



Dynamic Variability in complex, Adaptive systems

Deliverable reference: D6.1	Date: 16 March 2009	Responsible partner: Thales
Title: Case Study Specification and Requirements		
Editor(s): Dhouha Ayed Thomas Genßler		Approved by:
		Classification: public
Abstract / Executive summary: <p>The first purpose of this report is to provide individual definitions for the case studies of industrial partners CAS and Thales. Each definition describes the business and technological domains as well as the concrete materials that will be used to illustrate and assess DiVA technology.</p> <p>The second purpose is to describe an initial set of consolidated requirements for the technologies, tools and processes needed to support the management of complex, adaptive software systems.</p>		
<p>DiVA is a STREP FUNDED UNDER CONTRACT 215412 TO THE SEVENTH FRAMEWORK PROGRAMME, THEME 1.2: SERVICE AND SOFTWARE ARCHITECTURES, INFRASTRUCTURES AND ENGINEERING</p>		 <p>INTERNATIONAL DOCUMENT AVAILABLY: PARTNER + REPORT NUMBER ISBN</p>



Table of Contents

Case Study Specification and Requirements	1
Table of Contents	2
Version History	4
Contributing partners.....	5
1 Introduction	6
2 Thales industrial case: crisis management at an airport.....	7
2.1 MOTIVATION	7
2.2 CRISIS MANAGEMENT PHASES AND ORGANISATIONS.....	7
2.2.1 <i>The Observe phase</i>	7
2.2.2 <i>The Orient phase</i>	8
2.2.3 <i>The Decide phase</i>	8
2.2.4 <i>The Act phase</i>	8
2.2.5 <i>The Control room</i>	8
2.2.6 <i>Crisis management organisations</i>	9
2.3 SCENES.....	9
2.3.1 <i>Scene 1: crash of a plane on a runway</i>	10
2.3.2 <i>Scene 2: intrusion detection and car explosion</i>	12
2.3.3 <i>Context, variability and adaptations</i>	16
2.4 SYSTEM ARCHITECTURE	19
2.4.1 <i>Main use cases and system boundaries</i>	20
2.4.2 <i>Overall system architecture</i>	21
2.4.3 <i>Deployment structures</i>	23
2.5 CONCLUSION	25
3 CAS industrial case: service mash-ups in a hosted SaaS CRM application	27
3.1 CAS ACRONYMS AND TERMINOLOGY	27
3.2 MOTIVATION	27
3.3 SCENES.....	30
3.3.1 <i>Scene 1: office work versus mobile work</i>	30
3.3.2 <i>Scene 2: reconfiguration of the system due to heavy load</i>	31
3.3.3 <i>Context, variability and adaptations</i>	32
3.4 SYSTEM ARCHITECTURE	33



3.4.1	<i>System boundaries</i>	37
3.4.2	<i>Overall system architecture</i>	38
3.4.3	<i>QoS requirements and priorities</i>	40
3.4.4	<i>Deployment structures</i>	40
3.5	CONCLUSION	41
4	Requirements specification	43
4.1	REQUIREMENTS TEMPLATE	43
4.1.1	<i>Requirement status</i>	43
4.1.2	<i>Priority of requirements</i>	44
4.1.3	<i>Relationships between Requirements</i>	44
4.1.4	<i>Requirement category</i>	44
4.2	FROM REQUIREMENTS TO TECHNICAL SPECIFICATIONS	44
4.3	REQUIREMENTS	45
4.3.1	<i>Requirement analysis requirements</i>	45
4.3.2	<i>Modelling requirements</i>	46
4.3.3	<i>Runtime requirements</i>	48
4.3.4	<i>Verification and validation requirements</i>	54
4.3.5	<i>Non-functional / misc. requirements</i>	55
5	Conclusions	57
6	Appendix A: Thales crisis management use cases	58



Version History

Version	Description	Date	Who
0.1	Initial outline	11.4.2008	Dhouha Ayed
1.0	First case study versions	31.07.2008	Dhouha Ayed, Thomas Genssler
1.1	Updated outline	19.09.2008	Dhouha Ayed
1.2	Thales elaborated scenarios	28.10.2008	Anthony Assi, Dhouha Ayed
1.3	Thales QoS requirements and priorities	8.12.2008	Anthony Assi
1.4	Thales motivation and case study scenario	17.12.2008	Dhouha Ayed
1.5	Thales deployment structure and implementation	23.12.2008	Anthony Assi
1.6	Thales internally reviewed version	06.01.2008	Dhouha Ayed
1.7	CAS elaborated scenario	10.12.2008	Thomas Genssler
1.8	Update due to remarks from Dhouha	09.01.2009	Klaus Musch
1.9	Update due to remarks from the first internal review	19.01.2009	Dhouha Ayed
2.0	Update due to remarks from the 2nd internal review	11.02.2009	Dhouha Ayed
2.1	Update due to remarks from the technical coordination	06.03.2009	Dhouha Ayed
2.2	Internal review of requirements after previous modifications	09.03.2009	Anthony Assi, Dhouha Ayed



Contributing partners



Thales
Route départementale 128
91767 Palaiseau Cedex
France

Dhouha Ayed
Dhouha.Ayaed@thalesgroup.com

Anthony Assi
Anthony.Assi@thalesgroup.com



CAS Software AG
Wilhelm-Schickard-Straße 10-14
76131 Karlsruhe
Germany

Thomas Genßler
Thomas.genssler@cas.de



Lancaster University
Computing department,
InfoLab21, Lancaster University,
LA1 4WA,
United Kingdom

Reviewer: **Ruzanna Chitchyan**
rouza@comp.lancs.ac.uk



pure-systems GmbH
Agnetenstraße 14
39106 Magdeburg
Germany

Reviewer: **Michael K. W. Hentze**
michael.hentze@pure-systems.com

1 Introduction

The DiVA project aims at developing principles, methodologies and tools for the design and management of complex adaptive software systems. DiVA research will be validated based on real needs from the industry, i.e., the industrial partners in WP6 define case studies and the requirements, evaluate the technologies and tools in their contexts, and give feedback to technology providers to improve their products. This approach ensures that the research carried out is based on real-world evidence and is relevant to industry.

The first purpose of this document is to provide individual definitions for the case studies of the industrial partners CAS and Thales. Each definition will describe the business and technological domains as well as the relevant development processes. The second purpose is to describe an initial set of consolidated requirements for the technologies, tools and processes needed to support the management of complex, adaptive software systems.

The remainder of this document is organised as follows:

- Chapter 2 describes the Thales case study.
- Chapter 3 describes the CAS case study.
- Chapter 4 specifies an initial set of consolidated requirements.
- Chapter 5 concludes the document.



2 Thales industrial case: crisis management at an airport

2.1 Motivation

In crisis situations, many organisations need to come together to save lives and limit further losses. Many actors need to coordinate their actions, communicate their view of situation and jointly form themselves into an effective crisis management organisation.

Crises are highly unpredictable and dynamic situations. Proper crisis management persists in rapidly creating an understanding of the situation and its consequences and in quickly developing an organisation that can control the incident. In practice, this is a very complex process with many stakeholders, policies and procedures. The process relies heavily on experienced decision-makers and emergency responders getting good information and technological support. Not all types of crisis can easily be predicted, neither can the shape of the crisis management organisation. Supporting technology, therefore, used in crisis management decision support should be able to handle dynamic and novel situations.

This section describes the Thales case study about crisis management at an airport. Section 2.2 introduces the phases of a crisis management and the organisations that are involved in our scenario. Section 2.3 specifies two concrete crisis scenes we will use to illustrate the DiVA approach. Section 2.4 describes the architecture of the crisis management system that will be used to carry out the experimentations. Section 2.5 summarises the Thales case study.

2.2 Crisis Management Phases and Organisations

Several types of crises can occur at an airport such as a major fire, an explosion, or a chemical spill. The management of such crises requires a control room that centralises collaboration of the airport organisations with other external organisations such as hospitals and fire departments.

The common phases of the crisis management cycle are the **Observe-Orient-Decide-Act loop**. These phases take place differently according to the crisis type and its evolution. In this section, we introduce the crisis management case by detailing each of these phases, the role of the control room, and the crisis management organisations.

2.2.1 *The Observe phase*

The Observe phase gathers information about the environment, the status and the threat.

At an early stage, information about the accident reaches the control room through humans or sensors. Humans are generally airport employees or passengers who witness the beginning of the crisis. They activate an alarm or call the emergency call centre. Stationary sensors such as gas sensors, cameras, weather sensors or fire sensors send warnings to the control room.

However, since sensors can be unreliable and obtaining a clear idea about the crisis situation from a single person is difficult, additional information should be gathered and interpreted before conclusions can be made and warnings are issued. The control room assists with situation assessment using large amounts of relevant information that can be obtained from people and different types of sensor systems. Available information sources that are relevant to the assessment of the situation can be discovered dynamically. For example, a human alert of an explosion can result in a dynamic discovery and activation of all fire sensors of the airport in order to localise the explosion and monitor the fire propagation. An explosion alert can also require dynamic discovery and activation of gas sensors in order to detect gas emissions.



The information acquisition process and the interpretation of observations must be guided in an automatic and dynamic way such that reliable situation assessment is performed very quickly, thus improving efficiency and effectiveness of decision-making processes.

2.2.2 *The Orient phase*

The orient phase makes estimates, assumptions, analyses, and judgments about the situation to create cohesive situation awareness. This phase requires human skills, but some tools such as intelligent analysis tools and visual information sharing system can accelerate the process and help the crisis actors to create cohesive situation awareness.

2.2.3 *The Decide phase*

The Decide phase determines what needs to be achieved, whether it is an immediate reaction or a complex plan, for example the organisations and the actors that must be involved, the resources to deploy, etc. In this phase, human actors are assisted with decision making tools.

2.2.4 *The Act phase*

The Act phase executes the strategic decisions. While doing so, it continually adjusts operational plans to specific and changing circumstances that are encountered.

2.2.5 *The Control room*

The control room centralises all phases of the crises management (observe, orient, decide, and act). It is responsible for identifying the crisis, building the crisis management organisations according to the nature of the incident, collecting and providing crisis information to all the organisations dealing with crisis management. It receives real-time data from aerodrome and aircraft sensors and disseminates this data to internal or external organisations of the airport by subscription and on request.

In order to centralise the crisis management phases, the control room provides specific human actors and a crisis management system that provides the following systems:

- A visual information sharing system, called a “dashboard”: visualises the organisations, the deployed systems, and the crisis state.
- A communication interoperability system: stores and allows retrieval of static and dynamic information.
- A critical thinking tool: it is a decision making tool that helps to prevent tunnel vision (focus on one explanation to the exclusion of other possibilities) and information bias (focus only on supporting information) by proposing different possibilities of decisions.
- A dynamic adaptation system: adapts the crisis management system configuration according to the crisis type and evolution.
- A document sharing and a messaging system: allows crisis actors to write reports, to share them, and to exchange messages
-

The architecture of a crisis management system and its actors are detailed in section 2.4.1 and 2.4.2.



2.2.6 Crisis management organisations

After a major accident, it is crucial to build up an effective crisis management organisation as soon as possible. Crisis management organisations are ad hoc assemblies of multiple emergency organisations, such as police offices, fire departments, hospitals and the airport control room. These organisations need to work together to minimise the number of casualties and the damage level.

The process of scaling up should result in an organisation with the appropriate size and structure, relative to the nature, size, and complexity of the incident. The construction and scaling up of an organisation is a complex process: crisis management organisations are multi-disciplinary and have a complex command structure. When a crisis management organisation has been created, it is crucial to keep the organisation in line with what is happening at the scene, to prevent further escalation of the crisis.

According to the crisis type and its evolution, several types of organisations could be involved in a crisis. In the following, we give examples of such organisations:

- Fire departments: A fire department is involved as soon as the crisis may cause damages. The control room connects dynamically to the fire department in order to transfer information about the crisis, such as the location and the evolution of the crisis. The fire department actors decide on the resources (human, software and physical) to deploy and the actions to take. Rescuers that are deployed on the field have PDAs in order to collect information about victims and their evacuation and inform the control room about the crisis evolution. Their role is to rescue victims and act on the cause of the crisis in order to limit damages. They are also responsible for sorting the victims, and transferring them to hospitals.
- Hospitals: Hospitals are involved only if there are victims. The number of involved hospitals depends on the number of victims and the hospital type depends on the nature of the injuries. Information about victims that is collected from the rescuers (number, state, type of injury, emergency level, etc.) is sent to the hospitals in order to prepare necessary resources to receive the victims and take care of them.
- Police offices: police offices are involved when there are human losses, bomb attacks, suspicious behaviours, etc. The number of deployed police officers and their role depends on the crises nature and size.

Internal organisations of the airports are, for instance, the ATFCM (Air Traffic Flow and Capacity Management) organisation, The ATC (Air Traffic Control) organisation, or the Airport Operator organisation. The internal organisations are involved in order to determine the impact of the crisis on the airport capacity and to ensure that procedures are available to handle unplanned variations.

Other organisations could be involved in a crisis such as the ministry of defence, or a radio/television channel to inform people about the internal and external impact of the crisis (emissions of gas, traffic jams, etc.).

In summary, this section provided a general introduction to the Thales crisis management case in order to help understanding the concrete scenes presented in the next section. These concrete scenes will be used to illustrate the DiVA approach.

2.3 Scenes

The goal of DiVA is to provide a new tool-supported methodology with an integrated framework for managing dynamic variability in adaptive systems. In order to be able to

experiment and evaluate the DiVA technology, we specify, in this section, two concrete crisis management scenes that illustrate variability of context and required adaptations.

2.3.1 Scene 1: crash of a plane on a runway

Alex is a security manager who works in a control room at an airport. The authentication system he uses to get access to the crisis management system is a biometric one. Since Alex is connected locally to the system from the control room, encryption is not required.



Fig. 1 –Control room screens

The control room comprises a set of screens (see Figure 1) monitoring states of the airport sensors (gas sensors, cameras, weather sensors and fire sensors (see Figure 2)). A warning coming from a fire sensor on runway “A” appears on one of the screens. Alex identifies the alarm type and the system tries to localise it on the map. Since sensors can be unreliable, additional information should be gathered and interpreted before conclusions can be made and warnings issued. The available sensors that are on the runway (other fire sensors in order to monitor the fire propagation, gas sensors that detect gas emissions and camera videos that help to understand the situation) are determined and dynamically activated.



Fig. 2 – Airport Sensors



After analysing the captured videos of runway “A”, Alex understands that a plane has crashed there. He enters his conclusions about the crisis type and provides the first estimations of damages and the number of victims.

The crisis management system sends a message to the colleagues of Alex that are not in the control room to inform them about the crisis. A dynamic connection is automatically established with the fire department in order to transfer the current information about the crisis and first damages estimation.

John, the colleague of Alex, is at home. He can get access to the crisis management system from home via an internet connection. He requires authentication using a symmetric dynamic password and encryption should be performed on his connection with SSL-1024bit.

James is another security manager who is at the airport when the accident occurs. He uses a WIFI connection and authenticates with a symmetric dynamic password. His connection is encrypted with SSL-1024bit.

Since the situation is critical, John has to join the control room as soon as possible. He takes his car and uses his PDA to follow the crisis events. The display system adapts according to the PDA screen size.

The fire department decides to deploy the resources (human, software and physical resources) they have and to take their actions. Rescuers that are deployed on the field have two different profiles/ roles: “medical rescuer” and “physical rescuer”:

- Medical rescuers are responsible for rescuing the victims. They sort the victims (according to emergency and injury type) and transfer them to hospitals.
- Physical rescuers act on the causes of the crisis in order to limit damages.

The crisis management system dynamically deploys a lightweight application (client) on each of the rescuers PDAs. A deployed application adapts to the rescuers role, it:

- allows medical rescuers to send information about victims (injury type, required hospital, etc.), and
- allows physical rescuers to send information about the actions that are taken to reduce damages as well as information about the crisis evolution.

Several airport roles belonging to internal airport organisations such as the airport operator, the tower ground controller, and the flow manager must be informed about the evolution of the crisis. This information allows them to take decisions about the impact of this crash on the internal operational process of the airport (closure of runway A, update the airport capacity plan, update the departure sequences, adjust the slot times). Actors belonging to internal airport organisations use a non-encrypted Ethernet connection and static passwords to get access to the crisis management systems.

Each of the security managers of the control room and the internal airport actors can use an Ethernet or a WIFI connection.

Once the first estimations about the crisis reach the crisis management system, the system collects the hospital’s availability information and determines the number of hospitals that are necessary to take care of these victims. Then, the system establishes a dynamic connection with the nearest hospitals. As soon as first information about victims is sent by the rescuers, the system decides the hospitals to which it has to transfer this information according to the injury

types. The system is able to connect to new hospitals if none of the nearest existing hospitals is able to take care of victims.

After 30 minutes, the fire spreads over the airport and damages the air traffic control system. The crisis management system creates new instances of the services provided by this critical system, recovers the states they had just before their failure, deploy them on the replica of the air traffic control system and activates them. The air traffic control system is recovered in a few minutes.

2.3.2 Scene 2: intrusion detection and car explosion

This use case has been taken from a true incident that happened at Glasgow airport which handles around 9 million passengers a year¹. For special reasons, some of the scenario descriptions were altered in order to keep it as generic as possible.

On the 2nd of January, one of the busiest days of the year due to New Years Eve, celebrated the previous day, the international Airport of Glasgow became the target for a car bomb attack, which propelled the airport situation into the glare of CHAOS. We have divided this scene into multiple phases, in order to show how we want the system to dynamically adapt in each phase.

[Phase A-The gate] While the Security Administrator is on duty in the control room, staffs wish 'Happy New Year' to each other at the main hall. At 13:22 local time, a 4x4 vehicle approaches the gate that gives access to the inner forecourt of the airport. The inner forecourt of the airport is a restricted area because of its critical importance.

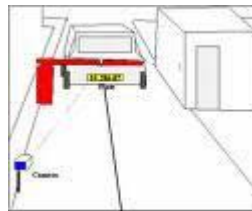


Fig. 3 – First entry point

For security reasons, an electronic gate handles gate manipulation, and is controlled with an access badge. There are three ways of opening this gate, either:

- activation from the control room, after verifying the identity of the entity requesting access to the site, using the CCTV cameras and the Video-phone, or
- by passing one's fingerprint on the finger scanner in front of the gate, or
- by scanning one's badge through the badge scanner, and entering the dynamic digital code.

¹[http://news.bbc.co.uk/2/hi/uk_news/scotland/6257194.stm]



Fig. 4 – Access control methods

The control room member is not yet aware of the presence of the 4x4 car in front of the gate, because the system has not detected any abnormal behaviour yet.

[Phase B- Wrong authentication] The driver tries to pass a badge to the scanner and enters a wrong digital code at two consecutive times. The System detects that a failed authorisation has been signalled at the main gate, it has activated the live feed camera broadcasting the scene, and projected the video stream on the main crisis management command screen.

[Phase C-Plate recognition] The driver enters again for the third time a wrong password, a vocal notification is then given, the system decides to deactivate his badge access to this gate, and it launches a warning in the control room. In the meantime the crisis management system has entered a prevention mode; it has decided to identify the vehicle using the “Automatic Number Plate Recognition” [ANPR] system in order to verify whether this vehicle belongs to the airport or not. If it does not, the system will check whether this car is on some federal blacklist (i.e.: police stolen cars black list, etc.).

In order to be able to load the ANPR system and to get access to the list of the local vehicles, the crisis management system must connect to the airport intranet using a multi-factor authentication based on asymmetric key and IP address. The system gets authenticated into the intranet portal, and gets the list of the airport vehicles plate numbers, but doesn't find any match for the plate number of the suspicious vehicle. Knowing that most of the attacks are made with stolen cars, the system decides to check if the plate number is in the polices' stolen cars list. The system loads the module in charge of that verification, which decides to connect to the police data base, using authentication based on client certificate. The system fetches the list of stolen cars plate numbers, and finds a match for the vehicle plate number!



Fig. 5 – The ANPR software

[Phase D-The retreat] The 4x4 vehicle gets away from the gate, as if it was retreating. It is now out of the camera view.

[Phase E-Intrusion detection] Suddenly, the cameras show the 4x4 vehicle heading towards the gate at high speed, going through the gate while breaking it into multiple pieces. At 13:24 we have an official breach of the security premises of the airport. The system has detected that its gate sensors is in an alarmed situation and has launched all necessary components to deal with this situation: it has established a connection dynamically with the Police Incident Room and has transferred a full description of the situation (type of the incident, alarm incident code, plate number of the car, etc...).

[Phase F-The collision] The vehicle manoeuvred to attempt a run directly at door two of the main terminal building and to gain entry into the main check-in area of the terminal. Flames issuing from the vehicle are seen onto the front of the building and the canopy above the doors. Once the system is in a critical situation, it has established a connection with all sensors in the breached area, and has started to log all events while pushing most relevant information to the control room's main control screen. All this logging will be used later on to reconstruct the incident and probably learn how to reinforce security.



Fig. 6 – Before the attack

[Phase G-Fire] At 13:27 the terminal fire alarm is activated and an evacuation is started, initially for the terminal in which the call was activated. As smoke permeates through the building, various smoke detectors are activated and additional areas of the building are evacuated. When smoke detectors are activated, the central system detects the location of the sensor, and tries to activate the required sensors in the same area.

The crisis management system, having gathered information from the different devices has decided to request some ambulances and reinforcements from the nearest emergency centre and police station.

[Phase H-The runaway] As the incident unfolds, two males exit from the car, with the driver on fire and heavily injured. The two individuals are wrestled to the ground and arrested by police officers, assisted by members of the public.

[Phase I-The explosion] Suddenly, at 13:38, a big explosion happens in the building, more fire alarms are initiated, additional smoke detectors are activated, etc.



Fig. 7 – After the blast



2.3.3 Context, variability and adaptations

The following table recapitulates the variability according to the context for both scenes outlined in the subsections above.

Relevant context and context changes	Adaptation	Comments
Location: in the control room or not, inside or outside the airport (internal or external network) Available hardware: fingerprint reader	Authentication: - static Password, - dynamic Password, - biometric authentication	Security, access control
Connection type: LAN Ethernet connection, WIFI connection, external WAN connection	Encryption: - no encryption, - encryption	Security
Roles: in-house personnel, visitors, on demand access, others	Gate control: - distance activation (CCTV cameras and video-phone) - fingerprint scanning - badge scanning plus dynamic digital code.	Security, gate access control
User location: inside/ outside the intranet	Multi-factor authentication (Airport systems and police systems) - Asymmetric keys + IP address - Client certificate	Security, access control
Location, required types of sensors	Dynamic activation of sensors: - Smoke sensors - camera - radiation detector - gas sensor	Search for surrounding/useful sensors
Normal situation, Wrong authentication, suspicious situation, intrusion detection, critical situation	Logging&Traceability: - normal - warning - push - critical - push broadcast	Log/ verbose level
Crisis type, crisis consequences: structural damages,	Dynamic connection to a system : - fire Department	Crisis Organisation building



human damages	- airport police - national police - hospitals - Ministry of Defence	
Users mobility, user preference, and network availability	Network connection: - LAN, - WIFI-WAN	Connectivity
Screen size	GUI: - big, - small	Usability
User role : medical rescuers physical rescuers	Support of several component versions	
Damaged system	System replacement	Availability

In this section we define examples of context types, the relevant contexts and the possible adaptations:

- User Location: Crisis Manager location is an important factor in the adaptation of the authentication mode which can have the following adaptation :
 - When the user is **in the control room**, authentication to the system is performed using the biometrics fingerprint reader.
 - When the user is **outside the control room** while being in the airport and such having a network connection to the airport intranet, he uses a static password authentication.
 - When the user is **outside the airport** such connecting through an external internet connection via VPN, he gets authenticated using a dynamic password.
- External Organisation Location: Depending on the external system location that the crisis system is willing to connect to, the authentication method is adapted as follows:
 - If the external organisation is located **inside** the airport (e.g. internal database system), the system gets authenticated using a combination of asymmetric key and IP address.
 - If the external organisation is located **outside** the airport (e.g., police database), a Client certificate authentication is used.

Note that, when a system server is moved from one location to another, the whole system should adapt to the new method of authentication without any manual reconfiguration.

- Sensors Location: Depending on the crisis or alert location, a group of interesting sensors is identified to lookup for and get its input as follow:
 - if for example a smoke alert is thrown in a specific building, all the building camera CCTV are being used to identify the crisis. As soon as



the floor where the crisis is happening is identified, all the cameras are disconnected except the ones on that specific floor.

- Connection type: The connection type that connects the user to the system affects the encryption adapted by the system as follows. In general we tend to have the highest encryption level while maintaining a good performance measure:
 - When the user is connected **through the internal crisis room LAN network**, basic encryption is applied, there is no need for a high level of encryption between the different crisis management systems located in the same room and on the same network.
 - When the user is connected **through any external network** other than the control room LAN, high encryption is applied.
 - When the user connects to the system using a **Wireless (WiFi) connection**, encryption is a high priority and the highest level of encryption is applied (ex: 1024 bits SSL encryption).
- Airport Roles: The profiles of the airport agents are grouped in multiple roles in order to adapt the gate control for each profile as follows:
 - **In-house personnel**, who are internal personnel of the airport, can open gates using their fingerprint intake.
 - **Visitors** have to use the Video-phone in order to be granted access to the inner forecourt of the airport. The in-house personnel who don't have a badge or whose badges have expired can also use this mean of authorisation.
- Rescue Roles: The rescuing profiles are grouped in two roles: medical rescuers and physical rescuers. The components deployed on the rescuers PDAs are adapted to their roles, they allow:
 - Medical rescuers to send **information about victims**
 - Physical rescuers to **send information about actions** that are taken to reduce damages as well as **information about the crisis evolution**.
- Screen size:
 - Depending on screen size, the number of the widgets placed on the user screen is adapted; most important widgets are put, and main information is visualised. The user can then click on each widget in order to highlight its contents. When the user is in front of the big screen, all the widgets (relevant to the current context) are placed on the screen, showing full information.
- Damaged system:
 - The system should be able to detect when a broker server is down, be able to look up a redundant one, and in case it fails, it should adapt to working without it.
- Crisis Evolution: The crisis status has a direct impact on the “Logging and Traceability” system. Logging is very useful in every crisis system because it

helps keeping trace of events, and can be studied later on for service improvement. The logging mode can be adapted to the context as follow:

- **Centralised Logging**, all system components store their logs in the same specific location. This type of logging is used when the airport is in normal situation (no crisis or alert in progress)
 - **Multicast logging** is used when an alert is triggered.
 - **Broadcast logging** is used when a crisis is in progress. This way all information is being distributed on the network.
- Crisis type: the crisis type has consequences on the external systems that must be connect to, for instance:
- An incident causing damage, for example a fire incident should launch the connection to the fire department.
 - An alert for human injuries should launch the connection to the health care department.

2.4 System architecture

This section describes the overall architecture related to the case study as well as deployment structures.

Figure 8 gives an overview of the Thales Airport Crisis Management System. It is divided into four parts:

1. The upper-left part of the image illustrates the crisis management system; it is located in the crisis control room which represents a room with restricted access. It has a lot of screens used for airport monitoring, gathering the input from the different sensors in the airport.
2. The upper-right part of the image illustrates the Server pool that hosts the crisis management system (and the DiVA Studio). It mainly consists of three dedicated servers. One is for the JBOSS-ESB application server, the second is for the DataBase and the third is for the messaging server that enables communications between the different actors.
3. The lower-left part of the image illustrates the sensor systems that monitor the airport situation.
4. The lower-right part of the image illustrates Actors and Organisations functionalities. Examples of actors can be fire fighters or medical rescuers while example of organisations can be Police or Fire Organisation. The actors, using the Laptop, PDA or Smartphone can interact with the system wirelessly. One of the adaptations that we can see is the adaptation of the system GUI depending on the actors' terminal, or even the encryption level depending on the actors' location.

In Section 2.4.1 we describe the actors and the functionalities of the crisis management system. In Section 2.4.2 we provide a high level composite architecture of the system. Section 2.4.3 details the deployment structures.

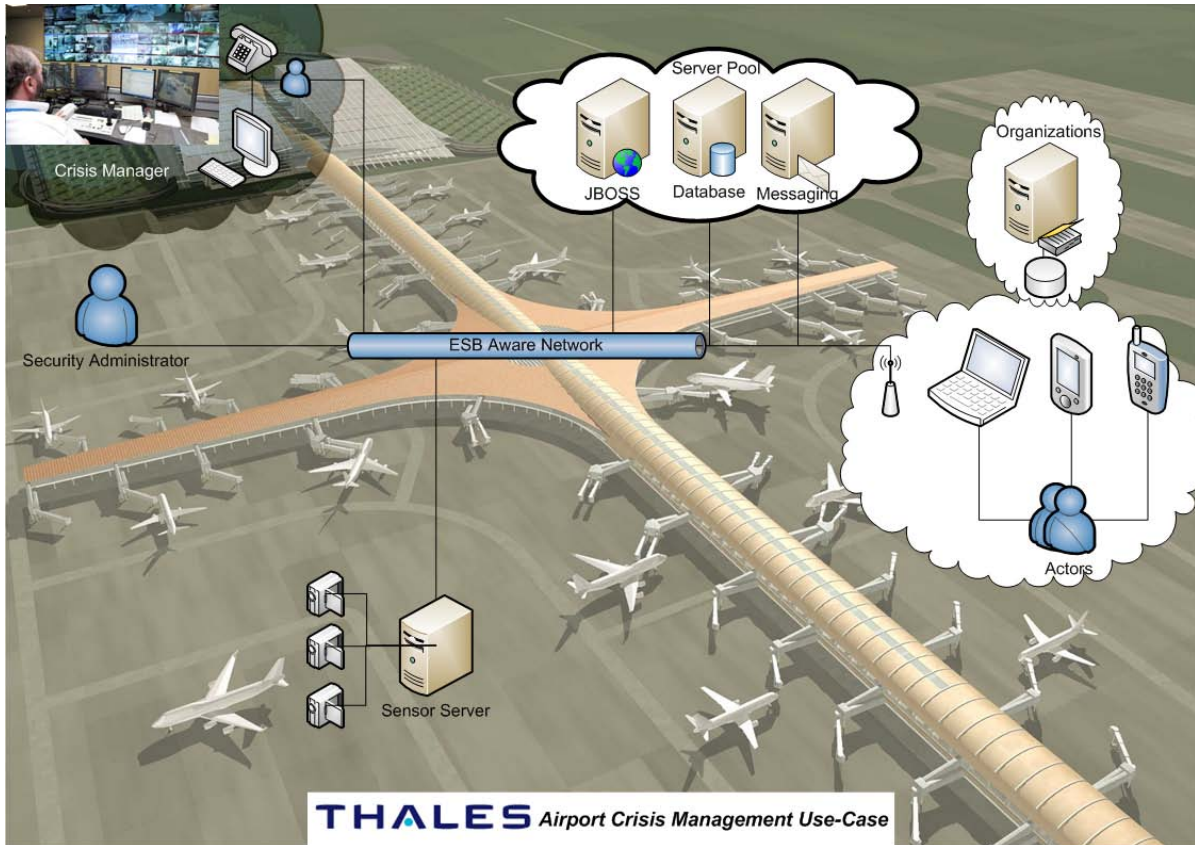


Fig.8 – Airport Crisis Management System Architecture

2.4.1 Main use cases and system boundaries

This section identifies archetypical actors, and the main functionalities offered by the crisis management system.

Three types of actors interact with the crisis management system:

- The Crisis Manager: the role of the crisis manager is to manage the phases of the crisis (observe, orient, decide and act phases). He is responsible for:
 - o Analysing Collected information (e.g., from the airport sensors).
 - o Estimating the crisis.
 - o Sending messages to organisations and their actors.
 - o Managing crisis management tasks.
 - o Managing interactions with organisations and information sharing.
- The Agent: An agent is a member of an organisation like, for instance, a hospital. He and his colleagues execute the tasks that are assigned to the organisation. In our scenario we consider the following types of Agents:
 - o Rescuer responsible for saving human-lives, member of a hospital or a fire department.
 - o Airport Officer: Airport employee responsible for the airport premises.



- Police officer: responsible for airport security.
- Fireman: responsible for fighting fire that hits buildings.
- The Security Administrator: the role of the security administrator is to configure the crisis management system, and make sure the system is functioning properly. Below we list some tasks he is responsible for:
 - Create the system users, and assign them profiles.
 - Configure the organisation contacts information.
 - Set-up authentication protocols and create the security keys used to authenticate the server.
 - Configure the sensors that are connected to the system.
 - Feed in the context rules.
 - Check the log files of the system.

In Appendix A, we give the use case diagrams related to each of the actors. These use cases represent the first step of analysis that allows us to identify the interfaces of the crisis management system (see Section 2.4.2).

2.4.2 Overall system architecture

This section provides the high level composite architecture of the crisis management system that specifies the partitioning of the system into components. It also identifies the interfaces and the relationships between the components and interfaces.

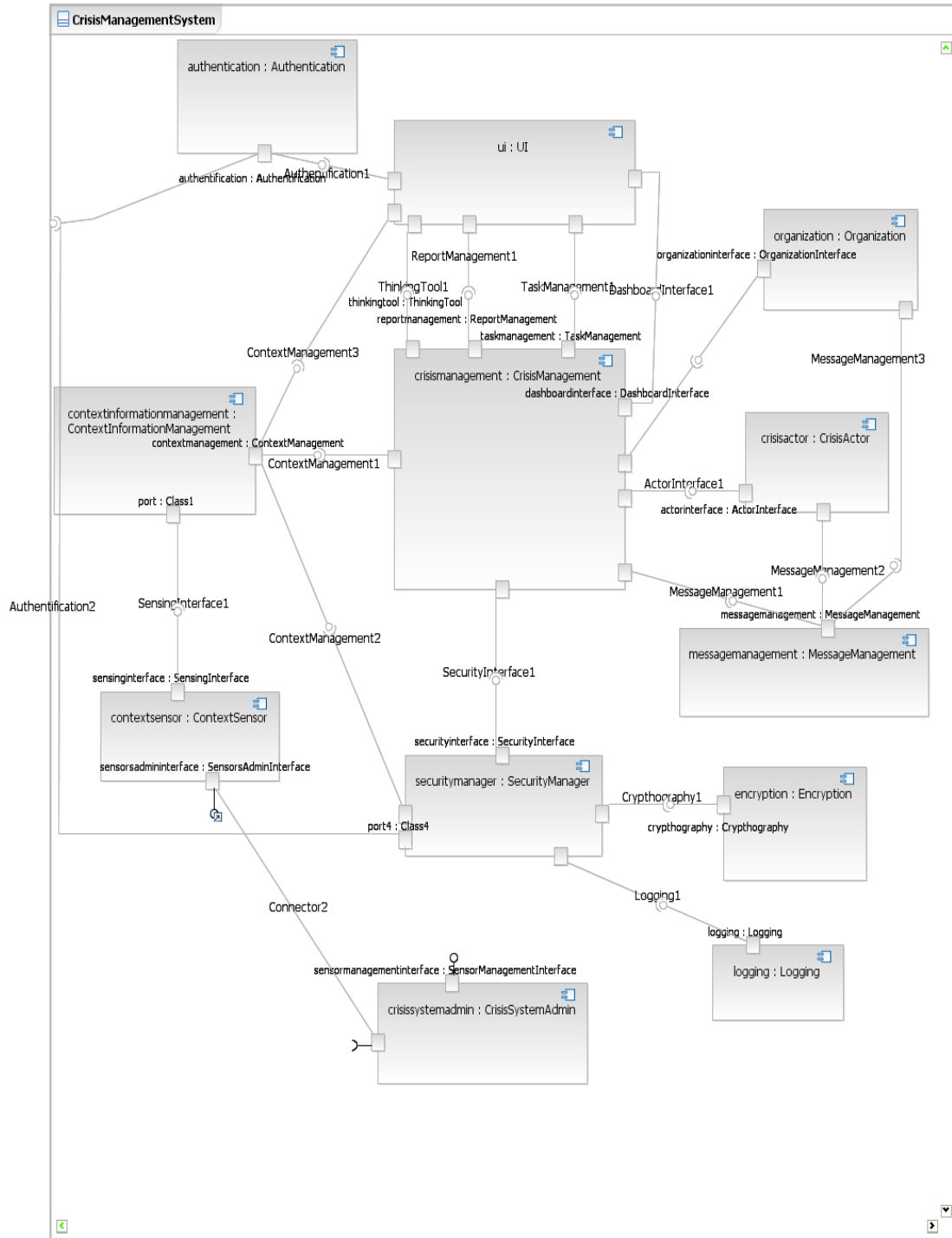


Fig. 9 – Crisis Management System Architecture

The crisis management system consists of twelve components (see Figure 9):

- The **CrisisManagement** component provides all the crisis management functionalities: a thinking tool, a task management facility, a reporting facility and a dashboard.
- **MessageManagement** component allows the crisis organisations and actors to exchange various types of messages such as emails, SMS and chat.



- The **SecurityManagement** component is responsible for processing security aspects such as encryption, logging and authentication. It uses the:
 - o The **Authentication component**.
 - o The **Logging component**.
 - o The **Encryption component**.
- The **Organisation** component allows an external organisation to interact with the crisis management system.
- The **UI component** represents the user interface that allows crisis actors to use the crisis management facilities.
- The **CrisisActor** component allows an actor to interact with the crisis management system.
- The **ContextInformationManagement** component is responsible for processing high level context information.
- The **ContextSensor** Component collects low level context information.

2.4.3 Deployment structures

Figure 10 presents the Deployment Diagram that shows the following nodes:

1. The Client Node hosts the components belonging to the Agents, such as Fire-Fighters, Medical rescuers, Policemen, etc. These components are the following:
 - **UI:** agent's user interface.
 - **Encryption:** the component responsible for encrypting user communication between his terminal and the crisis management system.
 - **Message Management:** the component that enables the messaging functionalities for agents between their colleagues and the system.
 - **Authentication:** the component used to authenticate the agents on the system.
2. The Application Server node hosts all the IT core servers (including DiVA studio) and the adequate components used to form the system:
 - **JBOSS-ESB Server** is the core IT application at the heart of the system. All the communications passes through the ESB.
 - **Database Server:** this is where the system will store all information about the contexts, the organisation's contacts information, all static information about the sensors descriptions and access codes, users information, etc.
 - **Logging:** A Logging component will be hooked-up on the server in order to log all the system activities. It provides traceability.
 - **Encryption:** the component responsible for encrypting the communication channels between the server and all the clients.
3. The Sensor Server is the server to which all the sensors report their state. It hosts the following components:



- **ContextSensor:** the component responsible of processing the sensors input and sending them to the central server for response actions.
4. An external Organisation is any organisation like hospital, police, etc. The following components are deployed on these types of nodes:
 - **Encryption:** the component responsible for encrypting the organisation communication channels.
 - **UI:** User Interface Component used as the organisation input interface.
 5. The Security Administrator node contains all the functionalities required by the Security administrator to configure the system for the first time, and assure its support:
 - **Crisis System Admin:** the component that offers all system functionalities for administrating the crisis system.
 - **UI:** Administrator's user interface component.
 - **Encryption:** the component responsible for encrypting the communication channels.
 - **Authentication:** the component used to authenticate the Security Administrator.
 6. The Crisis Manager System hosts all the necessary components to deal with a crisis:
 - **Crisis Management:** the component offering all the crisis management functionalities that the crisis manager can use while in the control room.
 - **Crisis Actor:** the component offering all functionalities for the crisis actor.
 - **Context Information Management:** the component used by the crisis manager to monitor the different contexts of the system (i.e.: sensor states).
 - **Encryption:** the component responsible for encrypting communication channels.
 - **Authentication:** the component used to authenticate the Security Administrator.
 - **UI:** The Crisis Manager's User Interface Component.
 7. The Messaging Server is the server responsible for handling the different messaging systems:
 - **Message Management:** the component that enables the messaging functionalities.

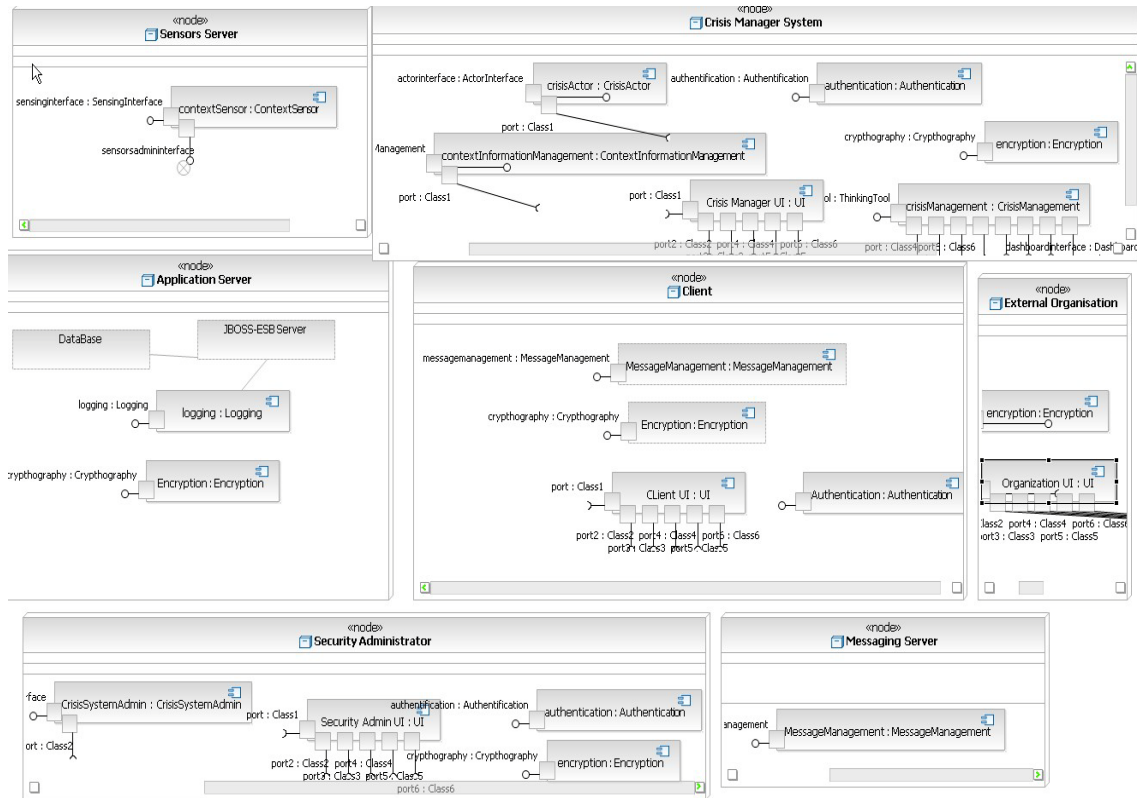


Fig. 10 – Deployment Diagram

2.5 Conclusion

The Crisis Management case study illustrates a complex adaptive system that covers various types of adaptations at several levels. It covers adaptation of functional and non-functional services by illustrating each of the architectural², behavioural adaptation (interaction of systems) and the reconfiguration of functional and non-functional properties. The variability dimensions³ and their variants according to the context are identified.

Engineering of complex adaptive systems is a relatively new discipline which largely remains a research topic. At present, tools and design methods are incomplete or missing for this new aspect of systems engineering. Based on these observations, Thales has highlighted a set of empirical research objectives related to engineering of complex adaptive systems to answer the following questions:

- How to design complex adaptive systems?
- How to manage complexity of runtime adaptation using for instance MDE based abstractions?
- How to verify and validate adaptive systems?

The **experimentation** will be guided by these empirical objectives and will follow these steps:

² dynamic addition, deletion and replacement of systems

³ example : authentication, encryption, logging, etc.



- Experimentation with the DiVA requirements engineering (**RE**) approach to evaluate its capacity to elicit and clarify requirements from the text provided in section 2.1-2.3. Alternative architectures based on the DiVA RE approach will be derived and compared to the architecture provided in section 2.4.
- Experimentation with the DiVA modelling approach for modelling the adaptation logic. To do that, we will use scenes of Section 2.3 and the architectural models of Section 2.4 as input.
- Runtime adaptation of complex systems is a challenging area. Concerns such as size, heterogeneity and dynamism all complicate the process. We will experiment the DiVA runtime adaptation by using the crisis management systems mock-ups and the architectural models. The performance of the runtime framework will be assessed and the usage of models and aspect-oriented techniques at the runtime level will be evaluated.
- Experimentation with the DiVA dynamic reasoning by assessing its performance and the consistency of the adaptation solutions.
- The use of models in the Verification and Validation (V&V) phase promises a reduction in the cost of V&V and an upward scaling of the V&V process to large-scale systems. Our goal is to study the feasibility and the added value of model-based V&V for adaptive systems and to test whether or not it improves the quality of complex systems. We also want to test whether or not it decreases the risk of failure.

A detailed description of the evaluation plan of DiVA results in the context of the Thales case study will be given in D.6.2.



3 CAS industrial case: service mash-ups in a hosted SaaS CRM application

This chapter describes the features, use cases and requirements of the CAS case study "Service Mash-ups and Mobility Support in a Hosted SaaS CRM Application". The chapter is structured as follows. The next section 3.1 defines the terminology for the case study description. Section 3.2 then gives a brief introduction into the business background and Customer Relationship Management. Section 3.3 introduces the CAS case study in more detail and describes the scenes. Section 3.4 gives an overview of the tools and the functional and technical architecture / infrastructure of the case study. Section 3.5 summarises the CAS case study.

3.1 CAS acronyms and terminology

This section gives an overview of terms used in the CAS case study.

- **SaaS:** Software-as-a-Service, a software delivery model, where clients use hosted software (usually over the Web) and pay a regular fee for the actual use of the software.
- **Client:** A commercial party paying a regular fee to use a SaaS system
- **User:** Representative of the client who is entitled to use the SaaS system. She has an account and password.
- **CRM:** (customer relationship management) is a type of business software that standardises sales processes and customer services.
- **Hosting company:** Company offering hosting infrastructure like server machines, data centre facilities etc. as well as hosting services as bandwidth, operation support and surveillance, software update and maintenance etc.

3.2 Motivation

The main purpose of CRM is to make customer information transparently available – always and anywhere throughout the entire company. Typical CRM goals include:

- Winning new Customers: Know and address your market segments and target groups.
- Knowing your Customers
 - Contact information, current needs and demands.
 - Contact history.
- Keeping your customers
 - Provide high-quality multi-channel services to the Customers.
 - Recognise/ address risks for customer defection.
- Optimising sales processes and customer care effectiveness
 - Buying probability/ sales prospect.
 - Tailor level of customer care to customer value/ escalation situation.

In order to achieve this, a CRM application typically covers the following functional areas:



- Address Management (address database)
- Address Qualification/ Profiling
- Contact Management/ Follow-up Management/ Customer care: Keeping track of each contact between the enterprise and its customers.
- Marketing Campaigns.
- Lead management.
- Sales Funnel/ Sales prediction.
- Escalation & Complaint Management.
- Mobile Salesforce Automation.
- Analytical CRM.
- Internal collaboration.

A CRM application can be seen as a central system offering a set of services to all connected users. Users access the system in various ways, e.g. with a client application, a standard web browser running on a desktop PC or a laptop, via mobile access from a PDA or a mobile phone or directly from other applications using its web service interface. The system itself provides its functionality either by accessing internal services and data sources or by accessing services from external partners (such as address profiling agencies).

CRM systems have to support a broad range of working patterns, ranging from back-office support at company premises to mobile or on site work at the customer. The success of CRM is strongly dependent on the availability of accurate and up-to-date information about customers and easy access to helpful services. It is very important to the user to have the same information available anytime, at any place, even if he is not at his personal computer in his office, for example because he is on a business trip.

CAS Software AG is the leading German specialist in the area of customer relationship management (CRM) for small and medium enterprises. State-of-the-art technology and very high adaptability has led our standard products and platforms to a very competitive level. More than 150.000 satisfied users are working on a daily basis with our powerful CRM solutions CAS genesisWorld and CAS teamWorks.

CAS genesisWorld is designed as a customisable platform solution. Other applications such as Office and ERP and also project management, archiving, document management and communication solutions, can be integrated via smart interfaces. Throughout our extensive partner network (more than 150 system and support partners) we offer more than 100 system extensions ranging from system connectors (e.g., for ERP systems) to workflow functionality.

A new CRM service platform EIM (Enterprise Information Manager) is currently under development. The first product based on this platform (CAS **PIA**) has been released in September 2008. This product will be used as case study system in the context of the DiVA project. It is explained in section 3.4.



We consider the Software-as-a-Service delivery model a crucial one. This consideration is backed with a number of recent studies⁴, indicating that the SaaS model is becoming a major trend for the cost-effective delivery of business applications, especially for small companies. In the SaaS model an application is centrally hosted and offered to clients on an “on-demand” basis. From a client’s point of view the SaaS application delivery model is typically characterised by very low initial costs and low operation costs, as no investment in infrastructure or additional software such as database systems or applications servers is necessary. During operation only the actual use of the system is charged.

From a technical point of view, our case study “Service Mash-ups and Mobility Support in a Hosted SaaS CRM Application” has the following properties:

No software distribution, no software installation: The entire application functionality (or at least the major share of it) is accessed through a standard web browser. The actual application together with necessary infrastructure (such as application servers and databases) is hosted in a data centre.

Multi-tenant capability: In order to keep operation costs low and to fully use the computing capacity of the machines, users of several clients (e.g., companies) typically share one set of server machines and one database / application server in the data centre.

Multi-channel access: SaaS applications offered through the Internet are usually supporting different interaction modes including classic page-oriented HTML GUIs, rich internet GUIs (e.g., AJAX) as well as access channels for mobile users such as WAP, data replication for offline use or speech control (e.g., to operate applications through a normal telephone).

Extendable service platforms: Modern SaaS applications are more and more evolving into open service platforms where third parties may deploy and offer additional services which can be integrated into user’s applications (so-called service mash-ups). The service platform has to ensure that services do not interfere with, or damage, other services in a way preventing these other services from fulfilment, especially in the hosted scenario where different users with different sets of services may share one application instance.

High scalability, availability and reliability: SaaS applications are typically operated in a data centre. In order to deal with variable number of users, SaaS applications have to be scalable that is, serving a bigger number of users can be done by adding another server computer, database and application server and without changing the application itself. As a consequence there is no 1:1 assignment of servers to users of a client. Based on load-balancing policies, users are assigned to any available server upon login.

SaaS applications promise a 24/7 availability from anywhere at any time. Thus, some sort of fallback mechanism has to be incorporated in the system, which transparently exchanges services that are no longer reachable or useable. Furthermore appropriate data backup and restore facilities have to be provided.

In order to provide high reliability, fall-back services have to be provided in case of service execution failure. This challenge is extremely important in case of a business service platform open to partners which may integrate their own services. Furthermore, dynamic update has to be supported, that is existing services may be replaced with new incarnations, without stopping the system and while obeying system and user context and process requirements.

⁴ E.g., “Enterprise Software Customer Survey 2008” (McKinsey & Sand Hill Group, 2008) or “Worldwide Applications 2008 Top 10 Predictions” (IDC, 2008)



High data security: Data in business applications such as CRM are especially sensitive, so appropriate security, data protection and authentication mechanisms have to be put in place.

In order to achieve the abovementioned properties (especially multi channel access and extendable service platform) under the constraints of high scalability and high availability / reliability there is a strong need for dynamic variability. SaaS applications cannot simply be stopped in order to be extended with new services or to change QoS properties. This adaptation has to be done dynamically at runtime.

Dynamically reconfigurable SaaS applications with service-mash-ups and ad-hoc services integration have put forth a whole new set of technical challenges such as

Ad-hoc-service integration and usage: Service mash-ups integrate additional distributed services, such as location-based services, pay-per-use-services or collaboration services on an ad-hoc basis with a set of core services. The functionality of core services may be temporarily extended with, or even replaced by ad-hoc services. The current system context has to be obeyed⁵. Furthermore, services in the system must not interfere with other services in a way preventing them from service fulfilment. This is a special challenge in a multi-tenant scenario where users of different clients may share the same application instance.

Context Awareness: The SaaS application needs to be aware of different contexts and situations, such as user or usage context (mobile, Web, LAN), user authorization for service usage⁶ in order to behave accordingly. This is especially important as there is no fix assignment between users of a particular client to a certain application instance due to load-balancing. While one client's users may be entitled to use a certain service (because the client pays for it), other client's users may not be entitled to use this service. As a consequence, one application instance may be used by users expecting partly different applications.

Security and Privacy: This challenge is twofold: First, in a hosted SaaS business application, critical business data is stored in a hosted database outside the company. Measures have to be taken to ensure data security and to prevent theft or misuse of private data (e.g., for industry espionage). This is already a challenge with today's applications. Secondly, and more important in the context of DiVA, when integrating services from other vendors, even more attention has to be paid to security and privacy. Access of services to private data (e.g., customer addresses) has to be strictly limited. Sandbox concepts have to ensure that a service gets only access to business data and user information which are minimally necessary to fulfil its task.

3.3 Scenes

3.3.1 Scene 1: office work versus mobile work

CRM user is connected to application from within the company network from a trusted workstation. The application proactively informs the user about incoming telephone calls or emails, about contacts of other employees of the organisation to his relevant customers (e.g., calls from technical support) and any escalation state of a customer (e.g., after a complaint call of customer has been recorded or a trouble ticket for a customer has been filed). User checks out files relevant for the customer visit for offline use by loading them from applications file storage into the application's local offline file cache. This happens transparently for the user as offline

⁵e.g., what other services are configured in the system? What are process requirements? What dependencies have to be fulfilled?

⁶ i.e., is the user authorised to use a particular service, which specific QoS characteristics have to be fulfilled



file cache is an integrated service on trusted workstations (e.g., company laptops). After a while user logs off to prepare for visit at a customer's site.

Once the customer is mobile (e.g. in his car), he can access application through his mobile phone (voice control) or smartphone (mobile access). That is, application **dynamically changes** UI to either voice control (if user only has a normal mobile phone available or is unable to use a graphical UI, e.g. while driving) or to smartphone access (reduced GUI, when user has smartphone or iPhone available).

The application knows from user's calendar that user is currently mobile and needs to be notified about incoming priority calls/ mails/ escalations on his mobile phone. That is, calls to user's office phone number **are automatically forwarded** to user's mobile phone, mails are pushed onto user's phone or read via phone call⁷ etc. In order to cope with bandwidth limitations, the escalation priority (i.e. the strategy that ranking server uses to rank information) **is adapted**, e.g., only information concerning those customers which are currently being visited by user or customers with high buying potential and open buying opportunities are being made available when mobile. For this purpose the **ranking server adapts** to the context change "user is mobile".

Another user cancels user's appointment with customer for the next day. Application sends notification to user about the cancellation using available communication channels (phone, text message, email). The communication channel is **dynamically chosen depending on user's current context** (use phone call, when user is in car; use text message, when user is in meeting, etc.)

After customer visit, user checks into a hotel and accesses application through a public terminal from within the hotel lobby. Application tries to **discover locally available services** (such as telephony via Skype, JaJah) and **incorporates them** if possible. User tries to checkout a file. Application **recognises** that user is connected from public terminal and there is no local offline file cache available in this situation. Application notifies user about the situation.

Application knows from user's calendar or other information (such as location of network access point) about user's physical location. Application also knows that there is no following appointment on this day. As it is early afternoon, application **looks up** location-dependent information services, adapts user's dashboard view and incorporates information about local places of interest (sights, historical sites etc.), restaurants close to the hotel and cultural event on that evening.

Once the user returns into his office, application **recognises** that user is connected to trusted network and synchronises modified files from offline file cache back into document store. Additionally, the normal working environment (telephony, email server, etc.) is **restored**.

3.3.2 Scene 2: reconfiguration of the system due to heavy load

The CAS PIA system consists of GUI server, application server and database. These three parts are put together via a connector-like mechanism. Calls between the subsystems can be made via different communication channels (Java method, RMI, web-service). The subsystem can be all on the same node or spread over several nodes. Each subsystem can be present only once or multiple times.

A system can be started with only one node, which holds all three subsystems (GUI server, application server, and database).

⁷ e.g., if user is driving and mail priority is high



As more and more users log in, average time consumed in the system for a request exceeds a certain threshold, let's say one second. The system identifies the GUI server as the bottleneck. So it adds an additional GUI server on a different node. New users will automatically be connected to the second GUI server.

Users who connect to the new GUI server do a lot of data mining, which leads to heavy load on the database server. System detects the new bottleneck and adds a new database server instance to the system. This means that the part of the system which monitors overall availability triggers the adding of the new database server instance, which is then used by the system. Only one of the database servers can do write operations, so maintaining contacts, documents, etc. is done by means of the first database server, while long running data mining operations are performed by means of the second database server.

Some users use long running calculations on the application server. If the application server load exceeds a certain threshold, another application server is automatically added to the system.

Currently, the re-configuration of the subsystems of CAS PIA is a manual process and cannot be done dynamically.

3.3.3 Context, variability and adaptations

Overall priorities:

Security and privacy are of high importance. The highest available security is always used. If a service requires a certain level of security (e.g. encrypting), which is not available, then the whole service is not available.

- Offline file caching
 - When user connects to system from within the company network or any other trusted computer, user can use automatic offline file caching facility. If user connects from a public terminal, file handling is done by the user (simple file download). However, automatic offline caching is not supported.
- UI mode
 - RAI⁸ GUI versus smartphone GUI versus voice control when mobile.
- Notification
 - Notification via pop-up window is used when RAI GUI is available.
 - Notification via phone call is only available when user is not busy.
 - When user is busy, notification via email (RAI GUI or smartphone) or SMS (mobile) is used.
- Encryption
 - Encryption (SSL) is used whenever possible. Communication between RAI GUI and GUI server is always encrypted (including authentication).
- Address broker service

⁸RAI is explained in section 3.4



- Address broker service with highest address quality and no costs is used first. If not available or address is not found, address broker with lower address quality but still without costs is used. If not available, address broker with costs is used.
- Weather service
 - Weather service with detailed data based on a regional base is used. If not available, weather service with data from a country wide base is used.
- Video on demand
 - Only Video-on-demand services which are reachable with the required bandwidth are used. Video-on-demand service reachable with the smallest net latency is used.
- Communication between subsystems
 - Java method vs. RMI vs. Web-service.
- Number of each subsystem
 - One or more instances of GUI server, application server and database server.
- Number of nodes and subnets
 - Subsystems are spread over one or more nodes and subnets.

3.4 System architecture

Figure 1 gives an overview of the case study. The core of the case study is the centrally hosted SaaS CRM service platform. This platform provides a number of CRM-related services to its users. These services can be internal services built into the application or external services integrated from third-parties. Users may access these services through different channels ranging from desktop computers with AJAX capable web browsers to mobile phones.

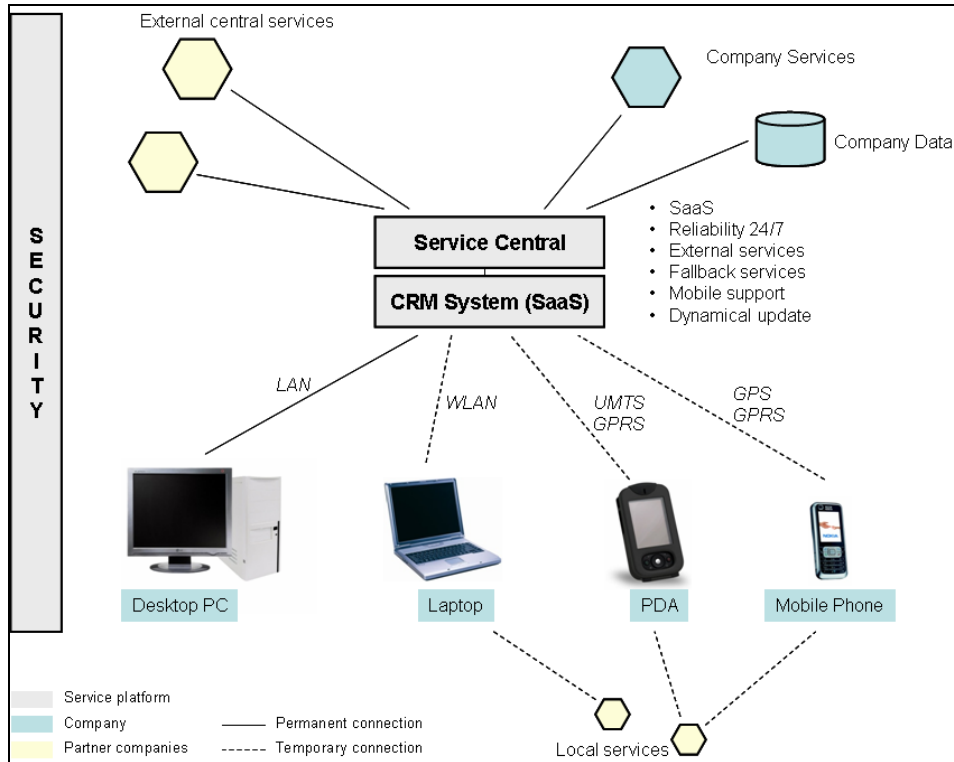


Figure 1 CAS case study

While interacting with the central SaaS applications, users may also want to integrate, and use, locally available services such as infrastructure on the client machine (mail clients, office applications) or services, or provide value, at a certain location or in certain contexts (printing service, booking services etc.).

The case study will be based on CAS PIA. CAS PIA is a hosted SaaS CRM application.

- CAS PIA features a multi-tenant⁹ architecture, i.e. multiple tenants share one installation of PIA which is operated by a hosting company. For each client, multiple users can make use of the CRM application.
- A client's user utilises the CAS PIA application with a web browser like Firefox or Internet Explorer. CAS PIA is written as a server based AJAX-style web application.
- CAS PIA is designed as a service integration platform. Third-party services can be integrated to tailor the functionality of the application to client's needs.

⁹ mult-client

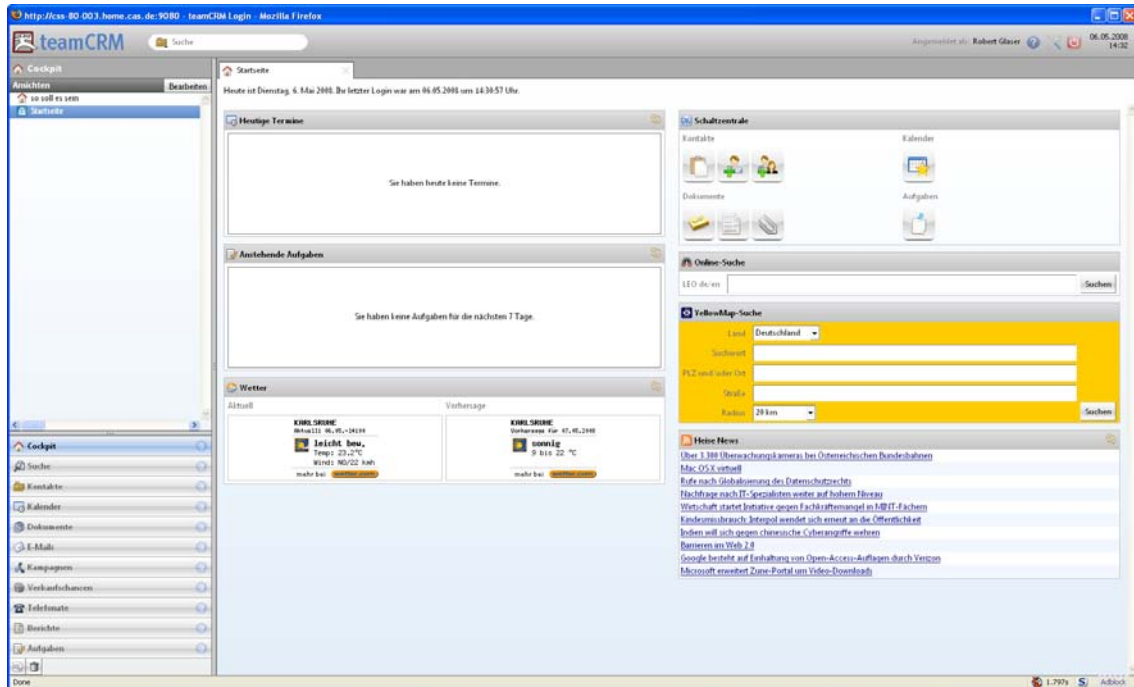


Figure 2 Cockpit view of CAS PIA

CAS PIA is a CRM system and includes the following typical CRM functionalities:

- Contact Management, contact history,
- Journaling (traceability of data changes),
- Groupware, E-Mail, telephony, Office integration,
- Document management,
- Campaigns, Reporting,
- Tagging, information linking,
- Service mash-ups,
- Information ranking (ranking server) for mobility support.

The different data types of CAS PIA are:

- Contacts (single contacts, organisations, representatives of organisations),
- Sales opportunities,
- Documents (notes, arbitrary documents such as pdf, images, office documents), serial documents.

CAS PIA has been built with cutting-edge technology. Figure 3 gives an overview of CAS PIA's architecture. CAS PIA has a layered architecture typical to most web application consisting of the following conceptual layers. Each layer communicates with the lower layer through a service-oriented interface. Both business server and presentation server are developed

in Java 5 Enterprise Edition and make use of the OSGi dynamic plug-in model known from Eclipse¹⁰ (the Equinox framework). This plug-in model already provides a powerful dynamic extension and re-configuration technique. Generative programming has been employed to generate database bindings and service interfaces for the different layers. The presentation server has been developed with RAP¹¹ (Rich Ajax Platform) and features GUI programming patterns known from Eclipse SWT and JFace¹². At client side (i.e. in the browser), RAP uses a state-of-the-art AJAX framework called Qooodoo¹³. This framework features Desktop-like user experience like keyboard navigation, focus and tab handling, Drag and Drop, Windowing, Theming etc.

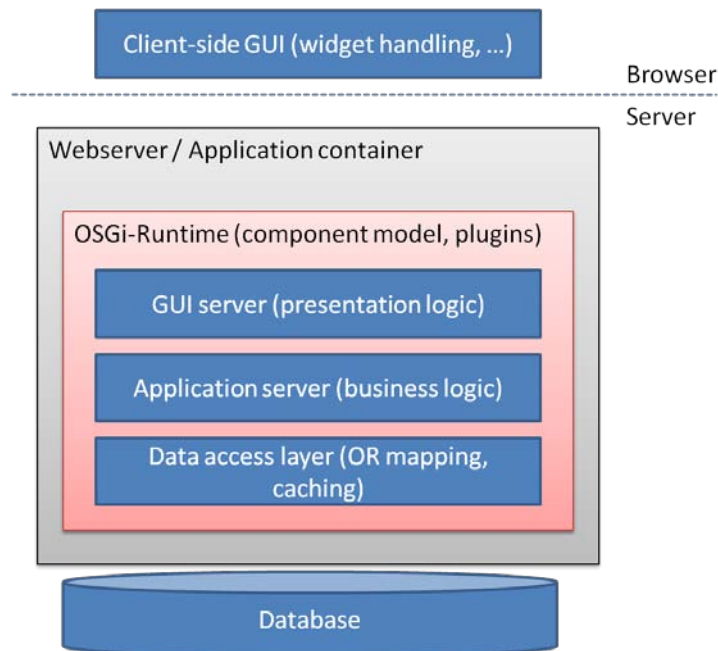


Figure 3 CAS PIA architectural layers

CAS PIA consists of the following conceptual layers:

- Client-side GUI: provides AJAX GUI including window management, widgets etc. The client-side GUI is implemented with Qooodoo.
- GUI Server/ Presentation Logic: provides server-side widget stubs, message handling facilities, error handling, widget lifecycle management (creation/ disposal of client-side widgets), input helpers, input checkers etc. The server side GUI has been implemented with RAP. For load-balancing reasons, the Presentation Server may be operated on a separate machine. In that case, remote-procedure calls are used to communicate between Presentation Server and Business Server. The GUI server is a state-full component. The GUI server keeps track of any GUI state of any user connected to it.

¹⁰ <http://www.eclipse.org>

¹¹ <http://www.eclipse.org/rap/>

¹² SWT and JFace are standard Eclipse widget libraries. RAP is API-compatible to SWT and JFace.

¹³ pronounced ['kɔksdu] like the German "Guckst Du!" which means "Look at this!", <http://qooodoo.org/>



- Application Server/ Business Logic: provides business functions including CRUD¹⁴ operations for business objects and complex business processes (e.g., execute marketing campaign). The application server is a stateless component.
- Data Access Layer: provides database encapsulation, object-relational mapping, access control, and distributed data caching.
- Database: provides persistency of user data (e.g., customer addresses, documents, etc.).

All server-side components are OSGi15-compliant bundles and run in an OSGi container. This container in turn is integrated into a Java web application container such as Tomcat.

For advanced functionality, a user might install a number of PIA Windows utilities offering local services such as local search functionality, Office integration, offline appointment notification, offline storage and desktop mail integration. These utilities come as plugins for Mozilla Firefox, Microsoft Internet Explorer, Microsoft Outlook and Mozilla Thunderbird or as a small supplementary Windows application (CAS Info@Click).

Additionally, external services can be integrated into the application. However, at this point it is not possible to integrate services dynamically. In order to integrate additional services the application has to be configured during application deployment.

3.4.1 System boundaries

Client (mandator)

CAS PIA distinguishes between clients and users. A client is e.g. a company. For each client there exist one or many users. A user is e.g., an employee of a company. When a user logs in, he has to provide both the client name (e.g. the name of the company) and the user name (the name of the employee).

This section describes the client's possible interactions with the system.

- Subscribe to use the system.
- Create users, define roles, and define access rights.
- Subscribe to third party services.

User (representative of the client)

This section describes the actions a user (representative of the client) can do in order to interact with the system.

- Login to the system.
- Read and write any kind of information that the user has access to (contacts, documents, emails, campaigns, etc.).
- Use third party services (when subscribed).
- Use different kinds of communication channels (web browser, PDA, mobile phone).

¹⁴ Create/ Read/ Update/ Delete

¹⁵ Open Services Gateway initiative



- Use offline data (documents, calendar, alerting, etc.).

System Administrator

This section describes a system administrator's possible interactions with the system.

- Check availability of system.
- Reconfigure the system (add additional GUI server, application server, database).
- Deploy new versions of (sub-)system.
- Restart system in case of breakdown.
- Do data recovery in case of data loss.

Third party service provider

This section describes a third party service provider's possible interactions with the system.

- Deploy third party services which automatically integrate into the system.

3.4.2 Overall system architecture

- Functional components
 - **Dashboard:** configurable information centre, integrates internal and external information sources, such as upcoming appointments, upcoming *Todos*, upcoming anniversaries, weather/ traffic information, stock information, system alerts, RSS feeds, easy access to most common tasks like new addresses, new appointment, new document, etc.
 - **Contact database** (including an intelligent service for parsing an address from e.g., an email footer, no need to enter the address field by field, link single contacts to companies, trip planner).
 - **Calendar:** manage appointments, find free slots, check whether user is available at a certain date/ time, book resources, move appointments via drag&drop.
 - **Task/ Todo database:** stores task such as todos, trouble tickets, follow-ups etc.; manages escalation procedures for over-due tasks etc.
 - **Document store:** store any kind of document, locking mechanism, automatic and manual versioning.
 - **Email:** send and receive Emails, archive Emails from external Mail-Client like Outlook and Thunderbird, serial Email.
 - **Campaigns:** planning and execution of campaigns via email, phone calls, appointments, letters etc., collecting reactions of customers, analysis of campaign costs and achieved turnover.
 - **Sales opportunities:** products, probability, alarm, next activity, contact people, competitors.
 - **Telephone calls:** plan and execute telephone calls, status.



- **Reports:** detailed reports of contacts, campaigns, sales opportunities, including different types of charts.
- General components
 - **Info ranking service:** ranks information stored in the CRM database such as contacts, appointments, documents, sales opportunities etc. according to their value for a certain user in a specific situation. Example: if CRM user visits customer X (e.g. known from calendar), information concerning customer X¹⁶ are rated higher.
 - **Links:** everything can be linked to everything. This gives a complete overview of all activities related to e.g. a single contact or appointment.
 - **Journal:** each data alteration can be watched (when, who, old value, new value)
 - **Telephony:** several kinds of telephony support (directly integrated into application, Skype, callto, etc.).
 - Notification service: call forwarding, event notification, phone call notes, etc.
 - **Offline file cache:** documents can be checked out for exclusive offline write access.
 - **Hosting operator interface:** creation of new mandators and users, database administration, deployment of updates, trouble shooting, etc.
 - **Authentication/ Authorization service:** extensive authorization concept including groups, explicit rights based on data object or implicit rights based on users.
 - **Templates:** can be created for all kind of data (contacts, todos, telephone calls, etc.).
 - **Views:** can be created for all kind of data (e.g., contacts in a certain region, todos in the next 10 days, documents created in the last 30 days, etc.).
 - **Search:** quick search for easy access, powerful field based search.
 - **Waste basket:** deleted data can easily be restored.
- Technical components
 - **GUI server:** communicating with user's browser (Internet Explorer, Mozilla Firefox, etc).
 - **Application server:** Rule&Action service, Index service, business operations, etc.
 - **Persistence layer:** database, document store.

¹⁶ latest appointment reports, current sales opportunities, current trouble tickets etc.

3.4.3 QoS requirements and priorities

All subsystems must be available at any time.

Time consumed in each server roundtrip must be acceptable by the user. The overall response time a user experiences consists of several parts (network delay, GUI server, application server, database, and browser). Time consumed in the subsystems under control of CAS PIA (GUI server, application server, database) should be less than one second.

3.4.4 Deployment structures

Figure 4 shows different coupling and distribution schemes of subsystems in CAS PIA.

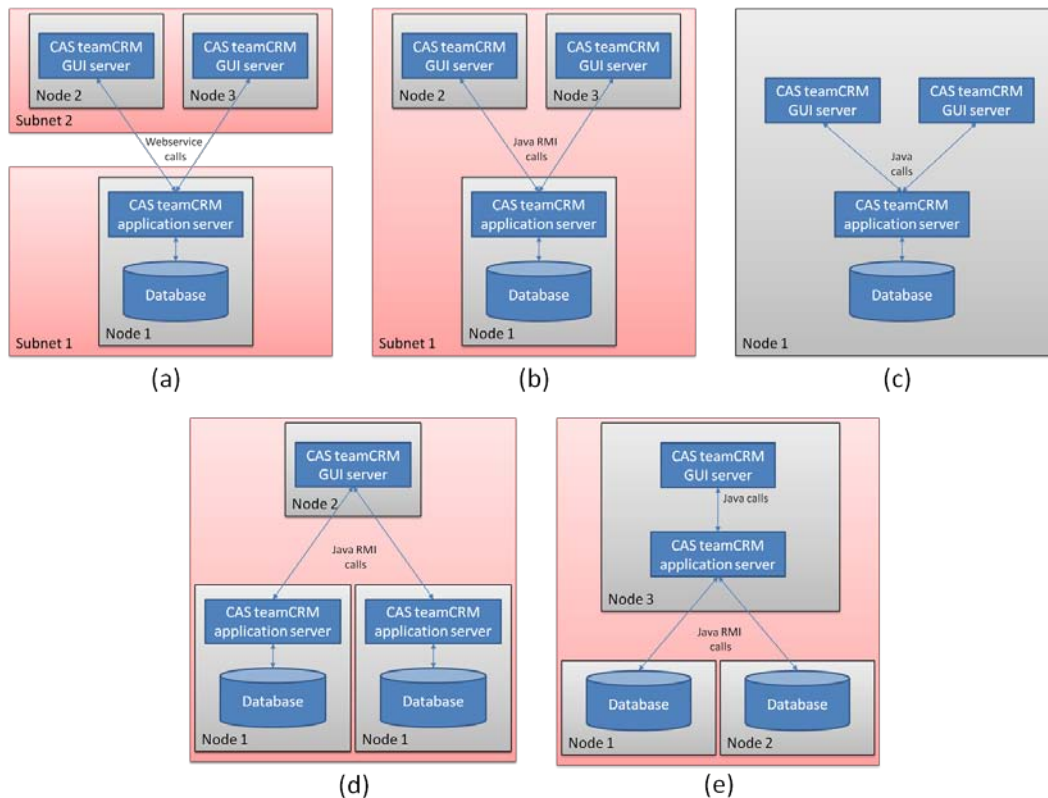


Figure 4 – Examples of coupling and distribution schemes in CAS PIA

Technically, CAS PIA supports a connector-like mechanism. All calls between subsystems are being made through a proxy object. This proxy object uses the Strategy pattern in order to choose the desired communication strategy (i.e., Java method calls vs. RMI vs. web-services) and communication parameters (node addresses, ports etc.).

However, currently the re-configuration of coupling and distribution of subsystems of CAS PIA is a manual process at deployment time that is, the operator sets the desired coupling/distribution scheme in a configuration script and (re-)starts the application. In order to support improved load-balancing within the system it is necessary to exchange the coupling scheme of, and communication protocols used between subsystems at runtime.

The actual decision which coupling/ distribution scheme to use depends on context factors, e.g.,:

- If the **GUI server is under heavy load**: move application server to another node and use remote communication. Add more GUI servers if necessary. However, users assigned to one GUI server cannot easily be moved to another GUI server as the GUI

server keeps track of GUI state per user. Thus only new users can be moved to new GUI servers. See Figure 4 (a) or (b).

- If **application server is under heavy load**: move to separate node, instantiate more application servers¹⁷ and use remote communication between applications servers and GUI servers. See Figure 4 (d).
- If **database layer is under heavy load**: move other subsystems (application server, GUI server) to another node, if necessary add more database subsystems and use remote communication between application server and database layer. See Figure 4 (e).

3.5 Conclusion

The CRM case study illustrates two different types of adaptations at several levels. It covers adaptations according to where a user is located and e.g., which communication channels are available. It also covers adaptations of the system according to which services are available to a client or which subsystem is under heavy load. The variability dimensions and their variants are identified.

As next step CAS will focus on the following research objectives:

- How to detect dynamic variability points from requirements specifications.
- How to model dynamic adaptability.
- How to perform safe re-configuration of systems at runtime.
- How to predict the behaviour of dynamically re-configurable systems and how to test/verify such systems.

The experimentation will be guided by these empirical objectives and will follow these steps:

- WP1 aims at providing tool support for the detection and derivation of dynamic variability points by means of semantics analysis of requirements and scenario documents by employing natural language analysis. CAS seeks to evaluate how well the WP1 approach and its associated tools can support the process of detection and derivation of dynamic variability points compared to the traditional, manual process.
- Experimentation with the DiVA modelling approach:
 - **Model expressiveness/ scalability**: can real-world systems be easily expressed in terms of the different models and model elements provided? Is it possible to express the configuration space together with relevant composition policies and constraints in a convenient way? Does the approach scale to real-world applications both in terms of manageability of variability and technical footprint? Do fundamental obstacles (such as combinatorial explosion of variants) exist?
The CAS teamCRM case study, a medium-sized business application will be used as a benchmark system in the context of this research question. This evaluation will be done qualitatively.

¹⁷ reminder: applications servers are stateless



- **Model consistency, change propagation:** Are dependant models kept consistent automatically in case of changes? If not: How difficult is it to keep them consistent manually. Is there at least partial tool-support?
- **Ease of use:** Ease of use will be qualitatively evaluated according to the level compliance of DiVA technology (e.g., notation, modelling) to industry standards and standard notations (such as UML) etc.
- CAS seeks to evaluate how well DiVA technology supports dynamic reasoning about the system context and how the results of this reasoning process can be used at runtime for recognizing the need for, and triggering and performance of dynamic re-configuration. This research question also seeks to evaluate the adaptation/ re-configuration techniques as such, i.e., can subsystems like services be added, updated, or removed; can the architecture and the collaboration of subsystems like services as well as quality-of-service properties be changed in a safe way without breaking the application?
- CAS seeks to evaluate how well testing and verification of adaptive systems is supported within DiVA studio.

A detailed description of the evaluation plan of DiVA results in the context of CAS case study will be given in D.6.2.

4 Requirements specification

The purpose of this section is to describe an initial set of consolidated requirements for the technologies, tools and processes needed to support dynamic variability of complex systems. This section also describes the methodology used to specify these requirements.

It is important to emphasise that the requirements presented here should be seen only as an initial baseline, agreed at month 6 in the execution of the DiVA project. The requirements mentioned in this section will be subjected to regular internal reviews within the DiVA consortium throughout the duration of the project. We anticipate them evolving and being further refined as the end users (industrial case providers) get to evaluate early deliverables from Work Packages (WPs) 1, 2, 3, 4 and 5.

The primary audience for this requirement specification is:

- DiVA partners involved in WPs 1-5. This section provides a starting point for defining more detailed technical specifications and road-mapping their work.
- DiVA case providers in WP 6. D6.2 will include the baseline experimentation plan proposing how each industrial case will demonstrate the validity of the deliverables provided by other work packages. It is very important that the experimentation plan considers the various end-user requirements that need to be validated.
- Product managers for other variability tools wishing to gain insight into the future needs of variability of complex systems.

The structure of this chapter is as follows:

First, we provide an overview of the philosophy and process followed to generate and document the common set of requirements by providing a template for the description of requirements.

Then we provide an overview of how requirements will be related to technical specifications in WPs 1-5.

Finally we present each individual requirement in a report format, and offer some conclusions and next steps.

4.1 Requirements template

The requirements in DiVA are based on a number of core principles that are the following: Technology and user-driven

- Precise
- Managed in an iterative way

To define and document the common set of requirements that respect these principles we have defined a template for the requirement description. In this section, we define the fields of this template.

4.1.1 Requirement status

The **Status field** is used to show the requirements lifecycle.

1. **New:** A requirement is first reported.
2. **Acknowledged:** Agreed among the stakeholders.

3. **Assigned:** Appointed person to be responsible for the specification.
4. **Closed:** Specification is approved.
5. **Stopped:** The requirement is cancelled.
6. **Reopened:** To handle major changes after the requirements are approved.

4.1.2 Priority of requirements

The **Priority** field reflects the stakeholders perceived important of the requirement as either:

- **High** – essential, must be addressed.
- **Medium** – important, but not essential.
- **Low** – nice to have.

4.1.3 Relationships between Requirements

We mainly use two types of relationships between requirements:

- **Parent of** and **Child of** are used where a more general requirement is further defined into several more specific requirements.
- **Related to** is used to denote other relations between requirements, such as between a functional requirement and a quality requirement or architectural constraint.

4.1.4 Requirement category

There are two main categories of product requirement:

- **Functional Requirements** describe which function the product will perform - “the what”. These requirements could be grouped under major functional areas.
- **Non-functional Requirements (Quality Requirements)** describe “how good“ a system needs to be in satisfying quality attributes such as availability, performance, usability and so on.

We define four subcategories of functional requirements according to the life cycle stage of an adaptive application:

1. **Requirements Analysis:** describes the requirements about the tools used to analyse the requirement of adaptive applications.
2. **Modelling:** describes the modelling functionalities required by adaptive applications.
3. **Runtime:** describes the functionalities that are required to support runtime variability.
4. **Verification and validation:** describes the functionalities that are needed to support verification and validation of systems.

4.2 From requirements to technical specifications

In this section, we describe the overall process of how requirements from WP6 migrate to technical specifications and then to solutions in WP 1-5. At present, the status of all requirements described in this section is marked as “**New**”. This reflects that the requirements have been created by the end users (case study partners) but are not yet assigned to the solution providers.

Within DiVA, the work package and task leaders of WP1-5 will initiate and manage an initial review of the baseline requirements. The objective is to translate the requirements into more

detailed technical specifications that can be assigned to individual technology and tool providers within the project.

This will be a two-way process. The baseline requirements have been consolidated and prioritised by industrial case providers based on “what” their case definitions need. Little consideration has been given to “how” those requirements will be implemented, or what the overall work effort on WP 1-5 will be. The work package leaders will refine their technical specifications and individual work plans to fulfil these requirements.

As each requirement is reviewed by WP 1-5, its status will change to “**Acknowledged**” and then to “**Assigned**” or “**Stopped**” depending on the result of the assessment. At the end of the process all requirements will be reviewed and work to fulfil them will have been planned, prioritised and assigned to individual technology providers.

Work package leaders from WP 1-6 also manage follow-up (quarterly) reviews of the requirements throughout the duration of the project to ensure that the current requirements and solutions continue to meet the needs of the industrial users.

4.3 Requirements

This subsection provides details of all categories of requirements defined at month 6 in the DiVA project. Consistent with the requirements process described in Subsection 4.1.4, the product requirements are categorised as **Requirement Analysis Requirements, Modelling Requirements, Runtime Requirements, Verification and validation Requirements** or **non-functional requirements**.

4.3.1 Requirement analysis requirements

R-1: Tool support for the derivation of dynamic variability spots			
Priority	Low	Status	new
Category	Requirements Analysis		
Date Submitted	07.04.2008		Last Update
Reporter	TG		Assigned To
Stakeholders	CAS		
Description			

4.3.2 Modelling requirements

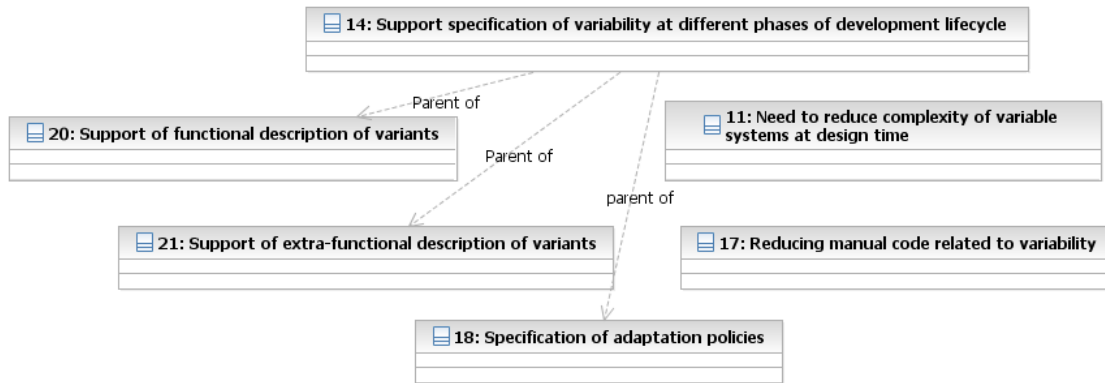


Figure 1 - Modeling requirements

R-14: Support specification of variability at different phases of development lifecycle			
Priority	High	Status	new
Category	Modelling		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	<p>We need modelling languages (meta-models) that support the specification of variability dimensions, configuration variants and composition policies. Several levels of abstraction and architectural views must be supported (such as PIM level, and PSM level). Composition policies must include:</p> <ul style="list-style-type: none"> - Dependencies between components/ services (X only in conjunction with Y, X not in conjunction with Y, versions, ...) - Cardinalities (such as "there must at least be one instance of a certain service/ component type") 		
Description	Using models to specify variability at design-time		



R-11: Need to reduce complexity of variable systems at design time			
Priority	High	Status	new
Category	Modelling		
Date Submitted	16.05.2008	Last Update	10.12.08
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	Support of handling and reducing exponential growth of the number of potential system configurations.		

R-20: Support of functional description of variants			
Priority	Low	Status	new
Category	Modelling		
Date Submitted	14.10.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	Designers must be able to describe the functional properties of each variant, for example the role of the variant and what is intended to do with.		

R-21: Support of non-functional description of variants			
Priority	Medium	Status	new
Category	Modelling		
Date Submitted	14.10.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	Designers must be able to describe QoS properties of each variant, for example information about performance, optimality, required resources.		



R-18: Specification of adaptation policies			
Priority	Medium	Status	new
Category	Modelling		
Date Submitted	14.10.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales, CAS		
Description	Ability to specify required configurations according to the context in terms of required functionalities and required QoS (Quality of Service).		

R-17: Reducing manual code related to variability			
Priority	High	Status	new
Category	Generation		
Date Submitted	07.04.2008	Last Update	10.12.2008
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	The MDA approach should reduce the manual code related to variability by generating a major part of it.		

4.3.3 Runtime requirements

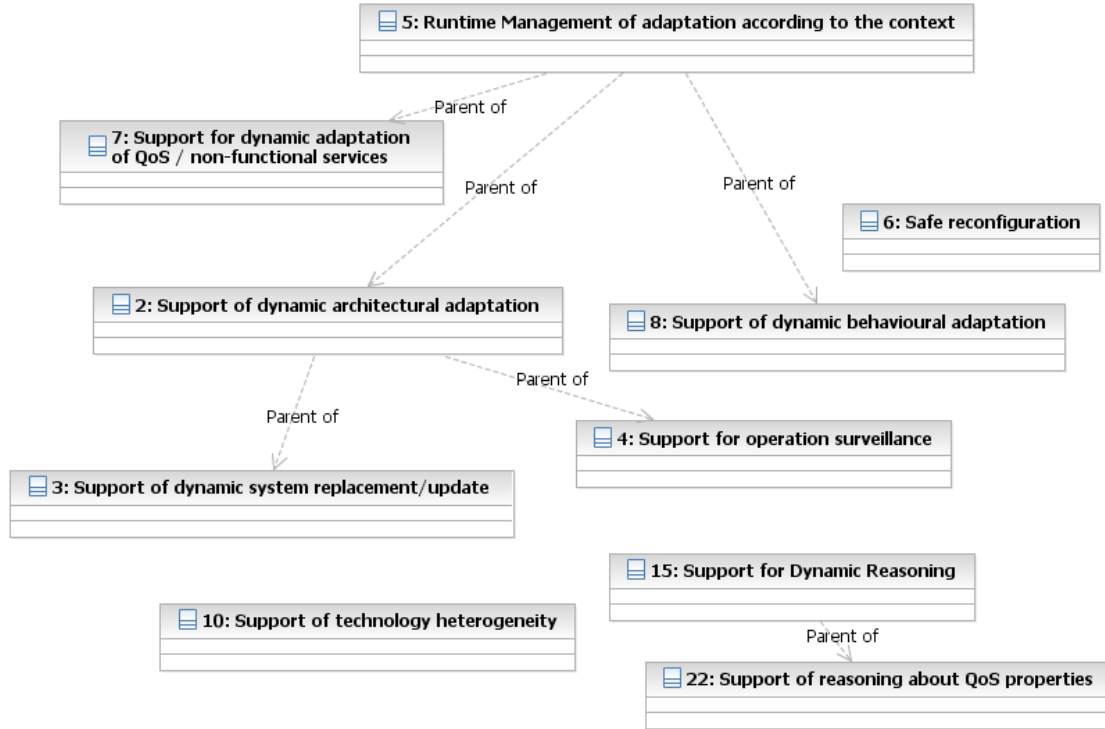


Figure 2 - Runtime requirements

R-5: Runtime Management of adaptation according to the context			
Priority	High	Status	new
Category	Runtime		
Date Submitted	16.05.2008		Last Update
Reporter	DA		Assigned To
Stakeholders	Thales		
Description	The adaptation framework must be able to collect context information and adapt at runtime.		



R-6: Safe reconfiguration			
Priority	High	Status	new
Category	Runtime		
Date Submitted	07.04.2008	Last Update	10.12.2008
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	A reconfiguration is safe if the system accomplishes the required functionalities after the reconfiguration without errors. A defined core process must come to a defined end state even if the overall process (i.e. Core process plus optional process parts) has undergone re-configuration. Re-configuration must not break core processes.		



R-2: Support of dynamic architectural adaptation			
Priority	High	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	Need to support runtime addition, removal, connection and disconnection of sub-systems. Dynamic service integration and removal during runtime. Dynamic replacement of connector components ¹⁸ in order to replace communication protocols between components/ services of the system.		

R-8: Support of dynamic behavioural adaptation			
Priority	Low	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	Support of variability in services interactions. Behavioural adaptation must be supported at non-functional level.		

¹⁸ the term 'connector' being used in the sense of architectural systems



R-3: Support of dynamic system replacement/update			
Priority	High	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	In a hosting scenario, dynamic component update without application outage is necessary: runtime robustness in case of component outage. Transparent hand-over to fallback component in case of component failure.		

R-4: Support for operation surveillance			
Priority	Medium	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	- QoS surveillance - Service state/ availability - Automatic detection of sub-system failure		



R-7: Support for dynamic adaptation of non-functional/QoS services			
Priority	Medium	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	DiVA@runtime could be able to support the adaptation of non-functional services such as security aspects.		

R-10: Support of technology heterogeneity			
Priority	High	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	We need the ability to assemble heterogeneous systems implemented with different technologies.		

R-15: Support for Dynamic Reasoning			
Priority	high	Status	new
Category	Runtime		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	- Analysis of the variability dimensions properties.		

	- Selection of feasible configuration.
--	----------------------------------------

R-22: Support of reasoning about QoS properties			
Priority	Medium	Status	new
Category	Modelling		
Date Submitted	14.10.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	The reasoning framework must be able to take into account QoS properties, like for instance performance priority.		

4.3.4 Verification and validation requirements

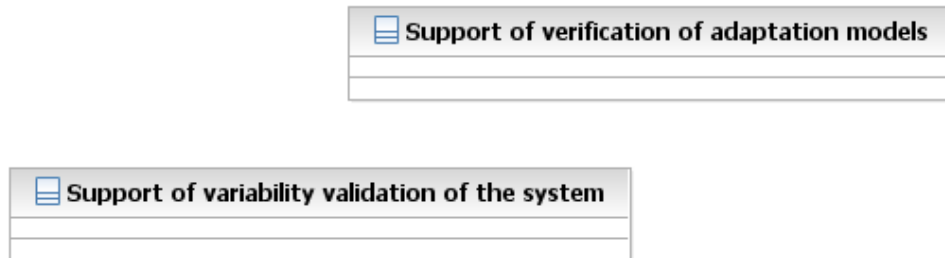


Figure 3 - Verification and validation requirements

R-19: Support of verification of adaptation models			
Priority	High	Status	new
Category	Verification/Validation		
Date Submitted	16.05.2008	Last Update	10.12.2008
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	Ability to check the correctness and consistency of the selected configuration (the configuration that adapts to the context). Ability to check if the selected configuration is the best one. Support for dynamic composition policy checking: Before components are added/ removed/		

	updated, policies must be checked whether system remains in consistent state (components should only be integrated into/removed from systems, if all necessary conditions are fulfilled after operation).
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

R-9: Support of variability validation of systems			
Priority	High	Status	new
Category	Verification/Validation		
Date Submitted	09.07.2008	Last Update	
Reporter	DA	Assigned To	
Stakeholders	Thales		
Description	Once deployed, the system that adapts to the context must be validated. Ability to check the correctness and consistency of the running system.		

4.3.5 *Non-functional / misc. requirements*



Figure 4 - Non-functional / misc. requirements

R-12: Technical infrastructure/ platform support			
Priority	High	Status	new
Category	Non-functional		
Date Submitted	16.05.2008	Last Update	
Reporter	TG, DA	Assigned To	
Stakeholders	CAS, Thales		
Description	DIVA technology must support the case study's technical infrastructure as specified in D6.1.		



R-13: Ease of use			
Priority	Low	Status	new
Category	Non-functional		
Date Submitted	28.07.2008	Last Update	
Reporter	TG	Assigned To	
Stakeholders	CAS		
Description	DIVA technology and tools should be reasonably easy to use, including but not limited to: - Documentation. - Integrated tool support, including user guidance.		

R-16: Compliance to standards			
Priority	Medium	Status	new
Category	Non-functional		
Date Submitted	28.07.2008	Last Update	
Reporter	TG	Assigned To	
Stakeholders	CAS		
Description	Notations, description languages etc. should be compliant to existing standards as much as possible.		

5 Conclusions

This document provides a description of two different case studies presenting two industrial domains, architectural styles, and variability dimensions.

An initial set of consolidated requirements for DiVA has been derived from the needs of adaptive complex software solutions across these two industrial domains, development environments and processes. This initial set of requirements will allow each of the case study providers to develop their individual experimentation plans to evaluate the DiVA technology.

The challenges facing us in performing the studies are:

- Availability of the required technology in scope and in time. Mitigation: Discuss the requirements with other WPs.
- More detailed definition of how to perform studies, subjects involved, data collection routines and metrics. Mitigation: Allocate enough resources and make a detailed plan with time schedule. Work is going on to make a time schedule for execution of the studies based on the DiVA roadmap.
- Changes in requirements. Mitigation: Iterate over the evaluation plan.

In closing, we intend to use the case studies presented in this document for exercising all other activities in the technical work packages of DiVA project by evaluating the respective tools and techniques developed therein.

6 Appendix A: Thales crisis management use cases

The following use case diagram shows the general actions that the Crisis Manager can take.

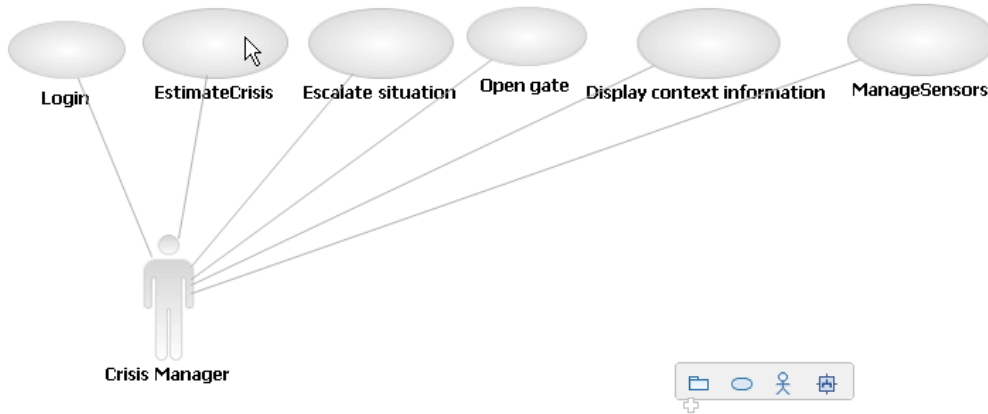


Figure 1 – Crisis Manager general actions diagram

The following use case diagram shows all the actions that the Crisis Manager can take to manipulate the Sensors and to handle Messages.

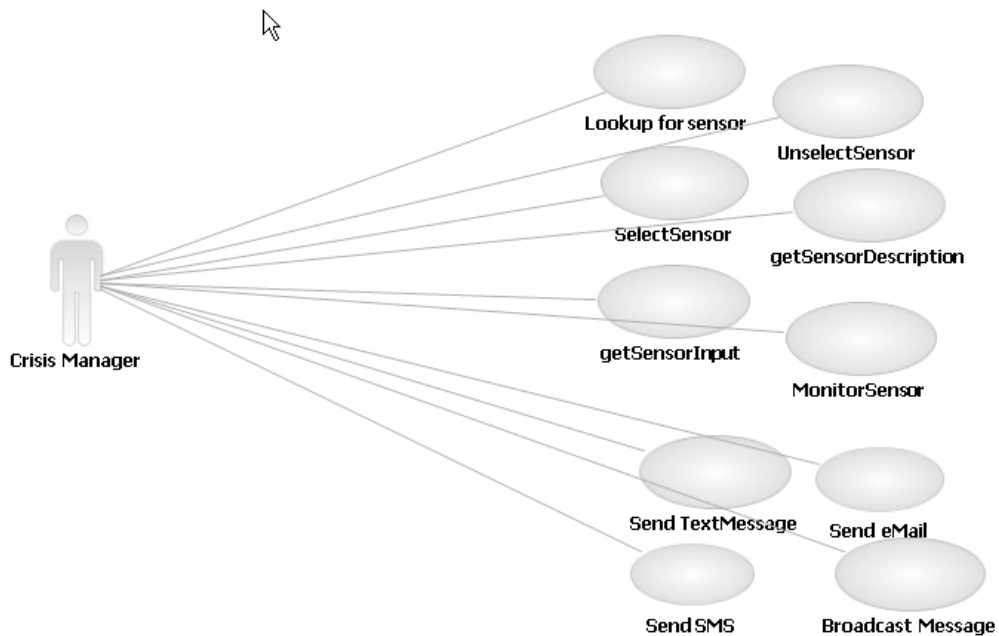


Figure 2 – Crisis Manager sensor&message diagram

The following Use Case diagram shows the general actions that the Crisis Manager can take to interact with the Organisations. An organisation consists of a group of different agents of the same type. Each organisation and agent is identified by contact information that consists of his name, description, location, phone number and email address.

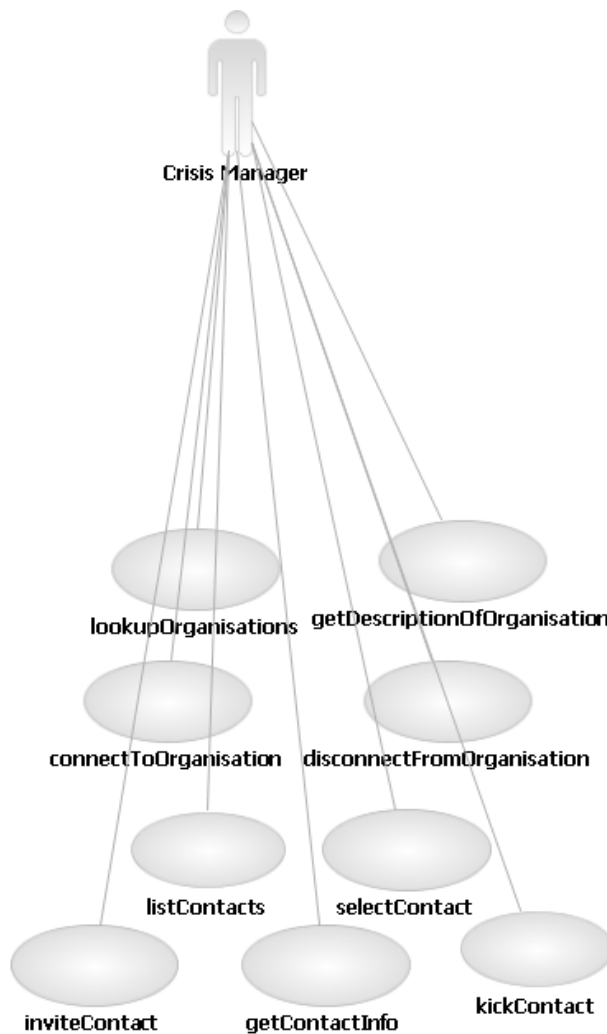


Figure 3 – Crisis Manager organization&contact diagram

The following use case diagram shows the general actions that the Crisis Manager can take to manipulate the Dash Board. A dashboard is a sort of a collaborative virtual workplace where agents can share information and files.

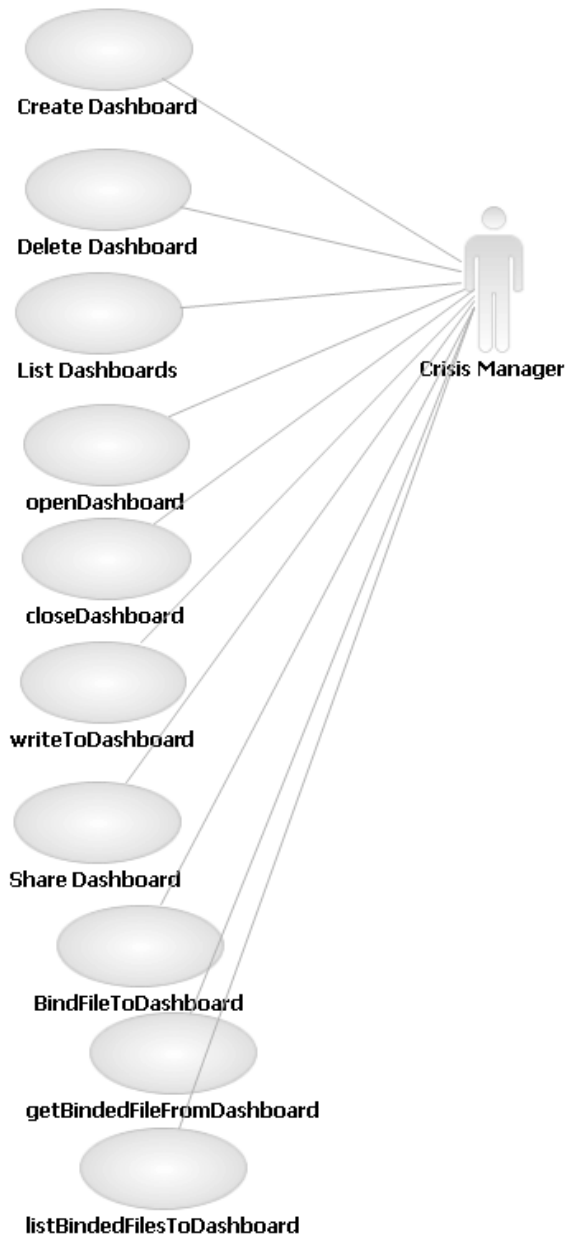


Figure 4 – Crisis Manager dashboard diagram

The following use case diagram shows the general actions that the Crisis Manager can take to manage the tasks. A task is a specific assignment that the crisis manager hands over to an organisation which is represented by its agents, like extinguishing a fire. Each task can be assigned to a specific agent, and the crisis manager can follow it up through these functionalities and update it.

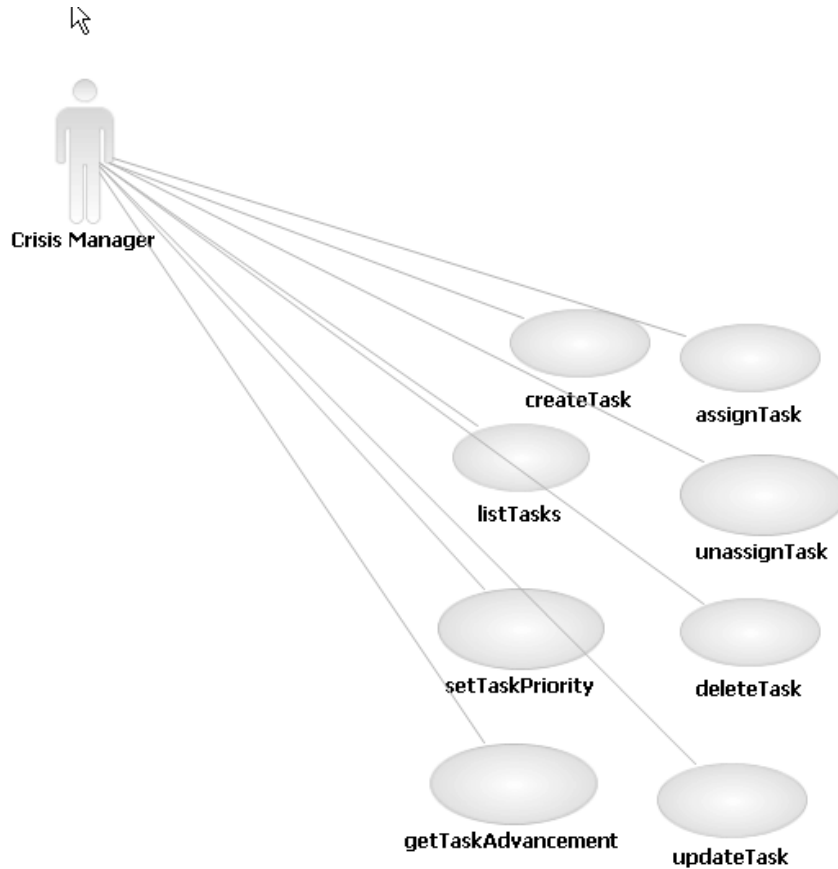


Figure 5 – Crisis Manager tasks diagram

The following use case diagram shows the general actions that an Agent can execute. For instance it can manage a task, make a report on the victims/ damages using its GUI implementation, and it can contact other agents or organisations by sending different types of messages.

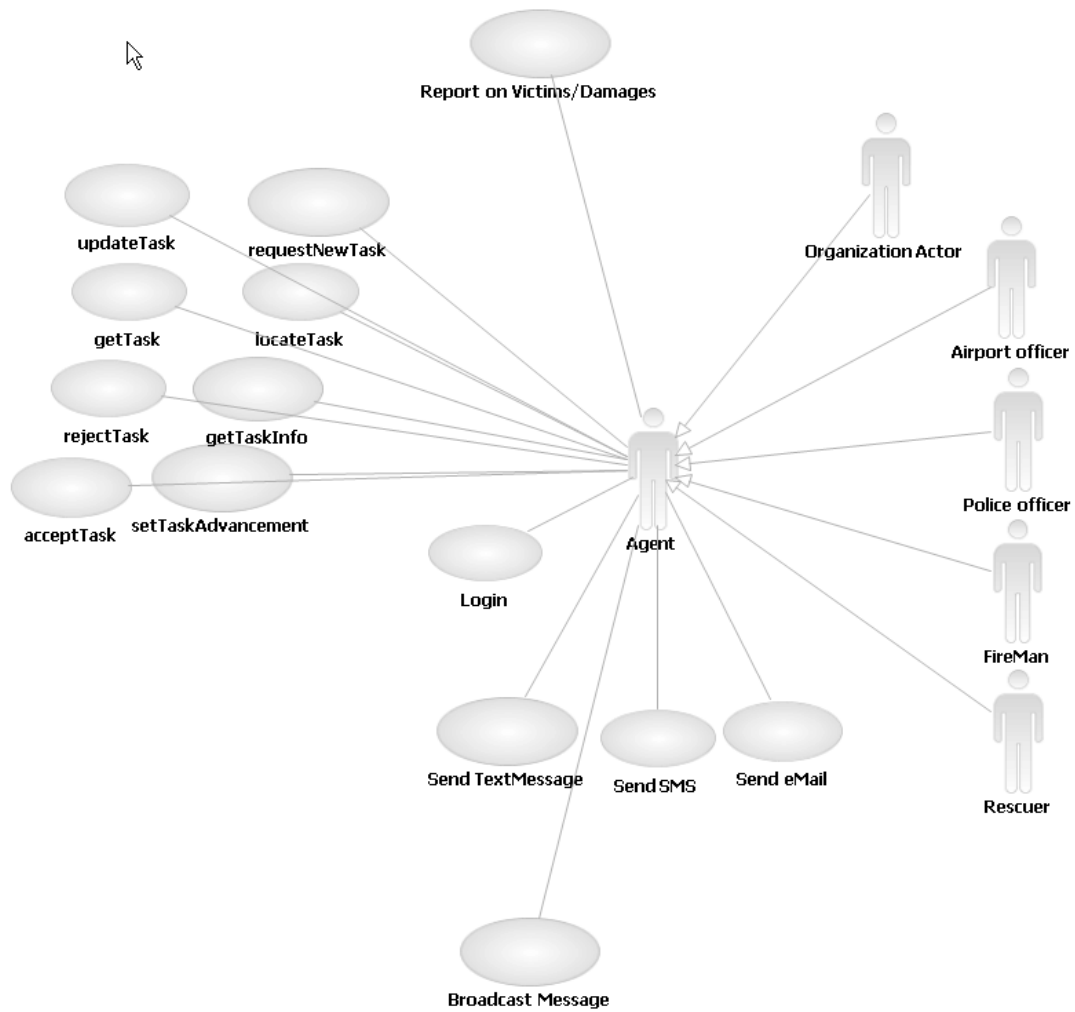


Figure 6 – Agent general actions diagram

The following use case diagram shows the actions that the Security Administrator can take; for instance, he can configure the system for the first time (create the system users and the GUI graphical placement), and he can setup the sensors that will interact with the crisis management system.

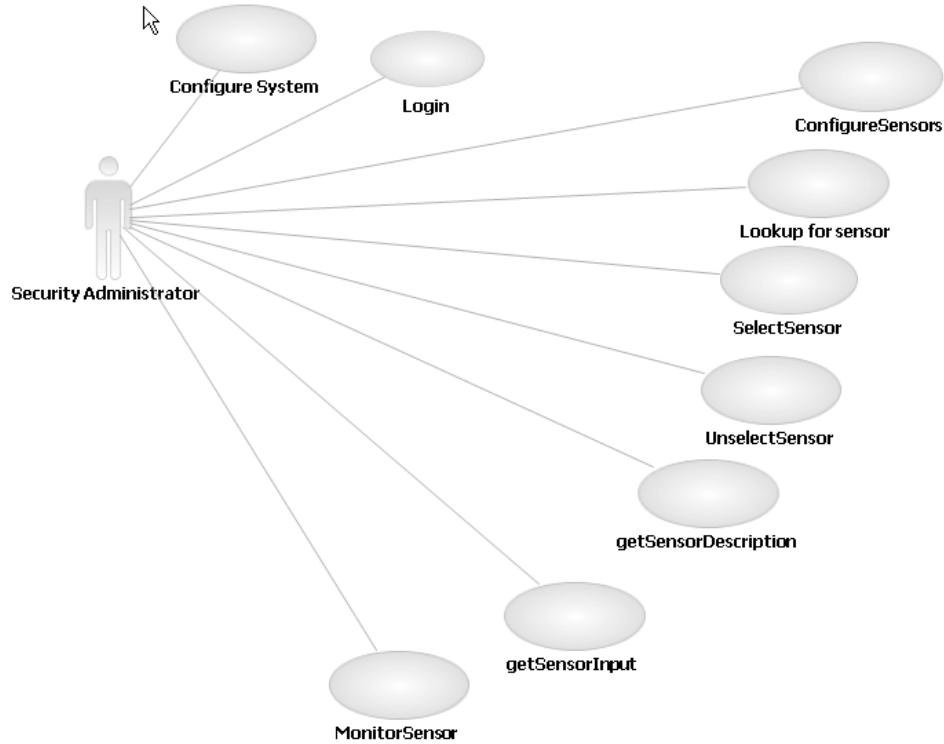


Figure 7 – Security Administrator general actions diagram