Pattern-Based Generation of AMF Configurations

Parsa Pourali

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master in Computer Science at

Concordia University

Montreal, Quebec, Canada

August 2014

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: Parsa Pourali

Entitled: Patter-Based Generation of AMF Configurations

and submitted in partial fulfillment of the requirements for the degree of

Master in Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. E. Shihab	Chair
Dr. G. Butler	Examiner
Dr. J. Rilling	Examiner
Dr. F. Khendek	Supervisor
Dr. M. Toeroe	Co-Supervisor

Approved by _____
Chair of Department or Graduate Program Director

August 2014 _____
Dr. A. Asif, Dean
Faculty of Engineering and Computer Science

ABSTRACT

Pattern-Based Generation of AMF Configurations

Parsa Pourali

Information Technology service providers aim at attracting customers by providing services that meet a high level of quality. They should not only satisfy functional requirements, but also non-functional requirements. An important non-functional requirement is the level of service availability. Service Availability Forum (SAForum), a consortium of communications and computing companies, has developed a set of services and standard Application Programming Interfaces (APIs) to address the issue of high availability for the Commercial-off-the-shelf (COTS)-based systems and enable portability. Among the services standardized by the SAForum, the Availability Management Framework (AMF) has the important role of ensuring the high availability of an application and its services, by managing the redundant application components deployed on the cluster. To achieve this task, AMF requires a configuration that represents the logical organization of the application components and their services.

The design of AMF configurations is a complex and error prone task. Automation of the process is the first step towards improving the quality of such configurations. It also enables exploring different potential solutions for a given set of requirements. An automated approach to generate configurations for applications to deploy on top of the SAForum middleware has been proposed in the context of the MAGIC project. This approach, however, may generate several configurations among which some may not meet the required level of service availability. Therefore, the system designer needs to evaluate the generated configurations using an availability

analysis tool to select an appropriate one for deployment. One may want to improve this process by targeting directly in the generation process, the configurations that can guarantee the requested level of service availability.

The objective of this thesis is to propose solutions to enhance this configuration generation process and generate configurations that can guarantee the required level of service availability without using advanced analysis tools. For this purpose, we propose configuration design patterns to improve the expected level of service availability and quantitative methods that eliminate some configurations that do not meet the availability requirement. The configuration design patterns improve the expected level of service availability by selecting the best configuration options. The methods estimate service availability for the different possible combinations of software components, which can provide the requested services, taking into account the properties of these components and the behavior of the SAForum middleware. As a proof of concept, we have embedded our proposed solutions into a prototype tool as an eclipse plug-in and validated our work with case studies.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Ferhat Khendek, for the patient guidance, encouragement and advices he has provided me during my study. Thanks for believing in me.

I have been very grateful to have Dr. Maria Toeroe as my research co-supervisor. Without her support, expertise and active participation in every step of the research, this thesis may never have been completed. Thank you very much for your encouragement and understanding over these past two years.

I am also thankful to my friends in the MAGIC group for all the great times that we had together. Thanks to them for sharing literature and their assistance.

This thesis is partially supported by the Natural Sciences and Research Council of Canada (NSERC), Ericsson Inc. and Concordia University. Many thanks to the faculty of Engineering and Computer Science of Concordia University that provided me with opportunities and facilities to achieve this work. My gratitude also goes to Ericsson Canada, the place that I could realize my ideas and develop my expertise.

Finally yet importantly, I give my special and deepest thanks to my family for their love, support, and encouragement. Without them, this thesis would never have been written. I dedicate this thesis to them.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ActiveCapabilityPerCST	Component capability to handle active assignments (per CST)
AIS	Application Interface Specification
AMF	Availability Management Framework
AOT	Average Outage Time of a component instantiation phase
AOTIWD	Average Outage Time for Instantiation attempts With Delay
AOTIWOD	Average Outage Time for Instantiation attempts Without Delay
API	Application Programming Interface
AppType	Application Prototype
CLC	Component Life Cycle
CLI	Command Line Interface
ClT	Clean-up Timeout for a component
CompType	Component Prototype
CR	Configuration Requirements
CSI	Component Service Instance
CSS	Callback time to set the component to the CSI active, standby or quiesced states
CST	Component Service Type
CSTO	The Cluster Startup Timeout
CSType	Component Service Prototype
CT	Component Type
dly	Delay time between delayed instantiation attempts

DSPN	Deterministic and Stochastic Petri Net
ETF	Entity Types File
GUI	Graphical User Interface
HA	High Availability
HPI	Hardware Platform Interface
IT	Instantiation Timeout for a component
NCF	Node CSIs Failover Timeout (The timeout to fail over of the CSIs assigned to the components of the active node, to the components of the standby node)
nia	Total number of instantiation attempts
NIWD	Number of Instantiation attempts With Delay
NIWOD	Number of Instantiation attempts Without Delay
NoOfActiveAssignments	Number of active assignments per each SI
NoOfCompsPerCST	Number of components of a CT that can provide a particular CST
NoOfCSIsPerCST	Number of CSIs of a CST in each SI of the SI template
NoOfNodes	Number of nodes in the cluster
NoOfSIs	Number of SIs of the SI Template
NoOfStandbyAssignments	Number of standby assignments per each SI
NST	Node Shutdown Time (The time to that a node takes to go down)
OIF	Outage time in case of all the Instantiation attempts Fail
PCNS	Probability of component Clean-up Not Successful (i.e. 1-PCS)
PCS	Probability of component Clean-up Successful
PINS	Probability of Instantiation Not Successful (Without Delay)
PINSD	Probability of a delayed Instantiation Not Successful

PIS	Probability of Instantiation Successful (Without Delay)
PISD	Probability of a delayed Instantiation Successful
SA	Service Availability
SAForum	Service Availability Forum
SG	Service Group
SGType	Service Group Prototype
SI	Service Instance
SOT	Switch Over Time of a non-faulty component(s)
StandbyCapabilityPerCST	Component capability to handle standby assignments (per CST)
SU	Service Unit
SUMaxCapPerCT	Maximum number of components of a given CT that can be put in the SU
SUType	Service Unit Prototype
XML	Extensible Markup Language

1 Introduction

This chapter introduces the context, the motivations and the contributions of this thesis.

1.1 Research Domain

Service Availability (SA) is an important requirement in today's technological world. It is required in many domains, such as mission critical and telecom systems. Any service outage in these systems can result in catastrophic damages or financial and reputation loss [1]. For instance, IBM Global Services has reported [2] that in 1996, service outage has cost around $4.54 billion loss of productivity and revenues for American businesses.

Service Availability is defined as the percentage of time the service is provided even in the presence of inevitable failures in the system [3]. High Availability (HA) is defined as providing a minimum of 99.999% service availability, which means having at most 5.26 minutes downtime in a year [3]. HA can be achieved by improving the reliability of the systems; however, no system is immune to failure [1]. For this reason, the goal of HA solutions is to minimize the time needed to recover services in case of a failure in the system [2]. This is typically achieved by using redundant resources (e.g. components) in the system and managing these redundant resources through repair and recovery mechanisms [1].

The Service Availability Forum (SAForum), a consortium of communications and computing companies, has developed a set of standard Application Programming Interfaces (APIs) to enable the development of Commercial-off-the-shelf (COTS) components-based highly available systems [4]. SAForum has introduced the Application Interface Specification (AIS) that

consists of several middleware services (See Figure 1-1) including the Availability Management Framework (AMF), which is the context of our work.

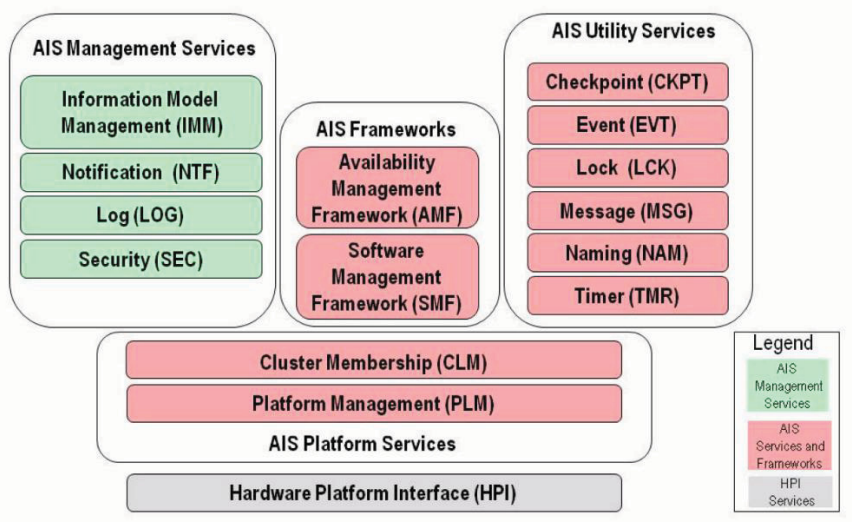

Figure 1-1 The SAForum middleware specifications and services [5]

Among the set of middleware services standardized by the SAForum, AMF has the important role of ensuring the availability of the services provided by a component-based system in a cluster. AMF achieves this task by re-assigning the workload of a faulty component to healthy ones. For this, AMF requires a configuration (AMF configuration) that describes the logical organization of application components and their services [6][7][1]. A system designer is generally responsible for designing the configuration and deploying it.

1.2 Thesis motivation

As mentioned, in today's world the general expectation is that online services are available 24 hours a day and seven days a week. Accordingly, the different stakeholders such as end-users and service providers specify a minimum level of service availability as part of their requirements [8]. System designers face the challenge of configuring the system and meet the required service availability. Since there are too many configuration options and many constraints to consider,

coming up with a configuration is a difficult and error prone task. This is true also in the context of the SAForum [4] middleware.

To alleviate the work of the system designer, an automated approach for AMF configuration generation has been proposed in [1]. The approach takes as input the requirements and available software catalogues known as Entity Types File (ETF)[1] [9][10]. It explores the ETF to select the software component prototypes that can satisfy the requirements. Then, it creates corresponding AMF types and entities based on the selected ETF prototypes. Furthermore, the approach generates multiple AMF configurations by exploring all the available decision points and options. The limitation of this approach is that, it does not take into account the availability requirement. Hence, the system designer has to evaluate the availability of generated configurations using analysis tools, and select the one that meets the availability requirement. One may want to improve this process by targeting directly in the generation process the configurations that can ensure the required level of service availability. This can be achieved by integrating configuration design patterns and methods into the generation process to filter out the configurations that will not provide the requested service availability.

1.3 Thesis contributions

In this thesis, we address the aforementioned issues by enhancing the generation process introduced in [1] in order to generate the configurations that can ensure the required level of service availability. We propose four configuration design patterns. In this thesis, a configuration design pattern is a general reusable solution that can improve the level of service availability in a given context. We also propose two quantitative methods to have an early estimation of the service

[1] ETF prototypes describe the capabilities and limitations of the software from the vendor's point of view [10].

availability that can be delivered by the configuration. The main contributions of this thesis can be summarized as follows:

- Four configuration design patterns that can improve the expected level of service availability:
    - ✓ ETF prototype adjustment configuration design pattern, which improves the level of service availability by minimizing the impact zone of a recovery action when a component fails. In other words, it ensures that in case of a component failure, the recovery action does not affect other components in the system.
    - ✓ Separation of Component Service Types configuration design pattern to customize the software components given by a vendor, so that if a component fails, not all of the services or functionalities of the system become unavailable.
    - ✓ Redundancy model selection configuration design pattern that selects the appropriate redundancy model based on the preferences for more general redundancy models over more specific ones.
    - ✓ Load-balancing entity distribution configuration design pattern to ensure the even distribution of entities (e.g. software components) over the cluster nodes.
- Two methods to estimate the achievable service availability and generate the configurations that can guarantee the required service availability:
    - ✓ Availability estimate-based prototype selection method for early elimination of the configuration options that cannot satisfy the required level of service availability.

- ✓ Availability estimate-based entities creation method to calculate the number of entities (e.g. application components) to be created and deployed on the cluster, with respect to the availability requirement.

- A proof of concept prototype tool of the proposed solutions and their validation.

1.4 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we review the background knowledge related to availability, SAForum middleware and related work, followed in Chapter 3 by a description of the configuration generation process [1] and its steps. We describe the main contributions of this thesis, namely configuration design patterns and availability estimate-based methods in the chapters 4 and 5 respectively. In Chapter 6, we present our prototype tool and the empirical validation of our work. Finally, we conclude this thesis in Chapter 7.

2 Background on Availability, SAForum Middleware and Related Work

In this chapter, we start by introducing the general definition for service availability before presenting the background information on SAForum middleware required later in this thesis. We also review the related work.

2.1 Service Availability

As mentioned previously, SA is defined as the probability of a service to be provided by a system [3] when requested. The two factors determining service availability of a system, are Mean Time to Repair (MTTR) and Mean Time to Failure (MTTF) [3]. The MTTF is the average time between two consecutive system failures, whereas the MTTR refers to the average time to repair the system, so that it can provide the service again. The service availability of a system can be defined as shown in Eq. (2-1) [3]:

$$Service\ Availability = \frac{MTTF}{MTTF + MTTR} \tag{2-1}$$

The MTTF and MTTR of the system depend on the MTTF and MTTR of its composing components.

2.2 SAForum Middleware

As briefly mentioned in Chapter 1, SAForum defined several interfaces and services to achieve and manage service high availability. These services are grouped into two categories, which are AIS and Hardware Platform Interface (HPI) services (See Figure 2-1) [5]. While, the AIS services can be used to manage the high availability of an application, the HPI services provide

the capability to monitor and control the hardware resources [5]. In the rest of this chapter, we will focus only on one of the AIS services, namely the AMF, as it is the main context for our work.

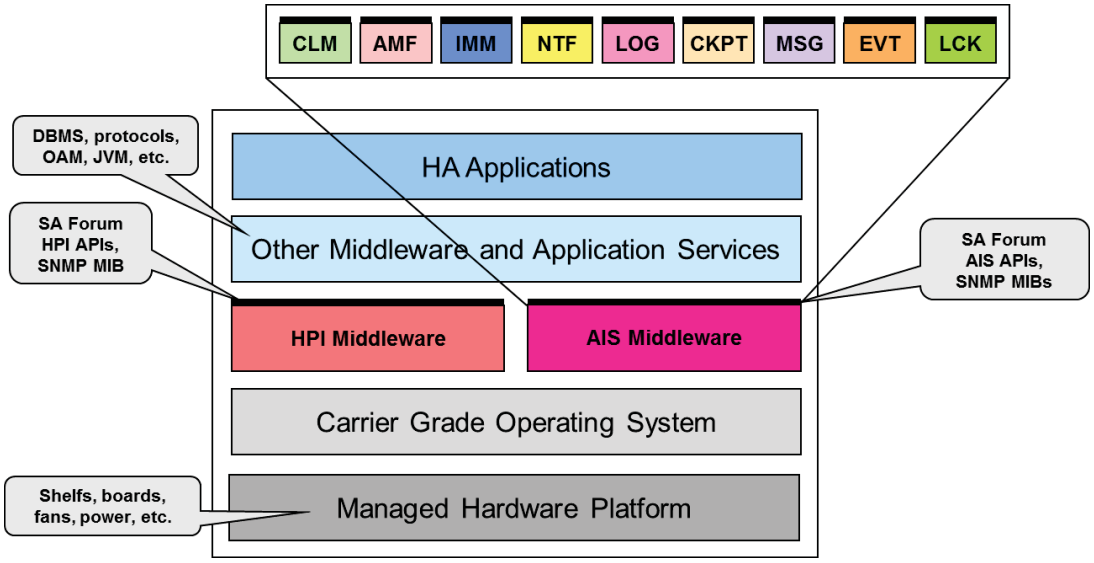

Figure 2-1 The SAForum middleware [5]

2.2.1 Availability Management Framework

Among the services that the SAForum middleware provides, AMF is the most important one as it manages the availability of application services. It is basically responsible for 1) assigning the workload to the application components, 2) managing the life-cycle of the resources under its control (e.g. software components), 3) reassigning the workload of a faulty component to a standby (and healthy) component, and 4) repairing the faulty component [6]. To achieve these tasks, AMF requires a configuration that is a logical organization of the entities (i.e. application's services and components). The AMF configuration model is defined in [1]. Each logical entity has a set of attributes that describe the properties of the entity [6]. Most of these logical entities are typed entities [6]. The AMF types represent a generalization of the entities [6]. We describe the AMF entities and types in the following subsection.

2.2.1.1 AMF Entities and Types

The AMF configuration contains logical entities, their attributes and relationships [6]. These logical entities are used by AMF to perform administrative and management operations [11]. Figure 2-2 shows these logical entities and their relations.

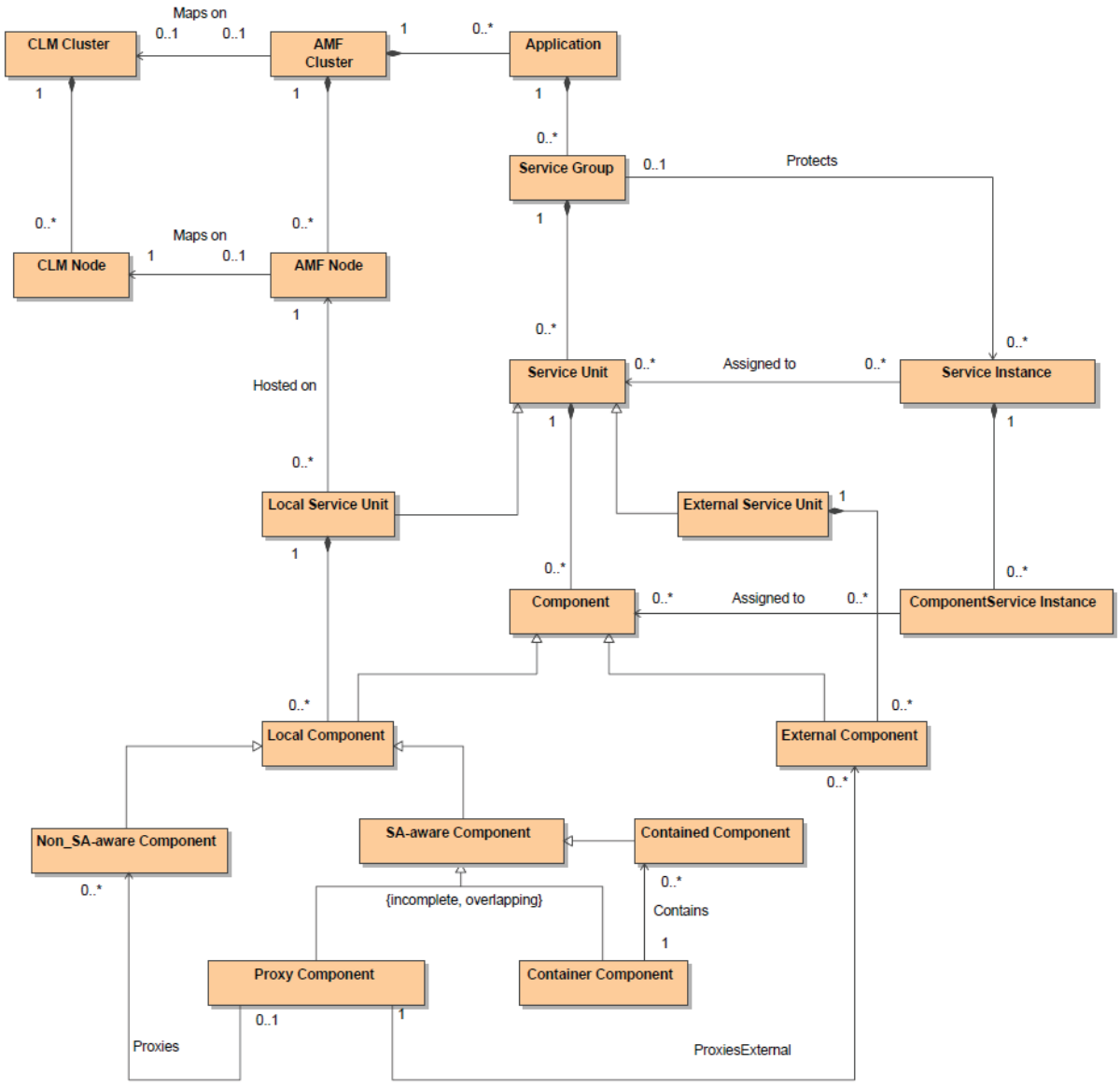

Figure 2-2 AMF logical entities and their relations [6]

Except for the node and the cluster, all other logical entities are typed entities. A type encapsulates the common characteristics that its instances share.

Component: A component is the smallest logical entity in the system on which AMF performs error detection, isolation and repair [6]. It represents a specific resource such as a process, which is capable of providing a set of functionalities [6].

- **Local and External Components:** Depending on the node that the component resides on, it can be local or external component. In other words, if the hosting node is being controlled by AMF, the component is a local component, otherwise it is an external component [6].

- **SA-aware components:** The local components that are under the direct control of the AMF are called SA-aware components [6].

    - **Contained and Container Components:** A contained component is a type of component that is run and controlled by another environment, referred to as container component [6], like a virtual machine for instance.

- **Non-SA-aware components:** The components that do not register directly with the AMF are called non-SA-aware components [6]. Typically, they are managed by an SA-aware component, known as proxy component, that mediates between the AMF and these components [6]. The components for which a proxy component mediates are called proxied components [6].

Component Type (CT): A CT is a particular version of software or hardware implementation. It represents the common characteristics that components of this CT share [6].

Component Service Instance (CSI): It represents the workload that AMF assigns to a component at runtime [6].

Component Service Type (CST): It represents the generalization of similar workloads (that is CSIs). AMF handles the CSIs of a CST in the same manner [6].

Service Unit (SU): An SU is an aggregation of several components that combine their individual functionalities to provide a particular service [6].

Service Unit Type: It defines the common characteristics that the SUs of this type share. It specifies a list of CTs that can be aggregated in the SU type. It also determines the number of components of each CT that an SU of this type can accommodate [6].

Service Instance (SI): An SI is an aggregation of basic workloads (CSIs). It is the workload that AMF assigns to an SU at runtime [6].

Service Type: It defines the list of CSTs that can be used to compose an SI of this service type [6].

Service Group (SG): SUs are grouped into SGs to protect SIs [6]. Any SU in the SG must be capable of accepting the assignment for any SI protected by the SG [6]. Each SG has a redundancy model that defines how the SUs in the SG should protect the SIs [6]. The redundancy models and their characteristics are described in the following.

Redundancy Models: There are five different redundancy models: 2N, N+M, NWay, NWay-Active, and No-Redundancy. Each of them has its own characteristics with respect to the number of active and standby assignments an SI may have, and the distribution of these assignments among the SUs within the SG. An SU may have the HA state of active or standby on behalf of an SI. Active means that the SU is the primary service provider for that SI, whereas standby means that the SU is acting as secondary (redundant) entity for that SI [6]. Additionally, if all the SIs' assignments to the SU are in the active state, the SU is called an Active SU, whereas

if all the SIs' assignments to the SU are in the standby state, the SU is called a Standby SU. If an SU has no assignment, it is a Spare SU.

In the following, we provide more details about the redundancy models.

- **2N Redundancy Model:** In an SG with 2N redundancy model, at most one SU can be assigned as active for all SIs (known as active SU), and at most one SU can be assigned as standby for all SIs (known as standby SU) [6]. In addition, it also allows for one or more spare SUs. According to this redundancy model, an SI can have only one active and one standby assignments [6]. Figure 2-3 shows an example of an SG with a 2N redundancy model. In this example, there are two SIs and each of them is composed of two CSIs. The SU1 and SU2 have the active and standby HA states for the SIs, respectively.

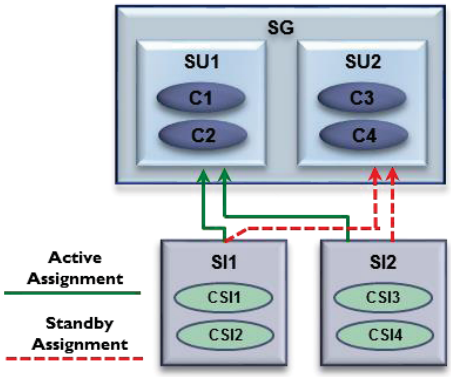

Figure 2-3 An example of an SG with a 2N redundancy model

- **N+M Redundancy Model:** In this redundancy model, N SUs can be assigned as active and M SUs can be assigned as standby on behalf of the SIs protected by the SG. Each SU of the SG, can only have one of the HA states (i.e. Active, Standby or Spare). An SI can have only one active and one standby assignments [6]. Figure 2-4 illustrates an example of an SG with an N+M redundancy model, with

four SUs. Two of the SUs are assigned as active and the other two SUs are assigned as standby to handle the SIs protected by the SG.

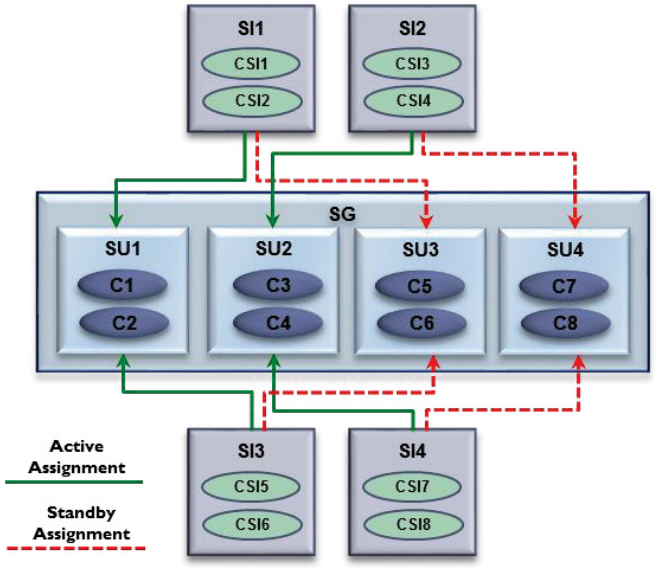

Figure 2-4 An example of an SG with an N+M redundancy model

- **NWay Redundancy Model:** In this redundancy model, there are N SUs that can be assigned simultaneously as active for a set of SIs and standby for other SIs. No SU can take the active and standby assignments on behalf of the same SI. Moreover, an SI can have only one active assignment, but it can have one or many standby assignments [6]. Figure 2-5 shows an example of an SG with an NWay redundancy model. In this example, there are three SUs to handle the three SIs protected by the SG. Every SI has one active and two standby assignments. As can be seen, each SU is assigned as active for one SI and standby for the other two SIs.

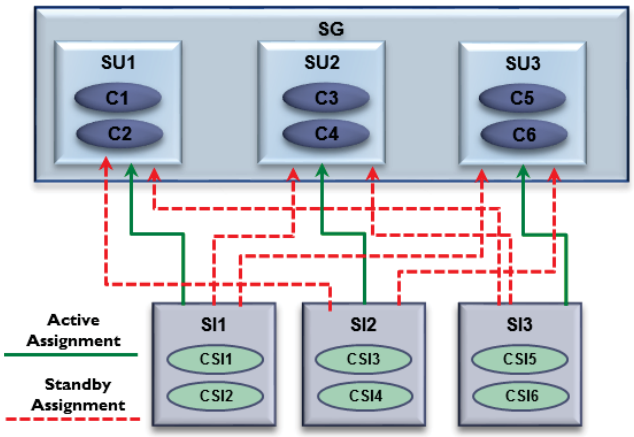

Figure 2-5 An example of an SG with an NWay redundancy model

- **NWay-Active Redundancy Model:** Unlike the 2N, N+M and NWay redundancy models, the NWay-Active redundancy model does not allow for standby assignments for an SI. However, it allows for an SI to be assigned as active to several SUs [6]. It means that, an SI can have one or many active assignments [6]. Figure 2-6 represents an example of an SG with an NWay-Active redundancy model. There are three SUs and three SIs. As shown in the figure, each SI has two active assignments that are assigned to different SUs.

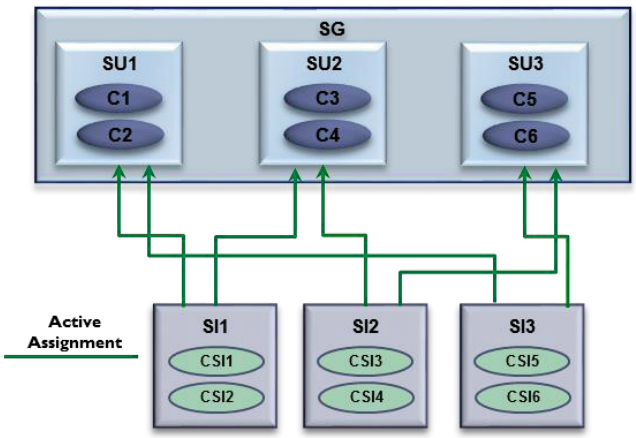

Figure 2-6 An example of an SG with an NWay-Active redundancy model

- **No-Redundancy Redundancy Model:** In this redundancy model, each SI can have at most one active assignment. Moreover, each SU can be assigned as active for at most one SI. Like other redundancy models, No-Redundancy redundancy model also allows for spare SUs in the SG [6]. An example of an SG with a No-Redundancy redundancy model with three active SUs (e.g. SU1, SU2 and SU3) and one spare SU (e.g. SU4) is shown in Figure 2-7.

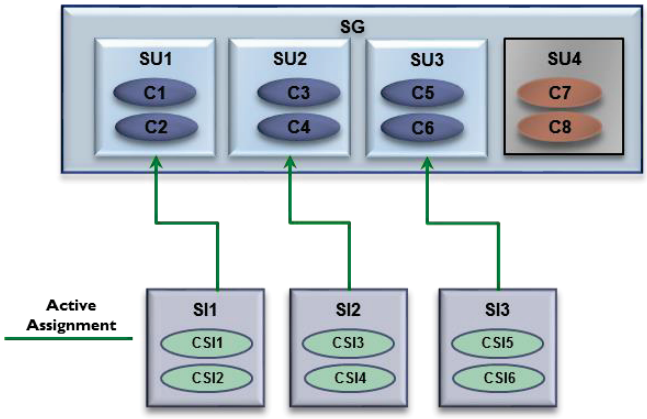

Figure 2-7 An example of an SG with a No-Redundancy redundancy model

Service Group Type: It defines the common characteristics that all the SGs of this type share. It also defines the list of SU types that can be supported by an SG of this type [6].

Application: A set of SGs form an AMF application. While an application can be composed of several SGs, an SG belongs to only one application [6].

Application Type: It defines the characteristics that all of the applications of this type share. It also defines the list of SG types that can be used to build an application of this type [6].

AMF Node: An AMF node is a logical entity in the cluster. AMF deploys components and SUs on the AMF nodes [6].

AMF Cluster: AMF nodes can be put together to form an AMF cluster [6].

2.2.2 An Example of AMF Configuration

An example of AMF configuration is given in Figure 2-8. In this example, there is one SG with a 2N redundancy model. The SG itself consists of two SUs and each SU is composed of two components. The application has two SIs protected by this single SG. According to the 2N redundancy model, at runtime AMF will assign both SIs to one of the SUs (e.g. SU1) in the active state and to the other (e.g. SU2) in the standby state. If a component in SU1 fails, depending on the applicable recovery procedure, the faulty component may be restarted or both active assignments will be failed over to SU2.

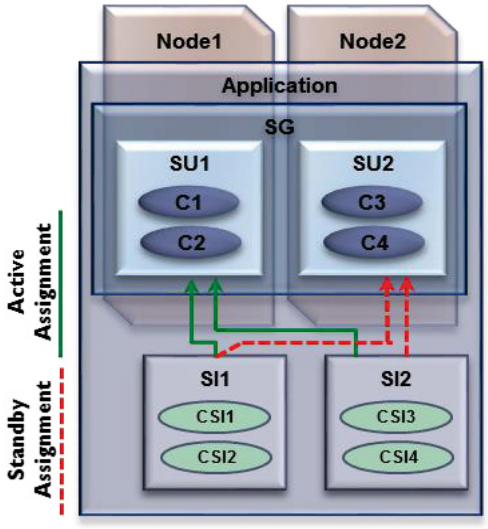

Figure 2-8 An example of AMF configuration

2.2.3 Entity Types File

The software vendor describes the software in terms of prototypes[1] in an ETF [9][6]. The AMF types are primarily derived from the prototypes that the software vendor provides to describe the features and limitations of their implementation. This description may include characteristics such as the component capability, the minimum and maximum number of components that can be

[1] Note that in this thesis we may use the terms ETF types and ETF prototypes interchangeably.

contained in an SU and the dependencies among the software components. In the rest of this subsection we describe the ETF prototypes and their characteristics [9].

Component Service Prototype (CSType): It specifies the attributes that characterize a particular workload that can be assigned to a component.

Service Prototype (ServiceType): A ServiceType specifies, if necessary, how CSTypes can be combined to build this ServiceType [9].

Component Prototype (CompType): A CompType describes the characteristics of a specific version of a software implementation that can be used to instantiate components. Most importantly the CompType defines the CSTypes such components can provide along with the dependencies among them [9]. The CompType also specifies some other characteristics. For example, it specifies whether a component of this type can be restarted or not in order to recover its services. The main characteristics that a CompType can specify are as follows:

- The CSTypes that a component of this CompType can provide (*providesCSType*) [9],
- For each CSType, the capability of a component of this CompType can be:
    - x active and y standby (*xactiveandystandby*),
    - x active or y standby (*xactiveorystandby*),
    - One active or x standby (*oneactiveorystandby*),
    - One active or one standby (*oneactiveoronestandby*),
    - x active (*xactive*),

- One active (*oneactive*).

Where x and y are the optional elements to specify the maximum number of active or standby CSIs that a component of this CompType can handle, respectively [9],

- The timeouts that AMF should wait for a component of this CompType to fulfill a life-cycle control or other callback commands such as component instantiation, cleanup and workload assignment (*defaultClcCliTimeOut* and *defaultCallbackTimeOut*) [9],
- The list of other CompTypes that this CompType relies on, to provide a particular CSType (*requiredCompType*) [9],
- The restartability attribute of a component of this type, which determines if the component can be restarted as a consequence of service recovery action (*disableRestart*) [9],
- The recovery action that the software vendor suggests to be performed by AMF in case of a component failure (*recoveryOnError*) [9].

Service Unit Prototype (SUType): An SUType limits the combination of CompTypes, and accordingly it lists the ServiceTypes that can be provided using instances of this SUType. Any limitation on the number of instances of a particular CompType within the SUType is defined as a range value. The SUType may also specify that in case of a component failure in the SU of this type, the entire SU should be failed over (*suFailOver*) [9].

Service Group Prototype (SGType): An SGType limits the SUTypes that can be used to build an SG of this SGType. It also specifies the redundancy model for its instances [9]. Using an

SGType, the vendor can recommend the attribute settings related to the repair and recovery escalation policies:

- The maximum number of times that the components of an SU can fail within the probation period, before the whole SU is restarted (*CompProbCountMax*) [9],
- The maximum number of times that an SU can be restarted in a probation period, before it is failed over (*SUProbCountMax*) [9],
- It can also specify whether AMF can be engaged to automatically repair the SUs within an instance of this SGType or not (*autoRepairOption*) [9].

Application Prototype (AppType): The AppType limits the SGTypes that can be used in its instances. It also specifies the name and the version of the AppType [9].

An ETF file describing a software implementation must contain at least the CompTypes and the CSTypes [9]. The other prototypes are optional and are used when the software vendor needs to specify constraints on the way the software is deployed (e.g. CompType A and CompType B must be collocated in the SUType C). Otherwise, all combinations and attribute settings are allowed. Hence, we refer to them as prototypes from which AMF types are derived. These AMF types must respect the constraints imposed by the prototype from which they are derived.

2.3 Related Work

From the configuration generation point of view, the most related work is reported in [1]. This work has automated the configuration generation process to alleviate the task of system designers. It generates multiple AMF configurations by considering all possible configuration options. The limitation of the work is that it does not take into account the availability requirement

as we do in this thesis. It addresses this issue by proposing a tool that transforms the generated configurations to a Deterministic and Stochastic Petri Net (DSPN) model [12], and then evaluate their respective availability using the TimeNET tool [13][14][15]. This is computationally expensive and time consuming.

The author in [10] proposed a model-driven approach to generate AMF configurations automatically. The approach generates configurations by using a set of transformations from ETF to AMF configurations. The approach considers the consistency and validity of the generated configurations. However, it only takes into account the functional requirements and does not tackle the non-functional requirements such as availability requirement. In this way, the generated configuration may not meet the requested level of availability.

On the other hand, numerous research papers addressed the issue of availability and reliability analysis, estimation and measurement. Usually, measurement-based methods focus and analyze the uptime and downtime of a system that is already in use, or the system that runs in a lab environment [8], to make an availability prediction. Other methods analyze the reliability or the availability of a system using statistical approaches or analysis models such as Markov models [16][17][18]. The shortcoming of these related works is that the simulation generally faces problems such as state explosion, long compilation time, etc. for simulating large-scale system models.

Most of the related work focuses on system availability instead of service availability and does not take into account an availability management middleware such as AMF and its specificities. From that perspective, close related works have been presented in [19] and [20]. The authors in [19] proposed a method for designing a system to meet service availability requirement.

This method lists all the possible arrangements of software/hardware entities to design a system, and it eliminates the ones that are not able to guarantee the requested service availability. Their main concern is to arrange the entities from hardware layer up to application layer, whereas our work tackles the problem of arranging the software entities composing an application, i.e. within the same layer. One of the shortcomings of the method in [19] is that it does not address the dependencies among entities and their impact on each other upon a failure.

The work in [20] targets partially AMF. The authors discuss the user-perceived service availability modeling and prediction. They consider service outage as the portion of time that the user perceives as such; the actual service outage may be higher. The approach considers other factors such as the user thinking time and the probability that the user sends another request at a certain time. This is more of system availability analysis taking into account the user behavior and perception. The method in [20] is applicable only to the configuration they have generated based on some assumptions. It is not reusable and for example, if the redundancy model changes, the model needs to be redesigned from scratch. The other shortcoming of this work is that they do not tackle actual recovery analysis, thus not all of the possible recovery actions are covered.

3 AMF Configuration Generation Process

This chapter focuses on reviewing the AMF configuration generation process proposed in [1]. It is important to discuss the approach prior to introducing the thesis contributions, since our work relies on it.

As shown in Figure 3-1, the AMF configuration generation process [1][7][21] consists of four main steps: 1. ETF prototype selection; 2. AMF type creation; 3. Creation of entities and setting of attributes; 4. Distribution of the entities. The input for the configuration generation process consists of the ETF and Configuration Requirements (CR) [10]. The CR describes the services to be provided by the application.

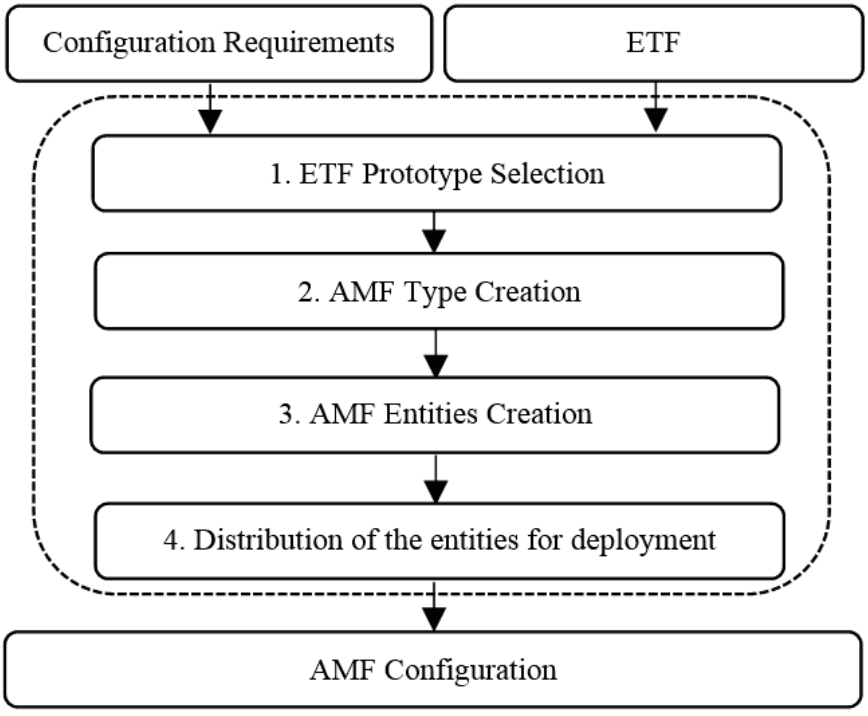

Figure 3-1 Configuration generation process steps

3.1 Input to the Configuration Generation Process

As mentioned, the input for the configuration generation process [1] consists of two parts:

- **The CR:** It has the information about the cluster (e.g. the number of cluster nodes), the number of SIs of each service type, the number of CSIs (of each CST) in each SI and information about the SGs that will protect the SIs [1].
- **The ETF:** It provides the available software catalogues to use for building the AMF application [1].

In the following, we describe the role of CR and ETF in the configuration generation process as introduced in [1].

3.1.1 CR Model

Through CR [1], the system designer defines the SIs and its CSIs as templates. The concept of template is used to create a number of similar SIs and CSIs in a generic way instead of creating each SI and CSI individually [1]. Each SI template encapsulates the information for the SIs that share common characteristics. For each SI template, the user should define the service type of the template, the number of SIs in the SI template and the number of active and standby assignments per SI.

An SI template groups one or more CSI templates. A CSI template defines a set of CSIs that are from the same CST and grouped in each SI of the SI template. For each CSI template, the system designer needs to define the CST and the number of CSIs of the CST (See Figure 3-2).

In addition, the system designer can describe the SGs that will protect the SI template by defining an SG template. He/she can determine the redundancy model of the SGs and the number of active, standby and spare SUs in each SG.

Many SI templates can be specified for the same application and form an administrative domain.

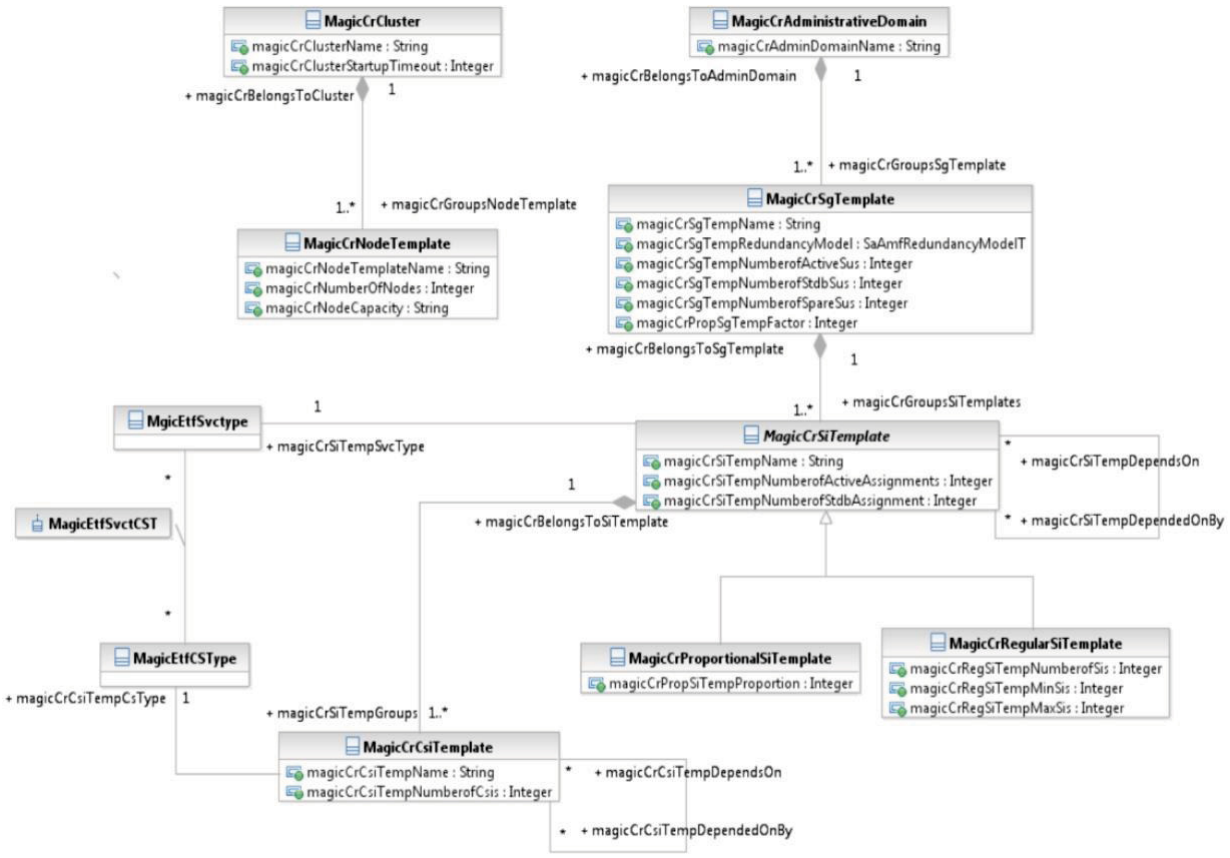

Figure 3-2 The CR domain model [1]

3.1.2 Entity Types File

As mentioned in Chapter 2, the AMF types are derived from the ETF prototypes. The ETF file is an Extensible Markup Language (XML) file that contains the software descriptions provided by one or several vendors. This XML file has to be created according to the ETF schema defined in [22]. While AMF types are described in terms of availability management and workload assignment, ETF deals with prototypes from the software deployment, capabilities, constraints, dependencies and limitations perspective [11]. Note that not all of the ETF prototypes are mandatory in a file, only CompType and CSType are mandatory [11]. If an ETF prototype is not

defined in the ETF, the system designer can build it in any valid way in the generation process. On the contrary, all the AMF types are required to be defined in the AMF configuration as they provide important information regarding the AMF entities and their relations [11].

3.2 Configuration Generation Process

As we mentioned, the configuration generation process [1] consists of four main steps, which are ETF prototype selection, AMF type creation, Creation of entities and setting of attributes and Distribution of the entities. In the following, we briefly explain these steps.

3.2.1 ETF Prototype Selection

This step is about finding all the ETF prototypes that can satisfy the CR (i.e. they can be used to provide the requested services). In this step, all the CompTypes and if specified the SUTypes, SGTypes and AppTypes that can provide the requested services are selected from the available ETFs. For this purpose, all available prototypes from the ETF are combined into a single graph as shown in Figure 3-3.

As one can notice, while an AppType has only one parent, the Root, other prototypes may have more than one. For example, SUType3 can be used to build instances of SGType2 or SGType3. Furthermore, some prototypes may be missing in the ETF. As an example, SGType3 can be used for any AppType other than AppType1, which specifies that only SGType1 and SGType2 can be used in its instances.

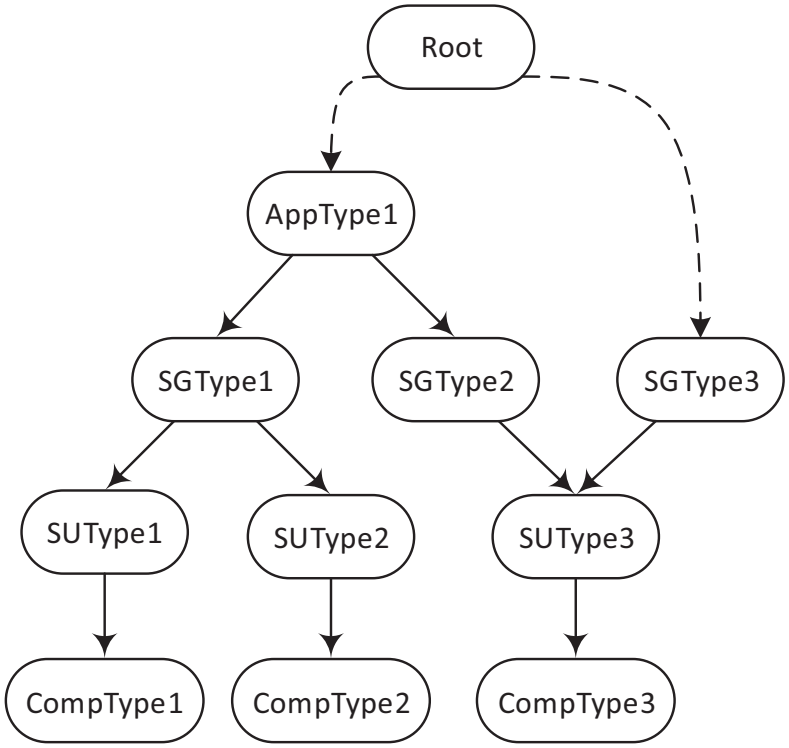

Figure 3-3 Examples of ETF Types

Often more than one suitable prototype or combination of prototypes can be found in the ETF. These are different sub-trees of the ETF starting from the Root all the way to the CompTypes. We refer to each of these prototype hierarchies as a type stack (TS).

Considering the example of Figure 3-3, we may find that either CompType1 or CompType3 is capable of providing the service requested in the CR. This leads to the three different type stacks shown in Figure 3-4.

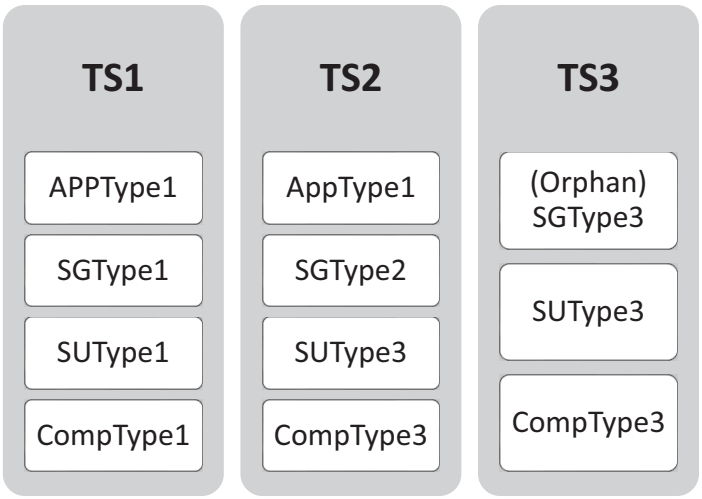

Figure 3-4 An Example of type stacks

These type stacks are passed onto the next step of the configuration generation process, the AMF type creation, where from the ETF prototypes the appropriate AMF types are derived. Based on type stacks shown in Figure 3-4, the generator can create at least three different AMF configurations that will be able to provide the requested service.

3.2.2 AMF Type Creation

In this step, the selected ETF prototypes in the type stacks are processed and the corresponding AMF types are derived. If there is no ETF prototype selected for an AMF type, at the level of SU, SG or application, one is created. The attributes of the AMF types are constrained by the attributes of the ETF prototypes they are derived from. Accordingly, if the ETF prototype specifies a particular value, it is used in the AMF type, otherwise the default provided by the ETF prototype, if any, is used. If no default is specified, the applicable default as defined in the AMF specification is used. Obviously, many other attribute settings would be possible and valid.

3.2.3 Creating Entities and Setting Their Attributes

Once the AMF types have been created, their corresponding AMF entities for the system are configured. For this, first the number of instances required for each AMF entity type is

determined. This number should take into account the CR as well as all the constraints imposed by their respective AMF type. For each AMF entity type the minimum number of instances that satisfy these constraints are created, which means that other choices are also possible and valid.

For all AMF entities, only the name and the attributes that need to be unique according to the specification are generated. The attributes that use by default a value in the corresponding AMF type are not set.

3.2.4 Distribution of Entities for Deployment

In the last step, the attributes that determine the distribution of the AMF entities (e.g. SUs) among the nodes of the cluster are set. Since multiple distribution options are possible, several configurations will be generated. As an example, consider a situation of having an SG grouping two SUs that can be distributed on three nodes. Figure 3-5 shows the deployment options to explore in this step. Each of the deployments (i.e. Deployment 1, Deployment 2 and Deployment 3) can lead to a different and valid configuration.

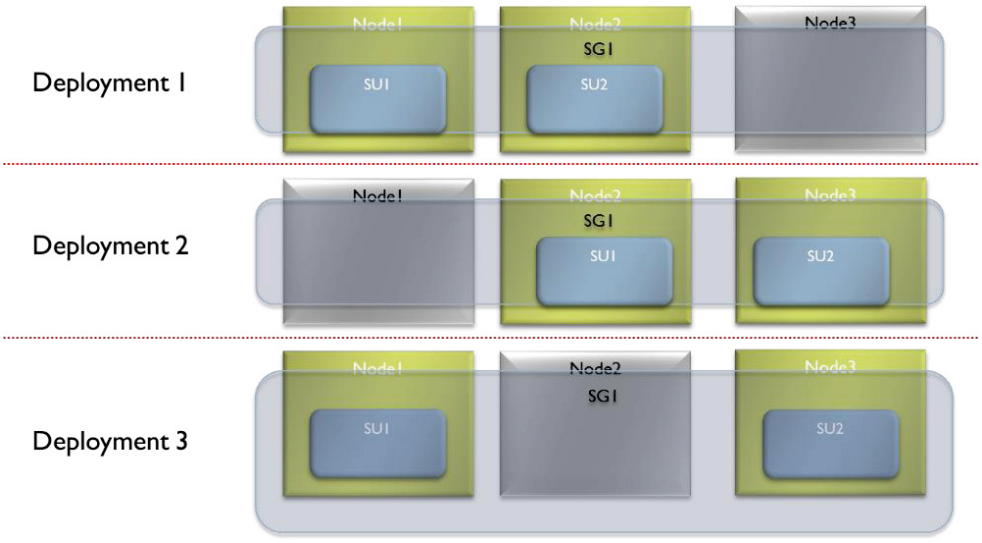

Figure 3-5 Multiple deployment options during configuration generation

3.3 Summary

In this chapter, we have reviewed the configuration generation process proposed in [1]. The process consists of four main steps. In each step, there are some decision points that can lead to generate multiple configurations. Our goal is to embed some configuration design patterns and methods in the generation process to improve service availability and eliminate the configurations that cannot satisfy the availability requirement. Moreover, if none of the configurations can satisfy the availability requirement, we select the one that can provide the highest level of availability.

4 Pattern-Based Configuration Generation

As discussed in Chapter 3, the automatic generation of AMF configuration(s) [1] is a process consisting of several steps. In each of the steps where we make a selection or settings, there might be rules or guidelines that can help us improve service availability. The purpose of this thesis is to define configuration design patterns and methods that can be embedded into the configuration generation process, in order to generate the configuration(s) that will meet the availability requirement or at least render a maximum service availability under the given constraints.

In order to define the configuration design patterns and methods, we have investigated the different options in each step of the configuration generation process. Some of these options are listed below:

- Step 1: Selecting multiple ETF prototypes to satisfy the CR
- Step 2: Creating multiple AMF types from the selected ETF prototype(s)
- Step 3: Creating different numbers of AMF entities for an AMF type
- Step 4: Choosing one of the multiple distribution options for AMF entities

In each decision point, we need to narrow down the choice we make in order to come up with the best configuration. The difficulty of this task is that in each step, the choices are based on some criteria, but not all of them.

Figure 4-1 summarizes the configuration design patterns and methods that we propose. We embed a total of four configuration design patterns and two methods into the steps of the

configuration generation process. These configuration design patterns and methods enhance the configuration generation process to meet the availability requirement.

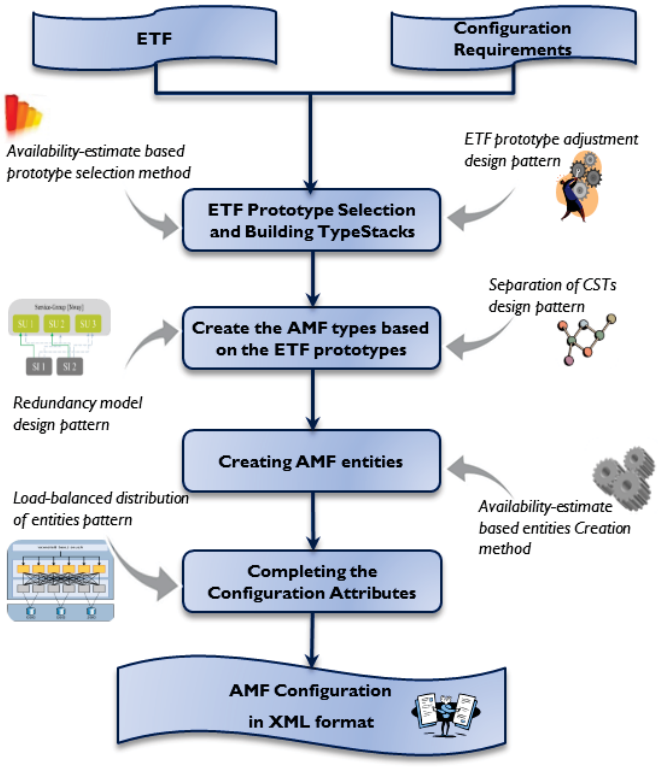

Figure 4-1 The pattern-based AMF configuration generation approach

We describe our contributions in two chapters. In this chapter, we discuss the configuration design patterns. We discuss the methods in Chapter 5.

4.1 Modifications to the Configuration Requirements

Prior to describing our configuration design patterns and methods, we will explain the changes we made on the CR. The CR in [1] does not include some of the attributes that we need in our work, such as the availability requirement. The elements added to the CR are as follows:

- **magicCrRegSiTemplateMinSA:** Since we are considering the expected level of service availability as a new requirement, we need to add it to the CR. By this

attribute, the system designer specifies the expected level of service availability for an SI template.

- **Components Global Attributes:** As part of Component Restart recovery action, AMF tries to reinstantiate the component. AMF first makes a predefined number of attempts to immediately reinstantiate the component. If all these instantiation attempts are unsuccessful, it tries to reinstantiate the component with another predefined number of attempts, this time with a predefined delay between each attempt [6]. The system designer should specify these predefined attributes. In the following, we introduce the corresponding attributes we defined in the CR:
    - **magicCrCompNumMaxInstantiateWithoutDelay:** This element defines the maximum number of immediate attempts that AMF should make to instantiate a component.
    - **magicCrCompNumMaxInstantiateWithDelay:** This element defines the maximum number of times that AMF should attempt to instantiate a component, with a delay between each attempt.
    - **magicCrDelayBetweenInstantiationAttempts:** The element defines the time (delay) that AMF should wait between the instantiation attempts.
- **magicCrNodeShutdownTime:** With this element, the system designer specifies the time the cluster nodes take to shut down. We need this attribute for our availability estimation methods.

The new CR domain model is shown in Figure 4-2.

Figure 4-2 The new CR domain model

4.2 ETF Prototype Adjustment Configuration Design Pattern

This section describes the ETF prototype adjustment configuration design pattern that is embedded in the first step of the configuration generation process. This configuration design pattern aims at minimizing the recovery impact zone imposed by a faulty component, thus improving service availability.

As mentioned in Chapter 3, in the first step of the configuration generation process, the generator selects the ETF prototypes that can satisfy the CR. Then, based on the selected ETF prototypes, it builds the corresponding type stacks. After, there is still a possibility to set/change the attributes of the ETF prototypes within the type stacks, because the software vendor may not set all the attributes of the ETF prototypes. We might be able to change the values, even if the software vendor sets the attributes. For instance, if the vendor does not provide any setting for the

component restartability option (i.e. *disableRestart=NULL*), we are allowed to set the attribute to either *False* or *True*.

We know that we can improve service availability by minimizing the number of impacted SIs in case of a component failure. To reduce the number of impacted SIs, we need to set the changeable attributes in a way that when a component fails, the recovery action affects the minimum number of SIs. Several attributes can determine the recovery impact zone imposed by a component failure. We refer to these attributes as recovery-related attributes. In the following, we introduce these recovery-related attributes and the guidelines to set them in order to improve service availability.

The most important recovery-related attribute is the recommended recovery action of a component specified in its CompType in the ETF. The recommended recovery action can be one of the followings: Component Restart, Component Failover, Container Restart, Node Switchover, Node Failover, Node Failfast, Cluster Reset and Application Restart. In some cases, no recommendation is provided [6]. When AMF detects a component failure, it starts with the recommended recovery action. Then, it continues with checking other configuration attributes that can alter this recovery action. We refer to this recovery action as actual recovery action [1]. Note that, the actual recovery action always has the same or a bigger impact zone than the recommended recovery action. The recovery-related attributes that can alter the recommended recovery action to a different actual recovery action are: component restartability (*disableRestart*), SU failover (*suFailOver*), probation counter maximum value for the SUs (*SUProbCountMax*) and components (*CompProbCountMax*) within the SG, SG auto reparability (*autoRepairOption*) and the SG redundancy model. In the following, we explain these recovery-related attributes and the way they determine the recovery impact zone of a faulty component.

One of the recovery-related attributes is the restartability attribute of the component. It is an important attribute because it determines if the component can be restarted to recover the service it provides. If the restartability of a component is disabled (i.e. *disableRestart = True*), AMF escalates the service recovery action. This means that AMF may decide to recover the service by reassigning SIs to another SU [6]. Enabling the restartability feature of a component is preferable, since it helps to keep the impact zone of a service recovery action to the faulty component. However, with respect to the other attributes, even if the restartability feature is enabled in some cases (i.e. *disableRestart = False*), AMF does not restart the component to recover the service. One of these attributes is the maximum allowed value to restart the components of the SG. It determines a threshold value equal to the total number of times that the components within an SG can be restarted in a given probation time [6]. If this value is set to zero, as a consequence of the service recovery action, AMF does not restart the components in the SG. In this case, AMF escalates the recovery action to the entire SU [6]. Consequently, if the maximum times that the SUs within the SG can be restarted is greater than zero (i.e. *SUProbCountMax>0)*, the SU containing the faulty component will be restarted. In contrast, if *SUProbCountMax=0,* the SU Failover will be performed as a recovery action [6]. Also, two other attributes directly let AMF restart the SU or not. One is the SG auto reparability, which determines if AMF should be engaged to restart the SUs in an SG as a repair or recovery action [6]. The other one is the SU failover attribute, which indicates that the whole SU must be failed over as a single entity in case of its constituent components failover [6].

Another important recovery-related attribute is the redundancy model of the SG. A redundancy model can alter the recovery action of a component indirectly. A recovery action should respect the characteristics and constraints of the redundancy model. For instance, the 2N

redundancy model does not allow for the failover of a single component within an SU and the recovery action will be escalated to the failover of the entire SU. On the other hand, the NWay redundancy model allows for the failover of some of the components within an SU.

Table 4-1 Pattern for setting the attributes of ETF prototypes

Row	CompType		SUType	SGType				*Recovery Impact Zone*
	Recommended Recovery Action	**disableRestart**	**suFailOver**	**autoRepair**	**Component ProbCountMax**	**SU ProbCountMax**	**Redundancy Model**	
1	No Recommendation OR Component Restart	False	X	X	> 0	X	No-Red., 2N, N+M,NWay, NWay-Active	***Component Restart***
2		False	X	True	0	> 0		***SU Restart***
3		False	X	True	0	0		***SU Failover***
4		True	X	X	X	X		***SU Failover***
5		False	X	False	0	X		***SU Failover***
6		False	False	True	X	X	NWay, NWay-Active	***Component & Siblings' Failover***
7	Component Failover	X	X	X	X	X	No-Red., 2N, N+M	***SU Failover***
8		False	False	True	X	X	NWay, NWay-Active	***Component & Siblings' Failover***
9		X	True	X	X	X		***SU Failover***
10		True	False	True	X	X		***SU Failover***
11		X	False	False	X	X		***SU Failover***

We have summarized the proposed configuration design pattern in Table 4-1,which shows how to set the recovery-related attributes, in order to minimize the recovery impact zone in case

of a component failure. As shown, we have categorized our proposed configuration design pattern into two parts, based on the recommended recovery actions provided for a CompType. The first category shows how to set the attributes if the recommended recovery action is no recommendation or Component Restart, whereas the second category is for when it is Component Failover. Other recovery actions such as Node Switchover, Node Failover, Node Failfast and Application Restart do not give us the opportunity to minimize the recovery impact zone. In other words, the recommended recovery action for a component may be at the node, application or cluster level, but configuration attributes never change their impact zone. This configuration design pattern can be used only if the redundancy model of the SGType is given in the ETF.

In Table 4-1, *X* stands for any value of the given attribute. That is, the attribute does not change the recovery action in the given context. For instance, consider a case where the recommended recovery action of a CompType is set to Component Failover, and the redundancy model of the SGType is 2N. In this case, none of the recovery-related attributes can change the fact that as the actual recovery action, the whole SU will be failed over.

The example in Table 4-2 helps us to understand the application of the design pattern. Let us assume we have a type stack and we want to set its attributes. As can be seen in Table 4-2, the vendor sets the *disableRestart* attribute of the CompType is set to *NULL.* We are allowed to set it to either *True* or *False*. We can see that in Table 4-1, Rows 1 and 4 are applicable. If we set the attribute to *False* (Row 1), the actual recovery action will be Component Restart, that is, only the component will be impacted in case of its failure. On the contrary, if we set it to *True* (Row 4), the actual recovery action becomes SU Failover (i.e. the whole SU will be impacted). Therefore, the pattern guides us to set the attribute to *False*, limiting the recovery impact zone to the component level, rather than setting it to *True*, which will expand the recovery impact zone to the SU level.

Table 4-2 Application of the ETF prototype adjustment pattern

CompType		SUType	SGType			
Recommended Recovery Action	**disableRestart**	**suFailOver**	**autoRepair**	**Component ProbCountMax**	**SU ProbCountMax**	**Redundancy Model**
Component Restart	NULL	False	True	3	5	2N

4.3 Separation of CSTs Configuration Design Pattern

In the previous section, we proposed the ETF prototype adjustment configuration design pattern that can be embedded into the first step of the configuration generation process [1]. In this section, we move to the next step of the generation process, namely, to the creation of AMF types. In this step, the generator derives the corresponding AMF types based on the ETF prototypes selected in the first step. We can derive multiple AMF types based on an ETF prototype, by either customizing the ETF prototype or using it as it is. The issue in this step is how to derive the AMF types from the ETF prototypes in order to improve service availability. We propose the separation of CSTs configuration design pattern. It aims at deriving multiple AMF CTs from an ETF CompType such that a component failure affects the minimum number of service types (i.e. SIs). Thus, service availability will be improved. This is similar to the principle of separation of concerns in Software Engineering.

Let us consider an application consisting of more than one service (i.e. more than one SI template). In the first step of the generation process, we select the ETF CompType(s) that can provide all the application services. Then, the generator derives the corresponding AMF CT(s) from the selected ETF CompType. We know that AMF uses the CSTs to assign the CSIs to the

components. It follows that if we create the CTs so that they can only provide the CSTs of a particular service type, we can keep the impact of a component failure on particular service only. Our configuration design pattern aims at deriving multiple AMF CTs by customizing the selected ETF CompType, in a way that each AMF CT can provide only a particular service. Hence, different services will be assigned to different components[1]. If a component fails, it will impact only one of the application services.

Note that the following constraints need to be satisfied for the application of this configuration design pattern:

1. The actual recovery action of the ETF CompType should be at component level, namely Component Restart or Component Failover. This is because if the actual recovery action is not at the component level (i.e. SU, application, node or cluster levels), all the services will be impacted anyways.
2. The dependent CSTs provided by an ETF CompType, should not be separated into different AMF CTs.

Let us explain this configuration design pattern with the example in Figure 4-3. The system designer requests an application with two SI templates (i.e. service types). From the first step of the generation process, we selected the ETF prototypes that can satisfy the CR and built the corresponding type stack. Note that in this example, we are also making the following assumptions:

- There is a dependency between the CSTB and CSTC,

[1] AMF uses the CSTs to assign the CSIs to the components. It assigns the CSIs (of a CST) to the components that can provide the CST.

- The actual recovery action for the ETFCT1 is determined as Component Restart.

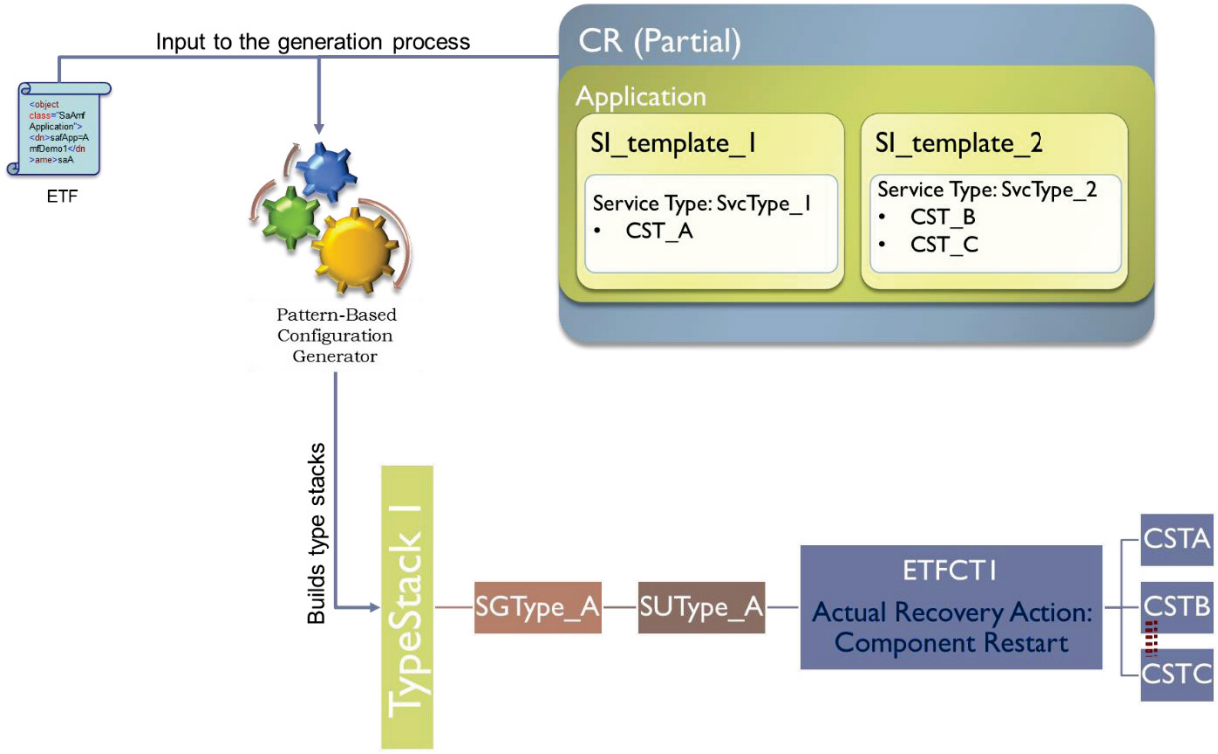

Figure 4-3 An illustration example for the separation of CSTs configuration design pattern

Once the type stack has been built, the generator derives the AMF types from the selected ETF prototypes. At this stage, we can derive the AMF CTs from ETFCT1 in four different ways. These are depicted as “Context A”, “Context B”, “Context C” and “Context D” in Figure 4-4. However, only one is the best solution in terms of service availability. We now discuss these four options and explain how our configuration design pattern suggests the best one.

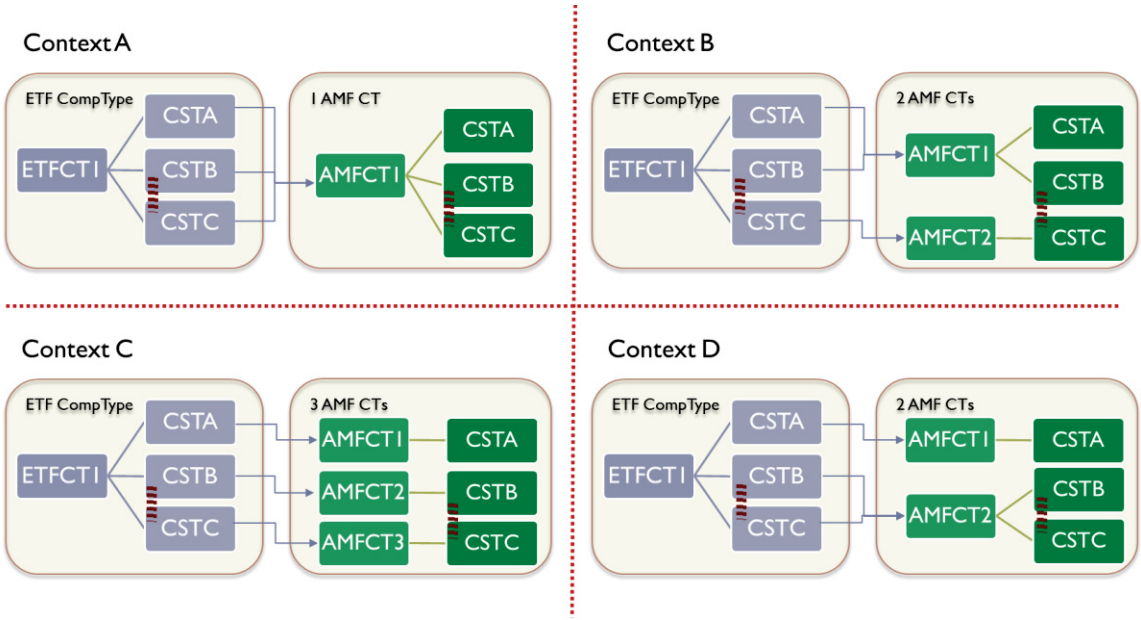

Figure 4-4 The different option to create AMF CTs for the example of CSTs configuration design pattern

- "Context A": Our configuration design pattern does not suggest this context. In this case, both SI templates (i.e. SI_template_1 and SI_template_2) will be assigned to an instance of AMFCT1. If the component fails, both SI templates (i.e. service types) will become unavailable (See Figure 4-5).

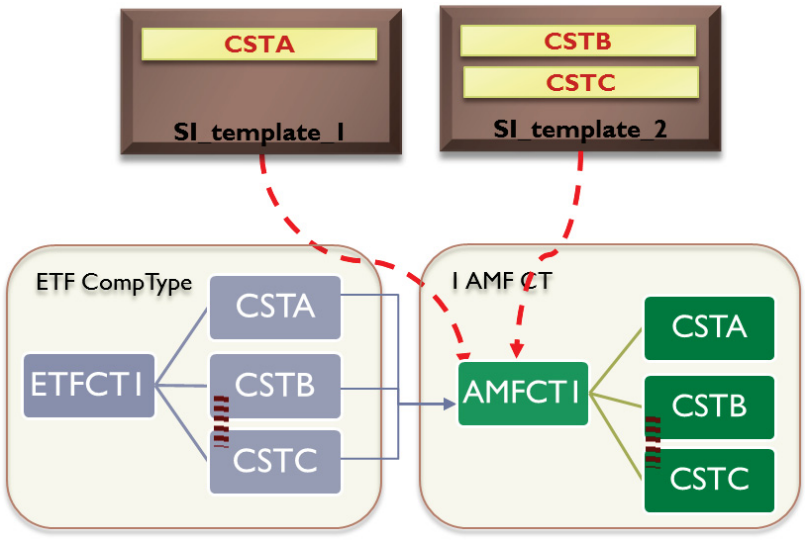

Figure 4-5 An illustration of SI templates' assignment to only one AMF CT

- "Context B": This is not the best solution, because of the following reasons:
    - Similar to "Context A", the SIs of the SI_template_1 and SI_template_2 will be assigned to the same component (i.e. an instance of AMFCT1). Thus, both SI templates will be impacted in case of component failure.
    - According to our constraints, if there is a dependency between the CSTypes that the selected ETF CompType can provide, we should not separate the CSTypes into different AMF CTs. Since there is a dependency between CSTB and CSTC provided by ETFCT1, we should not separate them into AMFCT1 and AMFCT2.

Figure 4-6 shows the assignment of the SI templates to the different components according to "Context B".

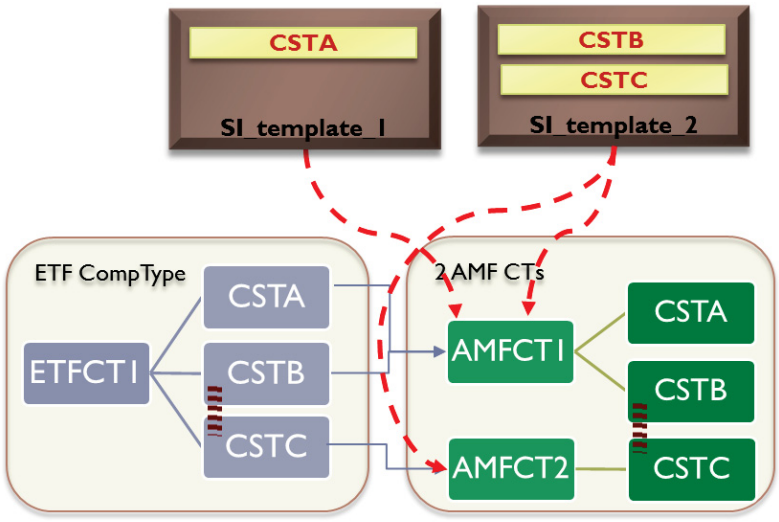

Figure 4-6 An illustration of SI templates' assignment to two AMF CTs with overlap

- "Context C": This is also a poor solution, since the dependent CSTypes (i.e. CSTB and CSTC) are separated into different AMF CTs (See Figure 4-7).

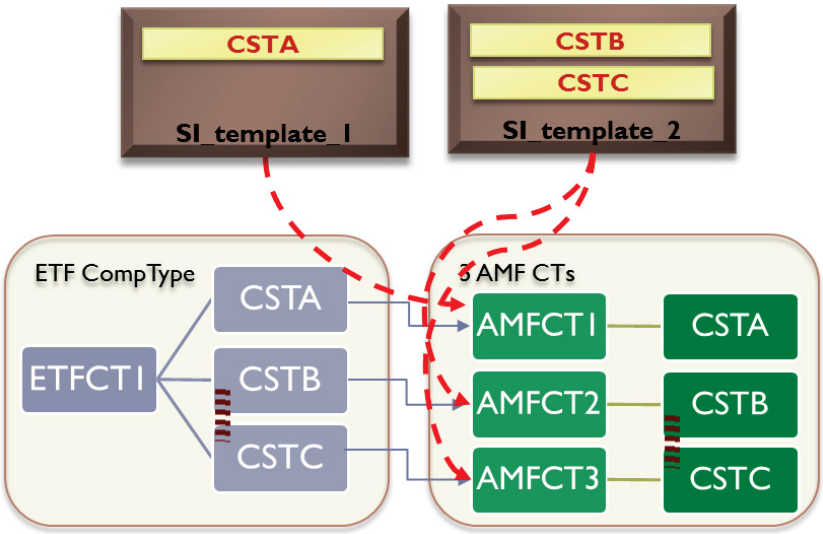

Figure 4-7 An illustration of SI templates' assignment to three AMF CT

- "Context D": This is exactly what our configuration design pattern suggests. The reasons to prefer this solution is as follows:
    - AMF will assign the SI_template_1 and SI_template_2 to the instances of AMFCT1 and AMFCT2, respectively. In this way, when one of the components fails, the other component can still provide the other service.
    - The dependency between the CSTB and CSTC has been taken into account. The same CT (i.e. AMFCT2) will provide both CSTB and CSTC.

Figure 4-8 illustrates the assignment of the SI templates to the different components according to "Context D".

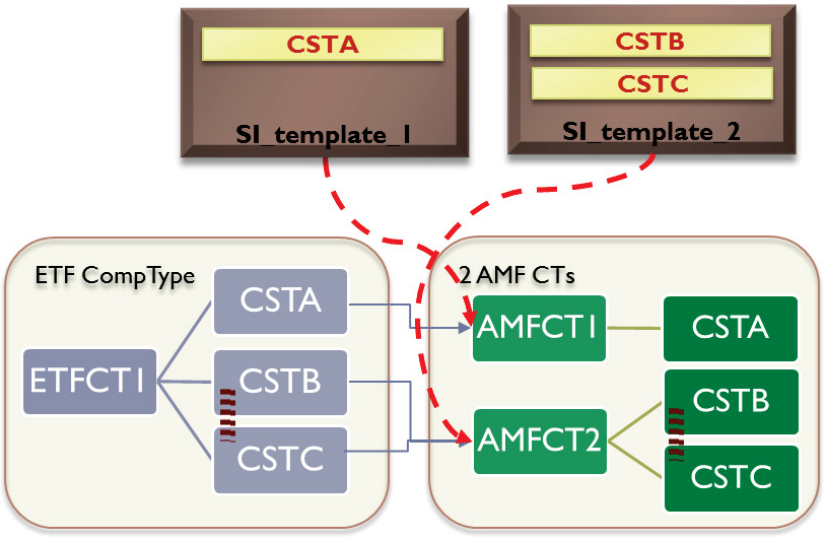

Figure 4-8 An illustration of SI templates' assignment to two different AMF CTs without overlap

4.4 Redundancy Model Selection Configuration Design Pattern

The redundancy model selection configuration design pattern is to select the appropriate redundancy model for an SG type in the second step of the generation process.

In the configuration generation process in [1], the system designer is responsible for specifying the redundancy model of the SGs as part of the input. In the ETF prototype selection step, the generator selects the ETF SGType(s) that has the requested redundancy model. After, the generator creates the corresponding AMF SG type based on the selected ETF SGType. Accordingly, the created AMF SG type will have the redundancy model as specified by the system designer. However, we believe that the redundancy model selection should not be the responsibility of a system designer, unless he/she desires a particular redundancy model. This is because he/she might not have any information to select the appropriate redundancy model at the time he/she is defining the CR. For instance, he/she may not know the redundancy model of the SGType or the capability model of the CompTypes in the ETF.

To alleviate the system designer's task, we propose the redundancy model selection configuration design pattern, which aims at selecting the appropriate redundancy model for AMF SG types. This redundancy model selection is based on the CTs' capability model and the redundancy models preferences. Note that this configuration design pattern is applicable only if the redundancy model of the selected SGType is not provided in the ETF. In other words, if the redundancy model of the selected SGType has been provided in the ETF, the derived AMF SG type must use the same redundancy model.

The AMF specification [6] defines five different redundancy models, namely No-Redundancy, 2N, N+M, NWay and NWay-Active. The 2N and No-Redundancy are the simplest redundancy models. The N+M generalizes the 2N redundancy model. Similarly, the NWay generalizes the N+M allowing an SU to have simultaneously active and standby assignments for different SIs. The NWay-Active supports the assignment of an SI as active to more than one SU. The NWay-Active and No-Redundancy are the only two redundancy models that do not allow standby assignments for SIs.

In order to select the redundancy model of an SG type, the capability model of the CTs within the SG type must be taken into account. The capability model of a CT is defined for each CST the CT provides [6]. It defines the number of active and standby CSIs that a component of a CT can handle for a particular CST [6]. A component capability model may fit more than one redundancy model. Our configuration design pattern consists of taking the most general redundancy model that applies according to the capability model of the CTs. We have ranked the redundancy models according to their availability, flexibility and upgradability. Moreover, we have taken into account the preliminary results on comparing the redundancy models that were

discussed in [23]. A summary of our redundancy models preferences with respect to the CTs' capability model follows:

1. We select the NWay-Active redundancy model, if none of the CTs has the capability to be standby for any of the CSTs they provide. This means that all of the CTs should have the capability of X_ACTIVE, 1_ACTIVE or non-preinstantiable for all of the CSTs they provide.

2. We select the NWay redundancy model, only if all of the CTs have the capability model of X_ACTIVE_AND_Y_STANDBY for all of the CSTs they provide.

3. We select the N+M redundancy model, if NWay and NWay-Active redundancy models cannot be selected. This means that at least one of the CTs has the capability of X_ACTIVE_OR_Y_STANDBY for one of the CSTs it provides.

4. The 2N and No-Redundancy redundancy models are used only when system designer asks for them.

Again, it is important to mention that the component capability is defined for each CST a CT can provide Thus, the redundancy model of an SG type has to respect the capability model of all the CSTs that can be provided by the CTs. The redundancy model selection using Algorithm 4-1 is shown hereafter.

```
SGT Redundancy Model Selection (SG type : sgt)

Begin

  if (system designer asked for 2N or No-Redundancy) then
     SelectedRedundancyModel = 2N or No-Redundancy based on what the designer asked

  Else
     int totalCountOfCSTs = 0        // to store the total count of CSTs
     int counterNWayActive = 0       // to store the total count of CTs that do not have standby capability
     int counterNWay = 0             // to store the total count of CTs that have the X_ACTIVE_AND_Y_STANDBY
     int counterNplusM = 0           // to store the total count of CTs that have X_ACTIVE_OR_Y_STANDBY

     For SUT:sut of sgt do
       For each CT:ct of sut do
          For each CST:cst of ct do

             Increment totalCountOfCSTs

             If (ct has the capability of X_ACTIVE, 1_ACTIVE or non-preinstantiable for the cst) then
                   Increment counterNWayActive

             Else If (ct has the capability of X_ACTIVE_AND_Y_STANDBY for the cst) then
                   Increment counterNWay

             Else
                   Increment counterNplusM

             End if

          End do // each CST
       End do // each CT
     End do // SUT

    // all of the CTs have the capability of X_ACTIVE, 1_ACTIVE or non-preinstantiable for all of the CSTs they provide
     If (totalCountOfCSTs = counterNWayActive) then
           SelectedRedundancyModel= NWayActive

     // all of the CTs have the capability model of X_ACTIVE_AND_Y_STANDBY for all of the CSTs they provide
     Else If (totalCountOfCSTs = counterNWay) then
           SelectedRedundancyModel= NWay

     // at least one of the CTs has the capability of X_ACTIVE_OR_Y_STANDBY for one of the CSTs it provides
     Else
           SelectedRedundancyModel= N+M
     End if

  End if

   Return SelectedRedundancyModel
End
```

Algorithm 4-1 Selecting the redundancy model of an SG type

4.4.1 An application of the redundancy model selection configuration design pattern

The following example illustrates our configuration design pattern. Figure 4-9 shows an example of an AMF SG type (i.e. SGT_A). As shown in Figure 4-9, the SG type has an SU type

consisting of two CTs: CT_A and CT_B. While CT_A can provide CST_A1 and CST_A2, CT_B can provide CST_B1 and CST_B2. We apply the steps of our configuration design pattern to select the appropriate redundancy model for SGT_A:

1. We cannot select the NWay-Active redundancy model, since only one of the CTs has the capability of X_ACTIVE.
2. We cannot select the NWay redundancy model, since not all of the CTs have the capability model of X_ACTIVE_AND_Y_STANDBY for all of the CSTs they provide.
3. We select the N+M redundancy model, because there is at least one CT (i.e. CT_A) that has the capability of X_ACTIVE_OR_Y_STANDBY for one of the CSTs it provides (i.e. CST_A1).

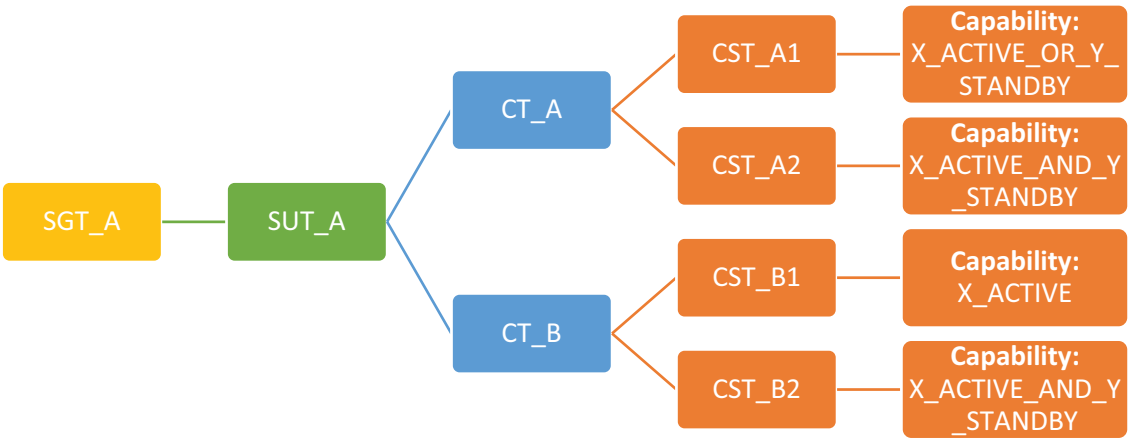

Figure 4-9 An application example of the redundancy model selection configuration design pattern

4.5 Configuration Design Pattern for Load Balancing

In the last step of the configuration generation process [1], the generator sets the configuration attributes related to the SUs distribution on the nodes. As mentioned earlier, in this

step we have a choice of multiple distribution options (See Figure 3-5). The issue in this stage is to select the best distribution option that will improve the expected level of service availability. We propose a load-balanced entity distribution configuration design pattern. It ensures the even distribution of SUs over the nodes, and furthermore guarantees a balanced load even after a single failure. An AMF-managed system is expected to tolerate the failure of one SU without causing a service outage [1]. As a result, we target load-balancing before and after a single failure and do not tackle load-balancing with multiple failures. Note that this solution is applicable only for distributing the SUs of the SGs with the 2N redundancy model. In our solution, we are making the following assumptions:

- The nodes have enough capacity to host the SUs,
- The active and standby SUs impose the same load on the nodes, and
- There are at least two nodes in the cluster.

This pattern consists of 3 steps as follow:

- Rank the SUs forming the SGs to define their active or standby role,
- Set the configuration attributes to ensure the even distribution of the standby SUs within the cluster, and
- Set the configuration attributes to ensure the even distribution of their corresponding active SUs among the other nodes.

In the rest of this section, we discuss the aforementioned steps in detail.

During the first step of the solution, we rank the SUs of each SG within the application. We know that in the 2N redundancy model there are at least one active and one standby SU[1]. AMF assigns the active or standby role to each SU of an SG at runtime according to the ranking of the SUs.

The rank of an SU is a positive integer and the lower the value, the higher the rank [6]. AMF assigns the active role to the highest-ranked SU and assigns the standby role to the lowest-ranked SU [6]. The configuration attribute that defines the rank of an SU is *saAmfSuRank*. We use a simple solution to rank the SUs of each SG in the cluster. For each SG, we set the *saAmfSuRank=0* for one of its SUs, and *saAmfSuRank=1* for the other SU. AMF will assign the active role to the SU with *saAmfSuRank=0* and the standby role to the SU with *saAmfSuRank=1*.

Once we have ranked the SUs forming the SGs, we need to distribute the standby SUs within the cluster. In order to distribute them over the cluster nodes evenly, we need to calculate the number of standby SUs that each node should host (*NodesStandbyLoad*). Since the total number of standby SUs might not be a divisor of the number of cluster nodes, some nodes might host the floor of *NodesStandbyLoad*, while others might host the ceiling of it. The number of standby SUs to be hosted on each node can be calculated as Eq. (4-1):

$$NodesStandbyLoad = \left\{ \left\lfloor \frac{TotalNumberOfStanbySUs}{NoOfNodes} \right\rfloor \middle| \left\lceil \frac{TotalNumberOfStanbySUs}{NoOfNodes} \right\rceil \right\} \quad \textbf{(4-1)}$$

where the *NoOfNodes* is the number of nodes in the cluster, and the *TotalNumberOfStandbySUs* is the total number of standby SUs from all of the SGs. For example, assume that there is an application with five 2N redundancy model SGs. We know that there is only one standby SU in a

[1] Note that there might be one or more spare SUs in each SG. However, we do not discuss the distribution of spare SUs in our solution.

2N redundancy model SG. As a result, the *TotalNumberOfStandbySUs* for five SGs will be equal to five.

Once we have calculated the number of standby SUs that each node should host, we need to set the configuration attributes to ensure the even distribution of the standby SUs over the cluster nodes (See Algorithm 4-2). To do so, we need to configure the *SaAmfSUHostNodeOrNodeGroup* attribute for each standby SU. AMF uses this attribute to map an SU on a node at runtime [6].

```
ConfigureStandbySUsDistributionOnTheNodes ()
Begin
NodesStandbyLoad = use equation (4-1) to find the maximum standby SUs to be hosted on each node
For each SG:sg do
    For each Standby SU:stdsu in sg do
        For each Node:N in cluster do
            If (total load on N is less than N.NodesStandbyLoad) then
                stdsu.SaAmfSUHostNodeOrNodeGroup=N // host the standby SU on N
                total load on N += 1
                break (Node:N)
            End if
        End do // each Node:N
    End do // each Standby SU:std
End do // each SG:sg
End
```

Algorithm 4-2 An algorithm to configure standby SUs distribution on the nodes

For the third step, we need to configure the active SUs distribution on the nodes. To do so, at first we need to calculate the maximum number of SUs (including both active and standby SUs) that each node should host, as Eq. (4-2):

$$NodesMaxLoad = \left\{\left\lfloor\frac{TotalNumberOfSUs}{NoOfNodes}\right\rfloor \middle|\middle| \left\lceil\frac{TotalNumberOfSUs}{NoOfNodes}\right\rceil\right\} \quad (4\text{-}2)$$

where the *TotalNumberOfSUs* is the total number of active and standby SUs in all the SGs. It can be concluded that the *NodesMaxLoad-NodesStandbyLoad* determines how many active SUs can be hosted on the node.

Afterward, we use Algorithm 4-3 to set the configuration attributes in a way that ensures the even distribution of the active SUs over the nodes. The algorithm considers the following:

- The maximum number of active SUs that a node can host, should be equal to the *NodesMaxLoad – NodesStandbyLoad*.
- No active and standby SUs of the same SG should reside on the same node.

```
ConfigureActiveSUsDistributionOnTheNodes ()
Begin
  // we have previously hosted the standby SUs on the nodes
  For each Node:node do
    total load on node = node.NodesStandbyLoad
  End do // each Node:node

  For each SG:sg do

    stdsu = the standby SU of the SG
    actsu = the active SU of the SG
    stdNode = stdsu.SaAmfSUHostNodeOrNodeGroup // the node hosing the standby SU
    For each Node:N in cluster do

      If (N is not stdNode) then // an active SU cannot reside on the same node as its standby resides

        If (total load on N is less than N.NodesMaxLoad) then

          actsu.SaAmfSUHostNodeOrNodeGroup=N // host the active SU on N

          total load on N += 1

          break (Node:N)

        End if
      End if

    End do // each Node:N

  End do // each SG:sg
End
```

Algorithm 4-3 An algorithm to configure the active SUs distribution on the nodes

4.5.1 An application of the load-balanced entity distribution configuration design pattern

In Figure 4-10, we have an application with twelve 2N redundancy model SGs. Let us assume the SUs of these SGs should be distributed over a cluster with five nodes (i.e. *NoOfNodes*=5). We also assume that we have ranked the SUs in the SGs. As a result, we know which SUs will be assigned as active or standby at runtime.

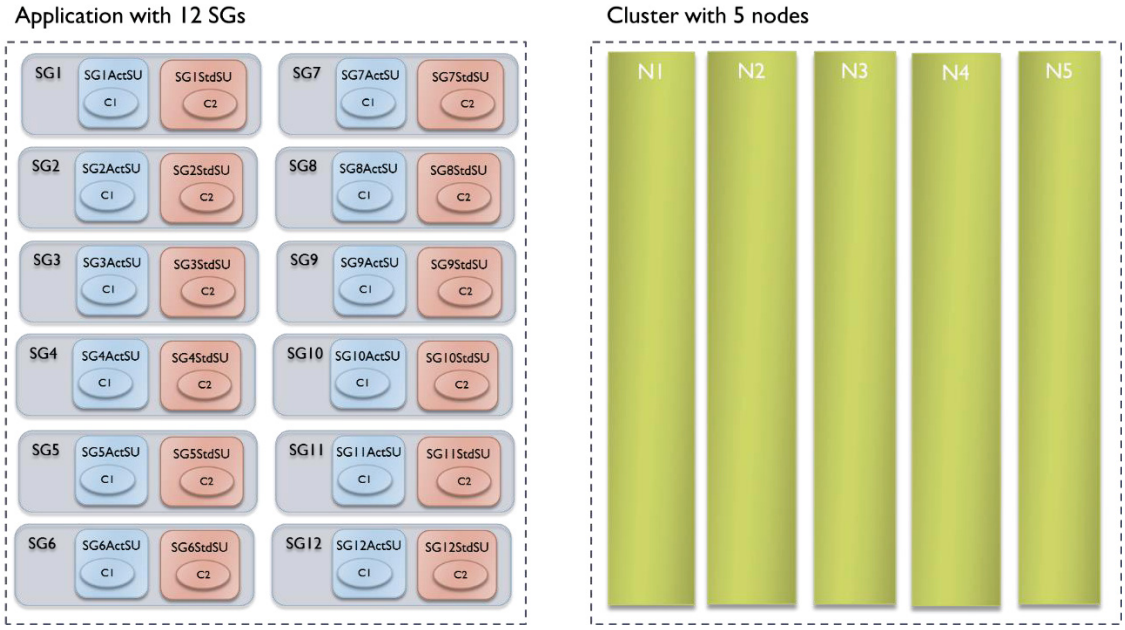

Figure 4-10 An application example of load-balanced entity distribution configuration design pattern

We need to configure the standby SUs distribution on the nodes. To do so, we need to calculate the total number of standby SUs that each node should host using Eq. (4-1).

$$NodesStandbyLoad = \left\{\left\lfloor\frac{TotalNumberOfStanbySUs}{NoOfNodes}\right\rfloor \middle| \middle| \left\lceil\frac{TotalNumberOfStanbySUs}{NoOfNodes}\right\rceil\right\}$$

$$= \left\{\left\lfloor\frac{12}{5}\right\rfloor \middle| \middle| \left\lceil\frac{12}{5}\right\rceil\right\} = \{2|3\}$$

As a result, three nodes should host 2 standby SUs, whereas the other two nodes should host 3 standby SUs.

Once we have calculated the number of standby SUs each node should host, we apply Algorithm 4-2 to set the configuration attributes and ensure a balanced distribution of the standby SUs over the cluster nodes (See Figure 4-11).

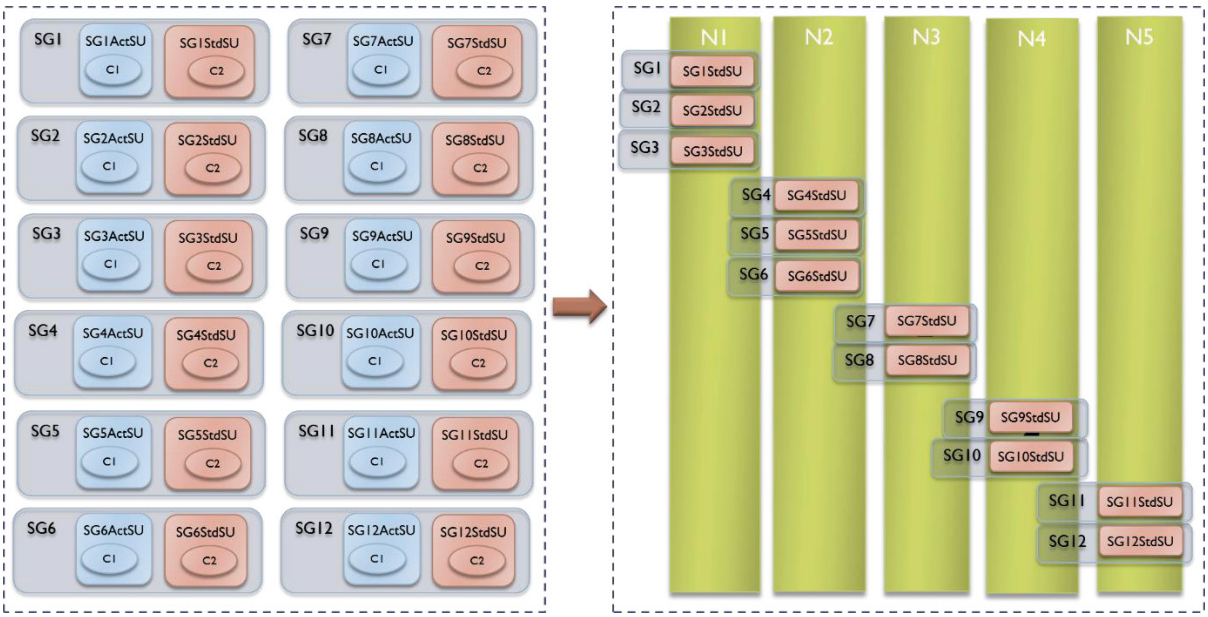

Figure 4-11 An example of distribution of standby SU using algorithm 4-2

We now need to set the configuration attributes to ensure an even distribution of the active SUs over the cluster nodes. At first, we calculate the maximum number of SUs (active and standby) that each node should host, by using Eq. (4-2):

$$NodesMaxLoad = \left\{ \left\lfloor \frac{TotalNumberOfSUs}{NoOfNodes} \right\rfloor \middle| \middle| \left\lceil \frac{TotalNumberOfSUs}{NoOfNodes} \right\rceil \right\}$$

$$= \left\{ \left\lfloor \frac{24}{5} \right\rfloor \middle| \middle| \left\lceil \frac{24}{5} \right\rceil \right\} = \{4|5\}$$

As a result, five nodes should host 5 SUs, and only one node should host 4 SUs. We use Algorithm 4-3 in order to set the configuration attributes related to the active SUs distribution over the nodes. For the discussed example, Figure 4-12 depicts an overall picture of the SUs distribution over the nodes.

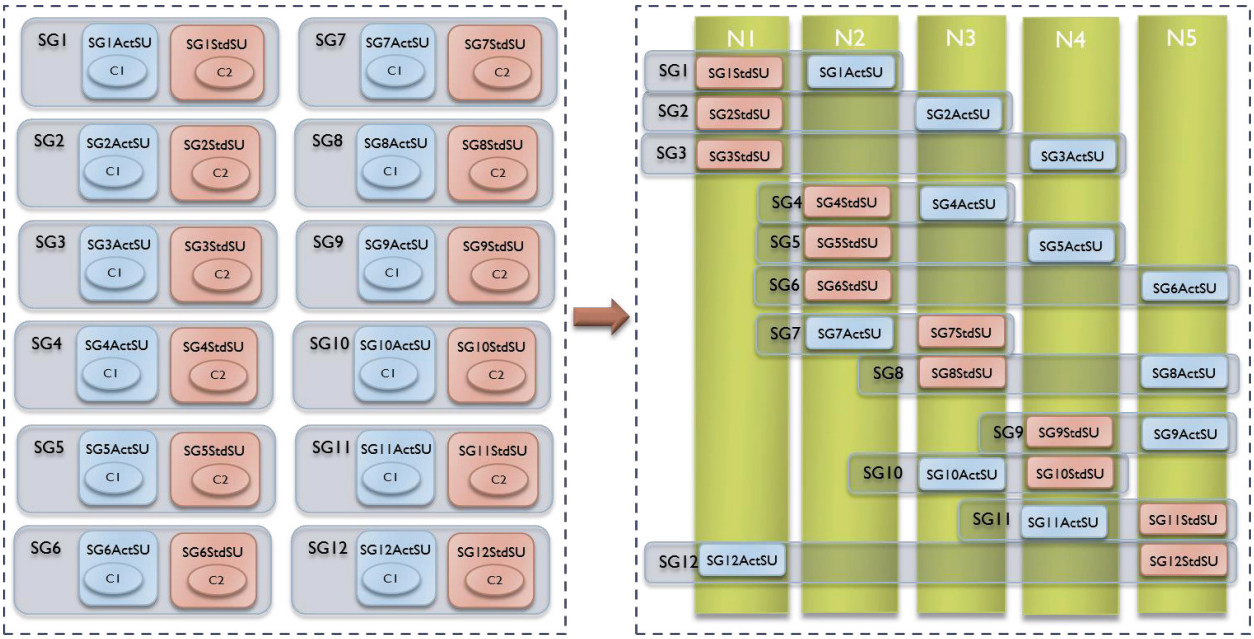

Figure 4-12 The overall picture of the SUs distribution over the nodes

4.6 Chapter Conclusion

In this chapter, we have introduced four configuration design patterns with the aim to improve the expected level of service availability of the generated AMF configuration. We have discussed some design decision points in the configuration generation process and the way our configuration design patterns make the best decisions to improve availability in each step. We summarize the proposed configuration design patterns as follows:

- ✓ We have proposed the ETF prototype adjustment configuration design pattern that is applicable in the first step of the generation process. The goal of this configuration design pattern is to set the attributes of the selected ETF prototypes in order to minimize the recovery impact zone of a faulty component. With it, we aim to decrease the number of SIs impacted during a component failure.
- ✓ We have also discussed two configuration design patterns that can be embedded in the second step of the generation process:

- The separation of CSTs configuration design pattern: the purpose of this configuration design pattern is to derive the AMF CTs from an ETF CompType in such a way that not all of the application services become unavailable due to a component failure. We aim to force the AMF to assign the SIs of different service types to different components. In this way, if a component fails, only the SIs of one service type will be affected.

- The redundancy model selection configuration design pattern: the objective of this configuration design pattern is to select an appropriate redundancy model for an AMF SG type. The configuration design pattern consists of selecting the most general redundancy model, taking into account the components' capability model. We can only use this configuration design pattern when the redundancy model of the SGType is not set in the ETF.

✓ As mentioned before, in the last step of the generation process, it is possible to choose among multiple distribution options. To select the best option, we have proposed the load-balanced entity distribution configuration design pattern. The objective of this configuration design pattern is to ensure a balanced distribution of the SUs over the cluster nodes before and after a single failure. Note that this configuration design pattern is only applicable to the SGs with a 2N redundancy model.

Each of the aforementioned configuration design patterns helps us to select the best solution in each step of the generation process to achieve superior service availability. In conclusion, we can generate only the configuration(s) that can provide the best level of service availability.

5 Availability Estimate Methods-Based Configuration Generation

The previous chapter covered the configuration design patterns that improve the level of service availability. Although using a configuration design pattern can improve the expected level of service availability, it is not guaranteed to generate only the configuration(s) that meet the requested service availability. In response, we can use a quantitative method to estimate the level of service availability and generate only the configurations that can satisfy the required level of availability. In this chapter, we discuss the two methods (See Figure 4-1) that can enhance the configuration generation process [1], and help us to generate the configurations that meet the availability requirement.

5.1 Availability Estimate-Based Prototype Selection Method

We propose an availability estimate-based prototype selection using partial configuration information without using full-fledged availability analysis methods. Our method can be applied at the first step of the configuration generation process [1] to prioritize or eliminate configuration options.

As mentioned in Chapter 3, in the configuration generation process [1] different configurations can be generated based on the different choices that can be made at each decision point. Some of these configurations may not meet the required availability specified in the CR. As author in [1] describes, to generate a configuration that satisfies the availability requirement, all the possible configurations need to be generated and analyzed (e.g. using simulation tools) to select the one with the required level of availability, as describe in [1]. However, we propose a method

that estimates service availability based on partial configuration information instead. This estimate can then be used to eliminate the type stacks that would lead to configurations that cannot satisfy the availability requirement. Our method allows us to filter out these type stacks and the corresponding configurations and proceed to the second step of the configuration generation process only with the type stack(s) that can achieve the required service availability. Thus, we avoid the use of full-fledged availability analysis methods and tools that are resource and time consuming and may not be usable for complex models.

As mentioned earlier, the deliverable service availability of a system can be determined by two factors, which are the MTTR and the MTTF of its composing components [3]. The MTTF is the average time that a component is expected to operate between two consecutive failures [3], whereas the MTTR refers to the average time that is needed to repair the component, so that it can provide the service again [3]. In this work, we consider the MTTF of a component as a constant failure rate provided by the vendor (i.e. ETF) for the CompType from which the component is derived. As a result, we only need to estimate MTTR.

In an AMF configuration, SUs are redundant and the configured or recommended recovery policies determine how the service is recovered after a component failure. We interpret MTTR as the mean time to recover the service rather than the mean time to repair the faulty component. To estimate it, we need to analyze the recommended recovery actions of the components in the context of the configuration to determine what actual recovery actions will be taken by AMF at runtime in case of a failure. Based on this actual recovery action, we can estimate the time needed to complete the associated procedures. With this and the failure rates of the components involved, we can estimate service availability. Lastly, we rank the solutions (type stacks) based on their estimated availability, and eliminate those that cannot satisfy the availability requirement (See Figure 5-1).

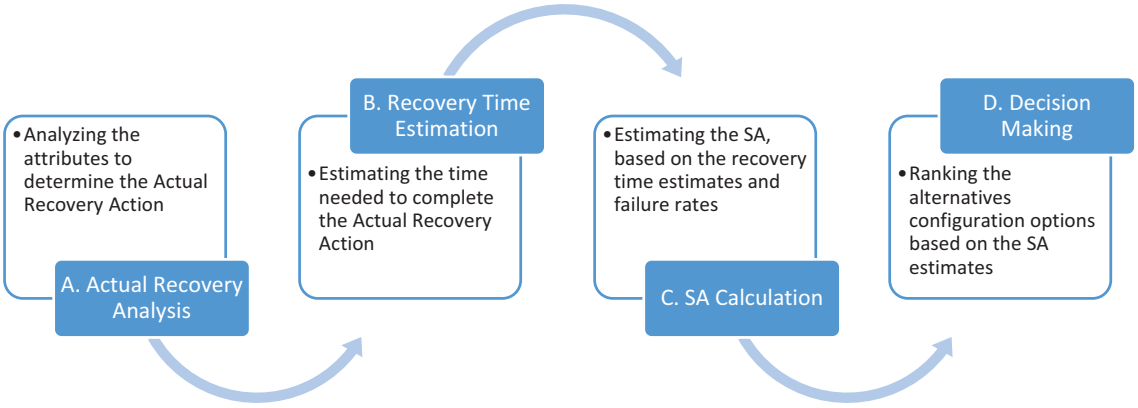

Figure 5-1 The steps of the Availability Estimate-Based Prototype Selection Method

The problem is that we want to apply this process before we have generated the configuration, i.e. when the information is only partial. We need to make assumptions about the configuration based on the ETF prototypes that we have selected in each type stack. Our assumptions are the following:

1. There is at least one component for each CompType that is needed to provide the service,
2. There is always a redundant SU hosted on a redundant node that can take over the service assignment if it is required so,
3. For an SG with the NWay-Active redundancy model, there is more than one assignment for each SI protected by the SG. Hence, we do not consider any outage time for a service protected by this redundancy model.
4. A node reboot will be performed if the instantiation or clean-up of a component fails, which means:
    a. *saAmfNodeFailfastOnInstantiationFailure is set to True*

b. *saAmfNodeFailfastOnTerminationFailure is set to True*

In the following, we provide the details of our availability estimation method.

5.1.1 Actual recovery analysis

As mentioned in Chapter 4, when AMF detects a component failure, it checks the applicable recommended recovery action of the component and it also checks all the other configuration attributes that can alter this recommendation and it determines the actual recovery action it will execute [1]. In this step, we analyze the recovery recommendations for all the CompTypes in the context of their type stack and determine the applicable actual recovery actions.

5.1.2 Recovery time estimation

After determining the actual recovery action for a CompType, we can estimate the time needed to complete this recovery. We need to determine the recovery action's procedures and their timing, in order to estimate the time required for completing the recovery action.

We start by explaining how we estimate the recovery times for the Component Restart, Component Failover, SU Restart and SU Failover. Later, in Section V we explain how the calculation can be applied to other recovery actions (i.e. Node Failfast, Node Failover, Node Switchover, Container Restart, Application Restart and Cluster Reset).

AMF accomplishes all its tasks through the management of the components of the system under its control. Accordingly, all recovery actions except for Node Failfast can and need to be decomposed into recovery actions applicable to components, which are: Component Restart, Component Failover, and Component Switchover. These component-level recovery actions can be further decomposed into a sequence of component life-cycle and API Callback operations. Table 5-1 presents the decomposition of the different component-level recovery actions.

Table 5-1 Decomposition of component-level recovery actions into component life-cycle and API Callback operations

Recovery Action	Operations Decomposition
Component Restart	Clean-up of the (faulty) component + Instantiation of the component + Set the assignment of the component to active
Component Failover	Clean-up of the faulty component + Set the assignment of the standby component to active
Component Switchover	Set the assignment of the active component to quiesced + Set the assignment of the standby component to active

All the operations in the table above are guarded by timers. These timers determine the maximum time that AMF waits before it considers the operation unsuccessful. It follows that, to estimate the recovery time we use the timeout values of these timers. The calculation depends on the procedure these operations apply to a configuration as we explain below. In the following, we present the details of our recovery time estimation for the different actual recovery actions.

❖ **Component Restart:** As illustrated in Figure 5-2 the Component Restart recovery action starts with cleaning up the faulty component. If the cleanup is successful, then AMF tries to re-instantiate the component. Once the component is instantiated successfully, to complete the Component Restart recovery action AMF sets the component's assignments as required to recover the services. As a part of the Component Restart recovery action, AMF attempts the instantiation of the component a number of times without delay (*NIWOD*), and if unsuccessful, it tries to instantiate it with delays between the attempts [6]. In our estimation, the cleanup and instantiation commands have their own success probability. We expect the number of instantiation attempts

without delay and the number of instantiation attempts with delay to be provided as input in the CR to the configuration generation process.

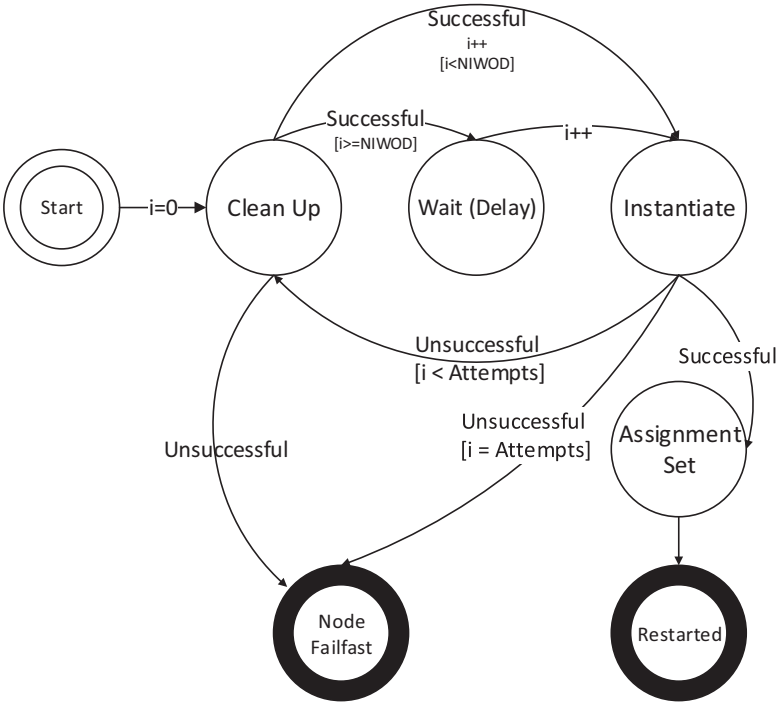

Figure 5-2 The Component Restart recovery action state diagram

A Component Restart recovery action fails if AMF fails to clean up the component or if the instantiation fails after all the allowed attempts. In these cases, (depending on the *saAmfNodeFailfastOnTerminationFailure* and *saAmfNodeFailfastOnInstantiationFailure* attributes of the node) AMF reboots the node hosting the component and all the assignments of the node are failed over to the standbys nodes (i.e. the nodes on which the components with the standby assignments reside).

Based on this procedure of the Component Restart recovery action, we can estimate the recovery time it requires. We take into account all the cases that may occur according to their probabilities. For example, a component can be restarted in the first attempt or, less likely, all of the instantiation attempts may fail. For this reason, our recovery time estimate for the recovery

action is between the time needed for the first successful attempt and the time of all the sequential unsuccessful instantiation attempts.

Figure 5-3 illustrates an example of a Component Restart recovery action. As shown, we allow only for one instantiation attempt without delay and for one with delay. In order to find the average recovery time, we determine the recovery time for each case. The tree in the example has five leaves. The leaves representing Case 2 and Case 4 are where the Component Restart recovery action is successful. The other leaves, Case 1, Case 3 and Case 5, correspond to where the Component Restart recovery action fails and it is escalated to node level. This means that all the assignments of the node should be failed over.

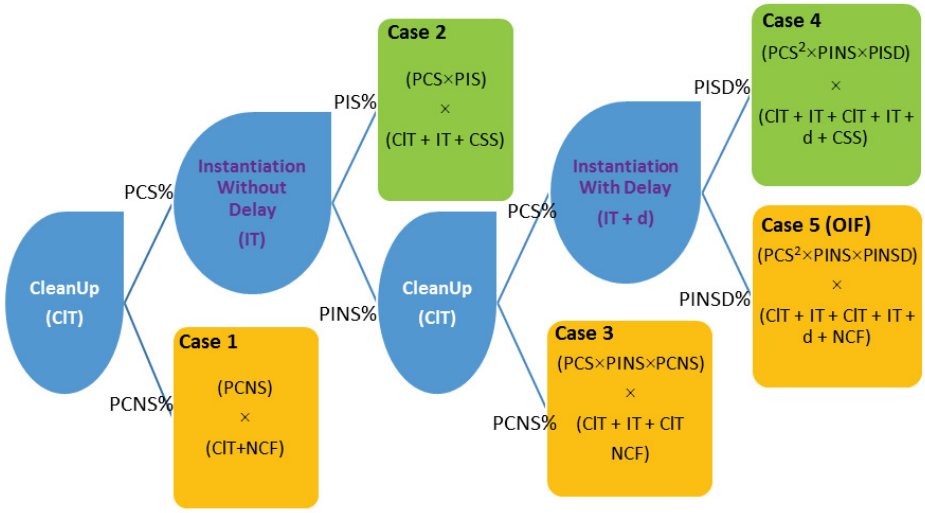

Figure 5-3 Component Restart recovery time estimation example

We can calculate the value of each leaf's probability based on the probabilities of the individual operations involved and the time they take based on their associated timeouts. For instance, the probability of Case 2 is equal to *PCS×PIS,* which represents the probability of a successful cleanup action multiplied by a successful instantiation attempt. Additionally, the time that is needed to complete each case, is the time required for all of the (unsuccessful) consecutive

actions prior to that case, and the time needed for the current action (e.g. *CSS*, *NCF*, *IT*). As also seen in the example, the average recovery time (i.e. outage time) for the Component Restart recovery action is equal to the summation of all of the probable cases.

According to the discussion above, first we need to calculate the *NCF*. The *NCF* represents the time that it takes to do a Node Failfast recovery action. As a consequence of a Node Failfast, the AMF reboots the node by invoking an administrative operation on the node without trying to terminate the components individually. When the node has been shut down for the node reboot, the AMF fails over the CSIs assigned to the components of the active node to the components of the standby node [6]. Moreover, the time that a node takes to reboot (i.e. *NST*) should be given an input (i.e. CR). If the CSIs failover is done in parallel, the NCF can be calculated as shown in Eq. (5-1):

$$NCF = NST + \underset{1 \le j \le N}{\text{Max}}\left(CSS_j\right) \tag{5-1}$$

where *j* varies from one to the number of components (prototypes) of the node (i.e. *N*).

Afterward, we can use Eq. (5-2) to calculate the average outage time due to the component instantiation attempts without delay, as part of the Component Restart recovery action estimation.

$$AOTIWOD = \sum_{i=1}^{NIWOD} \left[(PCS^{i} \times PINS^{i-1} \times PIS) \times \left(\left(i \times (CIT + IT)\right) + CSS\right)\right] + \sum_{i=1}^{NIWOD} \left[(PCS^{i-1} \times PINS^{i-1} \times PCNS) \times \left((i \times CIT) + \left((i-1) \times IT\right) + NCF\right)\right] \tag{5-2}$$

In Eq. (5-2) the term $(PCS^{i} \times PINS^{(i-1)} \times PIS)$ represents the probability of the i^{th} successful case while $(PCS^{i} \times PINS^{(i-1)} \times PCNS)$ represents the i^{th} unsuccessful case. To calculate the outage, these

probabilities are multiplied by the time that the particular case takes, which are the terms (*(i × (ClT + IT)) + CSS)* and *((i × ClT) + ((i-1) × IT) + NCF*).

Similarly, Eq. (5-3) calculates the average outage time due to the component instantiation attempts with delay, as the second part of the Component Restart recovery action estimation. In this equation, the term $PINS^{NIWOD}$ refers to the probability that all instantiation without delay attempts were unsuccessful.

$$AOTIWD = PINS^{NIWOD} \times \left[\sum_{i=NIWOD+1}^{n} \left[\begin{matrix} (PCS^{i} \times PINSD^{(i-NIWOD-1)} \times PISD) \\ \times ((i \times (ClT + IT)) + ((i - NIWOD) \times dly) + CSS) \end{matrix} \right] + \sum_{i=NIWOD+1}^{n} \left[\begin{matrix} (PCS^{i-1} \times PINSD^{(i-NIWOD-1)} \times PCNS) \\ \times \left((i \times ClT) + \left((i-1) \times IT\right) + \left((i - NIWOD - 1) \times dly\right) + NCF\right) \end{matrix} \right] \right] \quad \textbf{(5-3)}$$

As mentioned, it is possible that all the instantiation attempts made by AMF fail. In such a case, the average outage time can be calculated using Eq. (5-4).

$$OIF = PCS^{nia} \times PINS^{NIWOD} \times PINSD^{NIWD} \times [(nia \times (ClT + IT)) + \left(NIWD \times dly\right) + NCF] \quad \textbf{(5-4)}$$

The average outage time caused by the Component Restart recovery action is the sum of the partial time estimates defined by Eq. (5-2), Eq. (5-3) and Eq. (5-4). The estimated average recovery time for the Component Restart is given in Eq. (5-5).

$$MTTR = AOTIWOD + AOTIWD + OIF \quad \textbf{(5-5)}$$

❖ **Component Failover:** The Component Failover recovery action includes two operations, namely the cleaning up of the faulty component and the assignment of its active assignments to its standby components or to spares. Depending on the redundancy model and other configuration

attributes, the other components in the same SU of the faulty component, may be subject to a switchover or a failover of their assignments. The activity diagram of the Component Failover is shown in Figure 5-4.

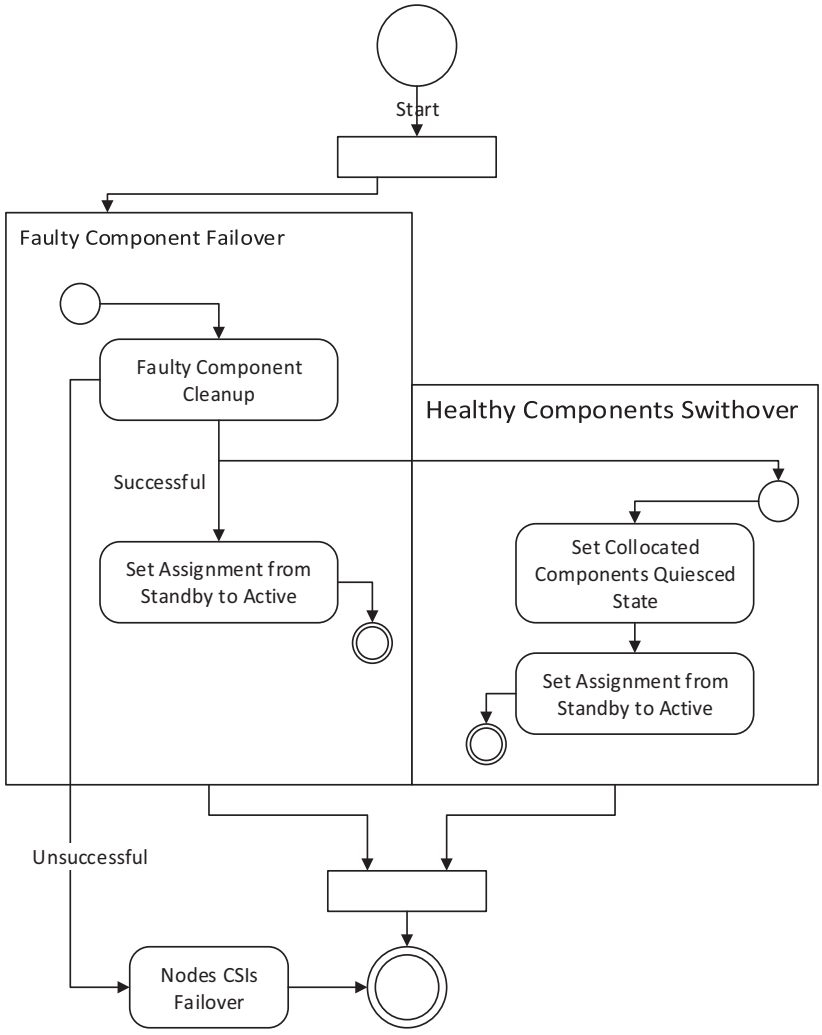

Figure 5-4 The Component Failover recovery action activity diagram

Figure 5-5 illustrates an example of a Component Failover recovery action. As shown, two different cases may occur. While Case 2 represents a successful failover recovery action, Case 1 shows where the failover of the faulty component fails and is escalated to the node level and all the assignments of the node should be failed over.

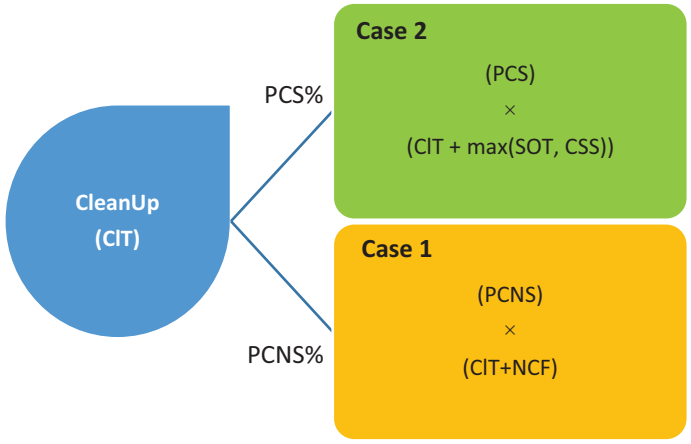

Figure 5-5 A faulty component failover example

As a Component Failover recovery action, AMF tries to clean up the faulty component. If the cleanup action is successful, then it fails over the faulty component's assignments, and at the same time, it switches over the non-faulty components' assignments. The switchover means that the component with the active assignment stops its task and moves to the quiesced state. Then AMF can move the active assignment to the corresponding standby or to a spare component [6]. Let *SOT* represent the time needed to switch over the active assignments of the non-faulty components. If AMF can switch over all components simultaneously (e.g. there is no dependency among components), *SOT* can be calculated as shown in Eq. (5-6):

$$SOT = \underset{1\le j\le N}{\text{Max}}\left(CSS_j\right) + \underset{1\le j\le N}{\text{Max}}\left(CSS_j\right) \quad \textbf{(5-6)}$$

$$\rightarrow SOT = 2 \times \underset{1\le j\le N}{\text{Max}}\left(CSS_j\right)$$

where *j* is iterating through the number of non-faulty components (i.e. *N*) in the SU. The first term in the equation, that is $\underset{1\le j\le N}{Max}(CSS_j)$, corresponds to the time that a component takes to move to the quiesced state, whereas the second term represents the time that it takes to move the active assignment to its standby or spare component.

As mentioned, if the cleanup is successful, the AMF fails over the faulty component's CSI(s), and switches over the non-faulty components (in the same SU) concurrently. We can then calculate the average outage time for a Component Failover recovery action using Eq. (5-7).

$$MTTR = [PCS \times (ClT + Max(SOT, CSS))] + [PCNS \times (ClT + NCF)] \quad \textbf{(5-7)}$$

❖ **SU Restart:** The SU Restart actual recovery action is the first level of escalation of the Component Restart recovery recommendation. The escalation occurs if within the probation time the number of component restarts exceeds the configured threshold. In our estimations we consider the escalation only for the case when the threshold is zero (i.e. *saAMFSGCompRestartMax=0*). For this reason, the first component restart triggers the SU Restart.

Conceptually, the SU Restart effects all of the components inside the SU. However, there are a few constraints to be respected. All the components must be cleaned up simultaneously, and if the cleanup is successful, then AMF simultaneously re-instantiates all of them and sets the same state assignments for all of the components as they had before the failure. The estimation of SU Restart recovery time can be formalized similarly to the Component Restart recovery time estimation. To take into account the constraints above we separate the average time calculation of the cleanup and the instantiation phases. Figure 5-6 illustrates the activity diagram for the SU Restart recovery actions.

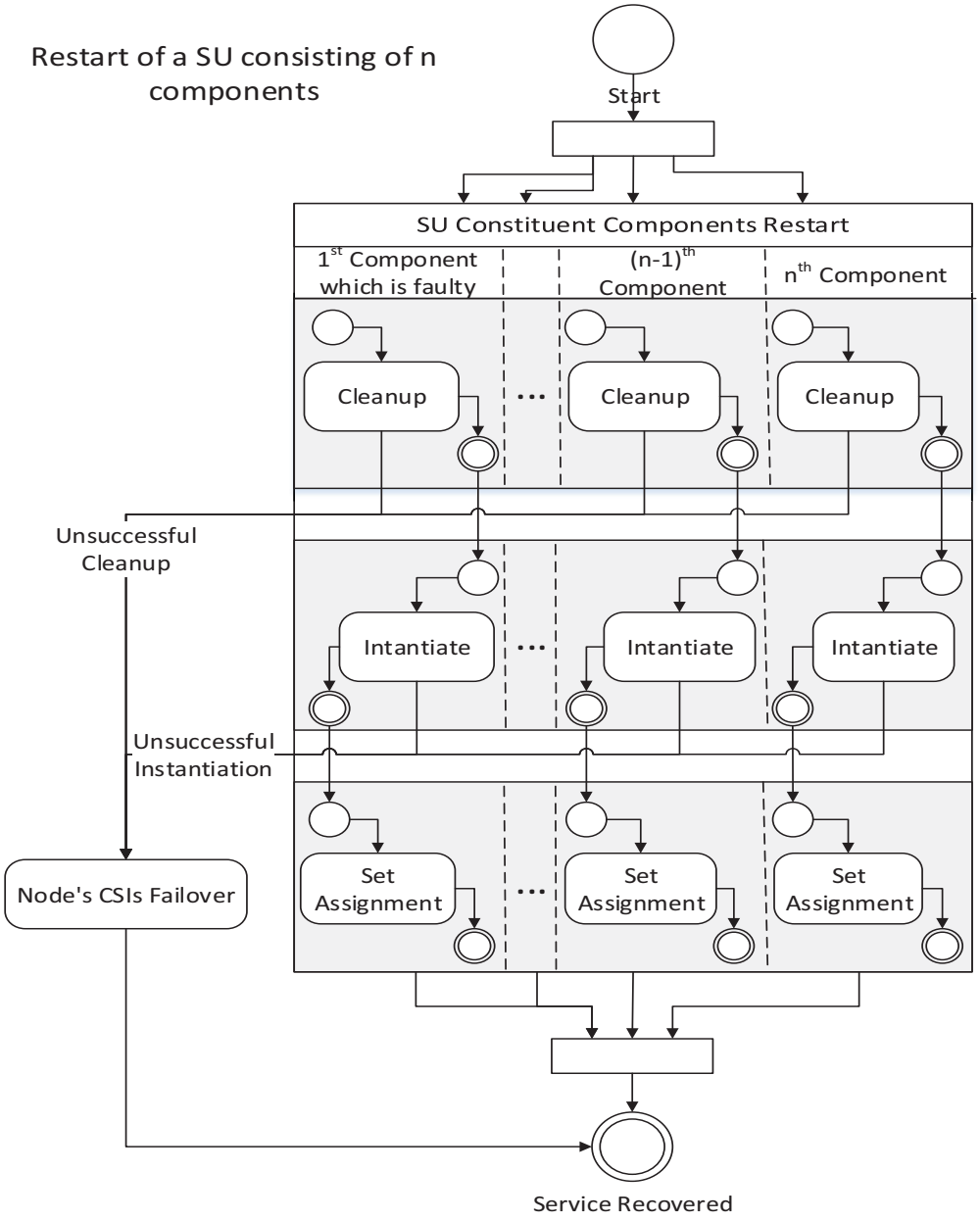

Figure 5-6 The SU Restart recovery action activity diagram

In order to calculate the average recovery time for each of the components within an SU, first we estimate the average time needed to clean up each of the components separately, according to Eq. (5-8).

$$CleanupTime = [PCS \times (ClT)] + [PCNS \times (ClT + NCF)] \quad \textbf{(5-8)}$$

Then, we estimate the instantiation time for a component. In this way, we calculate the average outage time for the instantiation attempts without delay as shown in Eq. (5-9).

$$AOTIWOD = \sum_{i=1}^{NIWOD} \left[(PCS^{i-1} \times PINS^{i-1} \times PIS) \times \left(\left(((i-1) \times ClT) + (i \times IT)\right) + CSS\right)\right] \quad \textbf{(5-9)}$$

$$+ \sum_{i=2}^{NIWOD} [(PCS^{i-2} \times PINS^{i-1} \times PCNS) \times ((i-1) \times ClT) + ((i-1) \times IT) + NCF)]$$

Similarly, we can calculate the average outage time for the instantiation attempts with delay for each component using Eq. (5-10).

$$AOTIWD = PINS^{NIWOD} \times \quad \textbf{(5-10)}$$

$$\left[\sum_{i=NIWOD+1}^{n} \left[\begin{array}{c}\left(PCS^{i-1} \times PINSD^{(i-NIWOD-1)} \times PISD\right) \\ \times \left(\left(((i-1) \times ClT) + (i \times IT)\right) + \left((i-NIWOD) \times dly\right) + CSS\right)\end{array}\right] + \sum_{i=NIWOD+1}^{n} \left[\begin{array}{c}\left(PCS^{i-2} \times PINSD^{(i-NIWOD-1)} \times PCNS\right) \\ \times \left(((i-1) \times ClT) + ((i-1) \times IT) + \left((i-NIWOD-1) \times dly\right) + NCF\right)\end{array}\right]\right]$$

As mentioned earlier, all of the instantiation attempts made by AMF might fail. Eq. (5-11) gives the applicable portion of the outage time.

$$OIF = PCS^{nia} \times PINS^{nia} \times [\left((nia-1) \times ClT\right) + (nia \times IT) + \left(NIWD \times dly\right) + NCF] \quad \textbf{(5-11)}$$

Let *AOT* denote the average outage time for the instantiation phase for a single component. We calculate the *AOT* as the summation of all the portions calculated by the equations (5-9), (5-10) and (5-11). Eq. (5-12) gives the *AOT* calculation.

$$AOT = (AOTIWOD + AOTIWD + OIF) \quad \textbf{(5-12)}$$

Note that the assignment setting time is embedded in the *AOT* calculation. We present the estimation above for one CompType. Separate calculations must be made for all of the CompTypes within an SUType. Then, their maximum should be considered as the SU Restart time, as given in Eq. (5-13), where *j* iterates through components (i.e. *N*) within the SUType.

$$MTTR = \underset{1 \le j \le N}{\text{Max}}(CleaupTime_j) + \underset{1 \le j \le N}{\text{Max}}(PCS_j \times AOT_j) \quad \textbf{(5-13)}$$

Again, we are assuming that there is only one component of each CompType in an SU. As a result, *N* is interpreted as the number of CompTypes.

❖ **SU Failover:** The SU Failover recovery action is either another level of escalation of the Component Restart recovery recommendation or it is the result of a Component Failover recommendation altered by the parent *SUFailOver=True*. The SU Failover actual recovery action performs the following operations: 1) AMF abruptly terminates (cleans up) all the components of the SU, then 2) if successful, it sets the CSI state to active for all the standby or spare components. Figure 5-7 represents an activity diagram for the SU Failover recovery action.

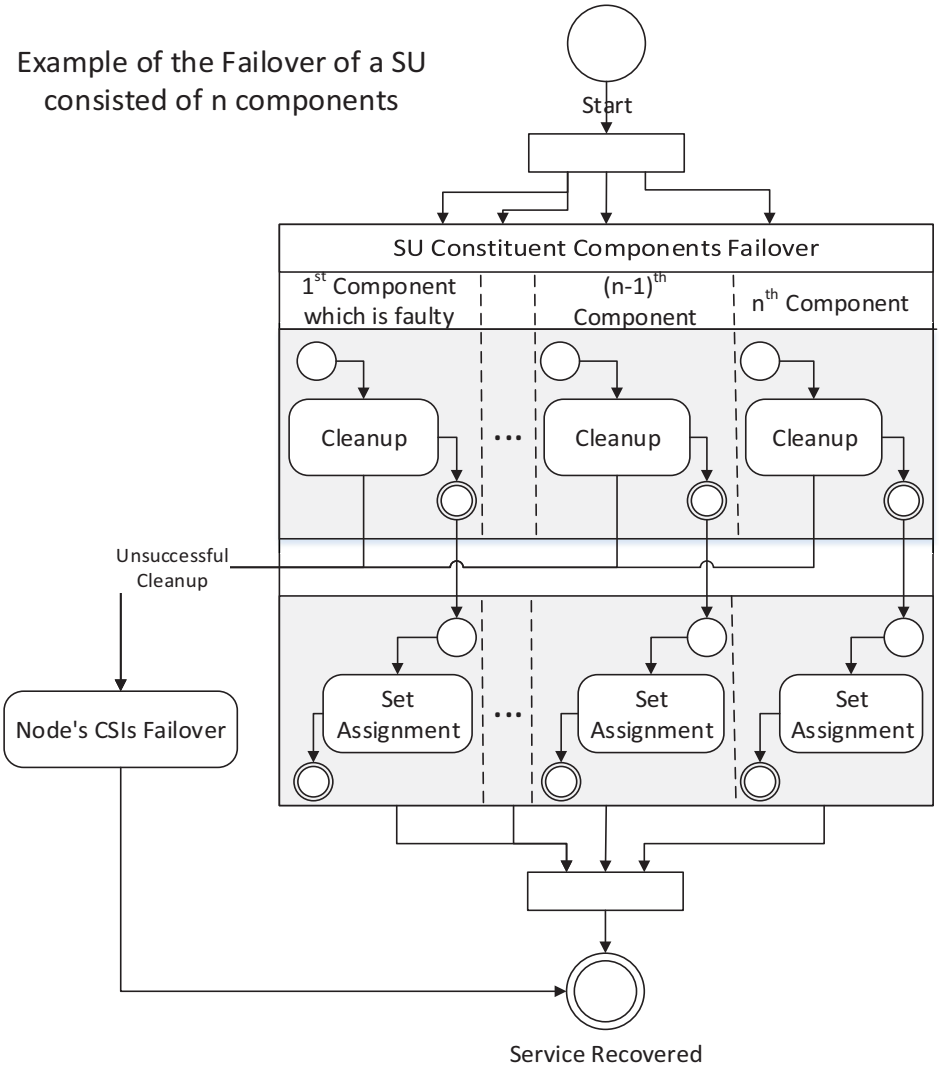

Figure 5-7 The SU Failover recovery action activity diagram

In order to estimate the recovery time needed for the SU Failover recovery action, we need to calculate the cleanup time needed for every component (prototype), using Eq. (5-8) from each of them. Since the actions are performed concurrently, the maximum of them should be used, for further calculations.

Once we calculate the cleanup time, we need to calculate the time needed to set the state of the CSIs of the standby or spare components to active (i.e. *CSS*). Similar to the cleanup action,

the assignment action is also done in parallel. We can estimate an SU Failover recovery time using Eq. (5-14):

$$MTTR = \underset{1 \le j \le N}{\text{Max}}(CleaupTime_j) + \underset{1 \le j \le N}{\text{Max}}(PCS_j \times CSS_j) \quad \textbf{(5-14)}$$

where *j* iterates through the CompTypes in the SUType, namely *N*, and the *CleanupTime* is a single component cleanup time calculated by Eq. (5-8).

❖ **Remaining Recovery Actions:** We have presented only the component and SU level recovery actions so far. The remaining recovery actions are: Node Failfast, Node Failover, Node Switchover and Application Restart.

With respect to the node-level recovery actions, our rationale is as follows: in case of a Node Failfast recovery action, AMF reboots the node and fails over all the SIs assigned to it to the nodes hosting the standby components, or spares. This is the same action used for the cleanup failure and the recovery time is equal to that of the *NCF*. The Node Failover recovery action is defined as the cleanup of all the components on the node, before failing over all the SIs assigned. We estimate the recovery time in the same way we do it for SU Failover, but for all SU (types) hosted on a node. In case of Node Switchover, AMF fails over the faulty component, but switches over the others that are hosted on the same node. For this reason, we estimate the recovery time using a formula similar to the one used to estimate the Component Failover.

The Application Restart recovery action is executed by terminating the whole application and starting it again. This cleans up all of the components first and then instantiates them again. Essentially, this is equivalent to the SU Restart recovery action performed simultaneously on all SUs of the application and coordinated across the cluster.

During the Cluster Reset recovery action, AMF reboots all the nodes that are part of the cluster without trying to terminate the components individually [6]. Therefore, we can consider the cluster reset time to be the nodes shutdown time (i.e. *NST*) added to their startup time (i.e. *CSTO*). The startup time is the maximum time that the AMF should wait, to ensure all required SUs are instantiated before assigning SIs to SUs [6]. The system designer gives this startup timeout as an input. We can calculate the Cluster Reset time using Eq. (5-15):

$$MTTR = \underset{1 \le j \le N}{\text{Max}}(NSH_j) + CSTO \tag{5-15}$$

where j iterates through the number of nodes in the cluster.

5.1.3 Determining Service Availability and Decision Making

In the previous subsections, we showed how to estimate the service recovery time (i.e. MTTR) after a component failure. Furthermore, we assumed that a component has its own failure rate (i.e. MTTF). Hence, by using the recovery time estimate and the failure rate of a component, we can estimate the availability of the CSIs that the component provides. Since we are interested in the availability of a service, we need to combine service availability of all the components that participate in providing that service.

In order to calculate the availability of the service provided by several components, we need to multiply the availability of the components that are participating in the provisioning of the service [24][25]. The rationale for this multiplication is that the whole service instance is available only if all of its CSIs are provided by some components and are therefore available. This also means that if two components derived from the same CompType are required, their availability is less than if only one was required. In other words, taking into account one component for each

CompType provides a higher estimate than the applicable and it is safe to eliminate type stacks that cannot provide the requested availability even by such an estimate.

Therefore, we estimate the availability of the SIs derived from a service type as given in Eq. (5-16),

$$STSA = \prod_{j=1}^{j \leq NP} \frac{MTTF_j}{MTTF_j + MTTR_j} \quad \textbf{(5-16)}$$

where *MTTFj* and *MTTRj* represent respectively the failure rate of *CTj* and the estimated time for the actual recovery action applicable in case of the failure of a component derived from *CTj*. *NP* is the number of CTs in the type stack that participate in providing the service type.

The calculations that we have shown are for estimating the availability of a particular service type. These calculations should be applied for all the service types defined for the application. Then, only the type stacks that can satisfy the requested service availability for all the service types will be passed to the next step of the generation process. Moreover, it is possible that none of the type stacks can satisfy the requested service availability. In such a case, we select the one with the higher level of availability, even though it cannot meet the availability requirement.

5.1.4 Discussion

So far, we have explained how to estimate service availability for the components that are independent. However, we need to consider the dependencies among the components (prototypes), as well. In the following, we discuss the dependencies among CompTypes. However, we do not handle all of them completely, since it is outside the scope of this thesis.

Instantiation-Level dependency: One of the complexities of estimating service availability is the instantiation-level dependency among the components of the same SU.

According to this dependency, the instantiation/termination of a component is a prerequisite for the instantiation/termination of another component [6]. This type of dependency is applicable only when instantiating or terminating an SU [6]. In the following, we explain this type of dependency and the way it can be handled in the availability estimation.

The instantiation-level dependency indicates the following:

- Within an SU, the AMF instantiates all components with the same instantiation level in parallel. Moreover, the AMF instantiates the components of a given instantiation level only when all components with a lower instantiation level have been instantiated successfully [6].
- Within an SU, the AMF terminates all components with the same instantiation level in parallel. Furthermore, the AMF terminates the components of a given instantiation level only when all components with a higher instantiation level have been terminated [6].

As mentioned, the instantiation-level dependency is only applicable during the instantiation and termination of an SU. Therefore, it is applicable for the SU Restart recovery action [6].

According to the discussion above, if there is an instantiation-level dependency among the CompTypes of an SUType, we need to use Eq. (5-17) instead of Eq. (5-13) to calculate the time of an SU Restart recovery action. Let M stand for the highest level of dependency among the CompTypes of the SUType. We can calculate the SU Restart recovery time as follows:

$$MTTR = \sum_{i=1}^{M} \underset{1 \le j \le N}{\text{Max}} (CleaupTime_{ij}) + \sum_{i=1}^{M} \underset{1 \le j \le N}{\text{Max}} (PCS_{ij} \times AOT_{ij}) \quad \textbf{(5-17)}$$

where i varies from one to the M, and j iterates through the number of CompTypes that have the same instantiation level in the SUType. For instance, the term $\underset{1 \le j \le N}{\text{Max}} (CleaupTime_{ij})$ in the equation above represents the maximum time needed to clean up the CompTypes of the i^{th} instantiation level.

Proxy-Proxied dependency: According to this type of dependency, if the proxy component fails, the AMF should find another proxy component to take over the proxying work [6]. In the meanwhile, the proxied components can continue providing services. Only if the proxy component and its proxied components fail at the same time, the proxied components fail to provide service [6]. Thus, we do not consider any additional calculations for this type of dependency, since the failure of the first proxy component does not indicate the failure of the other proxied component(s) [6].

CSI-CSI dependency: Due to the CSI-CSI dependency, the components will be assigned in a defined order. In our calculations, we are not handling this type of dependency, since we have assumed that the CSI assignments are done in parallel. However, if there is a CSI-CSI dependency, the CSIs assignment are done sequentially. This means that first the sponsor CSIs are assigned to the component and then their dependent CSIs. In such a case, the *CSS* of the CompType providing the dependent CSI should be increased so that the *CSS* of the CompType providing the sponsor CSI is included. The same logic applies for the CSIs removal, meaning that the sponsor CSI is removed from a component only when its dependent CSIs are removed from the assigned component(s) [6].

SI-SI dependency: An SI can be configured to depend on other SIs, in the sense that an SU can only be assigned as active for the dependent SI if all of its sponsor SIs are assigned [6]. The AMF defines the tolerance time as a configurable attribute of a dependency between SIs [6]. For example, if SI1 depends on SI2, the tolerance time indicates for how long SI1 can tolerate SI2 being in the unassigned state [6]. If this time passes before SI2 becomes assigned again, the AMF will remove the assignments of SI1 from the SU [6]. Accordingly, our calculations will be impacted in the following way: if the MTTR of the sponsor service is more than the tolerance time of the dependent service, the availability of the dependent service will be reduced by the availability of its sponsor. Thus, the estimation of the availability of the dependent service should be reduced, as shown in Algorithm 5-1.

```
CalculateTheSAOfDependentServiceType ()
Begin
   sponsorServiceMTTR = the MTTR that is calculated for the sponsor service
   if (sponsorServiceMTTR > tolerance time of the dependent service) then
        dependentServiceSA = the SA that is calculated for the dependent service
        sponsorServiceSA = the SA that is calculated for the sponsor service
        NewDependentServiceSA= dependentServiceSA * SponsorServiceSA
        dependentServiceSA = NewDependentServiceSA
   End if
End
```

Algorithm 5-1 An algorithm to calculate the availability in case of SI-SI dependency

5.1.5 An application of the availability estimate-based prototype selection method

Let us consider the example shown in Figure 5-8. As can be seen, through the CR, the system designer asks for an application that needs to provide one service type, namely "SvcTypeX". As the first step of the generation process, the configuration generator then selects the ETF prototypes that can satisfy the CR. It arranges the selects prototypes into two type stacks: TypeStack1 and TypeStack2. While in TypeStack1 the generator selected CompType1 and CompType2 together to deliver "SvcTypeX", in TypeStack2 it only selects CompType3 to provide the same service.

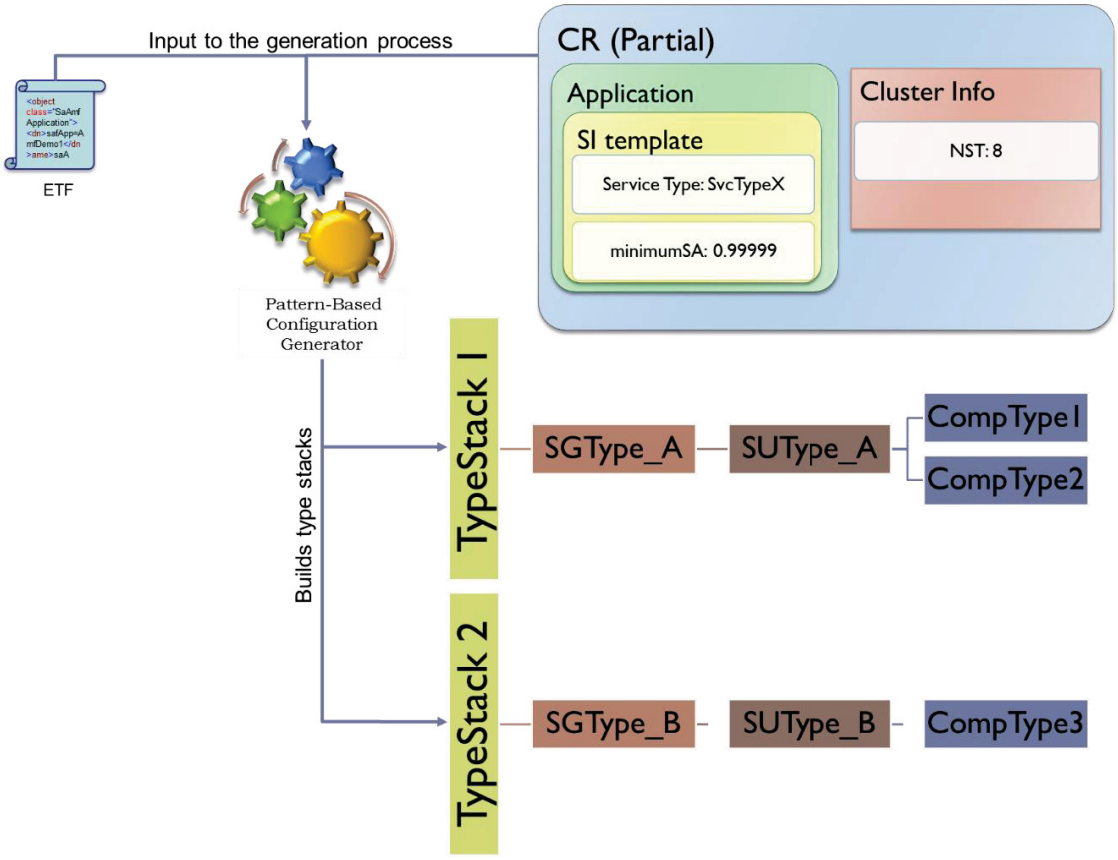

Figure 5-8 Selection of type stacks example

In Table 5-2, we present the attributes of the prototypes in the type stacks and then we show an application for our method.

Table 5-2 An example of type stacks and their attributes

Attributes	TypeStack1		TypeStack2
	SGType_A		SGType_B
	SUType_A		SUType_B
	CompType1	CompType2	CompType3
ClT (Sec)	3	4	3
IT (Sec)	1	2	2
CSS	3	1	2
PCS	0.7	0.8	0.9
PCNS	0.3	0.2	0.1
PIS	0.9	0.9	0.8
PINS	0.1	0.1	0.2
PISD	0.9	0.8	0.9
PINSD	0.1	0.2	0.1
NIWOD	2	2	2
NIWD	1	1	1
nia	3	3	3
dly	2	2	2
Failure Rate (Sec)	530000	920000	870000
Actual Recovery Action	Component Restart	Component Failover	SU Failover

In the table above, we have assumed that we have the result of the actual recovery analysis for each of the CompTypes. As a result, we only need to estimate their recovery time. To do so, we need at first to calculate the *NCF* that is the time the node takes to be shutdown, added to the timeout to fail over the CSIs of the components on the node. This means that the *NCF* for

TypeStack1 is the summation of the *NST* and the maximum of *CSS* of the CompType1 and CompType2. We use Eq. (5-1) to calculate the *NCF* for TypeStack1:

$$NCF = NST + \underset{1 \le j \le N}{\text{Max}}(CSS) = 8 + Max\,(3,1) = 11$$

Similarly, we calculate *NCF* value for TypeStack2 as:

$$NCF = NST + \underset{1 \le j \le N}{\text{Max}}(CSS) = 8 + Max\,(2) = 10$$

After, we continue to estimate the recovery time for the CompTypes.

We can calculate the recovery time for CompType1, for which the actual recovery action is Component Restart, by using equations (5-2) to (5-5). By using Eq. (5-2), we can calculate the average outage time for the portion of component instantiation without delay:

$$AOTIWOD = \sum_{i=1}^{NIWOD} \left[(PCS^{i} \times PINS^{i-1} \times PIS) \times \left((i \times (ClT + IT)) + CSS\right)\right]$$

$$+ \sum_{i=1}^{NIWOD} \left[(PCS^{i-1} \times PINS^{i-1} \times PCNS) \times \left((i \times ClT) + \left((i-1) \times IT\right) + NCF\right)\right]$$

$$= \sum_{i=1}^{2} \left[(0.7^{i} \times 0.1^{i-1} \times 0.9) \times \left((i \times (3+1)) + 3\right)\right]$$

$$+ \sum_{i=1}^{2} \left[(0.7^{i-1} \times 0.1^{i-1} \times 0.3) \times \left((i \times 3) + \left((i-1) \times 1\right) + 11\right)\right] = 9.473100$$

As the second part of the Component Restart recovery action estimation, we use Eq. (5-3) to calculate the average outage time due to the component instantiation attempts with delay.

$$AOTIWD = PINS^{NIWOD} \times$$

$$\left[\begin{array}{c}\sum_{i=NIWOD+1}^{n}\left[\begin{array}{c}\left(PCS^{i}\times PINSD^{(i-NIWOD-1)}\times PISD\right)\\ \times\left((i\times(ClT+IT))+((i-NIWOD)\times dly)+CSS\right)\end{array}\right]\\ +\sum_{i=NIWOD+1}^{n}\left[\begin{array}{c}\left(PCS^{i-1}\times PINSD^{(i-NIWOD-1)}\times PCNS\right)\\ \times\left((i\times ClT)+\left((i-1)\times IT\right)+\left((i-NIWOD-1)\times dly\right)+NCF\right)\end{array}\right]\end{array}\right]$$

$$= 0.1^{2}\times$$

$$\left[\begin{array}{c}\sum_{i=2+1}^{3}\left[\begin{array}{c}\left(0.7^{i}\times 0.1^{(i-2-1)}\times 0.9\right)\\ \times\left((i\times(3+1))+((i-2)\times 2)+3\right)\end{array}\right]\\ +\sum_{i=2+1}^{3}\left[\begin{array}{c}\left(0.7^{i-1}\times 0.1^{(i-2-1)}\times 0.3\right)\\ \times\left((i\times 3)+\left((i-1)\times 1\right)+\left((i-2-1)\times 2\right)+11\right)\end{array}\right]\end{array}\right]= 0.084819$$

To calculate the outage time for the case where all the instantiation attempts made by AMF fail, we use Eq. (5-4):

$$OIF = PCS^{nia}\times PINS^{NIWOD}\times PINSD^{NIWD}\times[(nia\times(ClT+IT))+\left(NIWD\times dly\right)+NCF]$$

$$= 0.7^{n}\times 0.1^{2}\times 0.1^{1}\times\left[\left(3\times(3+1)\right)+\left(1\times 2\right)+11\right]= 0.008575$$

Finally, to calculate the average outage time caused by the Component Restart recovery action we use the summation of the calculations above, as given in Eq. (5-5):

$$MTTR = AOTIWOD + AOTIWD + OIF = 6.9051 + 0.073059 + 0.005831 = 9.566494$$

Regarding CompType2, which has the actual recovery action of Component Failover, we estimate the recovery time as follows. First, we use the Eq. (5-6) to calculate the time needed to switch over other components (i.e. the component of the prototype CompType1) in the SU (prototype). The value of *N* is equal to 1, because it represents the number of non-faulty components in the SU.

$$SOT = \underset{1\le j\le N}{2\times\text{Max}}(CSS_{j}) = 2\times 3 = 6$$

Then, we use the Eq. (5-7) to calculate the average outage time for the faulty component's failover action (i.e. the component of the prototype CompType2):

$$MTTR = [PCS \times (ClT + Max(SOT, CSS))] + [PCNS \times (ClT + NCF)]$$

$$= [0.8 \times (4 + Max(6,2))] + [0.2 \times (4 + 11)] = 11$$

So far, we have shown the calculations for TypeStack1. TypeStack2 has only one CompType, namely CompType3, which has an SU Failover as its actual recovery action. We use Eq. (5-8) to calculate the cleanup time needed for a single component (prototype):

$$CleanupTime = [PCS \times (ClT)] + [PCNS \times (ClT + NCF)] = [0.9 \times (3)] + [0.1 \times (3 + 10)] = 5.7$$

Afterward, in order to estimate the recovery time of the CompType3, we use Eq. (5-14), where *N* denotes the number of components in the same SU as faulty component exists.

$$MTTR = \underset{1 \le j \le N}{\text{Max}}(CleaupTime_j) + \underset{1 \le j \le N}{\text{Max}}(PCS_j \times CSS_j) = (5.7) + (0.9 \times 2) = 7.5$$

Now that we have estimated the recovery times, we can go to the next step and calculate the availability of "SvcTypeX", based on each of the type stacks. Regarding TypeStack1 in which the two CompTypes (i.e. CompType1 and CompType2) are participating to deliver the service, we can calculate service availability by using Eq. (5-16).

$$STSA = \prod_{j=1}^{j \le NP} \frac{MTTF_j}{MTTF_j + MTTR_j} =$$

$$= \prod_{j=1}^{j \le 2} \frac{MTTF_j}{MTTF_j + MTTR_j} = \frac{530000}{530000 + 9.566494} \times \frac{920000}{920000 + 11} \cong 0.999982 \times 0.999988 \cong 0.999969$$

Similarly, we calculate the achievable service availability based on TypeStack2 as follows:

$$STSA = \prod_{j=1}^{j \le NP} \frac{MTTF_j}{MTTF_j + MTTR_j} =$$

$$= \prod_{j=1}^{j\leq 1} \frac{MTTF_j}{MTTF_j + MTTR_j} = \frac{680000}{680000 + 7.5} \cong 0.9999914$$

As shown in Figure 5-8, the minimum availability required for the service (type) is 0.99999. In this example, for TypeStack1, both CompType1 and CompType2 participate in delivering the service (type). As a result, we calculate the service availability they provide as the product of the availabilities of CompType1 and CompType2, which results in 0.999969. Applying the same formula to TypeStack2, the service availability delivered by CompType3 is only 0.999991. This shows that, in this example, we need to consider further only TypeStack2 since it is the only type stack that can satisfy the required service availability.

5.2 Availability Estimate-Based Entities Creation Method

In the current configuration generation process [1], the system designer is responsible for specifying the number of SUs and SGs as an input to protect an SI template. Then, the configuration generator takes into account the components' capability and calculates the number of components that need to be placed in each SU to provide the SI template. The issue here is that the system designer might not have enough information to specify the number of entities appropriately. On the other hand, the number of entities plays a very important role in the configuration in terms of service availability. More specifically, based on the number of components that provide service, service availability may increase or decrease. We propose the availability estimate-based entities creation method, which is applicable to the third step of the configuration generation process, namely the creation of AMF entities. The method aims at calculating the number of components, SUs and SGs that are no longer provided as input in the CR. This method calculates the number of entities so that the configuration meets the availability requirement. The assumptions that we make in our calculations are as follows:

- Only one CT in an SU type can provide a particular CST.
- The different service types can be provided by different sets of CTs. This is because we apply the separation of CSTs configuration design pattern prior to this step.

As mentioned, the method automatically calculates the number of SGs to protect an SI template and the number of SUs in each SG. It also calculates the number of components in each SU with respect to the requested level of service availability. Since each redundancy model has its own characteristics, the calculations may differ for each redundancy model[1]. In our calculations, we put the constraint that the maximum number of SUs for each SG should not exceed the number of nodes. We put this constraint to prevent hosting more than one SU of an SG on the same node.

5.2.1 Calculating the number of components

The number of components of a given CT in an SU should be calculated with respect to the requested availability of the SIs. In the previous section, we have shown the availability estimation of a service (type) assuming that there is only one component of each CompType. Now we show how far we can increase the number of components of each type (that is now AMF CT), with respect to the required level of availability.

We know that to estimate the availability of a service, we should multiply the deliverable availability of the components that are providing the service. We also know that, due to the multiplication of the components' deliverable availability, the larger the number of components, the lower the service availability. Therefore, to calculate the number of components in an SU, we should not increase the number of components to the point that the multiplication of the

[1] Note that since the redundancy model selection configuration design pattern is done prior to this method, the redundancy model of the SG type is selected at this stage.

components' deliverable availability becomes lower than the requested service availability. To reach this goal, we put the minimum number of components of each CT in an SU, and then we iteratively increase the number of components, unless the availability requirement is not satisfied.

The minimum number of components of a CT that should be put in an SU depends on the capability of the CT. Algorithm 5-2 defines the steps to calculate the minimum number of components of the CTs in an SU. We can then ensure that the SU will be capable of providing one SI. The algorithm finds the CT that can provide a particular CST of the service (type). Then, it finds the number of CSIs of the CST in one SI (i.e. *NoOfCSIsPerCST*). After, based on the CT capability, it calculates the number of components of the CT that are needed to provide the *NoOfCSIsPerCST*.

```
CalculateMinimumNumberOfCompsPerCT (ServiceType svc)
Begin
  For each CT:ct that participates to provide the svc do
      MinimumNoOfCompsPerCT= 1
      For each CST in ct do
          tempNumberOfCompsPerCST= Max ( Ceil (NoOfCSIsPerCST / ActiveCapabilityPerCST) and
                                         Ceil (NoOfCSIsPerCST / StandbyCapabilityPerCST) )
          if( tempNumberOfCompsPerCST > MinimumNoOfCompsPerCT) then
              MinimumNoOfCompsPerCT = tempNumberOfCompsPerCST
          End if
      End do // each CST
      ct.MinNumberOfComponents = MinimumNoOfCompsPerCT
  End do // each CT:ct
End
```

Algorithm 5-2 An algorithm to find the minimum number of components of a CT in an SU

We also need to calculate the maximum number of components of each CT that should be put in an SU. Algorithm 5-3 defines the steps to calculate the maximum number of components of the CTs. The maximum number of components of the CTs guarantees that the parent SU will be capable of providing all the SIs of the SI template (i.e. *NoOfSIs)*. According to the algorithm, at first we find the CT that can provide a particular CST of the service (type). Then, we find the number of CSIs of the CST in all SIs (i.e. *NoOfSIs* ×*NoOfCSIsPerCST*). Next, based on the CT

capability, we calculate the number of components of the CT that are needed to provide *NoOfSIs* ×*NoOfCSIsPerCST*.

```
CalculateMaximumNumberOfCompsPerCT (ServiceType svc)
Begin
  For each CT:ct that participates to provide the svc do
      MaximumNoOfCompsPerCT= 0
      For each CST in ct do
          If ( redundancy model is No Redundancy) then
              MaxSIsNo = 1          // an SU can provide at most one SI in no-redundancy model
          else
              MaxSIsNo = NoOfSIs in the SI template
          End if

          tempNumberOfCompsPerCST=Max(Ceil(MaxSIsNo*NoOfCSIsPerCST/ActiveCapabilityPerCST) and
                                      Ceil (MaxSIsNo*NoOfCSIsPerCST/StandbyCapabilityPerCST) )

          if( tempNumberOfCompsPerCST > MaximumNoOfCompsPerCT) then
              MaximumNoOfCompsPerCT = tempNumberOfCompsPerCST
          End if

      End do // each CST
      ct.MaxNumberOfComponents = MaximumNoOfCompsPerCT
  End do // each CT:C
End
```

Algorithm 5-3 An algorithm to find the maximum number of components of a CT in an SU

The actual number of components (of a CT) that we are targeting is between the minimum and maximum number of components. We use Algorithm 5-4 to calculate the actual number of components of the CTs in an SU, taking into account the requested service availability.

```
FindNumberOfCompsPerCT ()
Begin
  M = number of CTs
  // we find the proportion that the components of each CT needs to be put together, and Set the Initial value
     of NumberOfCompsPerCT to the MinimumNoOfCompsPerCT
  NumberOfComponentsPerCT[1: M] = CalculateMinimumNumberOfCompsPerCT ()
  i = 1
  do {
       NumberOfComponentsPerCT[1: M] = NumberOfComponentsPerCT[1: M] * i
       //number of components of a CT should not exceed the maximum number of components of the
         CT specified in ETF or by CalculateMaximumNumberOfCompsPerCT()
       If ( the number of components for none of the CTs exceeds their upper bound) then
       {

            SA = Calculate the service availability by multiplying the availability of all of the components
            If ( SA is more than the requested SA) then
            {
                 i++
                 continue;
            }
            Else // roll back the i value to its previous value and return the number of components
            {
                 i--
                 NumberOfComponentsPerCT[1: M] = NumberOfComponentsPerCT[1: M] * i
                 return NumberOfComponentsPerCT[1: M]
            }
            End if

       }
       Else // roll back the i value to its previous value and return the number of components
       {
            i--
            NumberOfComponentsPerCT[1: M] = NumberOfComponentsPerCT[1: M] * i
            return NumberOfComponentsPerCT[1: M]
       }
       End if

  } while ( SA >= CR.SA) // while calculated SA is higher than the availability requested in the CR
End
```

Algorithm 5-4 An algorithm to find the number of components of a CT in an SU

The logic behind the algorithm above is to first put the minimum number of components of each CT in an SU. This number also represents the proportion of the number of components of the CTs in the SU. Therefore, the number of components of a CT is always a coefficient of its minimum number. Afterward, we iteratively increase the number of components based on their proportion. For example, consider that the minimum and maximum number of components of a CT, namely “CT1”, are calculated as 4 and 16 respectively. In this case, the number of components

of "CT1" can be 4, 8, 12 or 16. In each iteration, we increase the number of components based on their proportion, and then we calculate the achievable service availability as shown in Eq. (5-18):

$$SISA = \prod_{j=1}^{j \leq N} \frac{MTTF_j}{MTTF_j + MTTR_j} \quad \textbf{(5-18)}$$

where N is the total number of components (i.e. the summation of the number of components of all the CTs) that participate in providing the service. In each step, if the calculated service availability ($SISA$) satisfies the availability requirement, we iterate again by increasing the number of components. We do this iteration multiple times, unless:

- The $SISA$ becomes lower than the requested service availability,
- Or the number of components of any of the CTs reaches its maximum number, which is determined by Algorithm 5-3.

5.2.2 Calculating the number of SUs and SGs

The calculation of the number of SGs and SUs that we propose are per CST. We apply separate calculations for each CST of the service type that the SG type protects. Then we choose the one that resulted in a greater number of SGs. If there is more than one CST by which the calculations result in the same number of SGs, we select the one that has the greater number of SUs. For example, if there is a service type that has three CSTs, we should use the calculations below for the three CSTs separately. Then we select the one that resulted in a greater number of SGs. Algorithm 5-5 defines the steps to calculate the number of SUs and SGs.

```
CalculateNumberOfSUsAndSGs ()
Begin

   FindNumberOfCompsPerCT() // uses Algorithm 5-4 to find the number of components of each CT in the SU

   NoOfSGs = 0
   NoOfSUs = 0

   For each CST:cst that is in the services (SI templates) that the SG type protects do

      NoOfCompsPerCST= find the CT that can provide the cst, and use the number of components of the
                       the CT, that has been calculated by FindNumberOfCompsPerCT()

      tempNoOfSGs = for the cst, calculate the number of SGs by using the equations (5-19) to (5-38)
      tempNoOfSUs = for the cst, calculate the number of SUs by using the equations (5-19) to (5-38)

      If ( tempNoOfSGs > NoOfSGs ) then
      {
          NoOfSGs = tempNoOfSGs
          NoOfSUs = tempNoOfSUs
      }
      Else if ( tempNoOfSGs == NoOfSGs ) then
      {
          If ( tempNoOfSUs > NoOfSUs ) then
              NoOfSUs = tempNoOfSUs
          End if
      }
      End if

   End do // each CST:cst

   Return NoOfSGs and NoOfSUs

End
```

Algorithm 5-5 An algorithm to find the number of SUs and SGs for an SG type

As mentioned, the calculations of the number of SUs and SGs for an SG type mostly depend on the redundancy model of the SG type. In the following, we explain the calculations for each redundancy model separately. Again, it is important to mention that we are making the constraint that no more than one SU of an SG can reside on the same node. It means that the maximum number of SUs within an SG should not exceed the number of nodes.

❖ **No-Redundancy redundancy model:** In this redundancy model, each SU can be assigned as active for, at most, one SI. For this reason, to handle all the SIs of an SI template, the number of SUs in an SG should be equal to the number of SIs. Due to our constraint on the maximum number of SUs in an SG, which should never exceed the number of nodes, if the number of SIs,

which can be the number of SUs, exceeds the number of nodes, we need to distribute the SIs into more SGs. We calculate the number of SGs as shown in Eq. (5-19):

$$NumberOfSGs = Ceil\ (\frac{NoOfSIs}{NoOfNodes - 1}) \quad \textbf{(5-19)}$$

In the equation above, we divide the total number of SIs to *NoOfNodes-1*. This is because we are reserving one node to put a spare SU on it later. Once we have calculated the number of SGs, we can calculate the number of SUs, including one spare SU, for each SG as shown in Eq. (5-20).

$$NumberOfSUsPerOneSG = Ceil(\frac{NoOfSIs}{NumberOfSGs}) + 1 \quad \textbf{(5-20)}$$

❖ **2N redundancy model:** In this redundancy model, we have at most one SU for all the SIs' active assignments and one SU to handle all the SIs' standby assignments. Hence, the number of active and standby SUs in each SG is equal to two. However, depending on other criteria, such as the number of components in an SU, one SG might not be able to protect all of the SIs of the SI template. To solve this problem, we can add more SGs to protect all of the SIs. To find the number of SGs needed, we at first need to find the maximum number of SIs that an active SU can handle, as shown in Eq. (5-21):

$$MaxNoOfSIsPerOneActSU = floor\left(\frac{NoOfCompsPerCST \times ActiveCapabilityPerCST}{NoOfCSIsPerCST}\right) \quad \textbf{(5-21)}$$

In the equation above, we use "*floor*" because the number of SIs that can be assigned to an SU has to be an integer. Once we have found the maximum number of SIs that an active SU can handle, we can find the maximum number of SIs that a standby SU is capable of handling, as shown in Eq. (5-22).

$$MaxNoOfSIsPerOneStdSU = floor\left(\frac{NoOfCompsPerCST \times StandbyCapabilityPerCST}{NoOfCSIsPerCST}\right) \quad \textbf{(5-22)}$$

The number of SIs that an SG can handle is the minimum of *MaxNoOfSIsPerOneActSU* and *MaxNoOfSIsPerOneStdSU*. For example, assume that in a 2N redundancy model, the active SU can handle five SIs and the standby SU can handle six SIs. In such a case, the number of SIs that can be handled by the SG is equal to five. We then calculate the number of SGs that are needed to handle all the SIs of the SI template, using Eq. (5-23) :

$$NumberOfSGs = Ceil\left(\frac{NoOfSIs}{\min(MaxNoOfSIsPerOneActSU, MaxNoOfSIsPerOneStdSU)}\right) \quad \textbf{(5-23)}$$

❖ **N+M redundancy model:** In this redundancy model, there are a total number of *N* active SUs and *M* standby SUs to handle the SIs protected by the SG (i.e. *N*+*M* SUs).

In order to find the total number of SUs in the N+M redundancy model, at first we must find the maximum number of SIs that can be assigned to one active and one standby SUs, as shown in Eq. (5-21) and Eq. (5-22) respectively. Once we have found the maximum number of SIs that an active SU can handle, we can calculate the total number of active SUs needed to handle all of the SIs, using Eq. (5-24)[1]:

$$NumberOfActSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneActSU}\right) \quad \textbf{(5-24)}$$

Similarly, the number of standby SUs needed to handle the all the SIs' standby assignments can be calculated as shown in Eq. (5-25):

[1] In this equation, we use Ceiling to avoid dividing one SI among more than one SU.

$$NumberOfStdSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneStdSU}\right) \quad \textbf{(5-25)}$$

Based on the equations above, to handle all the SIs' active and standby assignments, we need the total number of SUs equal to *NumberOfActSUs* + *NumberOfStdSUs*.

As we mentioned before, we constrain the maximum number of SUs in one SG to the number of nodes. Thus, if the total number of SUs (i.e. *NumberOfActSUs* + *NumberOfStdSUs*) exceeds the number of nodes, we need to distribute the SUs on a greater number of SGs. In order to distribute the SUs among SGs fairly, at first we need to find the proportion of active and standby SUs that we need to put together in an SG. For example, if the total number of active SUs is 100 and the total number of standby SUs is 50, the active proportion is 2 and the standby proportion is 1. This means that every two active SUs need one standby SU to be collocated with them in the SG. We can determine that the number of active and standby SUs in each SG should be a coefficient of their proportion. We find the active and standby proportions, as shown in Eq. (5-26) and Eq. (5-27) respectively.

$$ActProportion = \left(\frac{NumberOfActSUs}{\min(NumberOfActSUs, NumberOfstdSUs)}\right) \quad \textbf{(5-26)}$$

$$StdProportion = \left(\frac{NumberOfStdSUs}{\min(NumberOfActSUs, NumberOfstdSUs)}\right) \quad \textbf{(5-27)}$$

According to the above discussion, the number of SUs in an SG can be between the number of *ActProportion*+*StdProportion* and the *NoOfNodes*. In Eq. (5-28), we calculate the total number of active and standby SUs that can be put together to be compliant with the *NoOfNodes*. In this equation, we increase the number of SUs based on the *ActProportion* and *StdProportion*, to the

point that we reach to the *NoOfNodes*. For example, if *ActProportion=2, StdProportion=1* and *NoOfNodes=7*, we can have 6 SUs in each SG ((2+1)+(2+1)≤7).

$$NumberOfSUsPerOneSG = \min\left\{Ceil\left(floor\left(\frac{NoOfNodes}{ActProportion + StdProportion}\right) \times (ActProportion + StdProportion)\right), (NumberOfActSUs + NumberOfStdSUs)\right\} \quad \textbf{(5-28)}$$

Once we have calculated the number of SUs for one SG, we can calculate the needed number of SGs to group the total number of SUs, using Eq. (5-29):

$$NumberOfSGs = Ceil\left(\frac{NumberOfActSUs + NumberOfStdSUs}{NumberOfSUsPerOneSG}\right) \quad \textbf{(5-29)}$$

❖ **NWay-Active redundancy model***:* In this redundancy model, there are no standby assignments and SUs. Every SI can have one or more active assignment (i.e. *NoOfActiveAssignments>=1*). The *NoOfActiveAssignments* has to be given as an input (i.e. CR). In order to calculate the number of SUs and SGs in the NWay-Active redundancy model, at first we calculate the maximum number of SIs that can be assigned to an active SU, using Eq. (5-21). Knowing the maximum number of SIs for one active SU, we can calculate the maximum number of SIs that can be handled by an SG with SUs equal to *NoOfNodes*. The calculation is shown in Eq. (5-30).

$$MaxNumberOfSIsPerSG(With\ NoOfNodes\ SUs) = \min\left(MaxNoOfSIsPerOneActSU \times floor\left(\frac{NoOfNodes}{NoOfActiveAssignments}\right), NoOfSIs\right) \quad \textbf{(5-30)}$$

Now that we have calculated the maximum number of SIs that each SG can handle (i.e. *MaxNumberOfSIsPerSG*), we can calculate the number of SGs needed to handle the total number of SIs, using Eq. (5-31). For instance, if an SG can handle 10 SIs (i.e. *MaxNumberOfSIsPerSG =10*), and there are 50 SIs in the SI template (i.e. *NoOfSIs=50*), we need to create 5 SGs to handle all of the SIs (50/10=5).

$$NumberOfSGs = Ceil\left(\frac{NoOfSIs}{MaxNumberOfSIsPerSG}\right) \quad \textbf{(5-31)}$$

Finally, we calculate the number of SUs for each SG as shown in Eq. (5-32).

$$NumberOfSUsPerOneSG = NoOfActiveAssignment \times Ceil\left(\frac{MaxNumberOfSIsPerSG}{MaxNoOfSIsPerOneActSU}\right) \quad \textbf{(5-32)}$$

❖ **NWay redundancy model:** In the NWay redundancy model, the calculations are more complex than in the other redundancy models. Each SI can have only one active assignment to an SU and one or more standby assignments to other SUs (i.e. *NoOfStandbyAssignments>=1)*. The *NoOfStandbyAssignments* has to be given as an input (i.e. CR). Moreover, an SU can be assigned as active for some SIs and, in the meanwhile, it can be assigned as standby for the other SIs. There are no explicit active and standby SUs. However, in this section when we mention active and standby SUs, we mean the SUs that are needed to handle the SI's active and standby assignments respectively.

In order to calculate the number of SUs and SGs in the NWay redundancy model, we use the same calculations presented for the NWay-Active redundancy model. However, in the NWay-Active redundancy model, we have shown the calculations only for the active assignments. In the NWay redundancy model, we use the same calculations, but for both the active and standby

assignments. We choose the result from one of these assignments as the reference result of our calculations.

Active Assignments:

First, we use Eq. (5-21) to calculate the maximum number of SIs that can be handled by one SU. Second, we calculate the maximum number of SIs that can be handled by an SG with the maximum number of *NoOfNodes- NoOfStandbyAssignments* active SUs, as shown in Eq. (5-33):

$$MaxNumberOfSIsPerActiveSG = \min(MaxNoOfSIsPerOneActSU \times floor(NoOfNodes - NoOfStandbyAssignments), NoOfSIs) \quad (5\text{-}33)$$

In the equation above, we have considered that the maximum number of active SUs in each SG should be equal to *NoOfNodes-NoOfStandbyAssignments*. This is because *NoOfStandbyAssignments* is to survive simultaneous failures. For this reason, we must reserve *NoOfStandbyAssignments* nodes (or SUs) in addition to the active SUs, in order to handle the *NoOfStandbyAssignments* simultaneous failures.

Then, by using the maximum number of SIs that each SG can handle, we calculate the number of SGs that are needed to handle the total number of SIs, as shown in Eq. (5-34):

$$NumberOfActiveSGs = Ceil\left(\frac{NoOfSIs}{MaxNumberOfSIsPerActiveSG}\right) \quad (5\text{-}34)$$

Lastly, we use Eq. (5-35) to calculate the number of active SUs:

$$NumberOfActiveSUsPerOneSG = Ceil\left(\frac{MaxNumberOfSIsPerActiveSG}{MaxNoOfSIsPerOneActSU}\right) + NoOfStandbyAssignments \quad (5\text{-}35)$$

In the equation above, it can be seen that we have added those SUs that we have reserved for handling simultaneous failures (i.e. *NoOfStandbyAssignments* SUs).

Standby Assignments:

At first, we use Eq. (5-22) to find the maximum number of SIs that a standby SU can handle. Then we calculate the maximum number of SIs for one SG, as shown in Eq. (5-36). By using the equation, we assume that the SG can have a maximum of SUs equal to *NoOfNodes* – 1. We consider the maximum number of SUs as *NoOfNodes* – 1, because no active and standby assignments of an SI can be assigned to the same SU.

$$MaxNumberOfSIsPerStandbySG = \min(MaxNoOfSIsPerOneStdSU \times floor\left(\frac{NoOfNodes - 1}{NoOfStandbyAssignment}\right), NoOfSIs) \quad \textbf{(5-36)}$$

At this point, we can calculate the number of SGs that we need to handle the total number of standby assignments of the SIs, as shown in Eq. (5-37).

$$NumberOfStandbySGs = Ceil\left(\frac{NoOfSIs}{MaxNumberOfSIsPerStandbySG}\right) \quad \textbf{(5-37)}$$

Finally, we calculate the number of SUs in each SG as shown in Eq. (5-38).

$$NumberOfStandbySUsPerOneSG = Ceil\left(\frac{MaxNumberOfSIsPerStandbySG}{MaxNoOfSIsPerOneStandbySU}\right) + 1 \quad \textbf{(5-38)}$$

Once we have calculated the number of SUs and SGs for both active and standby assignments, we choose the reference result from the one that resulted in a greater number of SGs (i.e. max (*NumberOfActiveSGs, NumberOfStandbySGs*)). If both assignments result in the same

number of SGs, we choose the one that results in a greater number of SUs per SG (i.e. max (*NumberOfActiveSUsPerOneSG, NumberOfStandbySUsPerOneSG*)).

5.2.3 Discussion

We have shown how we calculate the number of components in an SU. However, in our service availability calculation (i.e. Eq. (5-18)), we do not consider the impact of the recovery action of a service on another service. The following example helps in clarifying this. Assume that SU1 should provide two services (i.e. SI templates), which are Svc1 and Svc2. In such a case, the recovery action of Svc1 can affect the availability of Svc2, if the recovery action (of Svc1) is at the SU, node, application or cluster levels. This means that if a component that provides Svc1 fails and has the actual recovery action of the aforementioned levels (e.g. SU Restart), the other components in the same SU that provide Svc2 will be subject to restart as well. Therefore, the Svc2 will have an outage as well as the Svc1. In such a case, the *SISA* calculation for Svc2 can be refined into Eq. (5-39):

$$SISA = \prod_{j=1}^{j \le N+M} \frac{MTTF_j}{MTTF_j + MTTR_j} \quad \textbf{(5-39)}$$

where N is the total number of components that are participating to provide Svc2, and M is the number of components that are providing Svc1 and have the actual recovery of the SU, node, application or cluster levels.

5.2.4 An application of the availability estimate-based entities creation method

Now we present an example of the availability estimate-based entities creation method. In this example, we calculate the number of AMF entities based on the partial CR information in Figure 5-9. As shown, there is an SI template with 12 SIs defined in the CR. Each SI should have

five CSIs of the FTP CST and five CSIs of the HTTP CST. The minimum level of service availability for the SI template is defined as 0.999. Also, the number of nodes in the cluster is set to five nodes.

◢ e uiAdministrativeDomain	
ⓐ magicUiAdminDomainName	myWebServiceApplication
◢ e magicUiSgTemplateSiTemplates	
ⓐ magicUiSiTempName	Web_Service_SI_Template
ⓐ magicUiSiTempSvcType	web service
ⓐ MagicUiCRMinimumSA	0.999
ⓐ magicUiRegSiTempNumberofSIs	12
◢ e magicUiSiTempCsiTemplates	
ⓐ magicUiCsiTempName	FTP_CSI_Template
ⓐ magicUiCSiTempNumberofCsis	5
ⓐ magicUiCsiTempCsType	FTP
◢ e magicUiSiTempCsiTemplates	
ⓐ magicUiCsiTempName	HTTP_CSI_Template
ⓐ magicUiCSiTempNumberofCsis	5
ⓐ magicUiCsiTempCsType	HTTP
◢ e uiClusterInfo	
ⓐ numberOfNodes	5

Figure 5-9 An example CR for the availability estimate-based entities creation method

Based on the CR, the configuration generator selects the ETF prototypes that can provide the requested service. Then it derives the corresponding AMF types based on the selected prototypes. Once it has derived the AMF types, it applies the availability estimate-based entities creation method to calculate the number of entities of each AMF type. Figure 5-10 demonstrates the AMF types derived from our example.

Application_Type	
magicEtfApptName	Web Server AppType
SGT_A	
magicEtfSgtName	Web Server SGType
magicEtfSgtRedundancyModel	SA_AMF_N+M_REDUNDANCY_MODEL
SUT_A	
magicEtfSutName	Web Server SUType
FTP_CT	
magicEtfCtName	FTP CompType
MTTF	53000
MTTR	8.20
CST	
magicEtfCstName	FTP CST
ActiveCapability	5
StandbyCapability	5
HTTP_CT	
magicEtfCtName	HTTP CompType
MTTF	67000
MTTR	6.65
CST	
magicEtfCstName	HTTP CST
ActiveCapability	3
StandbyCapability	3

Figure 5-10 An example of created AMF types

Calculating the number of components: As the first step of the method, we use Algorithm 5-2 to calculate the number of components of each CT (i.e. FTP_CT and HTTP_CT) that should be placed in an SU. Then we use Algorithm 5-3 to calculate the maximum number of components of each CT that we can put in an SU. The results given by using the algorithms are as follows:

- For FTP_CT:
  - Minimum number of components = 1
  - Maximum number of components = 12
- For HTTP_CT:
  - Minimum number of components = 2
  - Maximum number of components = 24

We use Algorithm 5-4 to calculate the number of components that should be put together in an SU, with respect to the required level of service availability (i.e. 0.999). After doing so, we put the minimum number of components of each CT in an SU. This means that we start with one component of FTP_CT and two components of HTTP_CT. Then we calculate the service availability, using Eq. (5-18):

$$SISA = \prod_{j=1}^{j \le N} \frac{MTTF_j}{MTTF_j + MTTR_j} = \frac{53000}{53000 + 8.2} \times \left(\frac{67000}{67000 + 6.65}\right)^2 = 0.999646$$

Once we have calculated the achievable service availability based on the number of components, we check if it can satisfy the availability requirement. As can be seen, *SISA*=0.999646 can meet the required availability (i.e. 0.999). Next, we repeat this step by increasing the number of components of each CT. We increase the number of components of each CT based on their proportions. Hence, for this iteration, the number of components of FTP_CT and HTTP_CT will be two and four respectively. Again, we calculate the service availability by using Eq. (5-18):

$$SISA = \prod_{j=1}^{j \le N} \frac{MTTF_j}{MTTF_j + MTTR_j} = \left(\frac{53000}{53000 + 8.2}\right)^2 \times \left(\frac{67000}{67000 + 6.65}\right)^4 = 0.999293$$

The result of the *SISA* shows that the estimated service availability can satisfy the availability requirement. We iterate the steps again, this time with three components of FTP_CT and six components of HTTP_CT. The equation below shows that in this iteration, *SISA*=0.998940 becomes lower than the service availability requested in the CR.

$$SISA = \prod_{j=1}^{j \le N} \frac{MTTF_j}{MTTF_j + MTTR_j} = \left(\frac{53000}{53000 + 8.2}\right)^3 \times \left(\frac{67000}{67000 + 6.65}\right)^6 = 0.998940$$

At this stage, we determine that these numbers of components cannot satisfy the requested service availability. In response, we must set the number of components of the CTs according to the previous iteration. This means that the number of components of FTP_CT and HTTP_CT in an SU should be equal to two and four respectively. The steps discussed above are illustrated in Figure 5-11.

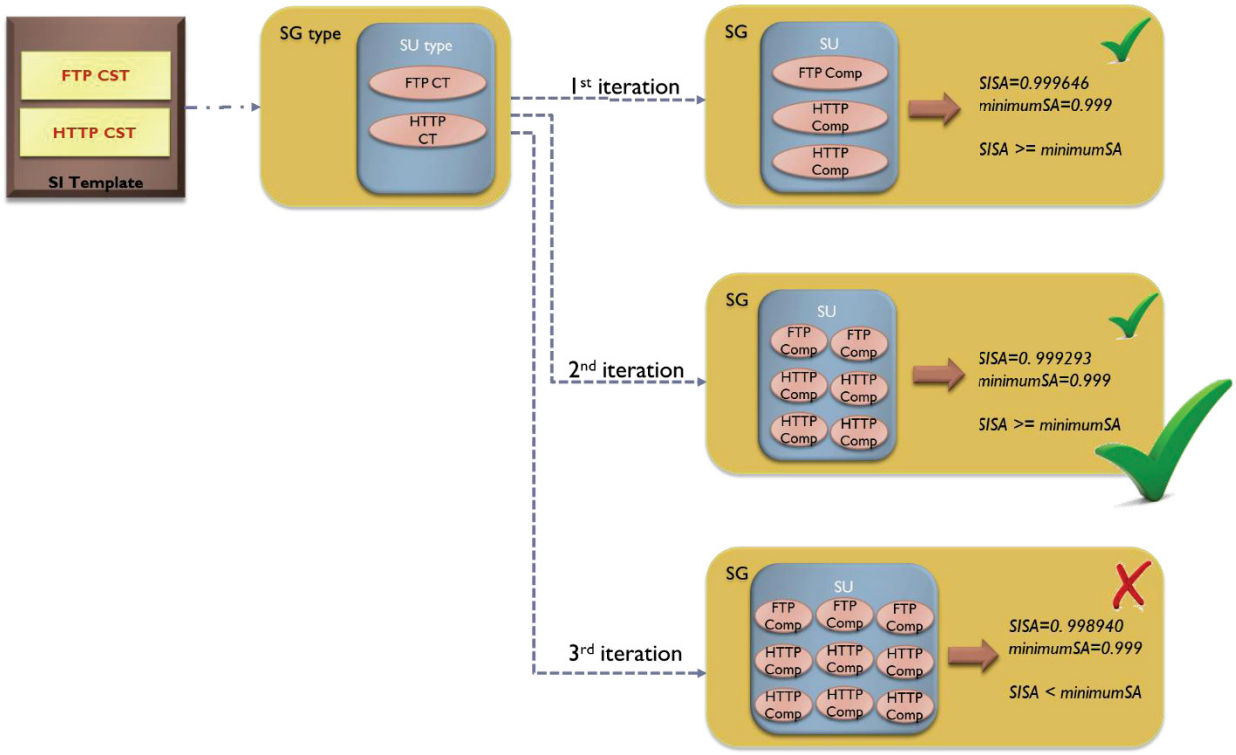

Figure 5-11 An illustration of calculating the number of components

Calculating the number of SUs and SGs: Once we have calculated the number of components of each CT in an SU, we must calculate the number of SUs and SGs. The following is the summarized list of the attributes and their value for this example, according to Figure 5-9 and Figure 5-10:

- *NoOfSIs = 12*
- Redundancy model of the SGs that that are going to protect the SI template: N+M

- For FTP_CST:
    - *NoOfCompsPerCST* = 2, is the number of components of FTP_CT, which has been calculated above.
    - *ActiveCapabilityPerCST* = 5
    - *StandbyCapabilityPerCST* = 5
    - *NoOfCSIsPerCST* = 5
- For HTTP_CST:
    - *NoOfCompsPerCST* = 4, is the number of components of HTTP_CT, which has been calculated above.
    - *ActiveCapabilityPerCST* = 3
    - *StandbyCapabilityPerCST* = 3
    - *NoOfCSIsPerCST* = 5

As mentioned, the number of SUs and SGs should be calculated for each CST. For this reason, we present the calculations for FTP_CST and HTTP_CST separately. Then we select one of them as the reference result.

- **Calculating the number of SGs and SUs for FTP_CST:**

To calculate the number of SUs and SGs for the N+M redundancy model, at first we need to calculate the maximum number of SIs that can be assigned to an active SU, using Eq. (5-21):

$$MaxNoOfSIsPerOneActSU = floor\left(\frac{NoOfCompsPerCST \times ActiveCapabilityPerCST}{NoOfCSIsPerCST} \right)$$

$$= floor\left(\frac{2 \times 5}{5} \right) = 2$$

Then we calculate the maximum number of SIs that can be assigned to a standby SU, using Eq. (5-22):

$$MaxNoOfSIsPerOneStdSU = floor\left(\frac{NoOfCompsPerCST \times StandbyCapabilityPerCST}{NoOfCSIsPerCST} \right)$$

$$= floor\left(\frac{2 \times 5}{5} \right) = 2$$

Next, we can calculate the total number of active SUs that we need to handle all the twelve SIs, by using Eq. (5-24):

$$NumberOfActSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneActSU} \right)$$

$$= Ceil\left(\frac{12}{2} \right) = 6$$

Similarly, we calculate the number of standby SUs needed to handle all of the twelve SIs' standby assignments, by using Eq. (5-25):

$$NumberOfStdSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneStdSU} \right)$$

$$= Ceil\left(\frac{12}{2} \right) = 6$$

Based on the calculations above, we conclude that the total number of 6 active and 6 standby SUs are needed to handle all the SIs. As a result, we need 12 SUs to handle all the SIs (6+6=12). However, we need to distribute the active and standby SUs in such a way that the total number of SUs in an SG does not exceed the number of nodes in the cluster (i.e. five nodes). To do so,

we use Eq. (5-26) and Eq. (5-27) to find the minimum proportion of the active and standby SUs that we need to put together in an SG.

$$ActProportion = \left(\frac{NumberOfActSUs}{\min(NumberOfActSUs, NumberOfstdSUs)} \right) = \left(\frac{6}{\min(6,6)} \right) = 1$$

$$StdProportion = \left(\frac{NumberOfStdSUs}{\min(NumberOfActSUs, NumberOfstdSUs)} \right) = \left(\frac{6}{\min(6,6)} \right) = 1$$

The above equations show that an active SU needs a standby SU to be placed with it in an SG. Then we use Eq. (5-28) to calculate the total number of active and standby SUs that can be put together in each SG, with respect to the number of nodes.

$$NumberOfSUsPerOneSG$$
$$= \min\left\{Ceil\left(floor\left(\frac{NoOfNodes}{ActProportion + StdProportion} \right) \times (ActProportion + StdProportion)\right), (NumberOfActSUs + NumberOfStdSUs)\right\}$$
$$= \min\left\{Ceil\left(floor\left(\frac{5}{1+1} \right) \times (1+1)\right), (6+6)\right\} = 4$$

Once we have calculated the number SUs for each SG, we use Eq. (5-29) to calculate the needed number of SGs:

$$NumberOfSGs = Ceil\left(\frac{NumberOfActSUs + NumberOfStdSUs}{NumberOfSUsPerOneSG} \right) = Ceil\left(\frac{6+6}{4} \right) = 3$$

According to the calculations above, we need 3 SGs, and each of them should group 4 SUs. The calculations above were only for FTP_CST. Now we need to do the same calculations for HTTP_CST.

- **Calculating the number of SGs and SUs for HTTP_CST:**

  First, we calculate the maximum number of SIs that can be assigned to an active SU:

$$MaxNoOfSIsPerOneActSU = floor\left(\frac{NoOfCompsPerCST \times ActiveCapabilityPerCST}{NoOfCSIsPerCST} \right)$$

$$= floor\left(\frac{4 \times 3}{5} \right) = 2$$

Then, we calculate the maximum number of SIs that can be assigned to a standby SU:

$$MaxNoOfSIsPerOneStdSU = floor\left(\frac{NoOfCompsPerCST \times StandbyCapabilityPerCST}{NoOfCSIsPerCST} \right)$$

$$= floor\left(\frac{4 \times 3}{5} \right) = 2$$

Afterward, we can calculate the total number of active SUs needed to handle all of the SIs:

$$NumberOfActSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneActSU} \right)$$

$$= Ceil\left(\frac{12}{2} \right) = 6$$

Similarly, we calculate the number of standby SUs needed to handle the standby assignments of all the SIs:

$$NumberOfStdSUs = Ceil\left(\frac{NoOfSIs}{MaxNoOfSIsPerOneStdSU} \right) \quad \textbf{(5-40)}$$

$$= Ceil\left(\frac{12}{2} \right) = 6$$

Then we find the minimum proportion of the active and standby SUs that we need to put together:

$$ActProportion = \left(\frac{NumberOfActSUs}{\min(NumberOfActSUs, NumberOfstdSUs)} \right) = \left(\frac{6}{\min(6,6)} \right) = 1$$

$$StdProportion = \left(\frac{NumberOfStdSUs}{\min(NumberOfActSUs, NumberOfstdSUs)} \right) = \left(\frac{6}{\min(6,6)} \right) = 1$$

Once we have calculated the *ActProportion* and *StdProportion* of the SUs in each SG, we calculate the total number of active and standby SUs that should be put together in each SG, with respect to the number of nodes.

$$NumberOfSUsPerOneSG$$
$$= \min\left\{Ceil\left(floor\left(\frac{NoOfNodes}{ActProportion + StdProportion}\right) \times (ActProportion + StdProportion)\right), (NumberOfActSUs + NumberOfStdSUs)\right\}$$
$$= \min\left\{Ceil\left(floor\left(\frac{5}{1+1}\right) \times (1+1)\right), (6+6)\right\} = 4$$

Then we calculate the needed number of SGs by using Eq. (5-29):

$$NumberOfSGs = Ceil\left(\frac{NumberOfActSUs + NumberOfStdSUs}{NumberOfSUsPerOneSG} \right) = Ceil\left(\frac{6+6}{4} \right) = 3$$

According to the calculations above, we need 3 SGs, and each of them should have 4 SUs.

At this point, we need to compare the results of the calculations for the CSTs and choose the one that resulted in a greater number of SGs. Following is a summarized result of the calculations for each CST:

- For FTP_CST:
    - *NumberOfSUsPerOneSG=4*

- *NumberOfSGs = 3*

- For HTTP_CST:
  - *NumberOfSUsPerOneSG=4*
  - *NumberOfSGs = 3*

As can be seen, the calculations resulted in the same number of SGs and SUs. Consequently, we can choose any of them as the reference result for the number of SGs and SUs. Figure 5-12 depicts the result of the example discussed above. Note that at runtime, the distribution of SUs on the nodes and SIs' assignments. As a result, we are showing only one possible distribution of the entities.

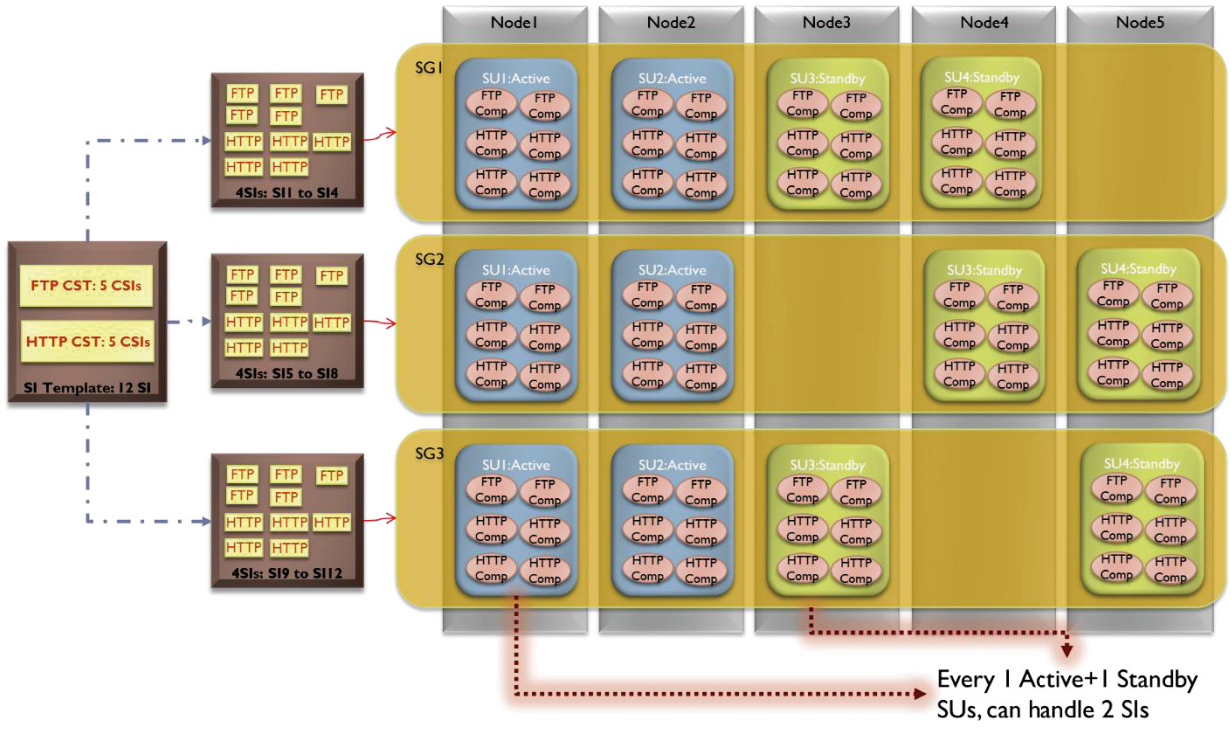

Figure 5-12 Application of the availability estimate-based entities creation method

5.3 Chapter Conclusion

In this chapter, we have discussed two methods, namely, availability estimate–based prototype selection and availability estimate-based entities creation methods.

- The availability estimate-based prototype selection method: As mentioned in Chapter 3, the configuration generator selects the ETF prototypes that can satisfy the CR. Then, based on the selected ETF prototypes, it builds type stacks and passes them to the second step of the generation process. The goal of the availability estimate-based prototype selection method is to estimate the deliverable service availability based on the available type stacks built in the first step of the generation process, and to eliminate the type stacks that cannot guarantee the availability requested in the CR. The method analyzes the actual recovery action to be performed by AMF at runtime. Then, it estimates the recovery time needed to complete the actual recovery action (i.e. MTTR). Using the MTTR and MTTF of the component prototypes, we estimate the achievable service availability for each type stack. Finally, we pass those type stacks that can meet the availability requirement.

- The availability estimate-based entities creation method: We know that in the configuration generation process proposed in [1], the system designer is responsible of providing the number of SUs and SGs to protect and SI template. Based on these numbers, the generator calculates the number of components. We prefer to release the system designer from this task, since he/she might not have enough information at the time to define the CR. We have proposed the availability estimate-based entities creation method in the third step of the generation process. The goal of this

method is to calculate the number of entities (i.e. SGs, SUs and components), taking into account the availability requirement.

The objective of the proposed methods is to generate the configuration(s) that can meet the level of service availability requested in the CR. While the availability estimate-based prototype selection method aims at eliminating the type stacks that cannot guarantee the availability requirement, the availability estimate-based prototype selection method aims at creating the entities such that the availability requirement is guaranteed.

In the next chapter, we demonstrate our prototype tool and also present a partial validation of the proposed configuration design patterns and methods.

6 Prototype Tool and Partial Validation

We have implemented our methods and embedded the configuration design patterns into the existing prototype tool [1]. In this chapter, we present our prototype tool and validate partially our configuration design patterns and methods.

6.1 Prototype Tool

The interactions between the different modules and the data flow in the prototype are shown in Figure 6-1. The prototype tool consists of four main modules, which are the Graphical User Interface (GUI), the Configuration Generator, the I/O Module and the Object Model.

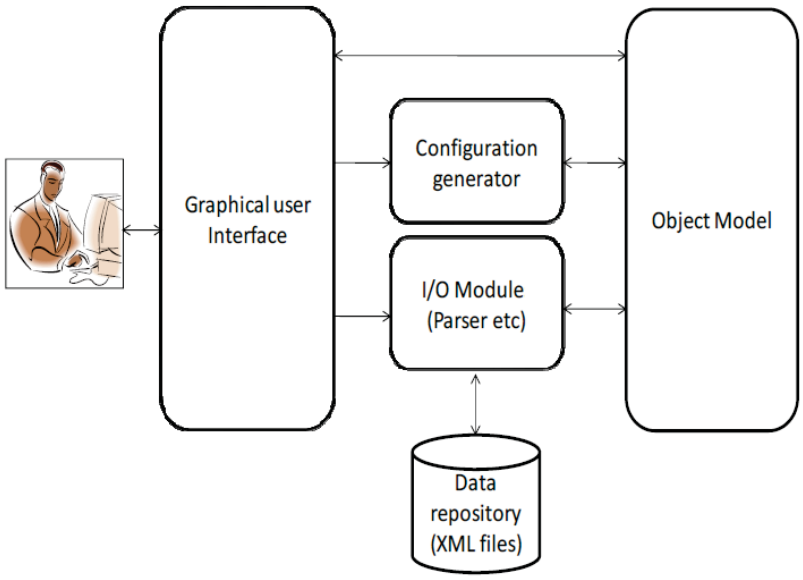

Figure 6-1 Overall architecture of the prototype tool [1]

Graphical User Interface: the GUI is used by the system designer to 1) run the configuration generation 2) provide the CR 3) provide the ETF XML file and 4) save the AMF configuration [1].

The Configuration Generator: the configuration generator module represents the implementation of the configuration generation process [1]. It consists of the generation steps, namely the ETF prototype selection, the AMF type creation, the AMF entities creation and the distribution of the AMF entities. Our proposed configuration design patterns and methods are also implemented and embedded in this module.

The Object Model: it contains the internal models, which are ETF, AMF, and CR models. These models are based on the ETF object model derived from the ETF schema [22], the AMF information model described in the AMF specification [6] and the CR concepts [1]. The Object Model is used to parse the input and output files (e.g. ETF and AMF configurations). In addition, it is used to check their consistency and perform validation [1].

The I/O Module: this module is responsible of fetching the input XML files (e.g. designated ETF file), and parsing them by using the proper Object Models [1]. Moreover, the I/O module is used to store the AMF configuration as an XML file. This XML file is created based on the internal AMF Object Model.

Figure 6-2 provides a refined view of the prototype tool with the different classes and the embedding of our patterns and methods.

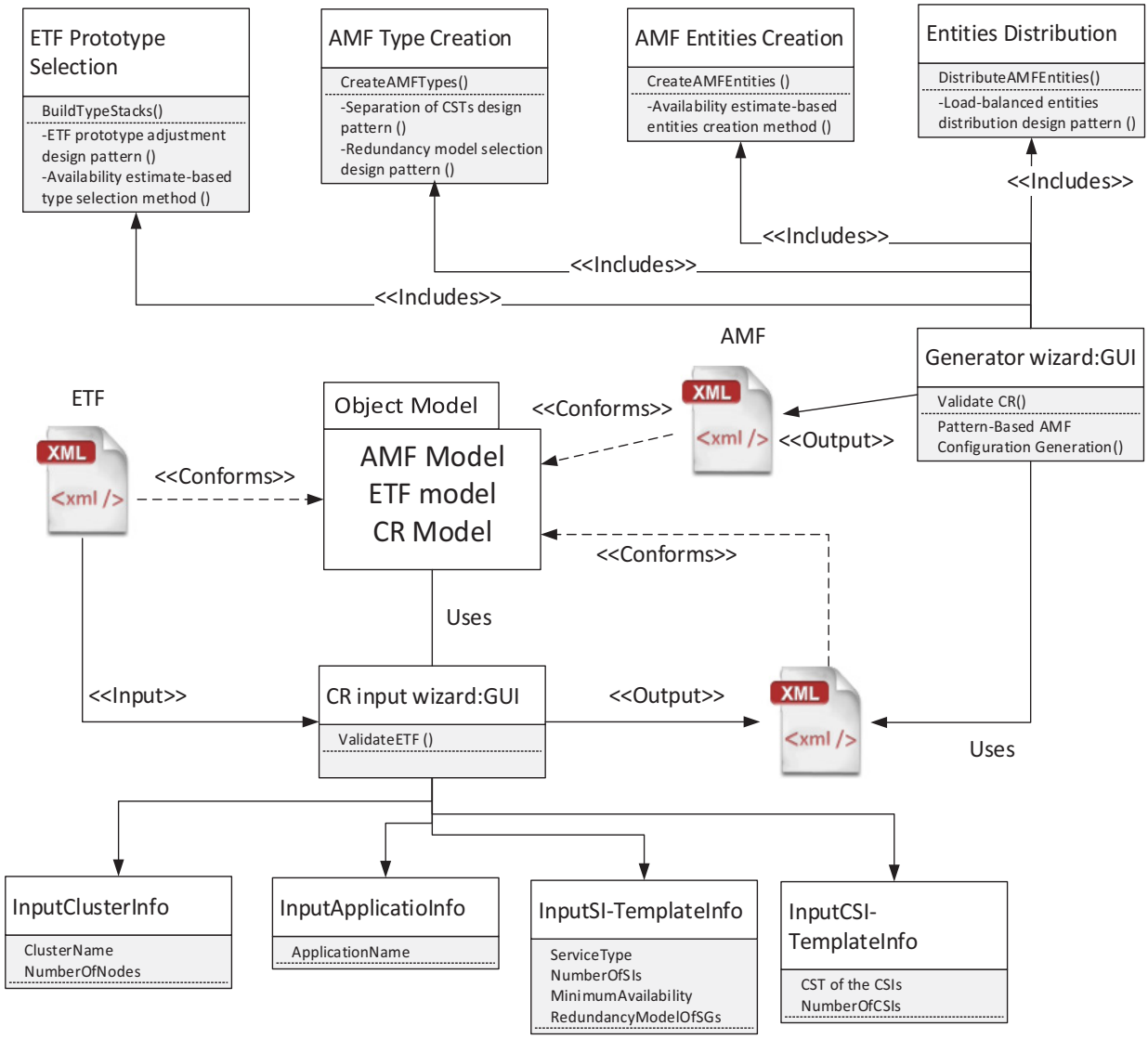

Figure 6-2 The pattern-based AMF configuration generation prototype tool

6.2 An application of the prototype tool

In this section, we generate an AMF configuration for a web server application, which provides a file transfer service using the FTP protocol.

Starting with the configuration generation tool, the system designer provides the ETF and the CR as input. Through the GUI, he/she chooses the ETF XML file from data repository (See

Figure 6-3). The I/O module uses the object module to parse the ETF file according to the internal ETF model. During this step, ETF file is checked for conformance.

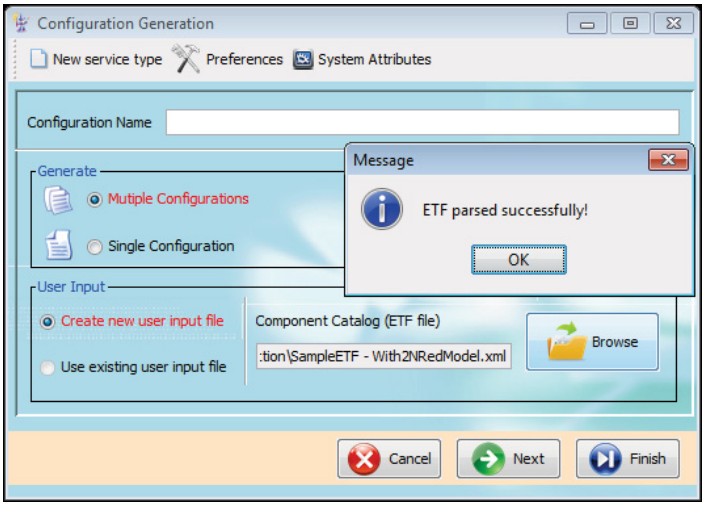

Figure 6-3 Providing the ETF file

The ETF file that we are using in this example is shown in Figure 6-4.

etfSGType	
magicEtfSgtName	Orphan Web Server SGType
magicEtfSgtGroups	//@etfSUType.0
magicEtfSgtAutoRepairOption	True
magicEtfSgtCompProbCounterMax	5
magicEtfSgtSuProbCounterMax	5
magicEtfSgtRedundancyModel	SA_AMF_2N_REDUNDANCY_MODEL
etfSGType	
magicEtfSgtName	Another Orphan Web Server SGType
magicEtfSgtGroups	//@etfSUType.0
magicEtfSgtAutoRepairOption	True
magicEtfSgtCompProbCounterMax	0
magicEtfSgtSuProbCounterMax	5
magicEtfSgtRedundancyModel	SA_AMF_2N_REDUNDANCY_MODEL
etfSUType	
magicEtfSutName	Web Server SUType
magicEtfSutSuFailOver	False
magicEtfProvidesSvcType	Web Server SvcType
magicEtfGroups	//@etfCtSut.0
etfStandaloneCompType	
magicEtfCtName	FTP CompType
magicEtfSupportedCsType	//@etfCtCSType.0
magicEtfCtVersion	Version-1
magicEtfRecoveryOnError	Component Restart
magicEtfCtCallbackTimeout	10
magicEtfCtClcCliTimeout	15
magicEtfCtFailureRate	53000
magicEtfCtProbabilityOfInstantiationSuccessful	0.9
magicEtfCtProbabilityOfInstantiationSuccessfulWithDelay	0.9
magicEtfCtProbabilityOfTerminationSuccessful	0.8
magicEtfCtDisableRestart	NULL
etfCSType	
magicEtfCstName	FTP
magicEtfSupportedby	//@etfCtCSType.0
etfCtCSType	
magicEtfMaxNumStandbyCsi	5
magicEtfMaxNumActiveCsi	5
magicEtfSupportingCt	//@etfStandaloneCompType.0
magicEtfSupportedCsType	//@etfCSType.0
magicEtfCompCapabilityModel	SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY
etfCtSut	
magicEtfMinNumInstances	1
magicEtfMaxNumInstances	
magicEtfMemberCt	//@etfStandaloneCompType.0
magicEtfGroupingSut	//@etfSUType.0

Figure 6-4 A sample ETF file

Next, the user provides the information regarding the CR. As part of the CR, the system designer should also provide the components global attributes, as we need them in our availability estimate methods. These attributes are the maximum number of components instantiation attempts

with or without delay and also the delay (in seconds) between each instantiation attempt (See Figure 6-5).

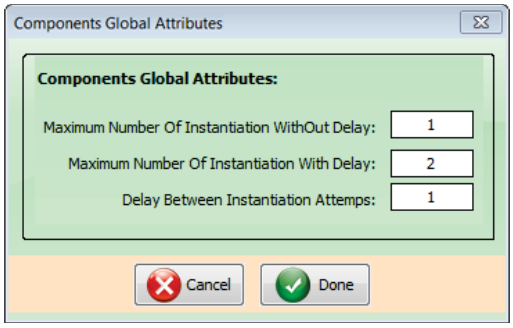

Figure 6-5 Components global attributes as part of the CR input

The user should also provide the cluster information including the number of nodes in the cluster and the time that each node takes to shut down (Figure 6-6).

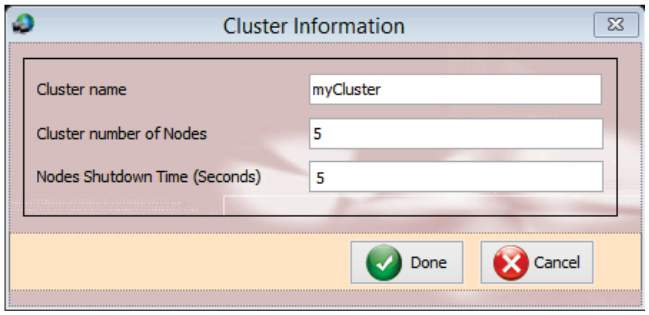

Figure 6-6 Custer information as part of CR input

Finally, the user should provide the necessary information needed to form the application. He/she should input the service type that the application should provide in terms of SI templates. Furthermore, he/she should specify the number of SIs in each SI template and their required level of service availability. In addition, the user may want to set the redundancy model of the SGs that will protect the SI template to 2N or No-Redundancy. The user also needs to define the CSI templates for the SI templates.

In this example, the user defines one SI template with 12 SIs with a minimum of 0.999 service availability. The service type of the SI template is “web service”. The user also asks for a 2N redundancy model for the SGs that will protect the SI template. Furthermore, according to the input, each SI should be composed of 5 CSIs from the FTP CST. Figure 6-7 shows the steps for defining the application and its services using the GUI.

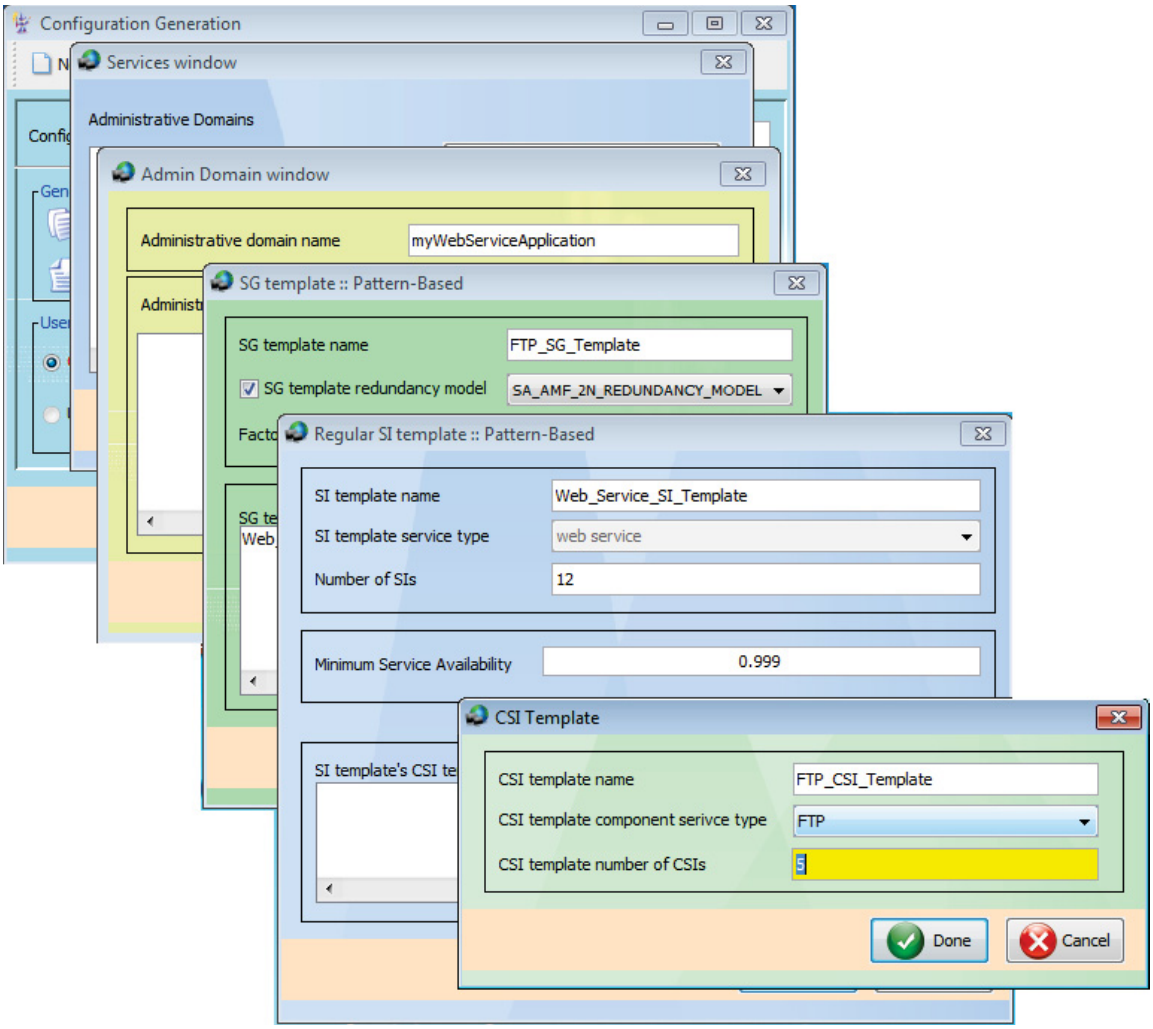

Figure 6-7 Application and its services information

The tool creates the CR file based on the user input. It stores the information regarding the cluster, the application and its services. Figure 6-8 shows the CR generated for this example.

uiAdministrativeDomain	
magicUiAdminDomainName	myWebServiceApplication
magicUiAdminDomainSgTemplates	
magicUiSgTempName	FTP_SG_Template
magicUiSgTempRedundancyModelLiteral	SA_AMF_2N_REDUNDANCY_MODEL
magicUiSgTemplateSiTemplates	
magicUiSiTempName	Web_Service_SI_Template
magicUiSiTempSvcType	web service
MagicUiCRMinimumSA	0.999
magicUiRegSiTempNumberofSIs	12
magicUiSiTempCsiTemplates	
magicUiCsiTempName	FTP_CSI_Template
magicUiCSiTempNumberofCsis	5
magicUiCsiTempCsType	FTP
uiClusterInfo	
numberOfNodes	5
clusterName	myCluster
NodesShutdownTime	100
uiSystemAttributes	
magicAMFCompNumMaxInstantiateWithDelay	2
magicAMFCompNumMaxInstantiateWithOutDelay	1
magicAMFDelayBetweenInstantiateAttemps	1

Figure 6-8 The CR for the web server application

In the first step of the configuration generation process, the generator selects the ETF prototypes that can provide the requested service and builds the type stacks accordingly. In our example, the user asked for a service type that is composed of FTP CST. The generator scans through the ETF file and builds two type stacks to provide the requested service as shown in Figure 6-9.

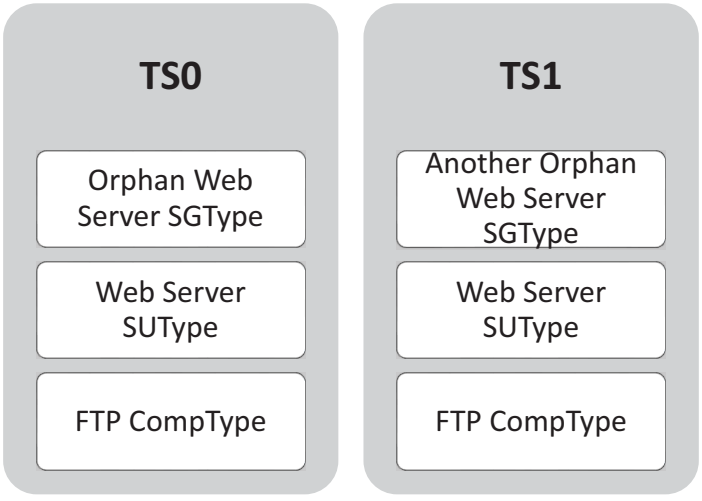

Figure 6-9 The type stacks for the web server application

Once the type stacks are built, the tool applies the ETF prototype adjustment configuration design pattern. As can be seen in the ETF, all of the recovery-related attributes of the prototypes are set by the vendor, except for the CompType *disableRestart* attribute. Therefore, we have a choice for the *disableRestart* attribute of the CompType. According to our configuration design pattern, we set the FTP CompType *disableRestart* attribute to *False*.

Next, the tool applies the availability estimate-based prototype selection method, to estimate the availability of each type stack and discard the ones that cannot meet the availability requirement. Figure 6-10 shows the results of applying this method to our type stacks. As shown, based on the user input and the ETF prototypes, the tool created two type stacks that can provide the requested service. However, since we asked for a minimum of 0.999 service availability, only TS0 is selected and TS1 is discarded. This is because the estimated service availability for TS1 is estimated at 0.99740777.

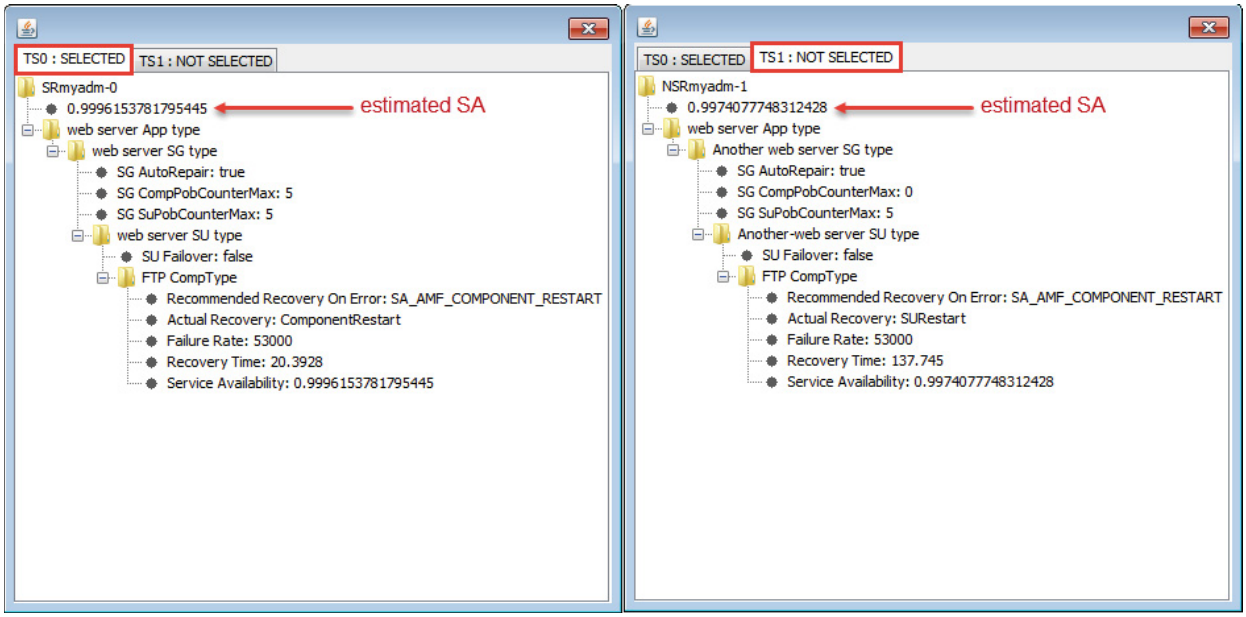

Figure 6-10 Estimated availability for the selected type stacks

The prototype tool passes TS0 to the next step of the configuration generation process namely AMF Type Creation. In this step, the generator creates the corresponding AMF types from the selected ETF prototypes in TS0. The CSTs separation and redundancy model selection configuration design patterns are also embedded in this step. In this example, since there is only one requested CST, the separation of CSTs configuration design pattern is not applicable. Hence, the generator derives only one AMF CT based on the FTP CompType. Likewise, the redundancy model selection configuration design pattern is not applicable, as the user asked for a 2N redundancy model explicitly.

As the third step of the generation process, the tool creates the AMF entities from the AMF types. In this step, the generator calculates the number of entities using the availability estimate-based entities creation method. In Figure 6-10, we have shown that the FTP CompType in the TS0 has an availability of 0.9996153. As a result, according to the method, the generator creates two components of the FTP CompType in each SUs, to respect the availability requirement. Moreover, since each component has the limited capability of handling 5 active and 5 standby CSIs, each SG can handle at most 2 SIs. Hence, we need 6 SGs to handle the 12 SIs of the SI-template. Note that the number of SUs in each SG is equal to 2, as the system designer asked for a 2N redundancy model. Figure 6-11 illustrates the partial AMF configuration resulting from the application of this method.

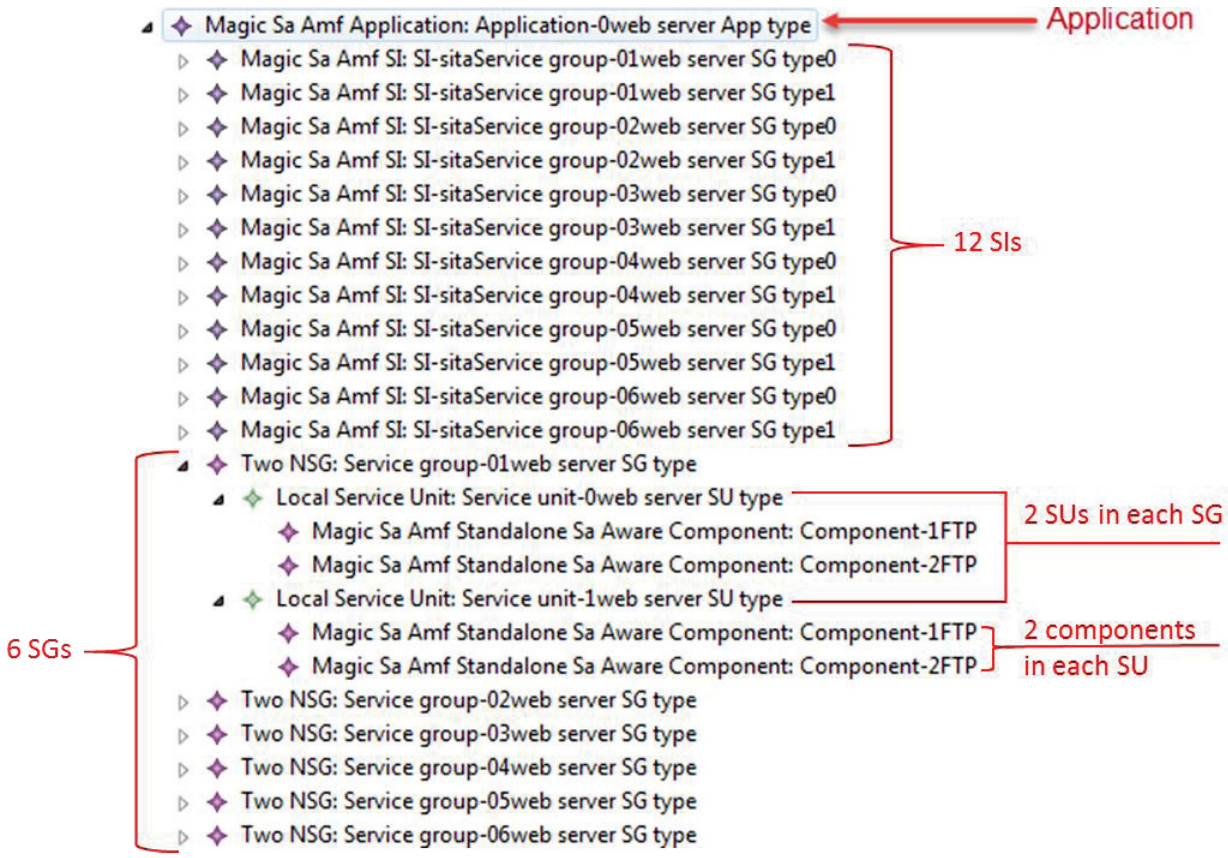

Figure 6-11 Partial AMF configuration to illustrate the created entities

Finally, the generator distributes the entities over the nodes. The load-balanced entity distribution configuration design pattern is applied in this step. In this example, the redundancy model of the SGs is set to 2N. Hence, the load-balanced entity distribution configuration design pattern is applicable. Figure 6-12 shows the result of the application of the configuration design pattern in this example.

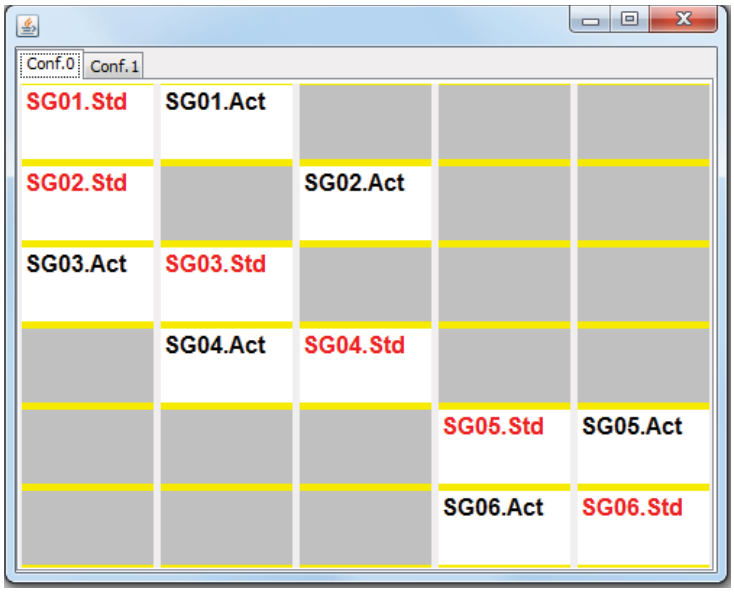

Figure 6-12 The result of the application of the load-balanced entity distribution configuration design pattern

The AMF configuration generation process is now completed. The tool uses the I/O module to export the configuration file to the repository (See Appendix).

6.3 Partial Validation

In this section, we partially validate our approach for generating configurations with respect to the availability requirement. For this purpose, we use the prototype tool presented in the previous subsection. We ran four different configuration generation experiments to test our approach. We have categorized the configurations based on the four different recovery actions, namely, Component Restart, Component Failover, SU Restart, and SU Failover. For instance, we have set the prototype attributes of the first experiment (Config. 1 and Config. 2) in a way that the components' actual recovery actions are all evaluated as Component Restart. We did the same for the other experiments. For all cases, we have asked for a minimum of 0.9999 service availability in the requirements. Also, we have applied all of the configuration design patterns and methods for the experiments.

As shown in Table 6-1, when we turned off our configuration design patterns and methods in the generation process, we generated eight configurations (two configurations per experiment). When we turned it on, we generated only four configurations, one for each experiment. Four configurations (Config. 2, Config. 4, Config. 6 and Config. 8) were eliminated when our method was used during the generation process.

To validate the availability rendered by each of the configurations, we have transformed each of the eight configurations to a DSPN model [12] and evaluated their respective availability using the TimeNET tool [13][14][15]. The transformation from the AMF configuration to a DSPN model is performed automatically by another tool [1]. The results of the availability analysis with TimeNET are also shown in Table 6-1.

Table 6-1 Comparison of generated configurations' availability and availability resulted by the simulation

Composed Components' Actual Recovery Action	Configurations	Estimated SA based on our tool	Eliminated based on our approach	TimeNET result
Component Restart	Config.1	0.999943		0.999921
	Config.2	0.999711	X	0.999651
Component Failover	Config.3	0.999918		0.999901
	Config.4	0.999807	X	0.999739
SU Restart	Config.5	0.999932		0.999905
	Config.6	0.999610	X	0.999699
SU Failover	Config.7	0.999719		0.999620
	Config.8	0.999601	X	0.999493

The availability analysis with TimeNET showed that indeed the service availability of 0.9999 was satisfied by all the configurations generated with our approach, except for one, namely Config. 7. The configurations that do not meet the availability requirement were eliminated. In the special case of Config. 7, although the configuration does not satisfy the required service availability, the tool did not eliminate it. The reason was that for the fourth experiment none of the configurations could satisfy the service availability requirement. In such a case, we select the configuration with the highest service availability estimate.

These preliminary results show that it is possible to reduce the solution space during configuration generation based on early estimate of service availability. However, a more thorough evaluation of our method is required in the future.

6.4 Chapter Conclusion

In this chapter, we have discussed the prototype tool implementing the configuration design patterns and methods proposed in Chapter 4 and Chapter 5. For illustration purpose, we have also presented an application of our prototype tool.

We have presented a partial validation of our approach using the prototype tool. We ran four different configuration generation experiments to test our approach. In each experiment, two configurations could be generated. However, our approach only generated the one that could satisfy the availability requirement. We have analyzed the eight configurations and their availability using the TimeNET simulation tool. The result of the availability analysis with TimeNET shows that our approach is capable of selecting the best configuration among all.

7 Conclusion

To conclude the thesis, we review the main research contributions followed by a discussion on potential future research work.

7.1 Research Contributions

In this thesis, we have proposed a set of configuration design patterns and methods to improve an existing configuration generation approach [1]. By embedding these patterns and methods into the configuration generation process, we aim to generate the best configuration in terms of service availability. We have proposed four configuration design patterns and two methods. The purpose of the patterns is to improve the expected level of service availability, whereas the methods have been proposed to eliminate the configuration options that cannot meet the requested service availability. The advantage of our work is that we estimate service availability in an early phase of the configuration generation process. As a result, we eliminate the need of availability analysis tools that are resource consuming and computationally expensive.

We have proposed the ETF prototype adjustment configuration design pattern to set or change the prototypes' recovery-related attributes in a way to improve service availability. It aims at minimizing the impact of a recovery action in case of a component failure on the services. In response, the service availability will be improved.

The ETF prototype adjustment configuration design pattern allows us to improve the availability for each prototype involved in a given context. However, this is not enough to compare the different configuration options as different components may have significantly different recovery/repair times. We have proposed the availability estimate-based prototype selection

method, to estimate the expected service availability of a partial configuration and eliminate the type stacks that do not satisfy the desired level of availability.

We have also proposed the separation of CSTs configuration design pattern for the second step of the configuration generation process. The objective of this configuration design pattern is to not affect all the functionalities of the system after a component failure. Also, we have proposed the redundancy model selection configuration design pattern to select the appropriate redundancy model for an SG type. For this, we have ranked the redundancy models based on the preferences for more general redundancy models over more specific ones. We select the appropriate redundancy model according to our preferences and components' capability model.

For the third step of the generation process, we have proposed the availability estimate-based entities creation method that is used to calculate the number of entities, with respect to the availability requirement. With our method, the system designer is released from providing the number of entities as an input. One should notice the difference between the two proposed methods. The availability estimate-based prototype selection method is used to eliminate the type stacks that cannot satisfy the availability requirement, whereas the availability estimate-based entities creation method aims at calculating the number of entities in order to meet the availability requirement.

We have proposed a configuration design pattern for load balancing. The goal of this configuration design pattern is to set the configuration attributes to ensure a load-balanced distribution of the SUs over the cluster nodes. The pattern guarantees load balancing before and after a single failure.

Finally, a partial validation of the work has been presented in Chapter 6. As a partial validation, we ran four experiments to generate configurations using our prototype tool. We have analyzed the generated configurations with the TimeNET tool and compared the analysis results with the results given by our prototype tool. The preliminary results of the analysis tool showed that our approach selected the best configuration for all of the experiments.

7.2 Potential Future Research Work

Although we have reduced the input information required from the system designer, still some elements in the CR, such as the number of SIs in each SI template, may not be determined easily. The authors in [26] proposed a model-driven approach to generate the CR from high-level user requirements. As a future direction, one may integrate our prototype tool with their prototype tool in order to develop an integrated tool that generates the AMF configurations directly from high-level user requirements.

This work needs to be validated further with more case studies. The future validation can be done for each configuration design pattern and method individually, to observe the impact of each of them on the deliverable service availability.

References

[1] A. Kanso, "Automated Configuration Design and Analysis for Service High-Availability," PhD Thesis, Concordia University, 2012.

[2] "Improving Systems Availability." IBM Global Services, Atlanta, USA, pp. 1–4, 1998.

[3] M. Toeroe and F. Tam, *Service Availability: Principles and Practice*, 1st ed. Wiley, 2012, p. 476.

[4] "Service Availability Forum." [Online]. Available: SAForum.org. [Accessed: 30-May-2014].

[5] SAForum, "Service Availability Forum, Service Availability Interface Overview," 2011. [Online]. Available: http://www.saforum.org/HOA/assn16627/images/SAI-Overview-B.05.03.AL.pdf. [Accessed: 30-May-2014].

[6] "Service Availability Forum, Application Interface Specification. Availability Management Framework SAI-AIS-AMF-B.04.01.," 2011. [Online]. Available: http://www.saforum.org/hoa/assn16627/images/SAI-AIS-AMF-B.04.01.AL.pdf. [Accessed: 30-May-2014].

[7] A. Kanso, M. Toeroe, F. Khendek, and A. Hamou-Lhadj, "Automatic Generation of AMF Compliant Configurations," in *5th International Service Availability Symposium, ISAS 2008*, 2008, pp. 155–170.

[8] A. Immonen and E. Niemelä, "Survey of reliability and availability prediction methods from the viewpoint of software architecture," *Softw. Syst. Model.*, vol. 7, no. 1, pp. 49–65, Jan. 2007.

[9] SAForum, "Service Availability Forum, Application Interface Specification. Software Management Framework SAI-AIS-SMF-A.01.02.AL.," 2011. [Online]. Available: http://www.saforum.org/HOA/assn16627/images/SAI-AIS-SMF-A.01.02.AL.pdf. [Accessed: 30-May-2014].

[10] P. Salehi, "A Model Based Framework for Service Availability Management," PhD Thesis, Concordia University, 2012.

[11] A. Kanso, "Automatic Generation of AMF Compliant Configurations," Master Thesis, Concordia University, 2008.

[12] M. Marsan and G. Chiola, *On Petri nets with deterministic and exponentially distributed firing times*, vol. 226. Springer Berlin Heidelberg, 1987, pp. 132–145.

[13] A. Zimmermann, "Modeling and evaluation of stochastic Petri nets with TimeNET 4.1," in *proceeding of 6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, 2012, pp. 54–63.

[14] R. German, C. Kelling, A. Zimmermann, and G. Hommel, “TimeNET: a toolkit for evaluating non-Markovian stochastic Petri nets,” *Perform. Eval.*, vol. 24, no. 1–2, pp. 69–87, 1995.

[15] “TimeNET.” [Online]. Available: http://www.tu-ilmenau.de/sse/timenet/. [Accessed: 18-Mar-2014].

[16] M. Kimura, S. Yamada, and S. Osaki, “Statistical software reliability prediction and its applicability based on mean time between failures,” *Math. Comput. Model.*, vol. 22, no. 10–12, pp. 149–155, Nov. 1995.

[17] B. Littlewood and J. Verrall, “A Bayesian reliability growth model for computer software,” *Appl. Stat.*, vol. 22, no. 3, pp. 332–346, 1973.

[18] W. Xie, H. Sun, Y. Cao, and K. S. Trivedi, “Modeling of user perceived webserver availability,” in *proceeding of 38th IEEE International Conference on Communications, 2003. ICC ’03.*, 2003, vol. 3, pp. 1796–1800.

[19] G. Janakiraman, J. Santos, and Y. Turner, “Automated multi-tier system design for service availability,” in *Proceeding of the First Workshop on Design of Self-Managing Systems*, 2003.

[20] D. Wang and K. K. S. Trivedi, “Modeling User-Perceived Service Availability,” *Serv. Availab.*, pp. 107–122, 2005.

[21] A. Kanso, M. Toeroe, A. Hamou-Lhadj, and F. Khendek, “Generating AMF Configurations from Software Vendor Constraints and User Requirements,” in *proceeding of the Fourth International Conference on Availability, Reliability and Security*, 2009, pp. 454–461.

[22] SAForum, “SAI-AIS-SMF-ETF-A.01.01.xsd (ETF schema describing the software bundle and the entity types’ relations and features.).” [Online]. Available: http://www.saforum.org/HOA/assn16627/images/sai-ais-smf-xsd-a.01.02.al.zip. [Accessed: 30-May-2014].

[23] A. Kanso, M. Toeroe, and F. Khendek, “Comparing Redundancy Models for High Availability Middleware,” *Computing Journal*, Springer, to apear in 2014.

[24] R. Billinton and R. N. Allan, *Reliability Evaluation of Engineering Systems*. Boston, MA: Springer US, 1992, p. 453.

[25] “System Reliability and Availability.” [Online]. Available: http://www.eventhelix.com/realtimemantra/faulthandling/system_reliability_availability.htm#Availability in Series. [Accessed: 18-Mar-2014].

[26] M. Abbasipour, M. Sackmann, F. Khendek, and M. Toeroe, “Ontology-based User Requirements Decomposition for Component Selection for Highly Available Systems,” in *proceedings of the 15th IEEE international conference on Information Reuse and Integration (IEEE IRI), 2014*.

Appendix (Generated Configuration in Chapter 6)

```
<?xml version="1.0" encoding="ASCII"?>
<DomainModel.MagicAmfConfiguration:MagicAmfRoot xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:DomainModel.MagicAmf="http:///DomainModel/MagicAmf.ecore"
xmlns:DomainModel.MagicAmfConfiguration="http:///DomainModel/MagicAmfConfiguration.ecor
e">
  <magicDataTypes>
    <saStringT content="Global Attributes"/>
    <saStringT content="All-Nodes"/>
    <saStringT content="web server App type"/>
    <saStringT content="Version-1"/>
    <saStringT content="web server App type"/>
    <saStringT content="web server SG type"/>
    <saStringT content="Version-1"/>
    <saStringT content="web server SU type"/>
    <saStringT content="Version-1"/>
    <saStringT content="FTP"/>
    <saStringT content="Version-1"/>
    <saStringT content="instantiate command path"/>
    <saStringT content="clean up command path"/>
    <saStringT content="Version-1"/>
    <saStringT content="FTP"/>
    <saStringT content="Ericsson-TSP"/>
    <saStringT content="HTTP"/>
    <saStringT content="Version-1"/>
    <saStringT content="instantiate command path"/>
    <saStringT content="clean up command path"/>
    <saStringT content="Version-1"/>
    <saStringT content="HTTP"/>
    <saStringT content="Version-1"/>
    <saStringT content="web service "/>
    <saStringT content="Another web server SG type"/>
    <saStringT content="Version-1"/>
    <saStringT content="Another-web server SU type"/>
    <saStringT content="Version-1"/>
    <saStringT content="FTP"/>
    <saStringT content="Version-1"/>
    <saStringT content="instantiate command path"/>
    <saStringT content="clean up command path"/>
    <saStringT content="HTTP"/>
    <saStringT content="Version-1"/>
    <saStringT content="instantiate command path"/>
    <saStringT content="clean up command path"/>
```

<saStringT content="Application-0web server App type"/>
<saStringT content="Service group-01web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-01web server SG type0"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-01web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="Service group-02web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>

<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-02web server SG type0"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-02web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="Service group-03web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-03web server SG type0"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>

<saStringT content="SI-Web_Service_SI_TemplateService group-03web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="Service group-04web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-04web server SG type0"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-04web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="Service group-05web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>

<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-05web server SG type0"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-05web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="Service group-06web server SG type"/>
<saStringT content="Service unit-0web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Service unit-1web server SU type"/>
<saStringT content="Component-1FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="Component-2FTP"/>
<saStringT content="FTP"/>
<saStringT content="Version-1"/>
<saStringT content="instantiate command path"/>
<saStringT content="clean up command path"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-06web server SG type0"/>

```
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="SI-Web_Service_SI_TemplateService group-06web server SG type1"/>
<saStringT content="CSI-FTP_CSI_Template0"/>
<saStringT content="CSI-FTP_CSI_Template1"/>
<saStringT content="CSI-FTP_CSI_Template2"/>
<saStringT content="CSI-FTP_CSI_Template3"/>
<saStringT content="CSI-FTP_CSI_Template4"/>
<saStringT content="myCluster"/>
<saStringT content="myCluster0"/>
<saStringT content="NodeSWBundle-"/>
<saStringT content="myCluster1"/>
<saStringT content="NodeSWBundle-"/>
<saStringT content="myCluster2"/>
<saStringT content="NodeSWBundle-"/>
<saStringT content="myCluster3"/>
<saStringT content="NodeSWBundle-"/>
<saStringT content="myCluster4"/>
<saStringT content="NodeSWBundle-"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
```

```
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
```

```
<saUnit32T content="1"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
<saUnit32T content="1"/>
<saUnit32T content="2"/>
<saUnit32T content="5"/>
<saUnit32T content="5"/>
<saUnit32T content="2"/>
```

```
  <saUnit32T content="1"/>
  <saUnit32T content="5"/>
  <saUnit32T content="5"/>
  <saUnit32T content="2"/>
  <saUnit32T content="1"/>
  <saUnit32T content="3"/>
  <saUnit32T content="3"/>
  <saUnit32T content="3"/>
  <saUnit32T content="3"/>
  <saUnit32T content="3"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="5"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="1"/>
  <saTimeT content="500"/>
  <saTimeT content="40"/>
  <saTimeT content="40"/>
  <saTimeT content="40"/>
  <saTimeT content="40"/>
  <saTimeT content="40"/>
  <saBoolT/>
  <saBoolT/>
</magicDataTypes>
```

```
  <magicCluster magicSaAmfClusterClmCluster="//@clmCluster"
magicSafAmfCluster="//@magicDataTypes/@saStringT.247"
magicSaAmfClusterStartupTimeout="//@magicDataTypes/@saTimeT.31"/>
  <magicApplications magicSaAmfAppType="//@magicAppTypes.0"
magicSafApp="//@magicDataTypes/@saStringT.36">
    <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.0"
magicSafSi="//@magicDataTypes/@saStringT.60">
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.61"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.62"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.63"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.64"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.65"
magicSaAmfCSType="//@magicCsTypes.0"/>
    </magicSaAmfApplicationProvides>
    <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.0"
magicSafSi="//@magicDataTypes/@saStringT.66">
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.67"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.68"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.69"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.70"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.71"
magicSaAmfCSType="//@magicCsTypes.0"/>
    </magicSaAmfApplicationProvides>
    <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.1"
magicSafSi="//@magicDataTypes/@saStringT.95">
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.96"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.97"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.98"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.99"
magicSaAmfCSType="//@magicCsTypes.0"/>
      <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.100"
magicSaAmfCSType="//@magicCsTypes.0"/>
    </magicSaAmfApplicationProvides>
    <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.1"
magicSafSi="//@magicDataTypes/@saStringT.101">
```

```
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.102"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.103"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.104"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.105"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.106"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.2"
magicSafSi="//@magicDataTypes/@saStringT.130">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.131"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.132"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.133"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.134"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.135"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.2"
magicSafSi="//@magicDataTypes/@saStringT.136">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.137"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.138"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.139"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.140"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.141"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.3"
magicSafSi="//@magicDataTypes/@saStringT.165">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.166"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.167"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.168"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.169"
magicSaAmfCSType="//@magicCsTypes.0"/>
```

```
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.170"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.3"
magicSafSi="//@magicDataTypes/@saStringT.171">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.172"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.173"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.174"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.175"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.176"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.4"
magicSafSi="//@magicDataTypes/@saStringT.200">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.201"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.202"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.203"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.204"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.205"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.4"
magicSafSi="//@magicDataTypes/@saStringT.206">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.207"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.208"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.209"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.210"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.211"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.5"
magicSafSi="//@magicDataTypes/@saStringT.235">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.236"
magicSaAmfCSType="//@magicCsTypes.0"/>
```

```
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.237"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.238"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.239"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.240"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicSaAmfApplicationProvides magicSaAmfSvcType="//@magicSvcType.0"
magicSaAmfSiProtectedbySG="//@magicApplications.0/@magicAmfApplicationGroups.5"
magicSafSi="//@magicDataTypes/@saStringT.241">
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.242"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.243"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.244"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.245"
magicSaAmfCSType="//@magicCsTypes.0"/>
    <magicAmfSIGroups magicSafCsi="//@magicDataTypes/@saStringT.246"
magicSaAmfCSType="//@magicCsTypes.0"/>
  </magicSaAmfApplicationProvides>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.0
//@magicApplications.0/@magicSaAmfApplicationProvides.1"
magicSafSg="//@magicDataTypes/@saStringT.37"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.38"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.25"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.1">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.0"
magicSafComp="//@magicDataTypes/@saStringT.39"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.29"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.28"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.7"
magicSaAmfCompType="//@magicCompTypes.4"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.1"
magicSafComp="//@magicDataTypes/@saStringT.44"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.33"
```

```
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.32"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.8"
magicSaAmfCompType="//@magicCompTypes.5"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.49"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.34"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.0">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.2"
magicSafComp="//@magicDataTypes/@saStringT.50"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.38"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.37"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.9"
magicSaAmfCompType="//@magicCompTypes.6"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.3"
magicSafComp="//@magicDataTypes/@saStringT.55"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.42"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.41"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.10"
magicSaAmfCompType="//@magicCompTypes.7"/>
    </magicAmfSGGroups>
  </magicAmfApplicationGroups>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.2
//@magicApplications.0/@magicSaAmfApplicationProvides.3"
magicSafSg="//@magicDataTypes/@saStringT.72"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.73"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.43"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.2">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.4"
magicSafComp="//@magicDataTypes/@saStringT.74"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.47"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.46"
```

```
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.11"
magicSaAmfCompType="//@magicCompTypes.8"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.5"
magicSafComp="//@magicDataTypes/@saStringT.79"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.51"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.50"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.12"
magicSaAmfCompType="//@magicCompTypes.9"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.84"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.52"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.0">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.6"
magicSafComp="//@magicDataTypes/@saStringT.85"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.56"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.55"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.13"
magicSaAmfCompType="//@magicCompTypes.10"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.7"
magicSafComp="//@magicDataTypes/@saStringT.90"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.60"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.59"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.14"
magicSaAmfCompType="//@magicCompTypes.11"/>
    </magicAmfSGGroups>
  </magicAmfApplicationGroups>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.4
//@magicApplications.0/@magicSaAmfApplicationProvides.5"
magicSafSg="//@magicDataTypes/@saStringT.107"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.108"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.61"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.0">
```

```
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.8"
magicSafComp="//@magicDataTypes/@saStringT.109"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.65"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.64"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.15"
magicSaAmfCompType="//@magicCompTypes.12"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.9"
magicSafComp="//@magicDataTypes/@saStringT.114"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.69"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.68"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.16"
magicSaAmfCompType="//@magicCompTypes.13"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.119"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.70"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.1">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.10"
magicSafComp="//@magicDataTypes/@saStringT.120"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.74"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.73"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.17"
magicSaAmfCompType="//@magicCompTypes.14"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.11"
magicSafComp="//@magicDataTypes/@saStringT.125"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.78"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.77"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.18"
magicSaAmfCompType="//@magicCompTypes.15"/>
    </magicAmfSGGroups>
  </magicAmfApplicationGroups>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.6
//@magicApplications.0/@magicSaAmfApplicationProvides.7"
```

```
magicSafSg="//@magicDataTypes/@saStringT.142"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.143"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.79"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.1">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.12"
magicSafComp="//@magicDataTypes/@saStringT.144"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.83"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.82"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.19"
magicSaAmfCompType="//@magicCompTypes.16"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.13"
magicSafComp="//@magicDataTypes/@saStringT.149"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.87"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.86"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.20"
magicSaAmfCompType="//@magicCompTypes.17"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.154"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.88"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.2">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.14"
magicSafComp="//@magicDataTypes/@saStringT.155"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.92"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.91"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.21"
magicSaAmfCompType="//@magicCompTypes.18"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.15"
magicSafComp="//@magicDataTypes/@saStringT.160"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.96"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.95"
```

```
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.22"
magicSaAmfCompType="//@magicCompTypes.19"/>
    </magicAmfSGGroups>
  </magicAmfApplicationGroups>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.8
//@magicApplications.0/@magicSaAmfApplicationProvides.9"
magicSafSg="//@magicDataTypes/@saStringT.177"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.178"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.97"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.4">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.16"
magicSafComp="//@magicDataTypes/@saStringT.179"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.101"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.100"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.23"
magicSaAmfCompType="//@magicCompTypes.20"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.17"
magicSafComp="//@magicDataTypes/@saStringT.184"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.105"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.104"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.24"
magicSaAmfCompType="//@magicCompTypes.21"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.189"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.106"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.3">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.18"
magicSafComp="//@magicDataTypes/@saStringT.190"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.110"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.109"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.25"
magicSaAmfCompType="//@magicCompTypes.22"/>
```

```
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.19"
magicSafComp="//@magicDataTypes/@saStringT.195"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.114"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.113"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.26"
magicSaAmfCompType="//@magicCompTypes.23"/>
    </magicAmfSGGroups>
  </magicAmfApplicationGroups>
  <magicAmfApplicationGroups xsi:type="DomainModel.MagicAmf:MagicAmfTwoNSG"
magicSaAmfSGSuHostNodeGroup="//@magicNodeGroup.0"
magicAmfSGProtects="//@magicApplications.0/@magicSaAmfApplicationProvides.10
//@magicApplications.0/@magicSaAmfApplicationProvides.11"
magicSafSg="//@magicDataTypes/@saStringT.212"
magicSaAmfSGType="//@magicSGTypes.0">
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.213"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.115"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.3">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.20"
magicSafComp="//@magicDataTypes/@saStringT.214"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.119"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.118"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.27"
magicSaAmfCompType="//@magicCompTypes.24"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.21"
magicSafComp="//@magicDataTypes/@saStringT.219"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.123"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.122"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.28"
magicSaAmfCompType="//@magicCompTypes.25"/>
    </magicAmfSGGroups>
    <magicAmfSGGroups xsi:type="DomainModel.MagicAmf:MagicAmfLocalServiceUnit"
magicSafSu="//@magicDataTypes/@saStringT.224"
magicSaAmfSURank="//@magicDataTypes/@saUnit32T.124"
magicAmfLocalServiceUnitConfiguredForNodegroup="//@magicNodeGroup.0"
magicSaAmfSUType="//@magicSUTypes.0"
magicAmfLocalServiceUnitConfiguredForNode="//@magicNode.4">
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
```

```
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.22"
magicSafComp="//@magicDataTypes/@saStringT.225"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.128"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.127"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.29"
magicSaAmfCompType="//@magicCompTypes.26"/>
      <magicAmfLocalServiceUnitGroups
xsi:type="DomainModel.MagicAmf:MagicSaAmfStandaloneSaAwareComponent"
magicAmfCompConfiguredby="//@magicCompGlobalAttributes"
magicSafSupportedCsType="//@magicCompCsTypes.23"
magicSafComp="//@magicDataTypes/@saStringT.230"
magicSaAmfCompNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.132"
magicSaAmfCompNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.131"
magicSaAmfCompDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.30"
magicSaAmfCompType="//@magicCompTypes.27"/>
    </magicAmfSGGroups>
   </magicAmfApplicationGroups>
  </magicApplications>
  <magicAppTypes safAppType="//@magicDataTypes/@saStringT.4"
magicAmfApptSGTypes="//@magicSGTypes.0 //@magicSGTypes.1"
magicAmfApptypeFor="//@magicApplications.0"
magicSafVersion="//@magicDataTypes/@saStringT.3"/>
  <magicSGTypes safSgType="//@magicDataTypes/@saStringT.5"
saAmfSgtValidSuTypes="//@magicSUTypes.0"
magicSaAmfSgtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2
//@magicApplications.0/@magicAmfApplicationGroups.3
//@magicApplications.0/@magicAmfApplicationGroups.4
//@magicApplications.0/@magicAmfApplicationGroups.5"
magicSafVersion="//@magicDataTypes/@saStringT.6"
magicSaAmfSgtDefAutoAdjustProb="//@magicDataTypes/@saTimeT.1"
magicSaAmfSgtDefCompRestartProb="//@magicDataTypes/@saTimeT.2"
magicSaAmfSgtDefCompRestartMax="//@magicDataTypes/@saUnit32T.5"
magicSaAmfSgtDefSuRestartProb="//@magicDataTypes/@saTimeT.3"
magicSaAmfSgtDefSuRestartMax="//@magicDataTypes/@saUnit32T.6"
magicSaAmfSgtMemberOf="//@magicAppTypes.0"/>
  <magicSGTypes safSgType="//@magicDataTypes/@saStringT.24"
saAmfSgtValidSuTypes="//@magicSUTypes.1"
magicSafVersion="//@magicDataTypes/@saStringT.25"
magicSaAmfSgtDefAutoAdjustProb="//@magicDataTypes/@saTimeT.4"
magicSaAmfSgtDefCompRestartProb="//@magicDataTypes/@saTimeT.5"
magicSaAmfSgtDefCompRestartMax="//@magicDataTypes/@saUnit32T.15"
magicSaAmfSgtDefSuRestartProb="//@magicDataTypes/@saTimeT.6"
magicSaAmfSgtDefSuRestartMax="//@magicDataTypes/@saUnit32T.16"
magicSaAmfSgtMemberOf="//@magicAppTypes.0"/>
  <magicSUTypes xsi:type="DomainModel.MagicAmf:MagicAmfLocalSUType"
safSuType="//@magicDataTypes/@saStringT.7"
magicAmfSutIsValidFor="//@magicSGTypes.0"
```

```
magicSafMemberCompType="//@magicSutCompTypes.0 //@magicSutCompTypes.1"
magicSafVersion="//@magicDataTypes/@saStringT.8"
magicSaAmfSutDefSUFailover="//@magicDataTypes/@saBoolT.0"
magicSaAmfSutProvidesSvcType="//@magicSvcType.0"
magicAmfLocalSutTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.0/@magic
AmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.1"/>
  <magicSUTypes xsi:type="DomainModel.MagicAmf:MagicAmfLocalSUType"
safSuType="//@magicDataTypes/@saStringT.26"
magicAmfSutIsValidFor="//@magicSGTypes.1"
magicSafMemberCompType="//@magicSutCompTypes.2 //@magicSutCompTypes.3"
magicSafVersion="//@magicDataTypes/@saStringT.27"
magicSaAmfSutDefSUFailover="//@magicDataTypes/@saBoolT.1"
magicSaAmfSutProvidesSvcType="//@magicSvcType.0"/>
  <magicSutCompTypes
magicSaAmfSutMaxNumComponents="//@magicDataTypes/@saUnit32T.7"
magicSaAmfSutMinNumComponents="//@magicDataTypes/@saUnit32T.8"
magicAmfGroupingSut="//@magicSUTypes.0"
magicAmfMemberCt="//@magicCompTypes.0"/>
  <magicSutCompTypes
magicSaAmfSutMaxNumComponents="//@magicDataTypes/@saUnit32T.11"
magicSaAmfSutMinNumComponents="//@magicDataTypes/@saUnit32T.12"
magicAmfGroupingSut="//@magicSUTypes.0"
magicAmfMemberCt="//@magicCompTypes.1"/>
  <magicSutCompTypes
magicSaAmfSutMaxNumComponents="//@magicDataTypes/@saUnit32T.17"
magicSaAmfSutMinNumComponents="//@magicDataTypes/@saUnit32T.18"
magicAmfGroupingSut="//@magicSUTypes.1"
magicAmfMemberCt="//@magicCompTypes.2"/>
  <magicSutCompTypes
magicSaAmfSutMaxNumComponents="//@magicDataTypes/@saUnit32T.21"
magicSaAmfSutMinNumComponents="//@magicDataTypes/@saUnit32T.22"
magicAmfGroupingSut="//@magicSUTypes.1"
magicAmfMemberCt="//@magicCompTypes.3"/>
  <magicSvcType safSvcType="//@magicDataTypes/@saStringT.23"
magicAmfSvcTProvidingSut="//@magicSUTypes.0 //@magicSUTypes.1"
magicSafMemberCSType="//@magicSvcTypeCsTypes.0 //@magicSvcTypeCsTypes.1"
magicSafVersion="//@magicDataTypes/@saStringT.22"
magicSaAmfSI="//@magicApplications.0/@magicSaAmfApplicationProvides.0
//@magicApplications.0/@magicSaAmfApplicationProvides.1
```

```
//@magicApplications.0/@magicSaAmfApplicationProvides.2
//@magicApplications.0/@magicSaAmfApplicationProvides.3
//@magicApplications.0/@magicSaAmfApplicationProvides.4
//@magicApplications.0/@magicSaAmfApplicationProvides.5
//@magicApplications.0/@magicSaAmfApplicationProvides.6
//@magicApplications.0/@magicSaAmfApplicationProvides.7
//@magicApplications.0/@magicSaAmfApplicationProvides.8
//@magicApplications.0/@magicSaAmfApplicationProvides.9
//@magicApplications.0/@magicSaAmfApplicationProvides.10
//@magicApplications.0/@magicSaAmfApplicationProvides.11"/>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.9"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.0"
magicAmfMemberOf="//@magicSutCompTypes.0"
magicSafVersion="//@magicDataTypes/@saStringT.10"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.11"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.12">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.16"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.1"
magicAmfMemberOf="//@magicSutCompTypes.1"
magicSafVersion="//@magicDataTypes/@saStringT.17"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.18"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.19">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.5"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.3"/>
```

```
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.27"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.29"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.3"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.28"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.2"
magicAmfMemberOf="//@magicSutCompTypes.2"
magicSafVersion="//@magicDataTypes/@saStringT.29"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.30"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.31">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.32"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.3"
magicAmfMemberOf="//@magicSutCompTypes.3"
magicSafVersion="//@magicDataTypes/@saStringT.33"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.34"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.35">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.5"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.3"/>
```

```
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.27"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.29"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.3"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.40"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.4"
magicSafVersion="//@magicDataTypes/@saStringT.41"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
0/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.42"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.43">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.45"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.5"
magicSafVersion="//@magicDataTypes/@saStringT.46"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
0/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.47"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.48">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
```

```
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.51"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.6"
magicSafVersion="//@magicDataTypes/@saStringT.52"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
0/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.53"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.54">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.56"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.7"
magicSafVersion="//@magicDataTypes/@saStringT.57"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
0/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.58"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.59">
```

```
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.75"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.8"
magicSafVersion="//@magicDataTypes/@saStringT.76"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
1/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.77"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.78">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.80"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.9"
magicSafVersion="//@magicDataTypes/@saStringT.81"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
1/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
```

```
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.82"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.83">
   <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
   <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
   <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
   <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.86"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.10"
magicSafVersion="//@magicDataTypes/@saStringT.87"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
1/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.88"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.89">
   <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
   <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
   <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
   <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.91"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.11"
magicSafVersion="//@magicDataTypes/@saStringT.92"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
```

```
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
1/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.93"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.94">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.110"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.12"
magicSafVersion="//@magicDataTypes/@saStringT.111"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
2/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.112"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.113">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.115"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.13"
```

```
magicSafVersion="//@magicDataTypes/@saStringT.116"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
2/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.117"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.118">
   <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
   <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
   <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
   <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.121"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.14"
magicSafVersion="//@magicDataTypes/@saStringT.122"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
2/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.123"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.124">
   <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
   <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
   <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
   <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
   <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.126"
```

```
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.15"
magicSafVersion="//@magicDataTypes/@saStringT.127"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
2/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.128"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.129">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.145"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.16"
magicSafVersion="//@magicDataTypes/@saStringT.146"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
3/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.147"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.148">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
```

```
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.150"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.17"
magicSafVersion="//@magicDataTypes/@saStringT.151"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
3/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.152"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.153">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.156"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.18"
magicSafVersion="//@magicDataTypes/@saStringT.157"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
3/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.158"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.159">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
```

```
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.161"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.19"
magicSafVersion="//@magicDataTypes/@saStringT.162"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
3/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.163"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.164">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.180"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.20"
magicSafVersion="//@magicDataTypes/@saStringT.181"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
4/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.182"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.183">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
```

```
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.185"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.21"
magicSafVersion="//@magicDataTypes/@saStringT.186"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
4/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.187"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.188">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.191"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.22"
magicSafVersion="//@magicDataTypes/@saStringT.192"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
4/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.193"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.194">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
```

```
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.196"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.23"
magicSafVersion="//@magicDataTypes/@saStringT.197"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
4/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.198"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.199">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.215"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.24"
magicSafVersion="//@magicDataTypes/@saStringT.216"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
5/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.217"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.218">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
```

```
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.220"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.25"
magicSafVersion="//@magicDataTypes/@saStringT.221"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
5/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.222"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.223">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.226"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.26"
magicSafVersion="//@magicDataTypes/@saStringT.227"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
5/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.228"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.229">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
```

```
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicCompTypes
xsi:type="DomainModel.MagicAmf:MagicAmfStandaloneSaAwareCompType"
safCompType="//@magicDataTypes/@saStringT.231"
magicSaAmfCtSwBundle="//@magicSwBundle.0"
magicSafSupportedCsType="//@magicCtCsTypes.27"
magicSafVersion="//@magicDataTypes/@saStringT.232"
magicSaAmfCtDefRecoveryOnError="SA_AMF_COMPONENT_RESTART"
magicAmfRegularSaAwareCtTypeFor="//@magicApplications.0/@magicAmfApplicationGroups.
5/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"
magicSaAmfCtRelPathInstantiateCmd="//@magicDataTypes/@saStringT.233"
magicSaAmfCtRelPathCleanupCmd="//@magicDataTypes/@saStringT.234">
    <magicSaAmfCtDefClcCliTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.2"/>
    <magicSaAmfCtDefCallbackTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
    <magicSaAmfCtFailureRate
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.18"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfInstantiationSuccessfulWithDelay
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.25"/>
    <magicSaAmfCtProbabilityOfTerminationSuccessful
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saStringT.26"/>
    <magicSaAmfCtDefQuiescingCompleteTimeout
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saTimeT.1"/>
  </magicCompTypes>
  <magicSvcTypeCsTypes magicAmfMemberCSType="//@magicCsTypes.0"
magicAmfGroupingSvct="//@magicSvcType.0">
    <magicSaAmfSvctMaxNumCSIs
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saUnit32T.2"/>
  </magicSvcTypeCsTypes>
  <magicSvcTypeCsTypes magicAmfMemberCSType="//@magicCsTypes.1"
magicAmfGroupingSvct="//@magicSvcType.0">
    <magicSaAmfSvctMaxNumCSIs
href="ui.xml#//@uiAssociatedETF/@dataTypes/@saUnit32T.2"/>
  </magicSvcTypeCsTypes>
  <magicCsTypes safCSType="//@magicDataTypes/@saStringT.14"
magicAmfCSTTypefor="//@magicApplications.0/@magicSaAmfApplicationProvides.0/@magicA
mfSIGroups.0
```

//@magicApplications.0/@magicSaAmfApplicationProvides.0/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.0/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.0/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.0/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.1/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.1/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.1/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.1/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.1/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.2/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.2/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.2/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.2/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.2/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.3/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.3/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.3/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.3/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.3/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.4/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.4/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.4/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.4/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.4/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.5/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.5/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.5/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.5/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.5/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.6/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.6/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.6/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.6/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.6/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.7/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.7/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.7/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.7/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.7/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.8/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.8/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.8/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.8/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.8/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.9/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.9/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.9/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.9/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.9/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.10/@magicAmfSIGroups.0

```
//@magicApplications.0/@magicSaAmfApplicationProvides.10/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.10/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.10/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.10/@magicAmfSIGroups.4
//@magicApplications.0/@magicSaAmfApplicationProvides.11/@magicAmfSIGroups.0
//@magicApplications.0/@magicSaAmfApplicationProvides.11/@magicAmfSIGroups.1
//@magicApplications.0/@magicSaAmfApplicationProvides.11/@magicAmfSIGroups.2
//@magicApplications.0/@magicSaAmfApplicationProvides.11/@magicAmfSIGroups.3
//@magicApplications.0/@magicSaAmfApplicationProvides.11/@magicAmfSIGroups.4"
magicAmfSupportedby="//@magicCtCsTypes.0 //@magicCtCsTypes.2 //@magicCtCsTypes.4
//@magicCtCsTypes.5 //@magicCtCsTypes.6 //@magicCtCsTypes.7 //@magicCtCsTypes.8
//@magicCtCsTypes.9 //@magicCtCsTypes.10 //@magicCtCsTypes.11
//@magicCtCsTypes.12 //@magicCtCsTypes.13 //@magicCtCsTypes.14
//@magicCtCsTypes.15 //@magicCtCsTypes.16 //@magicCtCsTypes.17
//@magicCtCsTypes.18 //@magicCtCsTypes.19 //@magicCtCsTypes.20
//@magicCtCsTypes.21 //@magicCtCsTypes.22 //@magicCtCsTypes.23
//@magicCtCsTypes.24 //@magicCtCsTypes.25 //@magicCtCsTypes.26
//@magicCtCsTypes.27" magicAmfMemberOf="//@magicSvcTypeCsTypes.0"
magicAmfProvides="//@magicCompCsTypes.0 //@magicCompCsTypes.1
//@magicCompCsTypes.2 //@magicCompCsTypes.3 //@magicCompCsTypes.4
//@magicCompCsTypes.5 //@magicCompCsTypes.6 //@magicCompCsTypes.7
//@magicCompCsTypes.8 //@magicCompCsTypes.9 //@magicCompCsTypes.10
//@magicCompCsTypes.11 //@magicCompCsTypes.12 //@magicCompCsTypes.13
//@magicCompCsTypes.14 //@magicCompCsTypes.15 //@magicCompCsTypes.16
//@magicCompCsTypes.17 //@magicCompCsTypes.18 //@magicCompCsTypes.19
//@magicCompCsTypes.20 //@magicCompCsTypes.21 //@magicCompCsTypes.22
//@magicCompCsTypes.23" magicSafVersion="//@magicDataTypes/@saStringT.13"/>
  <magicCsTypes safCSType="//@magicDataTypes/@saStringT.21"
magicAmfSupportedby="//@magicCtCsTypes.1 //@magicCtCsTypes.3"
magicAmfMemberOf="//@magicSvcTypeCsTypes.1"
magicSafVersion="//@magicDataTypes/@saStringT.20"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.9"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.10"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.0"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.13"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.14"
magicAmfSupportedCsType="//@magicCsTypes.1"
magicAmfSupportingCt="//@magicCompTypes.1"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.19"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.20"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.2"/>
```

```
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.23"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.24"
magicAmfSupportedCsType="//@magicCsTypes.1"
magicAmfSupportingCt="//@magicCompTypes.3"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.26"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.27"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.4"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.30"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.31"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.5"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.35"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.36"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.6"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.39"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.40"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.7"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.44"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.45"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.8"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.48"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.49"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.9"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.53"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.54"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.10"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
```

```
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.57"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.58"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.11"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.62"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.63"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.12"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.66"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.67"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.13"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.71"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.72"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.14"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.75"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.76"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.15"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.80"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.81"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.16"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.84"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.85"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.17"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.89"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.90"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.18"/>
  <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.93"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.94"
```

```
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.19"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.98"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.99"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.20"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.102"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.103"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.21"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.107"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.108"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.22"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.111"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.112"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.23"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.116"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.117"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.24"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.120"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.121"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.25"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.125"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.126"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.26"/>
 <magicCtCsTypes
magicSaAmfCtCompCapability="SA_AMF_COMP_X_ACTIVE_OR_Y_STANDBY"
magicSaAmfCtDefNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.129"
magicSaAmfCtDefNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.130"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfSupportingCt="//@magicCompTypes.27"/>
```

```
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.26"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.27"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.30"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.31"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.35"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.36"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.39"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.40"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.44"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.45"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.1/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.48"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.49"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.1/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.53"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.54"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.1/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.57"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.58"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.1/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
 <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.62"
```

```
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.63"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.2/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.66"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.67"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.2/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.71"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.72"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.2/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.75"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.76"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.2/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.80"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.81"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.3/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.84"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.85"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.3/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.89"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.90"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.3/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.93"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.94"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.3/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.98"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.99"
magicAmfSupportedCsType="//@magicCsTypes.0"
```

```
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.102"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.103"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.107"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.108"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.111"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.112"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.116"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.117"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.5/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.120"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.121"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.5/@magicA
mfSGGroups.0/@magicAmfLocalServiceUnitGroups.1"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.125"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.126"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.5/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.0"/>
  <magicCompCsTypes
magicSaAmfCompNumMaxActiveCSIs="//@magicDataTypes/@saUnit32T.129"
magicSaAmfCompNumMaxStandbyCSIs="//@magicDataTypes/@saUnit32T.130"
magicAmfSupportedCsType="//@magicCsTypes.0"
magicAmfProvidingComp="//@magicApplications.0/@magicAmfApplicationGroups.5/@magicA
mfSGGroups.1/@magicAmfLocalServiceUnitGroups.1"/>
  <magicSwBundle magicSafSmfBundle="//@magicDataTypes/@saStringT.15"
magicSafHostNode="//@magicNodeSwBundle.0 //@magicNodeSwBundle.1
//@magicNodeSwBundle.2 //@magicNodeSwBundle.3 //@magicNodeSwBundle.4"
magicAmfSwBundleFor="//@magicCompTypes.0 //@magicCompTypes.1
//@magicCompTypes.2 //@magicCompTypes.3 //@magicCompTypes.4
//@magicCompTypes.5 //@magicCompTypes.6 //@magicCompTypes.7
```

```
//@magicCompTypes.8 //@magicCompTypes.9 //@magicCompTypes.10
//@magicCompTypes.11 //@magicCompTypes.12 //@magicCompTypes.13
//@magicCompTypes.14 //@magicCompTypes.15 //@magicCompTypes.16
//@magicCompTypes.17 //@magicCompTypes.18 //@magicCompTypes.19
//@magicCompTypes.20 //@magicCompTypes.21 //@magicCompTypes.22
//@magicCompTypes.23 //@magicCompTypes.24 //@magicCompTypes.25
//@magicCompTypes.26 //@magicCompTypes.27"/>
  <magicNodeSwBundle
magicSaAmfNodeSwBundlePathPrefix="//@magicDataTypes/@saStringT.249"
magicAmfHostNode="//@magicNode.0" magicAmfInstalledSwBundle="//@magicSwBundle.0"/>
  <magicNodeSwBundle
magicSaAmfNodeSwBundlePathPrefix="//@magicDataTypes/@saStringT.251"
magicAmfHostNode="//@magicNode.1" magicAmfInstalledSwBundle="//@magicSwBundle.0"/>
  <magicNodeSwBundle
magicSaAmfNodeSwBundlePathPrefix="//@magicDataTypes/@saStringT.253"
magicAmfHostNode="//@magicNode.2" magicAmfInstalledSwBundle="//@magicSwBundle.0"/>
  <magicNodeSwBundle
magicSaAmfNodeSwBundlePathPrefix="//@magicDataTypes/@saStringT.255"
magicAmfHostNode="//@magicNode.3" magicAmfInstalledSwBundle="//@magicSwBundle.0"/>
  <magicNodeSwBundle
magicSaAmfNodeSwBundlePathPrefix="//@magicDataTypes/@saStringT.257"
magicAmfHostNode="//@magicNode.4" magicAmfInstalledSwBundle="//@magicSwBundle.0"/>
  <magicNode
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmf
SGGroups.1 //@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.0"
magicAmfGroupedby="//@magicNodeGroup.0"
magicSafInstalledSwBundle="//@magicNodeSwBundle.0"
magicAmfBelongsTo="//@magicCluster"
magicSafAmfNode="//@magicDataTypes/@saStringT.248"
magicSaAmfNodeSuFailOverProb="//@magicDataTypes/@saTimeT.32"
magicSaAmfNodeSuFailoverMax="//@magicDataTypes/@saUnit32T.133"
magicSaAmfNodeClmNode="//@clmCluster/@groups.0"/>
  <magicNode
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmf
SGGroups.0 //@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.0"
magicAmfGroupedby="//@magicNodeGroup.0"
magicSafInstalledSwBundle="//@magicNodeSwBundle.1"
magicAmfBelongsTo="//@magicCluster"
magicSafAmfNode="//@magicDataTypes/@saStringT.250"
magicSaAmfNodeSuFailOverProb="//@magicDataTypes/@saTimeT.33"
magicSaAmfNodeSuFailoverMax="//@magicDataTypes/@saUnit32T.134"
magicSaAmfNodeClmNode="//@clmCluster/@groups.1"/>
  <magicNode
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmf
SGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.1"
magicAmfGroupedby="//@magicNodeGroup.0"
magicSafInstalledSwBundle="//@magicNodeSwBundle.2"
```

```
magicAmfBelongsTo="//@magicCluster"
magicSafAmfNode="//@magicDataTypes/@saStringT.252"
magicSaAmfNodeSuFailOverProb="//@magicDataTypes/@saTimeT.34"
magicSaAmfNodeSuFailoverMax="//@magicDataTypes/@saUnit32T.135"
magicSaAmfNodeClmNode="//@clmCluster/@groups.2"/>
 <magicNode
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmf
SGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.0"
magicAmfGroupedby="//@magicNodeGroup.0"
magicSafInstalledSwBundle="//@magicNodeSwBundle.3"
magicAmfBelongsTo="//@magicCluster"
magicSafAmfNode="//@magicDataTypes/@saStringT.254"
magicSaAmfNodeSuFailOverProb="//@magicDataTypes/@saTimeT.35"
magicSaAmfNodeSuFailoverMax="//@magicDataTypes/@saUnit32T.136"
magicSaAmfNodeClmNode="//@clmCluster/@groups.3"/>
 <magicNode
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmf
SGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.1"
magicAmfGroupedby="//@magicNodeGroup.0"
magicSafInstalledSwBundle="//@magicNodeSwBundle.4"
magicAmfBelongsTo="//@magicCluster"
magicSafAmfNode="//@magicDataTypes/@saStringT.256"
magicSaAmfNodeSuFailOverProb="//@magicDataTypes/@saTimeT.36"
magicSaAmfNodeSuFailoverMax="//@magicDataTypes/@saUnit32T.137"
magicSaAmfNodeClmNode="//@clmCluster/@groups.4"/>
 <magicNodeGroup
magicAmfConfigureFor="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmf
SGGroups.0 //@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.1"
magicSaAmfNGNodeList="//@magicNode.0 //@magicNode.1 //@magicNode.2
//@magicNode.3 //@magicNode.4"
magicSafAmfNodeGroup="//@magicDataTypes/@saStringT.1"
magicAmfConfigureForS="//@magicApplications.0/@magicAmfApplicationGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2
//@magicApplications.0/@magicAmfApplicationGroups.3
//@magicApplications.0/@magicAmfApplicationGroups.4
//@magicApplications.0/@magicAmfApplicationGroups.5"/>
```

<magicCompGlobalAttributes magicSafRdn="//@magicDataTypes/@saStringT.0"
magicSaAmfNumMaxInstantiateWithoutDelay="//@magicDataTypes/@saUnit32T.4"
magicSaAmfNumMaxInstantiateWithDelay="//@magicDataTypes/@saUnit32T.3"
magicSaAmfDelayBetweenInstantiationAttempts="//@magicDataTypes/@saTimeT.0"
magicSaAmfNumMaxAmStartAttempts="//@magicDataTypes/@saUnit32T.0"
magicSaAmfNumMaxAmStopAttempts="//@magicDataTypes/@saUnit32T.2"
magicAmfConfiguredFor="//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.0/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.1/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.2/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.3/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.4/@magicAmfSGGroups.1/@magicAmfLocalServiceUnitGroups.1
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.0/@magicAmfLocalServiceUnitGroups.1

```
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.1/@magicAmf
LocalServiceUnitGroups.0
//@magicApplications.0/@magicAmfApplicationGroups.5/@magicAmfSGGroups.1/@magicAmf
LocalServiceUnitGroups.1"/>
  <clmCluster magicSaAmfCluster="//@magicCluster">
    <groups magicSaAmfNode="//@magicNode.0"
safNode="//@magicDataTypes/@saStringT.248"/>
    <groups magicSaAmfNode="//@magicNode.1"
safNode="//@magicDataTypes/@saStringT.250"/>
    <groups magicSaAmfNode="//@magicNode.2"
safNode="//@magicDataTypes/@saStringT.252"/>
    <groups magicSaAmfNode="//@magicNode.3"
safNode="//@magicDataTypes/@saStringT.254"/>
    <groups magicSaAmfNode="//@magicNode.4"
safNode="//@magicDataTypes/@saStringT.256"/>
  </clmCluster>
</DomainModel.MagicAmfConfiguration:MagicAmfRoot>
```