability may address more than one need. The perceived value of SOA is that it provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.

Weopedia

(n.) Abbreviated *SOA*, an application architecture in which all functions, or services, are defined using a description language and have invokable interfaces that are called to perform business processes. Each interaction is independent of each and every other interaction and the interconnect protocols of the communicating devices (i.e., the infrastructure components that determine the communication system do not affect the interfaces). Because interfaces are platform-independent, a client from any device using any operating system in any language can use the service

[26] The definition presented is a part extracted from [1]

TechEncyclopedia

(**S**ervice-**O**riented **A**rchitecture) An umbrella term for a standardized interface between software so that one program can utilize the functions (services) of another program. SOA typically refers to Web services.

Formerly called a "distributed objects" architecture, the SOA term was coined at the turn of the century as Web services were evolving. CORBA and DCOM are examples of earlier SOAs. See Web services, CORBA and DCOM.

SearchWebServices.com

A service-oriented architecture (SOA) is the underlying structure supporting communications between services. In this context, a service is defined as a unit of work to be performed on behalf of some computing entity, such as a human user or another program. SOA defines how two computing entities, such as programs, interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. Service interactions are defined using a description language. Each interaction is self-contained and loosely coupled, so that each interaction is independent of any other interaction.

Dearing, 2003

SOA takes the existing software components residing on the network and allows them to be published, invoked and discovered by each other. SOA allows a software programmer to model programming problems in terms of services offered by components to anyone, anywhere over the network"

Appendix B – WCF and Web Services

Table B:1 – Microsoft WCF Standard Bindings (msdn.microsoft.com)

Binding	Interoperability	Security (Default)	Session (Default)	Transactions	Duplex	Encoding (Default)
BasicHttpBinding	Basic Profile 1.1	(None), Transport, Message, Mixed	None, (None)	(None)	n/a	Text, (MTOM)
WSHttpBinding	WS	Transport, (Message), Mixed	(None), Transport, Reliable Session	(None), Yes	n/a	Text, (MTOM)
WSDualHttpBinding	WS	(Message), None	(Reliable Session)	(None), Yes	Yes	Text, (MTOM)
WSFederationHttpBinding	WS-Federation	(Message), Mixed, None	(None), Reliable Session	(None), Yes	No	Text, (MTOM)
NetTcpBinding	.NET	(Transport), Message, None, Mixed	Reliable Session, (Transport)	(None), Yes	Yes	Binary
NetNamedPipeBinding	.NET	(Transport), None	None, (Transport)	(None), Yes	Yes	Binary
NetMsmqBinding	.NET	Message, (Transport), Both	(None)	(None), Yes	No	
NetPeerTcpBinding	Peer	(Transport)	(None)	(None)	Yes	
MsmqIntegrationBinding	MSMQ	(Transport)	(None)	(None), Yes	n/a	

Interoperability	Names the protocol or technology with which the binding ensures interoperation.
Security	Specifies how the channel is secured: • None: The SOAP message is not secured and the client is not authenticated. • Transport: Security requirements are satisfied at the transport layer. • Message: Security requirements are satisfied at the message layer. • Mixed: Claims are carried in the message; integrity and confidentiality requirements are satisfied by the transport layer.
Session	Specifies whether this binding supports session contracts.
Transactions	Specifies whether transactions are enabled.
Duplex	Specifies whether duplex contracts are supported. Note this feature requires support for Sessions in the binding.
Encoding	Specifies the wire format of the message. Allowable values are: • Text: for example UTF8. • Binary • MTOM: a method for efficiently encoding binary XML elements within the context of a SOAP envelope.
Streaming	Specifies whether the message streaming is supported.

Table B:2 – Basic Profile 1.1 Core Web Services Standards

Core Web service standards included in the Basic Profile 1.1 provided by the Web Services Interoperability Organization
SOAP 1.1
WSDL 1.1
UDDI 2.0
XML 1.0 (Second Edition)
XML Schema Part 1: Structures
XML Schema Part 2: Data types
RFC2246: The Transport Layer Security Protocol Layer Version 1.0
RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile
RFC2616: Hypertext Transfer Protocol 1.1
RFC2818: HTTP over TLS Transport Layer Security
RFC2965: HTTP State Management Mechanism
The Secure Sockets Layer Protocol Version 3.0

Appendix C – Object Models and Schemas

C.1 CAP v1.1 XML Object Model

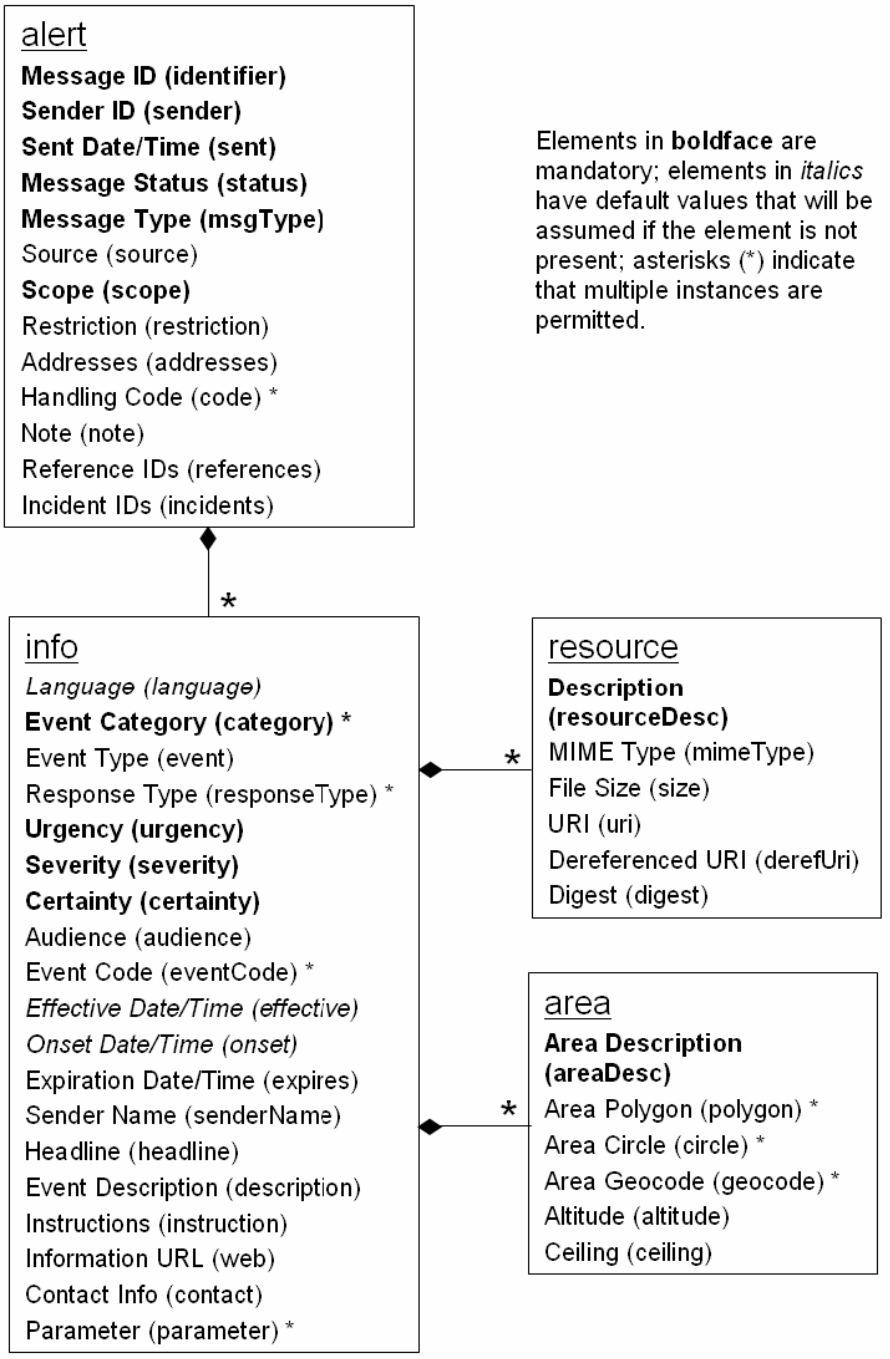

Figure C:1 – CAP XML Object Model [21]

C.2 EDXL XML Object Model

Elements in boldface are mandatory; asterisks (*) indicate that multiple instances are permitted [20].

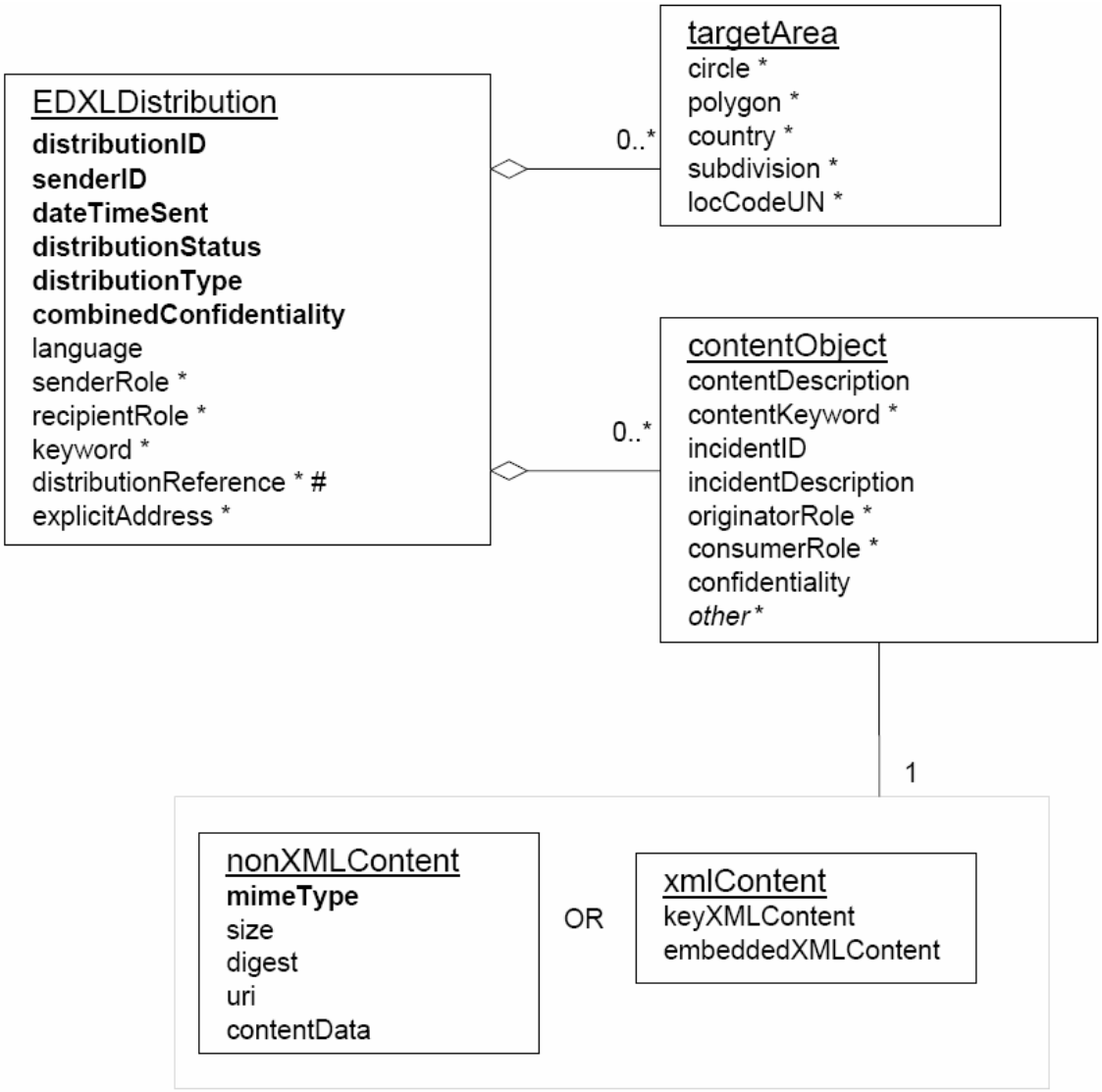

Figure C:2 – EDXL XML Object Model [20]

C.3 WCU Object XML Schema

complexType **Indication**

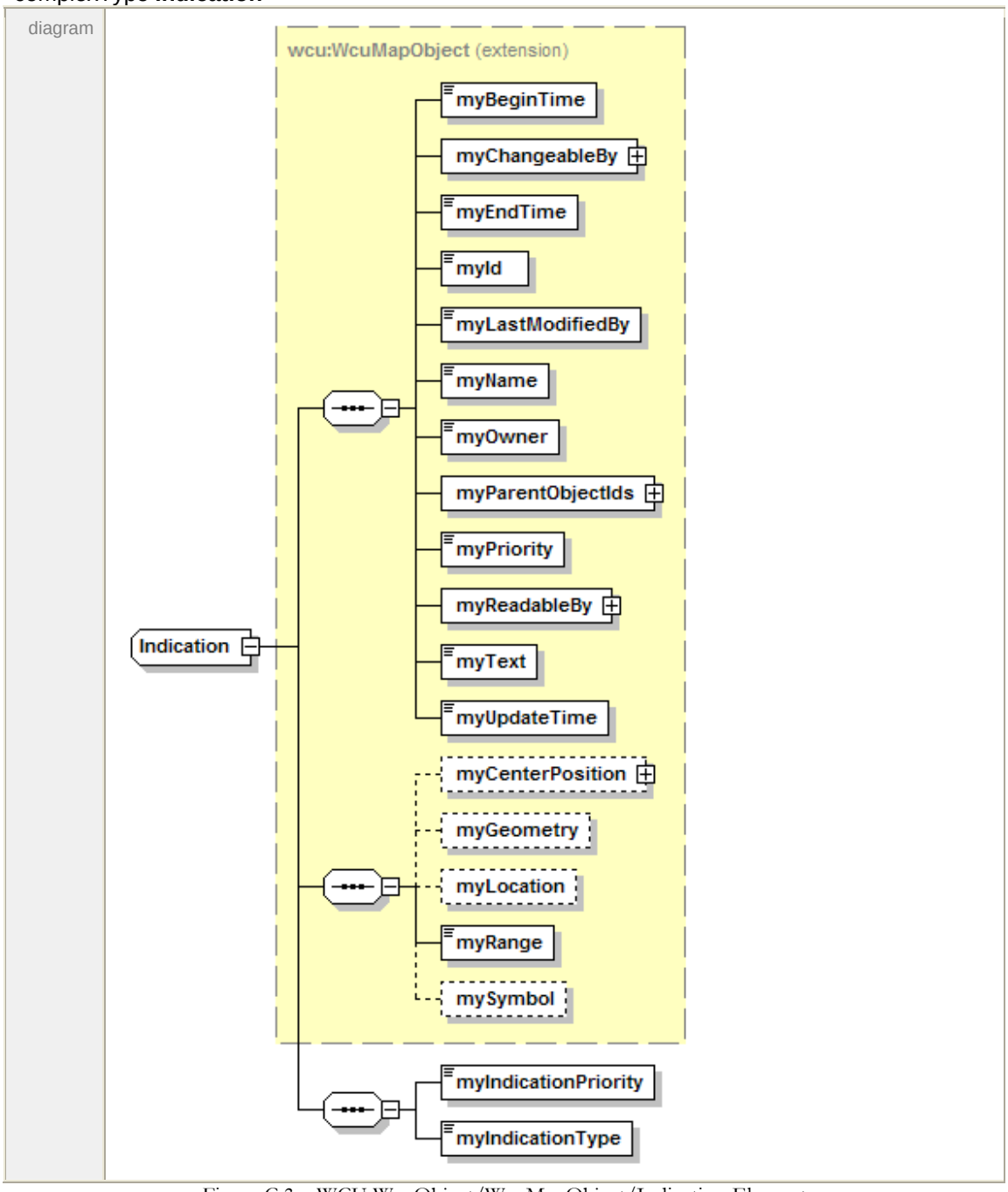

Figure C:3 – WCU WcuObject/WcuMapObject/Indication Element

C.4 SOS InfoServer XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CaseFolder" nillable="true" type="CaseFolder" />
  <xs:complexType name="CaseFolder">
    <xs:complexContent mixed="false">
      <xs:extension base="DBCaseFolder">
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1" name="Logs" type="ArrayOfLog" />
          <xs:element minOccurs="0" maxOccurs="1" name="Case" type="Case" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DBCaseFolder">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="CallCenterId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="CallCenterName" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="CaseFolderId" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="CaseFolderStatusId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="CaseFolderStatus" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="IncidentCommander" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="MedicalIncidentOfficer" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ArrayOfLog">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Log" nillable="true" type="Log" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Log">
    <xs:complexContent mixed="false">
      <xs:extension base="DBCaseFolderLog" />
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DBCaseFolderLog">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="Created" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="LogRowTypeid" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="LogText" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="OrderNo" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="RowCancelled" type="xs:boolean" />
      <xs:element minOccurs="1" maxOccurs="1" name="RowCancelledTime" type="xs:dateTime" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Case">
    <xs:complexContent mixed="false">
      <xs:extension base="DBCase">
        <xs:sequence>
          <xs:element minOccurs="0" maxOccurs="1" name="Missions" type="ArrayOfMission" />
          <xs:element minOccurs="0" maxOccurs="1" name="Logs" type="ArrayOfLog" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DBCase">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="PersonFirstName" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="SexId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="SexName" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="AgeMeasureTypeId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="AgeMeasureTypeName" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="Age" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="DiagnosisId" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="DiagnosisName" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="PhoneNumberCountryCode" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="PhoneNumberAreaCode" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="PhoneNumber" type="xs:string" />
```

```
<xs:element minOccurs="0" maxOccurs="1" name="PersonStreet" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonStreetNo" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonBlock" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonEntrance" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonFloor" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonApartment" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonLocality" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="PersonMunicipalityId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonMunicipalityName" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonPostCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonCommunity" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="PersonNote" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="StreetNo" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="Block" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="Entrance" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="Floor" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="Apartment" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="PublicPlace" type="xs:boolean" />
<xs:element minOccurs="0" maxOccurs="1" name="ToStreetNo" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ToBlock" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ToEntrance" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ToFloor" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ToApartment" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ToCommunity" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="ObjectTiedDatetime" type="xs:dateTime" />
<xs:element minOccurs="1" maxOccurs="1" name="ServiceCaseStartTime" type="xs:dateTime" />
<xs:element minOccurs="1" maxOccurs="1" name="ServiceCaseEndTime" type="xs:dateTime" />
<xs:element minOccurs="0" maxOccurs="1" name="AddressNote" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="AlarmArrivalTime" type="xs:dateTime" />
<xs:element minOccurs="1" maxOccurs="1" name="AlarmCategoryId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmCategoryName" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="AlarmCatOrderNo" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmDetectorCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmDetectorText" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmEventCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmEventText" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmHandlingTypeName" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmOriginalCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmResetCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmResetStatusName" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmSectionCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmSectionText" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="AlarmSimulated" type="xs:boolean" />
<xs:element minOccurs="1" maxOccurs="1" name="AlarmStatusId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmTransmitterCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmTransmitterPartCode" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="AlarmTransmitterTypeCode" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseId" type="xs:int" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseIndex1" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseIndex1Name" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseIndex2" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseIndex2Name" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseIndex3" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseIndex3Name" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseIndexComment" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="CasePriority" type="xs:int" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseStatusId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseStatus" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="CaseTypeArea" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="CaseTypeId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="Casetype" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="Community" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ComplCategoryText" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ContractNumber" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="Created" type="xs:dateTime" />
<xs:element minOccurs="1" maxOccurs="1" name="Finished" type="xs:dateTime" />
<xs:element minOccurs="0" maxOccurs="1" name="FinishedBy" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="HandOverTime" type="xs:dateTime" />
<xs:element minOccurs="0" maxOccurs="1" name="Locality" type="xs:string" />
<xs:element minOccurs="1" maxOccurs="1" name="MunicipalityId" type="xs:int" />
<xs:element minOccurs="0" maxOccurs="1" name="MunicipalityName" type="xs:string" />
<xs:element minOccurs="0" maxOccurs="1" name="ObjectName" type="xs:string" />
```

```
        <xs:element minOccurs="0" maxOccurs="1" name="Orderer" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="PersonName" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="PickUpTime" type="xs:dateTime" />
        <xs:element minOccurs="1" maxOccurs="1" name="PositionModified" type="xs:dateTime" />
        <xs:element minOccurs="1" maxOccurs="1" name="PositionRefSystemId" type="xs:int" />
        <xs:element minOccurs="0" maxOccurs="1" name="PostCode" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="Preordered" type="xs:boolean" />
        <xs:element minOccurs="0" maxOccurs="1" name="RouteDirections" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="SocialSecurityNumber" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="Street" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="ToCaseTypeArea" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="ToLocality" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="ToMunicipalityId" type="xs:int" />
        <xs:element minOccurs="0" maxOccurs="1" name="ToMunicipalityName" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="ToPostCode" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="ToStreet" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="CustomerNumber" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="Urgent" type="xs:boolean" />
        <xs:element minOccurs="1" maxOccurs="1" name="XCoordinate" type="xs:double" />
        <xs:element minOccurs="1" maxOccurs="1" name="YCoordinate" type="xs:double" />
        <xs:element minOccurs="1" maxOccurs="1" name="ZCoordinate" type="xs:double" />
        <xs:element minOccurs="0" maxOccurs="1" name="StatusName" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ArrayOfMission">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="Mission" nillable="true" type="Mission" />
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Mission">
      <xs:complexContent mixed="false">
        <xs:extension base="DBMission" />
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="DBMission">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1" name="StationCode" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="FalseAlarm" type="xs:boolean" />
        <xs:element minOccurs="0" maxOccurs="1" name="FalseAlarmReasonCode" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="ForeignCountyName" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="ForeignCountyTransport" type="xs:boolean" />
        <xs:element minOccurs="1" maxOccurs="1" name="LastStatusReport" type="xs:dateTime" />
        <xs:element minOccurs="1" maxOccurs="1" name="Mileage" type="xs:int" />
        <xs:element minOccurs="1" maxOccurs="1" name="MissionStarted" type="xs:dateTime" />
        <xs:element minOccurs="0" maxOccurs="1" name="ResourceCode" type="xs:string" />
        <xs:element minOccurs="0" maxOccurs="1" name="Status" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1" name="StatusId" type="xs:int" />
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Appendix D – Microsoft BizTalk™ Server

Microsoft's BizTalk™ Server 2006 is based on an XML Framework and is primarily for Business Process Management[27] (BPM) and application integration. It does not by itself add any new protocols but relies on XML and XML Schemas. BizTalk™ uses an internal XML-format for all communication that takes place inside the server.

D.1 Server Architecture

The BizTalk™ Server consists of two main parts: a messaging component (MessageBox) and an orchestration (logic) component [15]. The messaging component is for communication with other systems by using adapters, while the orchestration component provides all the logic. Three additional components are a business rules engine, a monitoring tool for the engine and orchestrations, and an authentication facility for mapping authentication information for use with non-Windows based systems. On top of this is a non-technical business activity monitoring tool, as well as a tool for set up and managing business activity services.

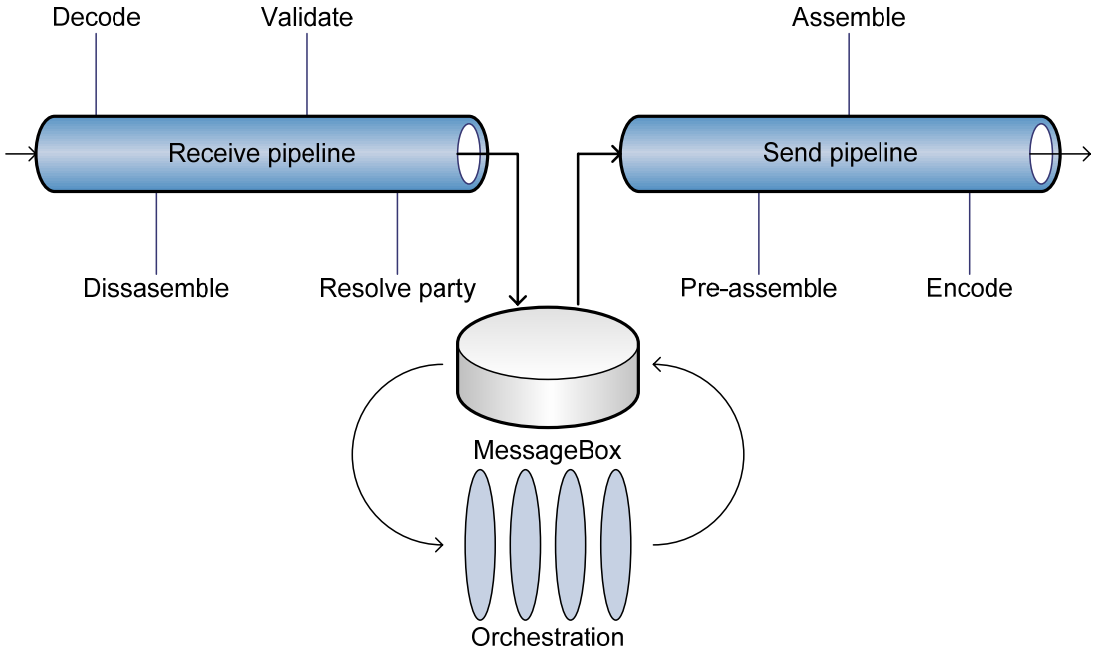

Figure D:1 – Microsoft BizTalk™ Server engine [15]

Figure D:1 describes the core functionality of the BizTalk™ Server engine. The receive pipeline consist of four stages of message processing; decode, disassemble, validate, and resolve party. When a message has been processed it is sent to the MessageBox. The message is then sent to the orchestration subscribing to the corresponding message and then processed according to the orchestration rules and sent to the send pipeline. The send

[27] Automation of manual processes and the subsequent optimization of those processes

pipeline consist of a pre-assemble stage, an assemble stage, and an encoding stage. Interested readers are referred to [15] for a more detailed description of the core engine.

D.2 Adapters and Accelerators

Microsoft BizTalk uses adapters and accelerators. Adapters are used to extend functionality by enabling the Microsoft BizTalk Server to connect to other servers and applications. At the moment 23 application and technology adapters are supported. Among the most interesting are adapters for IBM DB2, Microsoft Windows file system, FTP, HTTP, POP3, SMTP, SOAP, SQL, and Web Services. See Appendix B for a complete list of available adapters.

The so called accelerators in Microsoft BizTalk Server are used to provide developers with tools, data schemas, and processes to make development within these areas less time consuming. Table 7.1 lists the five accelerators available at the moment.

Table D:1 – BizTalk Server 2006 Accelerators

Accelerator	Explanation
Cactus GDS	The Global Data Synchronization (GDS) Network is a consortium of interoperable data pools and a global registry (the GS1 Global Registry™) that enable companies around the globe to exchange standardized and synchronized supply chain data with their trading partners.[28]
HIPAA	The Health Insurance Portability and Accountability Act of 1996 (HIPAA) compliance is a health insurance act used in the United States.
HL7	Health Level Seven was founded in 1987 and is from august 2006 an ISO standard framework for exchange, integration, sharing, and retrieval of health information.
SWIFT	The Society for Worldwide Interbank Financial Telecommunication (SWIFT) is an industry-owned co-operative supplying standardized messaging services and interface software to financial institutions.
RosettaNet	RosettaNet is a globally supported organization that develops universal standards for the supply chain.

D.3 Orchestrations and Business Rules Engine

The logic of business processes can be implemented without any programming using a graphical tool. This allows non-technical persons to be more active in the business process and to implement these as orchestrations. For developers, Microsoft BizTalk also has graphical tools for creating XML Schemas, for defining translations (using XSLT) between those schemas, and for implementing the logic of business processes.

D.4 Management and Monitoring

BizTalk™ has an administration console which provides monitoring machines and the engine to keep a record of what is going on as well as reporting failures.

[28] See www.gs1.org

D.5 Supported Platforms and Software Requirements

Microsoft Windows 2000, Microsoft Windows XP, or Microsoft Windows Server 2003 is required to run Microsoft BizTalk™ Server 2006. To use databases Microsoft SQL Server 2000 or 2005 is required. Since the BizTalk Server 2006 is based on .NET Framework 2.0 Microsoft's Visual Studio is required for the development of services.

D.6 Integration with Other Software

While Visual Studio may be used for developing Web Services, Microsoft BizTalk Server 2006 is more for orchestrating them into business processes. To use Windows Communication Foundation with Microsoft BizTalk instead of ASP .NET Web Services, an adapter is available for download.

D.7 Security Aspects

In Microsoft BizTalk security is achieved by protecting the privacy of the system elements; authenticating the information, participants, and processes that enter and leave the system; and authorizing the access to and use of resources in the system [16].

D.8 Microsoft BizTalk Server 2006 Conclusions

The Microsoft BizTalk Server 2006 provides many functions and a user-friendly interface for designing and implementing service orchestration and interoperability with other systems. Its strengths come with large systems that have many interacting parts and where business logic or other logic is involved in the exchange of messages. However, for simple systems with only one or a few interacting parts and where there is no complex orchestration needed, the same functionality can without difficulty be created without the use of Microsoft's BizTalk Server.

Microsoft's BizTalk Server could be configured to communicate alarm information directly with the WCU Server and at the same time provide several interfaces outwards. This would however mean that BizTalk has to perform all authorization issues in the same way that WCU clients does, something that is difficult to achieve since BizTalk has no support for stateful mode. This is unless BizTalk-specific services are introduced at the WCU Server. This is however something that is not desired since the idea is to keep the server lightweight.

Another reason for not using Microsoft BizTalk Server in the proposed design is the price associated with it. A higher cost for the WCU system would lead to fewer potential customers, something that is avoided if possible.

The final conclusion is that Microsoft's BizTalk Server will not be used in the proposed design.

D.9 Microsoft BizTalk Server 2006 Adapters

Table D:2 – Microsoft BizTalk Server 2006 Adapters[29]

Adapter	Description	Supported Versions
SAP	Enables exchange of Intermediate Document (IDOC), BAPI, and Request for Comments (RFC) messages between BizTalk Server and an SAP R/3 system.	SAP R/3 4.x and R/3 6.20 (Enterprise)
PeopleSoft Enterprise	Enables exchange of Component Interface (CI) messages between BizTalk Server and a PeopleSoft system.	PeopleTools Versions 8.17.02, 8.43, and 8.45
JD Edwards OneWorld XE	Enables exchange of Business Function messages between BizTalk Server and a JD Edwards OneWorld system.	B7.3.3.3 with SP23
JD Edwards EnterpriseOne	Enables exchange of Business Function messages between BizTalk Server and a JD Edwards EnterpriseOne system.	8.10 with Tools Release 8.94
ODBC Adapter for Oracle Database	Enables reading and writing information from and to an Oracle Server database.	Oracle 8i (8.1.6.0), 9i (9.2.0.1), or 10g
Siebel eBusiness Applications	Enables exchange of Business Components and Business Service messages between BizTalk Server and a Siebel eBusiness Application.	6.2.1 with patch 110 or higher, 7.0, 7.5.*, 7.7.*, and 7.8.*
TIBCO Rendezvous	Enables exchange of XML and binary data format messages between BizTalk Server and TIBCO Rendezvous.	7.3
TIBCO Enterprise Message Service	Enables exchange of XML and binary data format messages between BizTalk Server and a TIBCO EMS server providing a tightly integrated and reliable application infrastructure.	4.2
Host Applications*	Enables data exchange between BizTalk Server and IBM mainframe zSeries (CICS and IMS) and midrange iSeries (AS/400) server programs.	Not applicable
IBM DB2*	Enables reading and writing information from and to IBM mainframe DB2 for z/OS, IBM midrange DB2/400, and IBM DB2 Universal Database for open platforms (AIX, Linux, Solaris, and Windows).	Not applicable
Host Files*	Enables data exchange between BizTalk Server and IBM mainframe zSeries VSAM datasets and IBM midrange iSeries AS/400 physical files.	Not applicable
WebSphere MQ (Client Based)*	Enables exchange of messages between BizTalk Server and IBM WebSphere MQ using the WebSphere MQ Base Client (non-transactional) or WebSphere MQ Transaction Extended Client APIs.	5.3 with Fix Pack 10 or higher and 6.0 with Fix Pack 1.1 or higher
WebSphere MQ	Enables exchange of messages between BizTalk Server and IBM WebSphere MQ.	5.3 with Fix Pack 10 or higher and 6.0 with Fix

[29] See www.microsoft.com/biztalk for latest version

Adapter	Description	Supported Versions
		Pack 1 or higher
MSMQ/MSMQT	Enables sending and receiving messages by using BizTalk Message Queuing (MSMQT), an implementation of the Microsoft Message Queue (MSMQ) protocol that sends and receives MSMQ messages to and from the Message Box database.	2.0 and 3.0
Base EDI	Enables sending and receiving messages by using the American National Standards Institute (ANSI) X-12 and Electronic Data Interchange for Administration, Commerce, and Trade (EDIFACT) standards.	Not applicable
FILE	Enables reading from and writing to files in the Microsoft Windows file system.	Not applicable
FTP	Enables exchange of files between BizTalk Server and FTP servers.	Not applicable
HTTP	Enables sending and receiving information by using HTTP. The BizTalk Server 2004 engine exposes one or more URLs to enable other applications to send data to it, and it can use this adapter to send data to other URLs.	Not applicable
POP3	Enables receiving messages from a POP3 mailbox into BizTalk Server by using the POP3 protocol.	Not applicable
SMTP	Enables sending messages between BizTalk Server and an SMTP gateway by using Simple Mail Transfer Protocol (SMTP).	Not applicable
SOAP	Enables sending and receiving messages by using SOAP over HTTP enabling BizTalk Server to interact in a Web services world.	Not applicable
SQL	Enables reading and writing information from and to a Microsoft SQL Server database.	Not applicable
Web Services Enhancements (WSE) 2.0	Enables more secure Web services (WS-Security, WS-Trust, WS-SecureConversation, WS-SecurityPolicy, and WS-Policy) with BizTalk Server 2004.	2.0
Windows SharePoint Services	Enables the exchange of XML and binary messages between BizTalk Server and SharePoint document libraries.	Windows SharePoint Services 2.0 with Service Pack 2

Appendix E – CAP to WCU Translation

Elements marked with boldface in the following tables (Table E:1) indicate that the corresponding element is **required** by the CAP protocol. The information presented in Table E:1 might come to change since the specification for the internal WCU architecture is still under development.

Table E:1 – CAP to WCU Translation Tables

alert	Definition	WCU Object
identifier	The identifier of the alert message	Maps to ExternalIndication.ExternalIndicationId
sender	The identifier of the sender of the alert message	Maps to ExternalIndication.ExternalSystemName
sent	The time and date of the origination of the alert message	Maps to ExternalIndication.Sent
status	The code denoting the appropriate handling of the alert message	Actual, Exercise, System, Test, Draft. WCU Plug-in will make decisions from this value (Only Actual, Exercise and System will be forwarded to WCU)
msgType	The code denoting the nature of the alert message	Alert (NewObject), Update (Update referred ExternalIndication), Cancel (Delete?), Ack (Not used), Error (Not used). Decide action for WCU.
source	The text identifying the source of the alert message	Maps to ExternalIndication.ExternalSource
scope	The code denoting the intended distribution of the alert message	All messages are public in WCU.
restriction	The text describing the rule for limiting distribution of the restricted alert message	All messages are public in WCU.
addresses	The group listing of intended recipients of the private alert message	All messages are public in WCU.
code	The code denoting the special handling of the alert message	Not used in Wcu internally.
note	The text describing the purpose or significance of the alert message	Primarily for Cancel and Error messages. Not used in WCU internally.
references	The group listing identifying earlier message(s) referenced by the alert message	Used for decisions on msgType i.e. Update and Cancel.
incidents	The group listing naming the referent incident(s) of the alert message	Not used in Wcu internally. This is done manually by operator.

info	Definition	WCU Object
language	The code denoting the language of the info subelement of the alert message	Not used by WCU.
category	The code denoting the category of the subject event of the alert message	Maps to Indication. Indication Type.
event	The text denoting the type of the subject event of the alert message	!?
responseType	The code denoting the type of action recommended for the target audience.	Not used Wcu internally.
urgency	The code denoting the urgency of the subject event of the alert message	Maps to Indication.IndicationPriority
severity	The code denoting the severity of the subject event of the alert message	Maps to Indication.IndicationPriority
certainty	The code denoting the certainty of the subject event of the alert message	Maps to Indication.IndicationPriority
audience	The text describing the intended audience of the alert message	Not used by WCU.
eventCode	A system specific code identifying the event type of the alert message	Not used by WCU? Name value parameter
effective	The effective time of the information of the alert message	Not used by WCU.
onset	The expected time of the beginning of the subject event of the alert message	Maps to WcuObject.BeginTime.
expires	The expiry time of the information of the alert message	Not used by WCU.
senderName	The text naming the originator of the alert message	Maps to ExternalSystem.Name
headline	The text headline of the alert message	Maps to WcuObject.Name
description	The text describing the subject event of the alert message	Maps to WcuObject.Text
instruction	The text describing the recommended action to be taken by recipients of the alert message	Not used by WCU.
web	The identifier of the hyperlink associating additional information with the alert message	Not used by WCU.
contact	The text describing the contact for follow-up and confirmation of the alert message	Not used by WCU.
parameter	A system specific additional parameter associated with the alert message	Not used by WCU.

resource	Definition	WCU Object
resourceDesc	The text describing the type and content of the resource file	Not used by WCU.
mimeType	The identifier of the MIME content type and sub-type describing the resource file	Maps to WcuFile.MIMEType.
size	The integer indicating the size of the resource file	Not used by WCU.
uri	The identifier of the hyperlink for the resource file	Maps to WcuFile.FileName
derefUri	The base-64 encoded data content of the resource file	Mapps to WcuFile. File byte[]
digest	The code representing the digital digest (“hash”) computed from the resource file	Maybe used for error check

area	Definition	WCU Object
areaDesc	The text describing the affected area of the alert message	Mapps to WCUObject.Text
polygon	The paired values of points defining a polygon that delineates the affected area of the alert message	Mapps to WcuMapObject.Geometry Polygon property.
circle	The paired values of a point and radius delineating the affected area of the alert message	Mapps to WcuMapObject.Geometry Polygon property.
geocode	The geographic code delineating the affected area of the alert message	Mapps to a Location object (SwedishAddress or NamedArea)
altitude	The specific or minimum altitude of the affected area of the alert message	Not used by Wcu.
ceiling	The maximum altitude of the affected area of the alert message	Not used by Wcu.

Appendix F – Evaluation Results

Table F:1 – Processing time in seconds, based upon a number of test messages

Test Iteration	Number of sent messages							
	1000	600	400	200	100	50	20	1
Not used	2.721604	1.736195	1.532857	0.766429	0.531808	0.484883	0.344111	0.265904
2	2.721604	1.548499	1.282595	0.578732	0.312828	0.187697	0.125131	0.078207
3	2.705962	1.532857	1.110539	0.56309	0.312828	0.203338	0.10949	0.062566
4	2.67468	1.532857	1.063615	0.56309	0.312828	0.203338	0.10949	0.062566
5	2.690321	1.548499	1.047974	0.56309	0.328469	0.187697	0.10949	0.078207
6	2.721604	1.532857	1.063615	0.56309	0.312828	0.187697	0.125131	0.062566
7	2.690321	1.517216	1.079257	0.56309	0.312828	0.187697	0.10949	0.062566
8	2.737245	1.517216	1.063615	0.56309	0.312828	0.187697	0.125131	0.062566
9	2.67468	1.517216	1.079257	0.578732	0.312828	0.187697	0.125131	0.062566
10	2.737245	1.517216	1.079257	0.56309	0.312828	0.187697	0.125131	0.078207
Median	**2.705962**	**1.532857**	**1.079257**	**0.56309**	**0.312828**	**0.187697**	**0.125131**	**0.062566**
Variance	0.000612	0.00017	0.005165	0.000476	0.000272	0.000476	0.00068	0.000612

COS/CCS 2007-04

www.kth.se