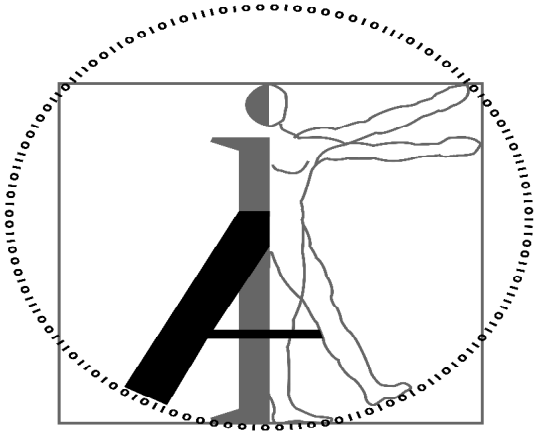
Initial Orchestration Plane Design

Deliverable D2.1

Autonomic Internet (AutoI) Project

FP7-ICT-2007-Call 1 - 216404



Abstract: This document describes the initial design of the Orchestration Plane (OP) within the AutoI Project. It extends the work done in deliverable D6.1 into a more detailed description of the AutoI Orchestration Plane. This deliverable provides a definition of Orchestration within AutoI and it defines the functions and requirements of such a plane. Next, the document presents a preliminary design for the OP as well as some initial implementation decisions.

Document Properties

Properties:

Document Number:	FP7-ICT-2007-CALL1-216404 AutoI/D2.1
Document Title:	Initial Orchestration Plane Design
Document responsible:	Guy Pujolle
Author(s) / Editor(s):	Guy Pujolle (LIP6), Zeinab Mohavedi (LIP6), Meriem Abid (Ginkgo), Daniel Fernandes Macedo (LIP6), Steven Davy (WIT), Abderhaman Cheniour (INRIA), Javier Rubio Loyola (UPC), Giannis Koumoutsos (UPatras), Alex Galis (UCL)
Reviewed by:	Zeinab Mohavedi (LIP6), Meriem Abid (Ginkgo), Daniel Fernandes Macedo (LIP6), Steven Davy (WIT), Abderhaman Cheniour (INRIA), Javier Rubio Loyola (UPC), Giannis Koumoutsos (UPatras), Alex Galis (UCL)
Dissemination Level:	Public Information
Status of the Document:	Final Version
Version:	1.0

This document has been produced in the context of the Autonomic Internet (AutoI) Project. The AutoI Project is part of the European Community's Seven Framework Program for research and is as such funded by the European Commission.

Revision History

Revision	Date	Issued by	Description
V0.1	13/10/08	Guy Pujolle	First draft
V0.2	02/11/08	Daniel Macedo	Improvements using Steve's comments, added text from INRIA, UPC and WIT.
V0.3	05/11/08	Daniel Macedo	Added text from UPC.
V0.4	10/11/08	Daniel Macedo	Added contribution from UPatras, changes proposed by UPC and worked on comments of the consortium.
V0.5	11/11/108	Meriem Abid	Contribution from Ginkgo.
V0.6	18/11/08	Steven Davy	Added content on Distribution and Policies for the Orchestration Plane.
V0.7	21/11/08	Daniel Macedo	Improvements on the text based on the Venice discussions.
V0.8	24/11/08	Daniel Macedo	Rework on the organisation of the text to reflect the evolution in the content.
V0.9	24/11/08	Zeinab Movahedi	Rework on previous comments, added initial deflexion regarding to the interface of KP/OP, Knowledge updater Behaviour , Federation task
V0.10	25/11/08	Daniel Macedo	Work on the introduction and the OP/KP interface. Added a picture and text showing the deployment scheme of the DOCs.
V0.11	26/11/08	Zeinab Movahedi	Added initial text for Interface of ANPI/OP, OP/Virtual Resources
V0.12	27/11/08	Daniel Macedo	Improvements on the text related to policies, adjustments of the text to fit the 26/11/08 telco decisions.
V0.13	07/12/08	Daniel Macedo, Zeinab Movahedi	Added a brief summary, Improvements based on Ginkgo, Patras comments. First review from LIP6.
V0.14	15/12/08	Zeinab Movahedi, Daniel Macedo	First round of reviews.
V01.5	07/01/09	Zeinab Movahedi, Daniel Macedo	Second round of reviews. Contributions from Zohra, Giannis, Javier, Alex.
V1.0	12/01/09	Daniel Macedo	Final formatting for the public version of the document.

Table of Contents

Executive Summary **4**

Abbreviations **5**

1 Introduction **6**

1.1 Definition of Orchestration within AutoI 7

1.2 Related Work 8

1.3 Interaction of orchestration, management and virtualization planes 9

1.4 The responsibilities of the OP in the AutoI Architecture 9

2 Orchestration Plane Functions and Requirements **11**

3 Preliminary Orchestration Plane Design **12**

3.1 Dynamic Planner 14

3.2 Behaviours 15

3.2.1 Federation Core Behaviour *17*

3.2.2 Distribution Core Behaviour *18*

3.2.3 Negotiation Core Behaviour *19*

3.2.4 Governance Core Behaviour *20*

3.2.5 AMS Behaviours *21*

3.3 Intra- and Inter- system views 22

3.4 Interfaces of the DOC 22

3.4.1 The DOC/Knowledge Plane Interface *23*

3.4.2 Interface SP/DOC, Virtual Resources/DOC and AMS/DOC *24*

4 Conclusions and Summary **25**

5 References **26**

Executive Summary

Next generation network and services require the optimisation of multiple types of orchestration mechanisms, enabling network operations (i.e. optimisation for initialisation, dynamic reconfiguration, adaptation and contextualisation, dynamic service deployment support and other tasks) and service tasks to be optimised. In the AutoI project, orchestration refers to the mediation of several Autonomic Management Systems with each other in an attempt to deal with the problem of management domain heterogeneity and integration. The later is a relevant innovation of the Autonomic Internet (AutoI) architecture as it provides the autonomic integration and federation of several autonomic management systems.

The objective of this deliverable is to describe the initial Orchestration Plane (OP) design within the overall AutoI architecture. This document defines the functions, requirements and the concepts behind the operation of the components that make the orchestration plane, the DOCs. A DOC, or Distributed Orchestration Component, is a functional entity of the AutoI architecture that deals with inter-domain management tasks, such as the federation, negotiation, governance, and distribution of management domains.

This deliverable is related to the following overall project objectives: *Objective 1 - Define an autonomic architectural solution for supporting Internet services.* The OP will allow the interoperation of different management domains, which may use different sets of business goals/policies, and which may have different QoS requirements. The OP will serve as a broker, arranging the interconnection of different management domains. It will automatically try to solve conflicts and suggest solutions to optimize the behaviour of autonomic FCAPS solutions. *Objective 7 - Control of virtual resource algorithms.* A new set of control algorithms will be defined providing a faster, more optimized response to changes on resources or to user demands. These control algorithms will be contained within an orchestration plane. This optimization is achieved via re-configuration of the parameters of the virtual resources (e.g. routing tables, attribution of classes to flows).

As future work beyond the initial design presented in this deliverable, we will specify the components and interfaces required for the OP to handle federation, distribution, negotiation and governance. The main function of the OP is to orchestrate the interaction of the AMSs, the complete definition of the DOC requires the instantiation of the AMSs. A detailed architecture will be produced in future deliverables. An open issue that will be addressed is the definition of the scope or nature of the orchestration policies. Working in conjunction with WP3, we will design the concepts (classes and relationships) supporting the orchestration part of the AutoI information model (AIM). Those classes will be used in the inter-DOC communication as well as in the negotiation of parameters and SLAs between DOCs and AMSs.

Abbreviations

AutoI	Autonomic Internet Project
AIM	AutoI Information Model
AMS	Autonomic Management System
DOC	Distributed Orchestration Component
DP	Dynamic Planner
FCAPS	ISO Telecommunications Management Network model and framework for network management. FCAPS is an acronym for Fault, Configuration, Accounting, Performance, Security, which are the management categories into which the ISO model defines network management tasks.
FE	Functional Element
FI	Future Internet
IBM	International Business Machines
FIN	Future Internet Network
IP	Internet Protocol
KP	Knowledge Plane
MIB	Management Information Base
MP	Management Plane
OP	Orchestration Plane
OS	Operating System
OSKMV	OSKMV planes: Virtualisation Plane (VP), Management Plane (MP), Knowledge Plane (KP), Service Enablers Plane (SP) and Orchestration Plane (OP).
OWL-S	Ontology Web Language for Services
QoS	Quality of Service
SAWSDL	Semantic Annotations for WSDL
SNMP	Simple Network Management Protocol
SP	Service Enablers Plane
SLA	Service level Agreement
vCPI	Virtualisation Component Programming Interface
vSPI	Virtualisation System Programmability Interface
VP	Virtualisation Plane

1 Introduction

Autonomic management systems, as initially defined in the IBM manifesto [9], have been defined as management systems of a single system. In networking, a number of autonomic management system have to perform different management tasks covering various nodes, links and services. Due to the existence of several management standards, different protocols and different vendors, managing a network is much more complex than managing a single isolated system. Thus, it is not practical to devise a single autonomic control loop that autonomically adjusts all the FCAPS (fault, configuration, accounting, performance and security) aspects of a network. This means that we need to define one or more autonomic loops for each of those management aspects in order to simplify the design of each control loop.

However, the operation of the network management system will depend on the interaction of all those control loops, which must ensure, amongst other key aspects, that the network operates within normal parameters set by the business goals of the operators. Also, the decisions of a control loop may go against the objectives of another one. As an example, an autonomic security component may use a heavier encryption scheme to improve the security of the network, however this encryption scheme may require too much processing and bandwidth, reducing the maximum throughput of the network to a level below the performance dictated on the SLA.

In networking, two sub-networks having different managers must interconnect. This requires that the protocols as well as the configuration of the network (i.e. security policies, QoS and SLAs) are compatible. If they are not, either a re-negotiation and re-configuration process is required, or translation services (gateways) must be installed in the border of the two networks.

In order to solve those problems we are introducing a new system or plane, the Orchestration Plane (OP), enabling cooperation of the various autonomic control loops ensuring their decisions are not orthogonal. This cooperation, or orchestration, ensures that the overall optimization goals of each autonomic component and control protocol are aligned with the goals and SLAs defined for the entire network, Orchestration also means that autonomic management domains run by different operators or administrators are able to automatically adjust their configuration to accommodate the federation of networks.

The need for an Orchestration Plane (OP) arises from the deployment of several autonomic control loops with different administrators or management goals, which would not be able to interoperate without a set of translation, negotiation, federation and deployment functions. Thus, orchestration deals with the meta-management of autonomic management systems, that is, the deployment and reconfiguration of autonomic management control loops in order to allow their interoperation. This is achieved based on a set of high-level goals, defined for each of the managed network domains that form the orchestrated network. The OP ensures the interoperation of management systems, even though those systems use different set of high-level goals and management standards. This process may be accomplished through the negotiation of new SLAs and policies, the deactivation of conflicting management systems followed by the activation of other management systems, or the migration of such systems or parts of them within the orchestrated network. The entire orchestration process is governed by Orchestration Policies, which dictate what are the compromises that each of the managed domains are willing to make for the sake of interoperability.

This deliverable presents the Orchestration Plane concept, a networking plane that implements the notion of orchestration of the management systems, as well as its initial design in the Autonomic Internet project. The orchestration concepts are realised in the AutoI architecture with the help of Distributed Orchestration Components (DOCs), which

support the federation and distribution of self-governing autonomic management systems (AMSs) through its negotiation activities/tasks.
Throughout this document the term high-level policies or business objectives are particularly linked to high-level aspects of management and control over a given system. This is, by no means, aligned to economical aspects such as pricing strategies.

1.1 Definition of Orchestration within AutoI

The purpose of the Orchestration Plane is to govern and integrate the behaviours of the network in response to changing context and in accordance with applicable high level goals and policies. It supervises and integrates all other planes behaviour insuring integrity of the Future Internet management operations. The Orchestration Plane can be seeing as a control framework into which any number of components can be plugged into or out in order to achieve the required functionality.
The Orchestration Plane would also supervise the optimisation and the distribution of knowledge within the Knowledge Plane to ensure that the required knowledge is available in the proper place at the proper time. This implies that the Orchestration Plane may use either very local knowledge to deserve a real time control as well as a more global knowledge to manage some long-term processes like planning. The Orchestration Plane would host several Autonomic Management Systems (AMSs). It is made up of one or more Distributed Orchestration Components (DOCs), and a dynamic knowledge base consisting of a set of data models and ontologies and appropriate mapping logic and buses.
Each AMS represents a set of virtual entities, which manage a set of virtual devices, sub-networks, or networks using a common set of policies and knowledge. The AMSs access a knowledge base, which consists of a set of data models and ontologies. A set of DOCs enable AMSs to communicate and cooperate with each other, by the use of Behaviours, which act as stubs in the DOC/AMS communication. Buses enable the Orchestration Plane to be federated with other Orchestration Planes. The Orchestration Plane acts as control workflow for all AMSs ensuring bootstrapping, initialisation, dynamic reconfiguration, adaptation and contextualisation, optimisation, organisation, closing down of AMSs. The Orchestration Plane provides assistance for the Service Lifecycle Management, namely during the actual creation, deployment, activation, modification and in general, any operation related to the application services and/or management services. The DOC enables the following functions across the orchestration plane:
Federation: The Federation enables a set of domains (AMS Domain or Orchestrated Domain) to be combined into a larger domain (Orchestrated Domain or two-combined orchestrated Domain) guided by common high level goals, while maintaining local autonomy.
AMS Federation: each AMS is responsible for its own set of virtual and non-virtual resources and services that it governs as a domain. Federation enables a set of domains to be combined into a larger domain (Orchestrated Domain) guided by common high level goals, while maintaining local autonomy.
Orchestration Federation: two orchestrated Domains federate to make a larger orchestrated Domain. Here, the federation should take into account the different goals of tow different orchestrated Domains which would be combined and decide if federation will be possible.
Negotiation: in AUTOI negotiation can take place between autonomous entities with or without human intervention. DOCs and AMSs are the main entities that can be engaged in negotiations to achieve their goals. Each AMS advertises a set of capabilities (i.e., services and/or resources) that it offers for use by other components in the Orchestration Plane.

The DOC performs negotiation between the AMSs for the fulfilment of a specific SLA, defined by the operators of the managed orchestrated domains.
Distribution: the DOC provides communication and control services that enable tasks to be split into parts that run concurrently on multiple AMSs within the Orchestration Plane.
Governance: each AMS can operate in an individual, distributed, or collaborative mode (i.e. in federation). The AMS collects appropriate monitoring data in order to determine if the virtual and non-virtual resources and services that it governs need to be reconfigured. High level goals, service requirements, context, capabilities and constraints are all considered as part of the decision making process.
System Views: The DOC is responsible for managing the system views that are stored and diffused using the knowledge plane (detailed in D4.1, section 3.3.4). DOCs will fetch the information required for their operation from the AMSs as well as the services and resources (through the vSPI interface defined in D1.1, section 7).

1.2 Related Work

The autonomic concept was proposed to overcome the growth of complexity of current and future networks. There are some new concepts that deal with the autonomic aspect and the generic self-* properties. The idea is to facilitate the management and/or the control of networks regarding the growth of complexity. D. Clark in his paper Knowledge plane [1] recommends the construction of a new generation of networks able to "self-manage" themselves given high-level objectives without any human intervention. Clark's proposal of a knowledge plane in fact can be seen of a junction of the management, orchestration and knowledge planes of the AutoI project. In the project we decided to separate those three planes in order to better tame the complexity.
Other autonomic architectures like Focale proposed by Motorola [2] extend the knowledge plane concept by introducing high-level goals. Strassner's inference plane is another proposal for the orchestration of networks.
Other architectures were proposed through European FP7 program but again the objective is different from AutoI:
- The ANA Project aims at exploring novel ways of organizing and using networks beyond legacy Internet technology [4]. Their focus is mostly in protocols, not in management and orchestration.
- The HAGGLE project deals with an innovative paradigm for autonomic opportunistic communication [6]. This project aims at developing a cross-layer network architecture exploiting intermittent connectivity by supporting opportunistic networking paradigm. In this project the needs for orchestration are on the device level, while in AutoI we are dealing with the orchestration of networks.
- The BIONETS project aims at a novel approach able to address the challenges of pervasive computing [7]. Learning from nature and society will allow to overcome heterogeneity and to achieve scalability via an autonomic peer-to-peer communication paradigm.
- The CASCADAS project has the objective of developing Component-ware for Autonomic, Situation-aware Communications and Dynamically Adaptable Services [5]. The project wants to propose an innovative architectural vision based on self-organized distributed components for autonomic and situation-aware communication.
- The Ambient Networks (AN) [10] is a FP6 project which envisaged the development of a software-driven network control infrastructure for wireless and mobile networks that will run on top of all current network physical infrastructures to provide a way for devices to connect to each other, and through each other to the outside world and to provide seamless service provisioning and roaming.

- 4D is a new architectural model for the Internet, where tasks are divided into 4 planes: Decision, Dissemination, Discovery and Data [3]. In 4D, the data plane is a simple plane, which only acts based on the configurations received by the decision plane. Decisions are taken based on the information recuperated by the Discovery plane, which constructs a view of the physical resources. Next, the decisions are sent to the Data plane using the Dissemination plane. The paper does not present any hard data, simulation or implementation to show the benefits of their architecture, however the authors argue that the main advantage of such an architecture is in the centralization of decisions into one single plane, removing the problems of multiple layers dealing with similar issues.
4D has two main differences with regards to the planes in AutoI. First, it fuses the management, control and orchestration planes into one, however it does not describe a framework or design patterns to make the design of autonomic networks a tractable task. Second, it does not deal with the fact that we cannot rely on one single management entity, once each domain will be operated by a different organization. The orchestration plane, however, accounts for this fact, allowing the negotiation and federation of different management domains.

1.3 Interaction of orchestration, management and virtualization planes

An overall AutoI autonomic management architectural model was described in deliverable D6.1 and its update in the D4.1. It consists of a number of distributed management systems described with the help of five abstractions - the OSKMV planes: Virtualisation Plane (VP), Management Plane (MP), Knowledge Plane (KP), Service Enablers Plane (SP) and Orchestration Plane (OP).
Together these distribute systems form a software-driven network control infrastructure that will run on top of all current network (i.e. for fixed, wireless and mobile networks) and service physical infrastructures to provide a way for devices or attachments to connect to each other, and through each other to the outside world and to provide seamless service provisioning.
The OP will interact with the management plane through Behaviours, defined in Section 3.2.5. Each DOC will control one or more AMSs, using the interfaces identified in the deliverable D4.1. Each DOC will deal with the orchestration issues related to the interoperation of the AMSs overseen by the DOC. For supporting these tasks, the DOCs will require information from the virtualization plane, using the vSPI interface (which is defined in WP1, a first version is available in D1.1) to fetch the required information.
Further, DOCs will aid in service deployment, starting up or closing down network and user services in the AutoI architecture. The DOC defines constraints on the deployment of new services, such as the set of virtual routers or networks where the service will be installed, as well as some of its execution parameters.

1.4 The responsibilities of the OP in the AutoI Architecture

The role of the Orchestration Plane is to govern, dynamically adapt and optimize autonomic control loops in response to changing Orchestration–aware context and in accordance with applicable high-level goals and policies. It supervises and it integrates all other planes' behaviour, ensuring integrity of the Future Internet management and control operations. Besides adapting the configuration of AMSs, the Orchestration Plane may also

bootstrap and close down AMSs when needed. The Orchestration Plane can be thought of as a control framework into which any number of components can be plugged into in order to achieve the required functionality.
The need for an Orchestration Plane (OP) arises from the deployment of several autonomic control loops with different administrators or management goals, which would not be able to interoperate without a set of translation, negotiation, federation and deployment functions. Thus, orchestration deals with the meta-management of autonomic management systems, that is, the deployment and reconfiguration of autonomic management control loops in order to allow their interoperation. This is achieved based on a set of high-level goals, defined for each of the managed network domains that form the orchestrated network. The OP ensures the interoperation of management systems, even though those systems use different set of high-level goals and management standards. Ontology translation and mapping techniques between different data models based on the common information model can be used to create the common language upon participating entities that can negotiate, federate etc. This process of interoperation of management systems may be accomplished through the negotiation of new SLAs and policies, the deactivation of conflicting management systems followed by the activation of other management systems, or the migration of such systems or parts of them within the orchestrated network. The entire orchestration process is governed by Orchestration Policies, which dictate what are the compromises that each of the managed domains are willing to make for the sake of interoperability.

2 Orchestration Plane Functions and Requirements

The OP collaborates with all the other planes and as such it requires the following key functions and requirements:

- Take into account high-level goals (represented as policies, for example) and customer needs. It does not consider low-level technical details or high-level business details.
- The knowledge required for intra- and inter- domain orchestration must be timely disseminated by the Orchestration Plane and related components of the architecture.
- Depending on the service to be provisioned, the reaction time of the DOC must be constrained within certain boundaries in order to achieve the defined SLAs.
- The Orchestration Plane must have interfaces to interact with the AMSs (i.e. to cope with environment changes and conflicts).
- It should provide support for the AMSs to define their dependencies on other components and services, as well as their expected operational conditions (e.g. based on policies describing their required services and virtual resources).
- The cooperation of the DOCs from different domains requires the use of open protocols and standardized information and data models.
- The Orchestration Plane must be aware of the state of virtual resources using the specific interface between virtualisation plane and orchestration plane, and the information stored on the Knowledge Plane (deliverable D4.1).
- Solve conflicts arising from orthogonal goals on different AMSs. Thus, it must be capable to reach a compromise, allowing the overall system to achieve its purpose.
- It acts as control workflow for all AMSs ensuring bootstrapping, initialisation, contextualisation, closing down of AMSs. It also controls the sequence and conditions in which one AMS invokes other AMS in order to realize some useful functions (i.e., an orchestration is the pattern of interactions between AMSs).
- Enhancement and evolution: The Orchestration Plane would allow relevant number of components to be plugged into or out in order to achieve the required functionality and without interruption of normal operation.

3 Preliminary Orchestration Plane Design

The orchestration plane concept in the AUTOI approach is composed by several Distributed Orchestration Components (DOCs). Each DOC will be responsible for the interaction of several subordinated AMSs. DOCs will interact among themselves whenever necessary, in order to guarantee end-to-end SLAs and SLOs. Communication between DOCs is achieved by a set of buses.
The Orchestration Plane governs the execution of the AMSs. It acts as control workflow for all AMSs ensuring bootstrapping, initialisation, dynamic reconfiguration, adaptation and contextualisation, optimisation, organisation, closing down of AMSs. It also controls the sequence and conditions in which one AMS invokes other AMS in order to realize some useful functions (i.e., an orchestration is the pattern of interactions between AMSs). The Dynamic Planner is responsible for those tasks. Finally, each DOC will interact with the Knowledge plane to fetch information regarding the controlled Behaviours (see section 4). This information will vary for each DOC, and will compose the situated view of the DOC. The Situated View, defined in Section 3.3.5 of D4.1, defines the information that is needed for the operation of an autonomic component and from where it must be collected.
The Orchestration Plane is made up of one or more orchestrated Autonomic Management Systems (AMSs). Each controlled AMS will have its associated Behaviour in the DOC. The Behaviour is a wrapper for the AMSs, providing the interfaces and functionality needed for the interaction with the DOC. There is one DOC for each orchestrated domain, which will communicate with the DOCs of other orchestrated domains to reach agreements, allowing the operation of the network as a whole.
A set of buses enables the federation of the Orchestration Plane with other Orchestration Planes. In this case, DOCs from two or more administrative domains negotiate their federation. The aspects of federation, negotiation and governance will be treated in detail in Sections 3.2.3 to 3.2.6.
A policy-based management system abstracts the behaviour and dynamic decisions of a system from its functionality. Therefore, the functionality of a system may stay the same, but its behaviour, specifically with respect to changing contextual conditions, may be varied. Policies are used in the Orchestration Plane as a means to adapt its behaviour with respect to changing contextual conditions, for example, changing business objectives, network resource availability or domain membership. As the orchestration plane is concerned with the orchestration of distributed management systems, its policies are directly associated with the requirements of those management systems, or AMSs in the case of AutoI. Essentially, the behaviour of the components of the Orchestration Plane, namely the DOCs, is dictated by the capabilities of the distributed AMSs. For example, a single AMS may be deployed using a DOC. The behaviour of this deployment may need to be dictated by the "owner" of the AMS who will define the policies with which the AMSs should be deployed, and the behaviour of the DOC with respect to any coordination with the deployed AMS. Other types of policies the DOC should enforce are those delegated to it by the AMS to perform negotiation, federation or distribution on its behalf. For example, an AMS may inform its associated DOC that information regarding its resources should not be exposed to other AMSs unless some strict authorisation requirements are met.
DOCs can also control AMSs that act directly on each virtual networked element. One example for this organization is the handover process for the application continuity. In this case, the handover decision can be controlled by the AMSs, while the definition of the parameters for the handover decision, i.e. the maximum number of clients to be accepted on each virtual network, the authentication process being employed, is defined by the high-level goals dictated by the DOC.

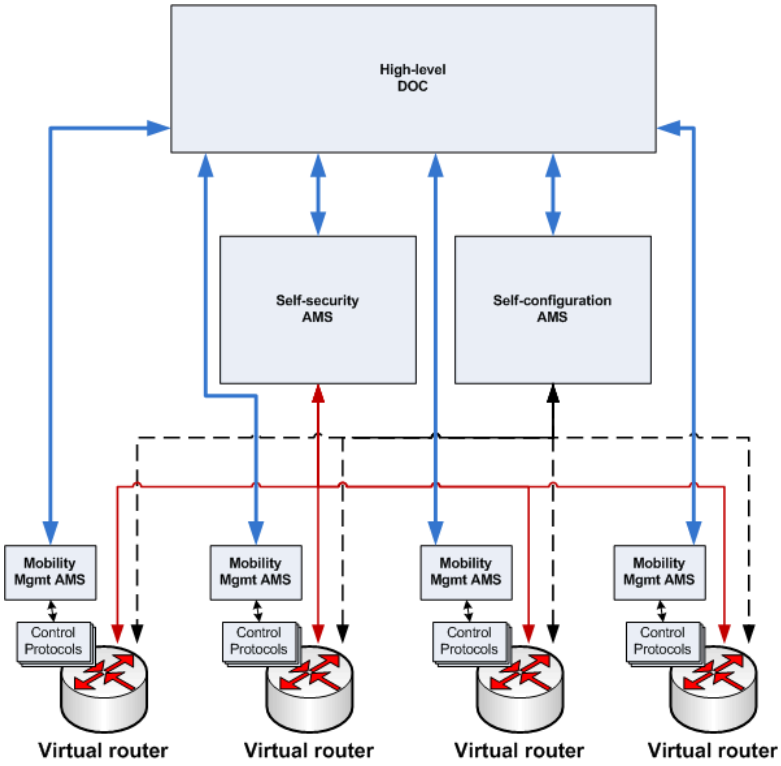


Figure 1 Deployment of the DOCs in the AutoI architecture.

This organization is shown in the Figure above. The high-level DOC controls the lower-level AMSs and the self-FCAPS AMSs (blue arrows), setting the policies and SLAs for those components. The AMSs act on the virtual resources (the red arrows and the dashed arrows) using the vCPI and vSPI interfaces, deploying services and executing FCAPS functions of the virtual resources and nodes. Finally, the lower-level AMSs control the near real-time decisions required for the operation of control protocols.

As defined in the deliverable D4.1, the AutoI architecture has several levels of policies, mapped into the policy continuum [8]. The orchestration plane uses Orchestration Policies, which are high-level policies that control the deployment and federation of autonomic control loops. In a sense, Orchestration High-level Policies are policies that act upon the inter-domain aspects of the autonomic management domains, controlling when and how two AMSs may federate. They may also dictate how the process to resolve conflicts will be carried out by the DOCs, as well as the deployment and distribution of the AMSs in the network. Meanwhile, the AMS High-level Policies will focus on the management of a single domain.

The DOC receives the Orchestration High-level Policies as well as the AMS High-level Policies from their corresponding administrative parties. Those policies are then processed and relevant information from this processing is stored in the System View, and are accessed to the required components as requested. For example, the policies required for the federation of two AMSs will be requested from the System View by a Federation Behaviour, while the policies related to security would be requested by the self-security autonomic AMS Behaviour.

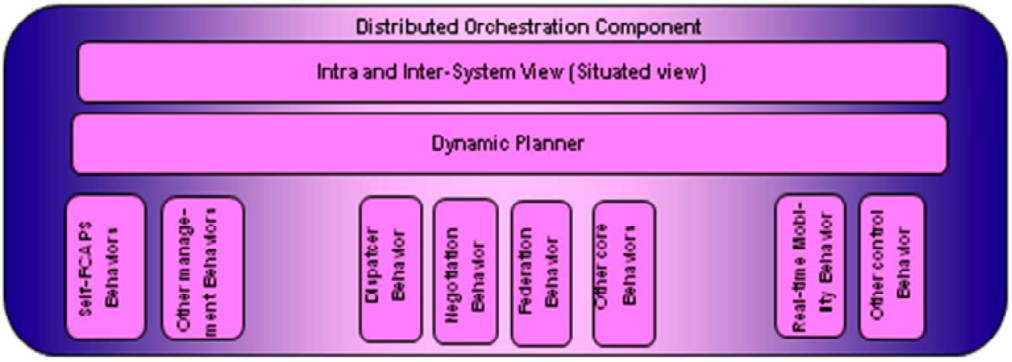


Figure 2 Design of the Distribution Orchestration Component

The design of the DOCs is outlined in Figure 2. It enables all components in the system managed by the Orchestration Plane to have plug-and-play behaviour. DOCs comprise three types of functional elements (FE):

Dynamic Planner FE. Acts as a workflow engine for execution of the Behaviours in a DOC. This process is necessary to decide what behaviours have to be achieved. The action of the Dynamic Planner will be dictated by policies.
Behaviours. Perform the specific/individual orchestration actions required to be performed by a DOC and also represent specific management tasks on the network. The main behaviours are Distribution, Federation, and Negotiation. Those Behaviours are described in detail in the following sections. Behaviours act as stubs for the AMSs. They also perform internal functions specific to the DOC (core Behaviours), i.e. negotiation among Behaviours.
Situated View FE. Is the "local window" of the DOC into the Knowledge Plane. This view is realized by the knowledge plane and has two parts. The **Intra-System View** provides an overall, composite view of the system as seen by the components within a single DOC. The **Inter-System View** provides an overall, composite view of the DOCs as a whole, that is, it provides a view of the entire Orchestration Plane.
The respective roles of these components are outlined in more details below.

3.1 Dynamic Planner

The Dynamic Planner acts as a workflow engine for the execution of the Behaviours. Such control includes the following tasks:

- Define the sequence and conditions on which the Behaviours must be bootstrapped for activating an AMS (a deployment of the AMS components defining their dependencies on other Behaviours; how and where to bootstrap them). It acts as control workflow for all AMSs ensuring bootstrapping, initialisation, contextualisation, closing down of AMSs. It also controls the sequence and conditions in which one AMS invokes other AMS in order to realize some useful functions (i.e., an orchestration is the pattern of interactions between AMSs).
- Monitor Behaviours, checking that the operation of an individual Behaviour does not conflict with others.
- Forward the advertised high-level policies for each orchestrated AMS and core Behaviour, which were provided by the operator or the owner of the network. Those policies are not processed by the DOC, since it is not its function to implement policy refinement.
- Identifiy conflicts in the initialization and reconfiguration of Behaviours. The Dynamic Planner will then start negotiation Behaviours to reconfigure the deployment plan.

- Trigger the initialization and closing down of Behaviours with a plug-and-play, unplug-and-play approach
- Trigger the dynamic reconfiguration of Behaviours.
- Facilitate the interaction between Behaviours. As an example of interaction, the negotiation Behaviour uses the Dynamic Planner to communicate with other Behaviours in order to solve configuration conflicts. Then, it is up to the Dynamic Planner to trigger a reconfiguration Behaviour.

The Dynamic Planner will be configured by policies, which will help the DP select the Behaviours (and hence which service and/or AMS) that must be bootstrapped, the service or AMS's parameters and SLAs.
The Dynamic Planner is similar to the Service Enablers Plane (SP) [D5.1] of the AutoI architecture since it provides the programmability and monitoring capabilities required for the autonomic operation of the network. However, while in the SP we deal with services, the DOC handles self-FCAPS management functions. Another difference among those is that AMSs are self-governing, while services are not. Thus, the DOC has to deal with negotiation aspects, which do not exist on the SP.
The autonomic control loop of the Dynamic Planner (see figure 3) is realized by the use of policies. Those policies define rules based on events and conditions that must be met by the virtual resources. Whenever those conditions are met, the Dynamic Planner executes the actions defined by the policy, which comprises the bootstrapping of one or more Behaviours. Those policies can be changed on the fly, allowing the DP to reconfigure itself and to adapt to changes in the SLAs of the network. Those policies are called in the AutoI architecture high-level orchestration policies. More specifically, the DP will use the distribution policies part of the orchestration policies to identify where to instantiate or migrate AMSs. Policies will also be used to decide which AMS or core Behaviour should be bootstrapped and when. The DOC will also use policies to enforce SLAs and goals into the AMSs.
The Dynamic Planner also acts as a mediator for the federation of the AMSs. Whenever the Dynamic Planner enforces a new configuration on the AMSs, those may refuse it due to their self-governance. In this situation, the AMSs will signal this rejection using the appropriate interfaces, and the DP will bootstrap a core Behaviour, responsible for conflict resolution, which will propose an alternative configuration.

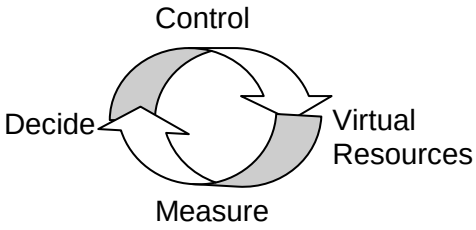


Figure 3. General autonomic control loop of the dynamic planner

3.2 Behaviours

Behaviours are sub-components of the DOC, which implement a certain orchestration task. As an example, the DOCs could implement Behaviours for the negotiation of high-level policies, the distribution of tasks, the creation and destruction of services and virtual routers. Behaviours interact with each other when necessary, i.e. the federation Behaviour may interact with a QoS Behaviour if the required QoS couldn't be met when two networks are joined. The lifecycle of a Behaviour is controlled by the Dynamic Planner, which

identifies, starts and stops the Behaviours necessary to accomplish a certain orchestration task.
We tentatively distinguish two types of Behaviours functional elements (FEs): the core Behaviour FEs - required for the proper operation of the Orchestration Plane and the AMS management Behaviours FEs , related to the control and management of the data plane. The core Behaviours support the operation of the dynamic planner, implementing the tasks necessary for the proper cooperation of AMSs and core Behaviours. Some examples of core Behaviours are listed below:

Distribution Behaviour FE: Sends and receives required data to different self-management orchestration Behaviour FEs.
Negotiation Behaviour FE: This FE implements algorithms to mediate between the AMSs in a DOC so that the AMSs can agree on common goals. After the negotiation is complete the AMSs should take autonomous and local actions under their corresponding administrative domain, converging to these negotiated goals. Negotiation can also occur between different DOCs.
Orchestration knowledge update Behaviour FE: Manages the dissemination of knowledge regarding the Orchestration Plane. It controls information regarding the core Behaviours required for the operation of the Orchestration Plane, as well as the information required by the Dynamic Planner.
Network federation Behaviour FE: This Behaviour controls the union/separation of virtual networks controlled by different AMSs.. This Behaviour identifies the steps necessary to compose/decompose different federated domains, proposing actions to the Dynamic Planner.
Knowledge update Behaviours FE: This class of Behaviours supervises the operation of the Knowledge Plane. They define the "What, When and Where" of the information: What information to collect, when to collect, and from whom (where). Those Behaviours are specific to each service, however the whole set of these Behaviours supervises storage of information in the Knowledge Plane. Each AMS requires pre-defined knowledge as well as runtime data. Mapping logic enables the data, represented in a standardized information model, to be transformed into knowledge and combined with knowledge represented by ontologies.
Bootstrap and Initialise Behaviour FE: This Behaviour is capable to bootstrap and initialise other Behaviours under supervision of the Dynamic Planner.
Reconfiguration Behaviour FE: This Behaviour is capable of dynamically reconfiguring and adapting other Behaviours under supervision of the Dynamic Planner.
Optimiser Behaviour FE: This Behaviour is capable of dynamically optimising and organising other Behaviours under supervision of the Dynamic Planner.
Closing down Behaviour FE: This Behaviour is capable of dynamically closing down other Behaviours under supervision of the Dynamic Planner.

Examples of Behaviour FEs, which have direct interworking with other management functions providing near real time reaction are listed below. Those Behaviours will act as a proxy to the AMSs, which will then implement each of those functions listed below. The associated Behaviours will deal with the interworking of those functions with others.

- Supervision of service life-cycle managers
- Supervision of Distribution/Federation/Negotiation of AMSs
- Supervision of interactions between AMSs
- Negotiation / Distribution of the high-level goals to different AMSs
- Monitoring of the AMSs
- Supervision of network consistency/integrity checks of the sequence of changes to networks made by separate AMSs

In the following sub-sections we provide a detailed description of the main core management Behaviours, which support federation, negotiation, governance and distribution of the management tasks. Next, we describe in more details the AMS Behaviours.

3.2.1 Federation Core Behaviour

Each AMS is responsible for its own set of virtual resources and services that it governs as a domain. Federation enables a set of domains to be combined into a larger domain, as well as breaking down a domain into smaller domains. With regards to the service enabler [SP], federation Behaviour is assumed to be important when conflicts between services are detected and services can't find a resolution themselves. Therefore, the Federation Behaviour is the only way to find a solution. We are considering two different networks A and B. The AMS governs each network as a domain, which can have different requirements, policies and constraints. Federation is set up into three different possibilities:
The first case is that AMS can federate the network by creating a domain coming from the union of both networks that are in conflict. The resulting domain inherits the characteristics of their union.
In the second case, we can assume that AMS can federate by creating a domain resulting from the intersection of both networks. In this case, the new network's requirements are the common ones from each network.
The third case, we might assume that AMS can't federate by the two previous possibilities. Moreover, in this case, the DOC creates a bridge, enabling to link the networks.

There are two aspects to federation in AutoI. The first deals with the agreements that must be shared among all participating members of the federations, and the second deals with facilitating the provision of services across federations. These two aspects are treated separately, but are tightly dependent on each other because if agreements cannot be made between participating members of the federation, then there can be no consensus on service provision. One of the major challenges facing the Internet of today is that dynamic agreement adaptation between independent administrative domains is difficult when business and technical concerns need to be addressed individually. In this case, the DOC will start up a special negotiation Behaviour to help in the negotiation of a new agreement, which will be driven by the decisions of the AMSs.
In AutoI, federation of AMSs is separated into two concerns, that are tightly coupled, namely business concerns and technical concerns. These are summarised hereafter:

Federation of High Level Objectives
A federation in AutoI is born when two or more AMSs need to come together under a common objective. Typically, this objective is to provide a common set of services with a guaranteed reliability across the boundaries of the AMSs. Each AMS will contain its own business objectives as defined by the policy continuum. They need to decide on a common set of business objectives that can be maintained across the federation. Once these set of common objectives are put in place, the business aspect of the federation is addressed. However, modifications to the federation business objectives can be proposed by any member, and any member can choose to leave the federation. Leaving a federation is part of the self-governance part of the AMSs and may entail a penalty if it breaches the terms of the federation. As each AMS can decide for itself to participate with other AMSs, the federation merely puts in place some common understanding between federation members about how they should interact with each other. One example could be to collude against a specific AMS that breaches the terms of the federation. Actually offering and consuming

services from within a federation is a technical concern and must be dealt with separately. This is to ensure that new services can always be introduced into a federation.

Technical concerns of a Federation
Once an AMS is participating in a federation of AMSs, it can consume and provide resources and services within the federation. The DOC in AutoI is concerned with the actual orchestration of the usage of these services and resources, and the joining and leaving operations of a federation. It is the challenge of the DOC to ensure that requested services of the AMS are made available in a way that abides by the terms of the SLAs and policies specified on the federations. To do this, the DOC must be able to assess whether the configurations of services are adequate enough to ensure the agreed terms of the federation are being upheld. The AMS must also abide by its own business objectives and if these objectives are no longer fulfilled it may decide to leave the federation. On leaving a federation, the associated DOC must signal this intention to the federation members.
Different types of services and resources required different technical solutions to ensure that they can be used effectively in a federation. It is up to the service creator to ensure that if the new service is to be used within a federation, that it be technically feasible to do so.

3.2.2 Distribution Core Behaviour

The DOC provides communication and control services that enable tasks to be split into parts that run concurrently on multiple AMSs within an Orchestration Plane, or even across multiple Orchestration Planes.

Business concerns of Distribution
The business concerns relate to whether each AMS requires the distribution of 1) information, 2) tasks, 3) code. Information distribution is important, as some AMSs may generate information that is unique to them and will need to be explicitly controlled. Access control policies may play a role here. Task distribution can be seen as the distribution of work that involves distributed processing across a number of processing elements. For example, the computation of the effective bandwidth of a particular film may be distributed across a number of machines, as this may be a time consuming process. The business concern here is that a particular AMS may request its associated DOC to distribute a processing task. The distribution of code is slightly different from the distribution of information and tasks in that there is typically no return expected from the distributing AMS. The distribution of code may be an enabler for an AMS to upgrade a particular service offering that relies on the federation of a number of AMSs.

Technical concerns of Distribution
The distribution of information may be carried out using the concepts being developed in the knowledge plane relating to the Context Information Service. However, the DOC needs to be involved as it can enforce strict access control over the information, and can thus instruct the distribution of information to abide by the high-level policies of the AMS. The high-level policies may authorise or prohibit information from leaving or entering its system.
The distribution of tasks can be technically carried out by the platform being used to distribute information. However, the task information may be in a specific format, where strict instructions may need to be in place to instruct the target AMSs of the requirements of the task. It is up to each AMS to decide whether to accept or reject the task being distributed. The AMS will need to inform the DOC of its intentions; the DOC then takes care of the distribution, and informs the AMS whether or not the intentions were meet.

The distribution of code can also be carried out by the platform being used to distribute information. Code in this case is a special form of information that can be distributed, in that it is mark to be executable. The code may be used to upgrade or deploy a new service within other AMSs. Therefore, the AMS instructs the DOC that code needs to be distributed and it is up to the DOC to handle the distribution of the code. This can be carried out using the Service Enabler Plane.

3.2.3 Negotiation Core Behaviour

In Future Internet Networks (FINs) approaches [1-6], it becomes mandatory for network and service providers to offer and publish their services so that more complex services can be provided. An important component that allows this requirement to be met in AUTOI is the negotiation component of the Orchestration Plane. This component acts as a virtual service broker that mediates between different AMSs, taking care of service requests and providing support so that the underlying service providers can negotiate SLAs, responsibilities, tasks, high-level goals, etc. In order to support such a functionality we must take into account the nature of the underlying service providers, the services they provide, their interests, their service qualities and other key aspects. All in all, this functionality should be fully automated with complex decision-making processes carried out by a dedicated negotiation inference engine.
In AUTOI each AMS advertises a set of capabilities (i.e., services and/or resources) that it offers for use in the Orchestration Plane. The negotiation functionality enables AMSs to reach agreements and form SLAs for selected, well-described services. DOCs mediate the negotiation process acting as trusted third parties or brokers for the AMSs since they have the advantage of a more holistic view of the network. Examples include using a particular capability from a range of capabilities (e.g., a particular encryption strength when multiple strengths are offered), being granted exclusive use of a particular service when multiple AMSs are competing for the sole use of that service, and agreeing on a particular protocol, resource, and/or service to use.
The negotiation functionality orchestrated between AMSs and DOCs has inherent business and technical concerns. The following elaborates on these two critical aspects.

Business and Technical Concerns for Negotiation
When the AMSs negotiate high-level goals under a DOC, or when different DOCs negotiate high-level goals with other DOCs, the negotiation finishes with participants aligning or not their internal business objectives. Agreement is reached when the participants converge to the negotiated high-level goals for which virtual and non-virtual resources must be allocated, managed and controlled autonomously in their respective domains (AMS and/or DOC). In the negotiation of business objectives, the responsibilities, benefits and penalties are also considered during the negotiation.
It is worth mentioning that an AMS and/or a DOC may negotiate business objectives with several AMSs and DOCs, sequentially or in parallel. The negotiation of business objectives is in turn influenced by technical concerns in the sense that AMSs compromise resources to fulfil the negotiated high-level goals. The governance capability of an AMS (and that of a DOC) defines with whom, why, and when to negotiate. The negotiation capability ensures that AMSs and DOCs can always negotiate with other entities in a federation. As AMSs and DOCs can have active negotiated agreements with several parties, there is a potential need to re-negotiate the high-level goals due to statistical changes in the resources committed to these goals when they cannot be fulfilled, or due to some internal decisions that lead to such corrective actions.
The DOCs may also trigger re-negotiation when the common business objectives are not fulfilled. Re-negotiation of high-level goals is a functionality that should be supported by the

Orchestration Plane. The renegotiation can be driven by utility functions, cost/benefit optimisations, etc. Again, during the re-negotiation of high-level goals, the responsibilities, benefits and penalties are also considered.
The Orchestration Plane provides the means for the mediation between AMSs and DOCs so that they can negotiate high-level goals as a result of a complex service request. As the AMSs and DOCs may belong to different administration domains, they may talk different languages, may express their high-level goals in different terms and so forth. Ontology translation and mapping techniques can be used to create the common language upon which participating entities can negotiate, federate, etc. The Orchestration Plane must provide the mechanisms for the negotiation to occur regardless of any of these technical aspects.
Another important technical concern of the OP negotiation functionality is that of convergence to optimal solutions. By definition, negotiation is needed when there is a complex service that demands the participation of more than one service provider. In multi-service provider environments, there may be several providers of the same service, with different qualities and in the middle of competitive environments. Under these circumstances of dynamic negotiation, a service request may involve several negotiation runs, and stability and convergence to optimal solutions are aspects that must be addressed by the DOCs.

3.2.4 Governance Core Behaviour

The governance functionality of an AMS deals with the self-interested actions that it takes to (re-) negotiate high-level goals with other parties and to take actions that can involve the commitment of virtual and non-virtual resources that may help provision of services across federations. Following on, each AMS can operate in an individual, distributed, or collaborative mode. In each case, it collects appropriate monitoring data in order to determine if the virtual and non-virtual resources and services that it governs need to be (re-) configured. Business objectives, service requirements, context, capabilities and constraints are all considered as part of the self-interested decision making process. DOCs also may be federated with other DOCs (Inter-systemsituated view) and as such they may act self-interestedly, namely they should have self-governance properties.

Business and Technical Concerns for Governance
In a federated environment, the DOCs provide support so that their underlying AMSs are aware of the needs of other AMSs within a federation. As an AMS is aware of its needs, it can decide which set of other AMSs it collaborates with. The nature and scope of the business objectives of an AMS would result in collaborations that can be different on the types of functionality and services that are the subject of the collaboration. The DOCs would provide support for these self-interested decisions to take place and to be acknowledged by the participants during the negotiation phase within a federation.
As introduced earlier, each AMS can operate in an individual, distributed, or collaborative mode:
The AMSs act as individual entities when they are not part of any federation, when they work isolated and autonomously, governing its own virtual and non-virtual resources and services driven by its own business objectives. No common goals are shared or responsibilities from any DOC are acquired. The DOCs should support this functionality by handling and transmitting the messages sent by the corresponding AMS to other AMSs and/or DOCs in the federation they are located in.
As a result of the distribution function of the Orchestration Plane, AMSs can work with other AMSs in a federation where complex services are provisioned by coordinating the activities, resources and services of each distributed AMSs after a negotiation phase. Relevant to the governance function is the fact that each AMS should provide a number of

contract interfaces to the DOCs, which use them to promote and mediate the negotiation of a complex service, its activation and maintenance. These interfaces could be interpreted as contract interfaces that isolate the internal structure and capabilities of the AMSs.
An AMS works in a collaborative mode when it acts as a local or a global collaborator. A local collaborator is responsible for coordinating the functionality of other AMSs in a given Orchestration Plane, this is when the DOC delegates part of its control to an AMS. An AMS works as a global collaborator when it coordinates the functionality of AMSs across different Orchestration Planes, namely amongst DOCs. This enables the emulation of client-server, n-tier, clustered, and peer-to-peer architectures.
In any case of operation, the OP should provide the means for an AMS to govern its virtual and non-virtual resources through a well-defined set of interfaces. The overall aim of the DOCs with this regard is to allow the AMSs to always decide on the action to take based on self-interested policies inside the AMS, driven by its business objectives and capabilities. Its decisions should also take into account the policies of other AMSs participating in the federation (see sections 3.2.3 and 3.2.5). However, the AMS can reject to abide to certain policies of the federation when those are not aligned with its own business objectives. In this case, it is up to the DOC to re-negotiate or change the policies of the federation.

3.2.5 AMS Behaviours

The DOCs use the AMS Behaviours as an interface to communicate with the AMSs. This encapsulation allows the DOC to see AMSs in a uniform way, having the same interfaces as the core Behaviours. All the communication of the AMSs with the DOCs follows the information models defined in WP3.
Further, AMS Behaviours are wrappers to the DOC, similar to stubs and skeletons in RPC (Remote Procedure Calls) or CORBA, once they provide a translation from the specific design issues of an AMS to the operational philosophy of the AMSs. One of the uses of such a wrapper is to hide from the DOCs the different implementations of the AMSs, for example ensuring a single communication point, even though the AMS may be a distributed component spread around several virtual nodes.
The wrapper also defines a set of commands that the AMSs must support to allow the DOC to orchestrate their federation and distribution. The interface provides mechanisms for the DOC to disseminate and renegotiate policies to the AMSs, allowing the AMS to be self-governing.
The AMSs can also support near real-time control of virtual resources and protocols. While the high-level self-FCAPS AMSs are mainly concerned with long-term management of resources, lower-level AMSs are deployed to react as fast as possible to changes in the AMS-aware context. Thus, lower-level AMSs use simple algorithms that act based on a pre-determined overall goal. Those goals are dictated by policies defined by the DOC. Those AMSs will act over a single virtual resource or a small set of virtual resources, due to the time constraints on their reaction. In a sense, lower-level AMSs may be seen as the first control loop of an autonomic system, acting based on pre-determined goals and with no sort of embedded learning. Examples of possible technologies that could be employed are state machines, PID controllers, fuzzy decisions, etc.
Low-level AMSs are associated with the maintenance of a QoS defined by the network SLA, controlling the parameters of the virtual nodes as well as their running algorithms, for example mobility management algorithms, the queuing discipline of QoS-aware MAC protocols or the admission and authentication of applications.

3.3 Intra- and Inter- system views

DOCs use the knowledge plane described in deliverable D4.1, section 3.1.8 to store and disseminate the information required for their operation. The information can be decoupled into two parts, or views, according to their relevance to a given DOC.
The Intra-System View concerns information required to orchestrate the services within the orchestration domain, while the Inter-System View deals with the orchestration of several orchestration domains. The Intra-System View contains information that enables DOCs to become aware of the particular situation that they are now in; the Inter-System View provides similar information for collaborating DOCs.
The Intra-System View thus deals with the services being run within the boundaries of a DOC, together with the information relevant for the operation of this DOC. This view is also called the Situated View. There is an important trade-off on the definition of the Situated View. Larger Situated Views will allow decisions to be taken using more knowledge describing the overall sate of the system. However, the cost of disseminating the knowledge among all the concerned nodes will be higher, as well as the processing cost. As a consequence, an optimal Situated View could be defined based on the problem being solved and on clear performance metrics.
The situated view will be detailed in the section 3.3.5 of Deliverable D4.1, once it is a part of the Knowledge Plane. It will also employ the ontologies and information models of D3.1 in the representation of the knowledge. The DOCs maintain the situated view, defining what information must be stored in this view, from which virtual nodes or resources, and with what frequency it must be updated. The interface of the Orchestration plane with the Knowledge Plane is described in 3.4.1.

3.4 Interfaces of the DOC

This section describes the interactions of the DOC with other elements of the AutoI architecture, as shown in Figure 3. We first describe the interface of the DOCs with the Knowledge Plane. The interfaces with virtual resources, the AMSs and the service enablers plane are just introduced here, since they will be defined in the deliverables respective to each of those functions.

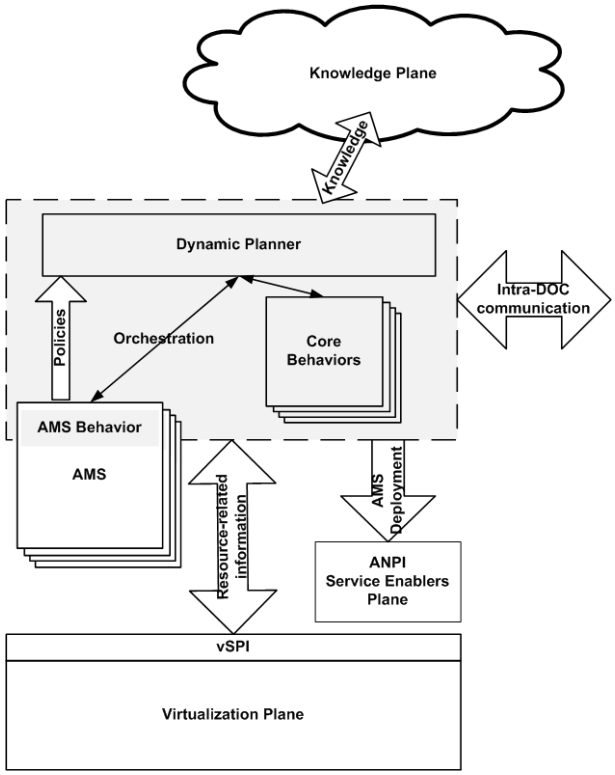


Figure 3 Interfaces of the DOC.

3.4.1 The DOC/Knowledge Plane Interface

The knowledge base consists of a warehouse in which all information and knowledge required for management and control tasks is kept. As each element within the autonomic architecture should be able to work autonomously, the Orchestration Plane should also be able to provide autonomously the information required for its functionality. The orchestration plane provides the DOCs with the required parameters regarding to which information should be monitored, how often it should take place and from where in the network it should be gathered. These parameters are decided by Dynamic Planner, based on the functionality of Orchestration plane and the state of the network, and raise a specific behaviour dedicated to this task. So, we can consider two types of knowledge within the knowledge base, those related to Management tasks and those related to Orchestration tasks.
One of the objectives of the OP is to identify what is the needed information that is required for autonomic decisions, and next to activate the required KP functions that will ensure the timely collection and delivery of this information to the DOCs or to a specific Behaviour. The format of the information requests, as well as the information that is produced by the OP follows the information models being produced in the Work Package 3 of the project. The following functions are required for the KP/OP interaction:

Lookup(Info): Requests the lookup of a certain piece of information (or knowledge) to the knowledge plane. This is used for fetching policies, SLAs as well as context and relevant configuration parameters of the AMSs being orchestrated.
Store(Info): Stores a certain information on the knowledge plane. This function is used for knowledge produced within the OP, which is then stored in the KP.

Subscribe(Event, Component): Defined by a condition on the stored information of the KP, this function allows the DOC to be notified of relevant events happening in the network. The events may define a condition and also a set of nodes or AMSs where such an event may happen. For example, the DOC may be interested in watching for the occurrence of certain faults to trigger the reconfiguration of an AMS responsible for fault management. The subscription also identifies which DOC and which of its components will process the information, that is, it can be the Dynamic Planner that requests the information, or it may be necessary for the operation of a certain Behaviour.
Watch(Info, SourceComponent, Periodicity, TargetComponent) and unWatch(Info): Used for the DOCs to define which are the information that must be periodically collected and disseminated by the KP. For example, the averaged used bandwidth of a network interface associated to a certain AMS may be monitored to verify if this component should be migrated to another node on the network. The information fetch is defined by the type of the information, its situated view (from which virtual nodes it must be collected, represented by the VNodes parameter), the periodicity of such an update and the component requiring this information. Once the DOC issued a Watch request, the KP is responsible for the maintenance of the information, delivering it directly to the concerned component. One example of use of Watch would be a DOC that defines that the number of active flows on the network should be communicated every 30 seconds to the autonomic performance optimization AMS. When this information is not needed anymore, the DOC will use the unWatch() function to free the KP resources.
Push(Information, DOCs): Used for synchronous message passing among the DOCs.

The Subscribe, Watch and Push interfaces of the DOC are used by the DOCs to create their inter-system view, used to feed the Dynamic Planner and the core Behaviours with their correct parameters. Those functions are also used for inter-DOC communication, once a DOC may watch the state of another DOC using the same functions. The access to information from other DOCs should be controlled by access policies, once DOCs may be controlled by different organizations, and thus it may be in the interest of the DOC to hide some aspects of the management of the network, e.g. due to competitive issues or to limit the knowledge of the topology and operation of the network to ward security attacks.

3.4.2 Interface SP/DOC, Virtual Resources/DOC and AMS/DOC

As described before, the ultimate task of the orchestration plane is to deploy AMSs within the network. The deployment of AMSs should be done considering the management requirements of services. Consequently, the orchestration plane performs this task of AMSs deployment, using the Service Enabler Plane (SP). However, the OP uses the SP to deploy AMSs, it does not deal with the deployment of services. The SP interface is defined in the deliverable D5.1.
In one hand, the orchestration plane needs to interact with virtual resources in order to provide the OP with physical management and control information regarding the state of the network and resources, which will be used for deploying AMSs Behaviours. This interface is useful since it provides the necessary information helping to deploy AMSs in the adequate places regarding the state of the network and resources. The interaction of the DOC with the virtualization plane will be described in the deliverable D1.1.
The last interface used by the DOC, which has been described before, is the interface of the DOC with the AMSs. This interface is indeed implemented by the AMS wrapper Behaviours, as described in Section 3.2.5 of this deliverable.

4 Conclusions and Summary

The AutoI project is proposes a new system or plane, the Orchestration Plane (OP), which enables the cooperation of the various autonomic control loops in the network, ensuring that the operates within the boundaries set by the business goals defined by the operators. This document presented the initial design of the orchestration plane in the AutoI project, defining its functions, requirements and the concepts behind the operation of the components that make the orchestration plane, the DOCs.

A DOC, or Distributed Orchestration Component, is a functional entity of the AutoI architecture that deals with inter-domain management tasks, such as the federation, negotiation, governance, and distribution of management domains. The DOCs have two main building blocks, the Dynamic Planner, which serves as an autonomic policy-based dispatcher. The Dynamic Planner creates and destroys Behaviours, which implement the interfaces for the AMSs and the core inter-management functions. The second building block are the Behaviours, which have two functions. The first one, carried out by the Core Behaviours, is to implement orchestration tasks. The second, implemented by the AMS Behaviours, is to act as a proxy for the communication of the DOC with the AMSs that it orchestrates.

In the second year of the project we will specify the components and interfaces required for the OP to handle federation, distribution, negotiation and governance. The main function of the OP is to orchestrate the interaction of the AMSs, the complete definition of the DOC requires the instantiation of the AMSs. At this stage of the project, the AMSs are not completely specified (D4.1), this deliverable only presented an initial description of the orchestration components. A detailed architecture will be produced in future deliverables.

An open issue that will be addressed is the definition of the scope or nature of the orchestration policies. This task is linked with the work done in the definition of the AMSs and its functions, as well as the types of policies used in the AMS. The amount of complexity and abstractions necessary for orchestration policies will depend on the degree of intelligence of the AMS, as well as how much the AMSs are willing to accept the guidance of the DOCs. While the AMSs represent different administrative domains, they might completely reject the policies and configuration parameters defined by the OP. The occurrence or not of such a situation in the AutoI architecture will define the level of intelligence of the DOC, and thus the complexity of the policies.

In order to allow the DOCs to identify and cope with conflicts between AMSs, we will derive means to describe the Behaviours, with their preconditions, provided services and service constraints. This could be done with OWL-S or SAWSDL, for example. Such a description is necessary for the negotiation and proper identification of which Behaviour to be bootstrapped, the federation of AMSs and the enforcement of SLAs, those characteristics indicate what tasks an AMS can and cannot perform, and what are its requirements.

Working in conjunction with WP3, we will design the concepts (classes and relationships) supporting the orchestration part of the AutoI information model (AIM). Those classes will be used in the inter-DOC communication as well as in the negotiation of parameters and SLAs between DOCs and AMSs.

5 References

[1] Clark, D. Partridge, C. Ramming, J. Wroclawski, J.T. (2003), A Knowledge Plane for the Internet, ACM SIGCOMM'03.
[2] Strassner, J. Foghlú, M.Ó. Donnelly, W. Agoulmine, N. (2007), Beyond the Knowledge Plane: An Inference Plane to Support the Next Generation Internet, GIIS200.
[3] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, H. Zhang (2005), A clean slate 4D approach to network control and management, SIGCOMM Computer Communications Review, volume 35 no. 5.
[4] The ANA Project: Autonomic Networking Architecture, http://www.ana-project.org/
[5] The CASCADAS Project: Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services, http://www.cascadas-project.org/
[6] The Haggle Project, http://www.haggleproject.org/
[7] The BIONETS Project: Bio-Inspired Service Evolution for the Pervasive Age, http://www.bionets.eu/
[8] S. Davy and B. Jennings and J. Strassner, The policy continuum-Policy authoring and conflict analysis, Elsevier Computer Communications, volume 31, number 13, 2008, ISSN 0140-3664, pages 2981-2995, Butterworth-Heinemann, Newton, MA, USA.
[9] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era, IBM Systems Journal, volume 42, number 1, pages 5-18, 2003.
[10] The Ambient Networks Project, http://www.ambient-networks.org/.
[11] Andreas Fischer, Andreas Berl (editors), Autonomic Internet Project, Deliverable D1.1 – Initial Virtual Plane Design, 2009.
[12] Zohra Boudjemil (editor), Autonomic Internet Project, Deliverable D3.1 – AutoI Information Model, 2009.
[13] Alex Galis, Lefteris Mamatas (editors), Autonomic Internet Project, Deliverable D4.1 – Initial Management Plane Design, 2009.
[14] Abderhaman Cheniour, Laurent Lefevre (editors), Autonomic Internet Project, Deliverable D5.1 – Initial Service Enablers Plane Design, 2009
[15] Alex Galis (editor), Autonomic Internet Project, Deliverable D6.1 – Initial AutoI Framework, 2008.