HYBRID CLOUD SECURITY CERTIFICATION

SPYROS KATOPODIS

A Thesis Submitted for the Degree of Master of Philosophy (MPhil) in Computer Science

City University London

Department of Computer Science

School of Mathematics, Computer Science & Engineering

June 2015

Table of Contents

Table of Figures

List of Tables

Acknowledgement

I would like to thank my supervisors and everybody in City University London for giving me the opportunity to make this experience unforgettable.

I am most grateful to my family and my friends for all their support and their belief in me. They are the ones who made me have faith and trust in myself and reminded me to always follow my dreams.

Declaration

"I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement."

Abstract

In this report, I introduce a hybrid approach for certifying security properties of cloud services that combines monitoring and testing data. This report argues about the need for hybrid certification and examines the basic characteristics of hybrid certification models.

The certification of cloud service security has become a necessity due to the on-going concerns about cloud security and the need to increase cloud trustworthiness through rigorous assessments of security by trusted third parties. Unlike the certification of security in traditional software systems, which is based on static forms of security assessment (e.g., the Common Criteria model), the certification of cloud service security requires continuous assessment. This is because cloud services are provisioned through dynamic infrastructures operating under security controls and other configurations that may change dynamically introducing unforeseen vulnerabilities. Cloud service security can also be compromised because of attacks on co-tenant services.

Recent work on cloud service certification applies dynamic forms of security assessment, notably dynamic testing or continuous monitoring. These overcome some of the limitations of traditional security certification and audits (e.g. they produce machine readable certificates incorporating dynamically collected evidence). However, there are cases where existing approaches cannot provide an adequate level of assurance. Testing, for instance, may be insufficient for transactional services, as it is normally performed through a special testing (as opposed to the operational) service interface. Monitoring-based certification may also be insufficient if there is conflicting or inconclusive evidence in monitoring data; such data may, for example, not cover all traces of system events that should be seen to assess a property.

To overcome such problems, I am working on a hybrid approach for certifying cloud service security that can combine both monitoring and testing evidence. For that reason, I designed a new cloud certification approach supporting the automated and continuous certification of security properties of cloud services based on the combination of dynamically acquired testing and monitoring evidence that can deliver the high level of assurance and can overcome the limitations of assessments based on each of these types of evidence in isolation. My approach is based on the cloud certification framework of the CUMULUS EU FP7 project.

1. Introduction

1.1 Overview

In this report, my research along with my work on hybrid security certification of cloud services is presented. More specifically, the subject of my research, the aim, the scope and the objectives behind my research are described in thorough detail, in addition to the problems that I am aiming to solve and the solutions to existing problems that my contribution provides. Finally, a literature review about the current approaches on hybrid certification is provided.

1.2 Motivation and definition of the research problem

Cloud computing is a highly emerging market. However, there are numerable concerns about the security of cloud services and the privacy of data that resides in the cloud. For that reason, the cloud security needs to be increased through rigorous assessments of security by trusted third parties. In cloud computing, the certification of the cloud relies on dynamic forms of assessment, as cloud services are provisioned through dynamic infrastructures operating under security controls and other configurations that may change dynamically introducing vulnerabilities. Additionally, the potential attacks on co-tenant services can compromise the service security. The notion of dynamic assessment includes continuous assessment, unlike the certification of security in traditional systems, where the certification is based solely on static forms of assessment. Recent work on cloud service certification applies dynamic forms of security assessment, namely dynamic testing or continuous monitoring, but the existing dynamic certification approaches are unable to provide the required level of assurance for cloud service security. To overcome the problems mentioned, I am working on a hybrid approach for certifying cloud service security that combines both monitoring and testing evidence. My objective is to certify the security of cloud services by providing executable certification models that can be used to drive the certification process and generate certificates. The evidence that will be used in the certification process is collected through testing and monitoring of these cloud services. The approach described will cover cases where testing is triggered by monitoring and vice-versa, along with cases where testing and monitoring are independent and the outcomes from each form of assessment (i.e. testing or monitoring) do not trigger the other form of assessment.

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [9]. Cloud technology refers to an Internet-based technology through which information is stored in servers and provided as a service (Software as a Service, or SaaS) and on-demand to clients (from the "clouds" indeed) and offers the opportunity for efficient utilisation of resources through the provision of compute, data, storage, and network capabilities as services on demand [1][2]. Over the past decades, several things have changed. At the moment, cloud computing represents a continuing evolution away from the decentralised IT systems of the previous decades and is already transforming providers of IT services, changing the way other industrial sectors provision their IT needs, as well as the way citizens interact with their computers and mobile devices [3]. At the same time, cloud computing is continuously evolving and has emerged as a popular paradigm for harnessing a large number of commodity machines in the cloud, increasing the capacity and adding new capabilities dynamically without investing in new infrastructure, training new personnel or licensing new software [4][5][6].

Cloud computing is one of the most fast growing segments in the IT industry. But as more and more information is placed on the cloud, concerns about security, data privacy and protection start to arise. One of the main concerns is related to the physical location of data stored on the cloud, as cloud computing data spreads across multiple geographic areas [6][7]. Companies outsource the management of security to third parties that host their IT assets. The co-existence of assets of multiple tenants who use the same instance of service, in the same location, along with the lack of security guarantees in the Service Level Agreements (SLAs) between cloud providers and cloud consumers, poses a big threat to the security and the protection of data in the cloud. The probability of attacks when this set of assets is available publicly makes security questionable [8]. Other vulnerabilities, risks and threats associated with security in cloud computing contain loss of governance, data, application and service portability when it comes to migration of data and services, failure of mechanisms separating storage, memory, routing and even reputation between different tenants, compliance risks, data protection issues, information leakage, malicious attacks and management interface compromise, application security and security related to third-party resources [6][9][11][12].

The worldwide public cloud services reached a total market size of $45.7 billion and the International Data Corporation (IDC) expects this market to grow at a compound annual growth rate (CAGR) of 23% until 2018 [13][14]. The certification of cloud service security has become

a necessity due to on-going concerns about cloud security and the need to increase cloud security through rigorous assessments of security by trusted third parties [1][9]. The certification of security in traditional software systems is based on static forms of security assessment (e.g., the Common Criteria model, ISO 27001, ISAE 3402 /SSAE 16 Type II, and EuroCloud Star Audit [15][17]). However, the nature of cloud computing introduces certain characteristics that make the current certification approaches ineffective [16][17]. Cloud computing can deliver a vast array of IT capabilities in real time using many different types of resources. Cloud computing systems are on-demand, automated, elastic, location-independent, and multi-tenant, providing dynamic scalability, on-demand self-service and multitenancy and geographical dispersal of services [16][18][19][20][21]. Consequently, it is important to take into account the dynamics of the cloud and rely on a dynamic certification mechanism.

The certification of cloud service security requires continuous assessment [26]. This is because cloud services are provisioned through dynamic infrastructures operating under security controls and other configurations that may change dynamically introducing unforeseen vulnerabilities (e.g., a zero-day vulnerability in HyperVM server virtualization was exploited to delete 100,000 websites using it for hosting purposes [22]). Furthermore, as cloud services share the infrastructures they are deployed in with other services, their security can be compromised as a consequence of attacks on their co-tenant services (fate sharing effect). An example of this was the permanent data loss of all co-tenant services of Megaupload following the suspension of one of its services due to copyright infringement [23]. Cloud service security cannot be fully guaranteed, according to the existing research, as cloud providers offer different services, such as Software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a service (IaaS), and, each service has its own security issues [28]. Additionally, there is an uncertainty about the security of customers' services once they are migrated to the cloud, as it is not easy for an organisation to control the processes and data when data is outside the perimeter of the company [9][29]. Assuring cloud security is a costly process and cannot be afforded by small enterprises. Last but not least, the introduction of vulnerabilities such as breach of integrity, confidentiality and customer privacy; reduced software protection, application and data availability; and authentication, authorization and accounting vulnerabilities, along with the opaque service offerings that do not allow users to monitor the systems, make the problem of assuring cloud security even stronger [9][29][30][31]. Under such circumstances, the desired security properties need to be assessed continually whilst a service is in operation [25].

It is worth mentioning that the security verification is affected by the interference between the features and the behaviour of inter-dependent services at any of the layers in the cloud stack [9]. The privacy of the data processed by cloud software services may be affected by dynamic changes in them, resulting in non-regulatory compliance and absence of compliance with organisational security policies [1][31]. More specifically, changes in cloud services that happen during run-time can result in unexpected liabilities and inter-dependency issues. At the moment, there is no cloud provider that can guarantee no interaction between cloud features, so the final user of cloud services may accuse the cloud service supplier of being responsible for failing to withhold cloud service properties. Some of these vulnerabilities and violations, though, may be caused by the dependencies between the services and the cloud, that might affect to a great extent the performance of the cloud and the availability of the services at all layers in the cloud, leading to ownership problems [1][9][31]. For that reason, while traditional certification techniques focus on monolithic systems (e.g., the Common Criteria model [15][35][36]) there is the need for service-based certification, which has already been addressed, but it focuses only on certifying services during design-time, failing to address the certification of inter-dependent services, including static and monolithic proofs for services [33][34].

In response to this need, recent work on cloud service certification applies dynamic forms of security assessment, namely dynamic testing (e.g. [24]) or continuous monitoring (e.g. [27][37]), as the certification of different types of cloud services can be effective only if it relies on dynamic forms of assessment, overcoming some of the limitations of traditional security certification and audits (e.g. they produce machine readable certificates incorporating dynamically collected evidence) [38]. Unlike traditional certification approaches, in dynamic cloud service certification, the aim is to produce machine-readable certificates containing a specification of the security property that they certify and the cloud service asset that this property refers to (aka target of certification (TOC)), as well as evidence demonstrating the satisfaction of the examined security property [26][38].

Existing dynamic certification approaches overcome some of the limitations of the traditional security certification and audits. However, under certain circumstances they may also be unable to provide the required level of assurance for cloud service security. Testing, for instance, may be insufficient for transactional services, as it is typically performed through a special interface of the service that is being certified, rather than its normal operational interface [38][39]. Monitoring based certification may also turn out to be insufficient in cases where there is

conflicting evidence within the monitoring data or in cases where such data do not cover all potential traces of system events that should be seen in order to assess a given property [26][38].

In order to overcome the problems above, I have developed a hybrid approach for certifying cloud service security that could combine both monitoring and testing evidence. My approach is based on the cloud certification framework of the CUMULUS EU FP7 project that provides models, processes and tools supporting the certification of compliance and security properties of all types of cloud services, through the use of multiple types of evidence including testing and monitoring [1][38]. The basic idea behind a hybrid certification model is to cross-check evidence regarding a security property that has been gathered from testing and monitoring. If there is no conflict between testing and monitoring evidence, the evidence will be combined. For example, let us suppose that we want to certify cloud service availability. If availability is measured as the percentage of the calls to service operations for which a response was produced with a given time period *d*, a monitoring check should verify exactly this condition. However, the trace of service calls that has been examined by the monitoring process might not cover all the operations in the service interface or the expected peak workload periods of the underlying infrastructure. In such cases, before issuing a certificate for service availability, it would be necessary to test any of the above service usage conditions that have not been covered yet.

1.3 Aims and Objectives

The problem I face is *to enable the automated and continuous certification of security properties of cloud services*. The research objective is *to design a new cloud certification approach and develop a framework supporting the automated and continuous certification of security properties of cloud services based on the combination of dynamically acquired testing and monitoring evidence that can deliver the high level of assurance required for trustworthy certificates, and can overcome the limitations of assessments based on each of these types of evidence in isolation, as monitoring-based approaches are insufficient when there is conflicting evidence, and test-based certification approaches are not effective when there are transactional services.* The hybrid approach that I am proposing will reduce uncertainty arising from reliance on any of the testing and monitoring evidence in isolation.

The steps that will lead to the achievement of the research objective are presented below:

- **Literature Review**

The current security certification mechanisms will be reviewed and I will provide an analysis of the certification models that exist and are used in order to certify service security in the cloud. I will present the different test-based and monitoring-based certification models that are used for cloud service certification, the current approaches, the related work and the state of the art. My analysis will cover the description of the different frameworks, along with their strengths and weaknesses, focusing on the existing issues that will be resolved with my research.

- **Develop a schema for defining hybrid certification models**

Firstly, I will define a schema that will support the definition of executable certification models that can be used to drive the certification process and generate the certificates. The certification of security properties will be based on the collection of monitoring and testing evidence. Monitoring will be continuous, so that it can cover changes at any layer of the cloud that might affect properties already certified, and will cover any changes that might occur. The monitoring evidence will be combined with testing evidence to cover dynamic certification.

- **Develop a certification infrastructure that will enable certification based on hybrid models**

In order to create hybrid certificates, a hybrid certification infrastructure is essential. For that reason, automatic monitoring combined with testing, needs to be supported. The data collected from monitoring and tested need to be analysed to provide the evidence for the hybrid model. The data will be correlated in different ways according to the security property examined. So, I will develop tools and mechanisms that will support the analysis of testing and monitoring data that are gathered and are used as evidence in order to generate and issue the hybrid certificates, based on hybrid certification models. Additionally, the mechanisms integrated will support the management of certificates, making the certificates available to cloud customers and providers.

To achieve this, the existing CUMULUS framework will be used, but it will be amended to support the hybrid certification.

- **Evaluation of the hybrid independent approach**

Research will be accompanied with the evaluation of the hybrid certification approach that will confirm the reliability, trustworthiness and real applicability of the proposed model to the industry.

1.4 Research Contributions

The research presented in this report provides novel hybrid security certification models for certifying services in the cloud, extending the current certification approaches that exist and are currently used in the industry, most of which are based on monitoring or testing.

For that reason, a schema for defining hybrid certification models will be introduced to enable the automatic certification of service properties based on testing and monitoring evidence. Additionally, certain security properties of services in the cloud will be assessed and certified according to the proposed model. The realisation of the above will be made with the provision of a hybrid infrastructure that will support hybrid certification. The proposed models aspire to resolve the gaps arising from current certification models that are based on testing or monitoring.

Additionally, I will contribute to the development of a dynamic certification mechanism that will be able to certify cloud services in any layer of the cloud stack. My approach is based on the automation of collecting data through testing and monitoring, thus making the certification process efficient, effective and easy to be customised by the user.

1.5 Report Outline

The rest of this report is structured as follows.

The second chapter includes the literature review that has been used for my research. The topics that are covered include the software and service certification, the test-based certification, the monitoring-based certification, along with current dynamic certification approaches and the state of the art. Additionally, the strong and weak points of each method are presented, and I describe what issues are solved by the use of the proposed hybrid approach. Furthermore, a comparison between the traditional approaches and the proposed framework is presented, showing how my approach can add value to existing certification schemes. A distinction between trustworthiness and security is presented in the end of this chapter. It is worth clarifying that my research provides higher security in certifying services in the cloud, but advanced security does not always imply advanced trustworthiness, as explained in detail in the end of the second chapter.

The third chapter includes the description of the hybrid approach. In this chapter I explain why a certification authority should use hybrid certification models, and I describe the different modes of hybrid models that exist, showing how they fill current gaps in security certification. Moreover, the framework of CUMULUS that is required to realise the hybrid models and the new capabilities of the framework, will be introduced. Also, a hybrid certification model schema for realising the security properties to be certified, the architecture of the certification infrastructure, and a number of examples of the hybrid certification of security properties, follow.

In the last chapter, the evaluation of my approach is presented. In this chapter, I outline how my proposed approach improves certain gaps that arise from the use of popular security certification approaches. Also, I introduce a section of conclusions where I evaluate my project and describe how it answers the research questions and how it meets its aims and objectives. Finally, the areas for future investigation and research, along with my published work, are presented.

2. Literature Review

2.1 Research Methods and evaluation

The objective of this chapter is to present the relevant literature review and underline the need of a hybrid certification approach. For the purposes of this chapter, I reviewed the state of the art and the existing approaches in the literature and identified the existing limitations, demonstrating how my approach can add value and bridge the existing gaps. First of all, I grouped the certification approaches into distinctive categories depending on the kind of certification they support, their use and their nature. Additionally, I demonstrated the evolution from service and software certification to dynamic certification, where testing is combined with monitoring, paving the way for hybrid certification. A comparison between traditional approaches and my proposed framework is presented in the end of this chapter, underlying the need for hybrid certification mechanisms. The biggest strength of this approach is the fact that a well-rounded overview of the existing certification methods and the auditing mechanisms, is identified, giving the opportunity to the readers of this report to gain a good understanding about different kinds of certification. On the other hand, a limitation of this research method is that there might be approaches with limited references that have been escaped my critical review.

2.2 Overview

Cloud certification is the approach to assess security properties and check one basic set of security requirements of an entity, which can be a software artefact, a service or a whole system. Cloud service certification is a topic that has gained a lot of ground the last years, as security concerns of cloud services arise. The majority of cloud services are complex, making it hard and infeasible for individual customers to check their security requirements, so the concept behind a certification scheme is to have a security certification scheme check all the security requirements for all customers, and, provide sufficient evidence for the validity and assurance of the security properties of the entities to be certified, confirming security accountability and liability [41].

The certification process of services in the cloud supports users in their decisions coping with all issues related to the security and trust in the provision of cloud systems, by having an accredited service authority assess the security properties and the underlying security mechanisms, so that the users can finally select only the services that are compliant to their requirements or specific

privacy frameworks [24]. The content of each certificate depends on the software artefact and the security property examined for this artefact, representing all the entities that are involved in the certification process, such as the entity making the assertion (i.e. certification authority), the software artefact that is being certified (i.e. offered service/platform) and the group of properties that need to be certified [43].

At the moment, there are standard contracts between the customers and the cloud service providers, but they do not protect the customers, as sudden and unexpected changes may arise in the cloud services. Additionally, the Service Level Agreements (SLAs) between the customers and the providers represent the contract, which captures the agreed guarantees, but the specifications of existing SLAs for cloud services are not designed for flexibly handling the performance and the technical requirements of consumer applications [42]. Additionally, SLAs usually offer the expected target values for the services offered without addressing the legal risks or the operational issues that might appear. Traditional certification schemes consider monolithic software components and focus on machine-readable, system-wide certificates to be used at deployment and installation times that do not prove to be suitable for service-oriented architecture environments, so there is the need for a dynamic certification scheme that manages the intrinsic dynamics of services [44]. However, recently, certification has moved to Service Oriented Architecture in order to cover the certification of both functional and non-functional properties and adapt to new environments [45].

At this chapter, I will present the current research and the literature review concerning the different kinds of certification and I will focus on the need for a certification-based approach for cloud service security. I will take into consideration the emerging audit approaches that are based on the logging and reporting of evidence of cloud operations, where certificates are issued by certification authorities, who are independent from the cloud service provider and the consumers.

2.3 Software and Service Certification

Software certification is a very popular concept in certification and contains a wide range of formal, semi-formal, and informal assurance techniques, requiring several dissimilar mechanisms [46]. Recent research has placed much importance on certifying software quality, although it is difficult to certify software quality without a measurement framework/environment

that simplifies the software quality measurement [47][48]. Software security certification builds on existing software assurance, validation, and verification techniques, but introduces the notion of explicit software certificates, which contain all the information that is necessary for an independent assessment of the certified properties [49].

In 1985, the first evaluation criteria appeared in the US under the name "Trusted Computer Systems Evaluation Criteria (TCSEC)", which was followed by a similar standard called "The Information Technology Security Evaluation Criteria (ITSEC)" in Europe in 1991. The four countries that formed this standard were UK, Germany, France, and the Netherlands. The U.S. Department of Defence created the Trusted Security Evaluation Criteria (TCSEC) standard for software security in 1993 [50], which was followed by the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) security certifications standards [51], but they were both very costly as they were national certification schemes. At 1993, the US made an international standard for the security evaluation of IT products and established the Federal Criteria (FC) in corporation with Canada and presented it in the European Commission. Eventually in 1993, the Evaluation Criteria merged the Federal Criteria with its own standard ITSEC and introduced the Common Criteria (CC) for the first time [54]. So, the Common Criteria is considered the first international software certification standard that appeared, and was developed by the governments of Canada, France, Germany, the Netherlands, the UK, and the U.S. [52]. However, it is worth mentioning that the Common Criteria approach is a security standard that can achieve comparability between the results of independent security evaluations of IT products and requires a comprehensible documentation of the software product, including a detailed threat analysis [53]. Common Criteria has two main disadvantages. The first one is the abstraction in the IT product evaluation process, and, the second one is the fact that it is a time consuming process, so eventually it is a costly process [54].

More research in the area of software certification includes a Software Component Certification framework, with the objective of acquiring quality in software components. This framework adopts the Goal Question Metric (GQM) approach to track the software product proprieties, defining a set of metrics to track the properties of the components in a controlled fashion. These metrics will help to measure the component properties, the level of the efficiency of the certification techniques that are used to evaluate the component characteristics and the reliability of the software components [72][73].

In terms of international standards, three international organizations have responsibility for a wide range of technology standards: the International Organization for Standardization (ISO), which is an independent, non-governmental membership organization giving world-class specifications for products, services and systems to ensure quality, safety and efficiency, the International Electrotechnical Commission (IEC), which is the world's leading organization for the preparation and publication of International Standards for all electrical, electronic and related technologies and provides a platform to companies, industries and governments for meeting, discussing and developing the International Standards they require and, finally, the International Telecommunication Union (ITU), which is the United Nation's specialized agency for information and communication technologies (ICT) who developed the technical standards that ensure the interconnection between networks and technologies and improved the access to ICTs to underserved communities worldwide [56][59][60][61].

In terms of national standards of software certification, certain countries have standards bodies that cover part or the whole standards spectrum [55]. One example is the American National Standards Institute (ANSI), that developed the procedures that standard developing organizations (SDOs) use to develop standards, reviewing also the procedures and processes that the SDOs use, and finally approving the standards that SDOs develop to become ANSI standards [56]. The Deutsche Institut für Normung (DIN) in Germany is recognized by the German government as the official national standards body, representing German interests at the international and European levels with responsibility for all standards [56][57]. Additionally, the European Community has a regional standards body, which is the European Committee for Standardization (CEN) and supports the standardisation activities in relation to a wide range of fields and sectors [58].

The previous standards and methods are focused on stable and monolithic systems and not on dynamic environments, which means that they fail to certify services in the cloud, but paved the way for service certification. The wide release of software applications as web services has sparked a big interest in the definition of assurance techniques for services, which mostly focuses on security [62]. Software certification is, thus, becoming the solution to prove security properties [63][64][65], as Service-Oriented Architecture (SOA) applications can support the dynamic environment of service-based scenarios [66][67], supporting the dynamic changes of applications. Each web service has functional, non-functional and behavioural characteristics and the three elements that describe it are the following: the Web-Service Description Language (WSDL), the Simple Object Access Protocol (SOAP), which is a transport protocol for exchange

of information, and the Universal Description, Discovery and Integration (UDDI), which is a registry used to store services [68]. An industry standard execution language that has been implemented for specifying the actions within business processes and web services, is the "Web Services Business Process Execution Language (WSBPEL)", which is a standard executable language for specifying actions within business processes with web services [69].

In service certification, an outcome of a service security certification is a security certificate of a service, which is realized by a language that enables the representation of a certificate in a structured, machine-processable manner, that enables automated reasoning to be performed on them, and, makes it feasible for certified security features to be part of any Service-Oriented Computing scenario [74][75].

The Assert4SOA EU FP7 project is providing a general service certification framework, notation and architecture for supporting certificates in service discovery and composition [70]. The Assert4SOA project produces novel techniques and tools that are fully integrated within the SOA lifecycle for expressing, assessing and certifying security properties for complex service-oriented applications, providing a framework for handling Advanced Security Service Certificates, called ASSERTs [75][76], but cannot support the certification of cloud services. The SERENITY EU FP7 project focuses on providing security and dependability in Ambient Intelligence, but the certificate outcomes fail to include information about the verification of the service-based systems [71]. Another project that deals with the certification of web services is the EU project AVANTSSAR, which proposes a rigorous technology for the formal specification and Automated VAlidatioN of Trust and Security of Service-oriented Architectures and automates this technology into an integrated toolset (AVANTSSAR Validation Platform). [77]. The AVANTSSAR project is a follow-up to the EU AVISPA project, which aims at developing a push-button, industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications [78].

2.4 Test based certification

In service-based environments we need certification schemes that will support the dynamic nature of services and can be integrated within the runtime service discovery, selection, and composition processes [45]. A lot of research has been made on web services testing and the difficulties of automatically generating test cases and assessing the correct functioning of them

[79]. This topic has been considered a very popular area of research and investigation, as web service testing differs from standard testing, because of the interaction with the service during the process. Also, testing is not used for bug-fixing, but for collecting the necessary evidence to prove certain security properties.

Non-functional testing is another area of research covered by several researchers who have focused on testing the non-functional characteristics and requirements of software systems [81][82][83]. Non-functional testing is used for performing automatic approach based on attack scenarios. In non-functional testing, search-based techniques are used to generate the test cases that violate the Service Level Agreements and the Quality of Service (QoS) measured when executing the service with the generated inputs guides the test [84]. Non-functional testing is of great importance, as the QoS of certain functionalities agreed between service providers and service consumers might not be met and, also, because the lack of service robustness, or the lack of proper recovery actions for unexpected behaviours can cause side effects. One more reason underlying the importance of non-functional testing, is the fact that the exposure of services to the Internet might lead to security attacks compromising the security of the services [79]. However, although non-functional testing is not costly (i.e. the developer does not have to pay when testing their own service), non-functional testing isn't realistic, because it doesn't account for the provider's and the consumer's infrastructure, and for the network configuration or load [84].

Specification-based testing for web services has evolved, as WSDL has been extended to foster web service testing, by including input-output dependency, invocation sequence, hierarchical functional description, and concurrent sequence specification [85]. Robust testing ensures that exceptional behaviour is properly handled and that the service reacts properly to such behaviour, but, usually, the error recovery code is not properly tested [79][84][86][87]. There are many works about the testing of conformance of the service implementation with its WSDL specification [88][89] and others based on symbolic transition system to model and test the web service coordination and automatically generate runtime test cases that are applicable to software composition [90][91][92][93][94].

SOA shifted the development perspective from monolithic applications to applications composed of services, distributed over the network, developed and hosted by different organizations, cooperating together according to a centralized orchestration, making it crucial to test services for interoperability, i.e., to perform service integration testing [79]. The area of

security certification has been investigated widely, but the existing works focus on providing human-readable certificates used at deployment and installation time, failing to support service-based scenarios [45]. The first attempt to assess and certify the correct functioning of SOA using security certificates based on signed test cases was made by Damiani [96] and, later, the U.S.-based Software Engineering Institute (SEI) defined a web service certification and accreditation process for the U.S. Army [97]. The works of Papazoglou and Ryu discuss the problem of managing evolving services that are subject to dynamic changes and variations, introducing the need of continuous redesign and refinement of services, as service evolution may in fact invalidate the certificate awarded to a service, thus triggering recertification [66][67][84].

The Quality of Service of web services has been researched by Al-Moayed and Hollunder [97] and a novel QoS model, that aims to perform flexible service selection by relaxing clients QoS preferences, has been proposed [84][98]. Concerning SLA testing, it is worth mentioning that SLA testing deals with the need for identifying conditions for which a service cannot be able to prove its functionality with a desired level of SLA [79]. An approach has been proposed by Di Penta for SLA testing of atomic and composite services using Genetic Algorithms (GAs), according to which GAs generate combinations of inputs and bindings for the service-centric system causing SLA violations [79].

Statistical testing has also been used for the certification of web services. In statistical testing, a model is developed to characterize the population of uses of the software, and the model is used to generate a statistically correct sample of all uses of the software [102]. In statistical testing, a software "usage model" characterizes the population of intended uses of the software in the intended environment and testing, based on a software usage model ensures that the failures that will occur most frequently in operational use will be found early in the testing cycle. The usage model is based on the software specification [102]. Statistical Testing addresses the problem of determining feasible sub-sets of test cases, while maintaining at the same time adequate and sufficient test coverage [103][104][105]. The purpose of statistical testing is to predict the reliability of the test object. Statistical testing is used to measure the reliability and the quality assurance of a software system and web services, as it helps testers prioritise testing effort based on usage scenarios, frequencies for individual web resources and navigation patterns to ensure the reliability of web applications [103]. The usage models are represented via Markov chains. Usage models are built from specifications of the system, by user guides and by observing the user interactions with the existing software. Test cases (i.e. paths through the usage models) are generated from the model taking into consideration the transition probabilities. The next step is

to statistically analyse the results of the tests in order to determine the expected reliability of the system [106]. In general, statistical testing uses a formal experimental paradigm for random testing according to a usage model of software [107]. The biggest advantage of statistical testing is that the model helps us to determine a feasible sub-set of test cases. Statistical testing also guarantees that the failures that occur most frequently in operational use will be found early in the testing cycle. However, statistical testing cannot be used for establishing confidence in predicting that a special failure, which has been uncovered in one service-based application instance, could also occur in other instances. Furthermore, it requires a very high number of test cases to produce statistical sound data, it may not thoroughly test the software's ability to handle exceptional conditions and, on average, the most frequently used operations receive the most testing [104][107][108].

As for dynamic testing, Foster has proposed a run-time certification and compliance mechanism combining static and dynamic model-checking techniques with event monitoring and violation detection, but without reference to cloud computing, focusing only on SOA environment and not considering multi-tenancy, location independency and on-demand provisioning [33]. Additionally, Kuo et al. implemented a dynamic risk assessment mechanism using SaaS web service, but the proposed mechanism is designed for internal use in organizations in order to help security managers realise the security awareness and vulnerability assessment in end-client devices [100].

Regardless of the test method, testing a service-centric system requires the invocation of actual services on the provider's machine and, this leads to several drawbacks, such as prohibitive costs for the users if they have to pay for services on a pay-per-use basis, massive testing that can cause a denial-of-service phenomenon for service providers and high bandwidth use [84]. For these reasons, dynamic testing of web services that will take into account the dynamic nature of web services combined with monitoring techniques is needed. One testing strategy that is suitable for testing service-based applications has been introduced [117]. According to this strategy, changes are captured at the service interface and the system is able to identify changes that occur in service operations and operational arguments in a service description of a test candidate, and the architecture is able to respond to responds to changes of service operation, operation arguments and service composition changes [117].

2.5 Cloud services certification

The certification of cloud services presents a number of challenges, related to the complexity of a scenario where different service delivery models are offered in dynamic environments. One challenge is the fact that the freedom of choice given by dynamic software provisioning on the cloud needs to be suitably reconciled with the apparently conflicting requirement of verifying that software systems have the appropriate assurance levels for the intended purpose. Also, influential security guidelines mandate users to check at the time of use that software systems hold the desired set of security properties [24].

In cloud services, the assurance evidence is collected by a wide range of formal, semi-formal, and informal evaluation methods, including formal verification of compliance to policies, system simulation, testing and monitoring [24]. It should be mentioned that security certification criteria, such as Common Criteria (ISO 15408) evaluate third-party software and provide a widely adopted practical solution to address the trust of cloud services [35], but these criteria are based only on testing and fails to certify security properties that require continuous monitoring, such as the *data-integrity-at-rest* security property, as defined in [160].

Also, ISO/IEC 27002 provides best practice recommendations on information security management for use by those responsible for initiating, implementing or maintaining information security management systems (ISMS) [32], but does not support the constant provision of information about the security of cloud services, failing to certify security properties that require continuous monitoring, such as the *data-alteration-detection* security property, as defined in [160]. Dhiyanesh and Thiyagarajan have focused on third party auditability in cloud storage systems via simultaneous integrity check in order to ensure cloud data storage security, but they do not mention anything about the certification process [109]. Additionally, a framework has been produced for secure cloud computing through IT auditing using checklists, but it does focus on the certification process [110]. Risk Assessment-as-a-service has been introduced to enable the evaluation of security risks in the cloud environment [111], but is only focused on the risks associated with security. A ComCert approach for automated compliance certification of cloud-based business processes has been proposed, but it only focuses on regulatory requirements and not on automated certification [112]. Providing security assurance through IT audits has been the norm in industrial practice, but, IT audits focus on providing guidelines for inspection of security controls on IT and cloud infrastructures, and, they are not automated [110][113]. Testing has been also used to develop a test based security certification

scheme for cloud services, but fail to certify security properties that require continuous monitoring such as the *data-integrity-at-rest* security property [39][160].

The TRUSTe EU Safe Harbor Seal Program is a European Program that verifies the compliance with the Safe Harbor Framework, including alternative dispute resolution, and helps organizations to get ready for self-certification with the U.S. Department of Commerce. The "TRUSTed Cloud Privacy Certification", but focuses only on certifying the privacy functionalities supported by service providers [114].

2.6 Monitoring based certification

Monitoring observes services or service-based applications during their actual use or operation [118][119]. Monitoring can provide statements about a service-based application's current execution and can uncover failures which have escaped testing, because the concrete input that lead to the current execution trace might have not been covered by any test case [118], in contrast to testing which aims at providing more general statements about services or service-based applications [120]. With monitoring, run-time verification is enabled and we can easily determine if the current execution preserves specified properties [118]. Generally, a monitor is a system that observes the behaviour of a system and determines if it is consistent with a given specification, by taking an executing software system and a specification of software properties and checking that the execution meets the properties. Monitoring is concerned with actual transitions between states, and not with possible transitions [118][121].

According to existing monitoring approaches, there are different techniques used, ranging from monitoring service compositions and workflows to monitoring infrastructures for serviced-based systems, monitoring individual software services and monitoring cloud services. Concerning cloud monitoring, the most widely covered topic is performance monitoring [125][126][127][128][130][131]. A configurable cloud certification framework allowing the definition and realisation of different monitoring based cloud certification models, is described in [37], in addition to systems that support cloud security monitoring [135][136][137]. With monitoring, cloud incidents can be detected by different key points of cloud infrastructure such as VMs of cloud users and data storage components [136][137]. Additionally, systems that support the monitoring of network level metrics and the detection of SLA violations have been developed [131][132][133]. The EVEREST EU FP7 Project is an open source system supporting

the monitoring of security properties of software and cloud services, including basic security properties such as confidentiality, integrity, availability, and the application of appropriate access controls at different layers [37]. At a hypervisor level, monitoring can provide incident detection, even if monitoring agents are not able to communicate with monitoring systems, as happens in the Amazon's CloudWatch case [138].

According to the existing state of the art, monitoring fails to support the certification of cloud services, as there are vulnerabilities in the provision of cloud services that are related to breaches of integrity, confidentiality and privacy due to multi-tenancy of services, interference between security mechanisms at different layers in the cloud stack and interference between security and cloud virtualisation and optimisation, as well as the dependences between services at different layers in the cloud and their potential dynamic evolution [1][9][31]. It is necessary that monitoring is continuous. There are existing methods that focus on systems with a stable structure operating under stable conditions [15]. Dynamic configuration of the monitoring infrastructure have been introduced in [144] in order to provide uninterrupted monitoring services when the monitoring capabilities that are available in a service based system and/or cloud infrastructure change as a result of dynamic changes in the constituent services of such systems. The on-going certification of cloud services is presented in [39][44][62], where there are references about the certification of cloud-based systems whose conditions can change dynamically, but ignore platform services and the infrastructure involved in the cloud service provision. In terms of international standards, NIST's SCAP specifications and Cloud Security Alliance's Cloud Trust Protocol provide interfaces for extracting monitoring data from clouds [25][145].

2.7 Dynamic certification

The notion of dynamic certification, where testing is combined with monitoring to provide security certifications of cloud services, is at a very early stage. An approach based on service monitoring and statistical testing has been proposed for the certification of web services. In this approach, monitoring data is augmented with results from online testing to make failure predictions with confidence [108]. Kuo et al. propose and implement a dynamic risk assessment mechanism using Software-as-a-Service (SaaS) web services, but the proposed mechanism is designed for internal use in organizations in order to help security managers realize the security

awareness and vulnerability assessment in end client devices [115]. Additionally, monitoring has also been combined with model checking techniques to assess properties of software cloud services [33], by combining static and dynamic model-checking techniques with event monitoring and violation detection, but there is no reference to cloud computing. The only reference is to SOA environments.

A dynamic approach has been proposed in [16]. This approach is based on standards like the Cloud Control matrix by Cloud Security Alliance (CSA CCM), the International Organisation for Standardisation (ISO) 27001/27017, the National Institute for Standards and Technology (NIST) SP800-53 and the EuroCloud Audit questionnaire, focusing on the continuous monitoring of critical parameters of cloud services and data centre organization, on automation of the certification/evaluation process, and on appropriate technical, organizational, legal, and economic conditions for the integration of the dynamic certification into the regular operations of cloud. According to this approach, customers of cloud services can be constantly informed about the actual security and quality state, as well as the compliance with the requirements of the certification, when continuous monitoring and automation is applied to the certification process [116], but this approach does not include testing techniques.

The CUMULUS EU FP7 Project has proposed one more dynamic approach. CUMULUS aims to address the cloud limitations by developing an integrated framework of models, processes and tools supporting the certification of security properties of infrastructure (IaaS), platform (PaaS) and software application layer (SaaS) services in cloud. CUMULUS framework brings service users, service providers and cloud suppliers to work together with certification authorities in order to ensure security certificate validity in the ever-changing cloud environment [117].

2.8 Current Approaches and comparison between traditional and dynamic certificate models

There are several research works in the area of cloud security and many security certification schemes that are currently used. The most famous is the accepted worldwide ISO 27001 for security audit, which provides best practice recommendations on information security management for use by those responsible for initiating, implementing or maintaining information security management systems (ISMS) [24][57][59]. At the same time, security certification criteria, such as Common Criteria (CC), provide a widely adopted practical solution

to address the trust deficit when evaluating and purchasing third-party software [24]. One more solution proposed for certifying cloud services is the Security, Trust and Assurance Registry (STAR) self-assessment certification framework developed by CSA [146], which is requesting from solution providers, non-profit organisations and individuals to enter into discussion about the current and future best practices for information assurance in the cloud. This framework is based on the Open Certification Framework (OCF) structured by three levels of trust [27]. However, the STAR self-assessment certification framework is only limited to monitoring. The Cloud Standards web site is collecting and coordinating information about cloud-related standards under development by the groups [147]. The Open Web Application Security Project (OWASP) maintains a list of top vulnerabilities to cloud-based or SaaS models which is updated since the threat landscape changes [148]. The Open Grid Forum publishes documents about security and infrastructural specifications, along with information for grid computing developers and researchers [149].

The Cloud Controls Matrix (CCM) developed by CSA contains a comprehensive set of controls to assess the information security assurance in clouds, and maps controls to existing frameworks. The CCM has been developed through the Open Certification Framework (OCM). The CCM provides a framework of controls that gives detailed understanding of security concepts and principles that are aligned to the CSA guidance in 13 domains [150]. The Cloud Audit protocol, also developed by CSA provides an automated query interface to cloud services for audit [151]. The EuroCloud Star Audit is a certification scheme for cloud providers that creates more transparency in the market and helps businesses find the right services for their cloud projects [152]. Additionally, ENISA's Cloud Computing Information Assurance Framework [9], along with the Security Recommendations For Cloud Providers by German Federal Office for Information Security (BSI) [154] and the Committee of Sponsoring Organizations of the Treadway Commission (COSO) [153], represent current audit frameworks, but they offer limited degree of automation and do not provide continuous monitoring. Furthermore, the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) method is an approach used to assess an organization's information security needs, focusing on information assets, where an organisation's important assets are identified and assessed, based on the information assets to which they are connected [155]. The previous certification approaches presented fail to support automated and continuous certification of security properties of cloud services and prove to have several limitations. For that reason, they cannot deliver the high level of assurance required for issuing certificates that guarantee the security of services.

At this point, a comparison between hybrid and traditional certification of cloud services is presented. Traditional approaches for certifying security properties rely on manual inspections and audits, proving to be static, inflexible, non-automated and unable to realise the economic dimension that the Cloud entails [16]. As already stated, the STAR self-assessment certification framework allows cloud providers to submit self-assessment reports, when fully implemented [27]. The CCM facilitates regulatory compliance and provides organizations with the needed structure, detail and clarity relating to information security tailored to the cloud industry. It is also specifically designed to provide fundamental security principles to guide cloud vendors and to assist prospective cloud customers in assessing the overall security risk of a cloud provider, integrating the ISO/IEC 27001 management systems standard, but CCM is human-centric, requiring from the companies that adopt it to address the issues that they define critical concerning cloud security and to pre-assess the level of maturity of their systems. The CCM enables the integration, monitoring and management of cloud services through a framework that can take care of the elementary issues regarding cloud security, but it does not support certification as an automated service in the cloud [27][157].

Traditional certification models (i.e. ISO/IEC 27001, NIST) also require manual inspections and are unable to provide the required level of assurance in cloud computing and to fit the dynamic nature of the cloud, focusing on monolithic software components and failing to address on-demand self-service, dynamic allocation of resources and multi-tenancy [16][17][24]. Additionally, traditional certification models lack in trust, transparency and accuracy, as they do not support the constant provision of information about the security of cloud services, unlike my hybrid approach that relies on continuous monitoring and testing and it is focused on cloud services. Security auditing approaches that provide security assurance focus on providing guidelines and are not automated [110][113]. One more drawback is that they require that the consumers rely on third-party auditors for security assurance. Common Criteria (CC) certification uses Evaluation Technical Reports [75]. Consumers can specify their requirements in a document called “Protection Profile” (CC-PP), and vendors can build products that conform to a CC-PP, but automation is not supported [158]. A PP defines an implementation-independent set of IT security requirements for a category of properties [159]. However, CC has a human-centric approach, which is not designed to support automated security certification, targeting static, monolithic systems and requiring a large investment of resources. Also, traditional CC certification does not focus on activities that can be automated, but considers analysing documental evidence provided by the developer.

Compared to non-hybrid models for certifying availability, the hybrid model I will introduce can produce availability assessments of higher confidence as the monitoring and testing evidence can be cross-checked before being used in an assessment (and certificate) and can both be included in a certificate depending on the chosen validation checks. Hybrid models offer also a more extended pool of evidence and possibilities to decide which data are relevant and of sufficient quality, so that they can be taken into consideration for issuing a hybrid certificate. Apart from increasing the confidence level of assessments, hybrid models are also more customisable than traditional certification models, since they offer the choice of deciding how test and monitoring evidence should be correlated, cross-checked and used in assessments.

2.9 Trustworthiness and security cloud certification

In this section, I will outline how security in cloud certification is related to trustworthiness. First of all, trustworthiness in cloud services is considered a very big concern, as trustworthiness is often compromised due to lack of control over the underlying infrastructure when data resides in the cloud. When it comes to selecting cloud services, cloud providers and cloud consumers do not have sufficient knowledge about each other, and this leads to uncertainty because of an implied established level of trust from the side of the cloud consumers, who expect that the adopted cloud services will fulfil their needs and requirements [171]. Trustworthiness represents the perceived level of confidence that data from a resource and, therefore, the particular resource itself, have not been compromised. Determining these aspects is hard due to the dependencies between pieces of data, potential changes in data and resources over time, and the fact that resources potentially conspire with each other [182].

The most popular techniques for selecting cloud services include static descriptions and reputation of these services, not taking into consideration other trust factors, thus, making in it difficult to compare the trustworthiness of cloud services, as it is quite expensive and time-consuming to evaluate trustworthiness, and, also, there is not a universally accepted standardised metric system for the trustworthiness evaluation [171][172]. However, the concept of trustworthiness is very generic and there are several aspects that need to be evaluated when we assess the trustworthiness of cloud services.

Transparency of providers' practices, capacities and processes help to generate evidence-based confidence that everything that is claimed to be happening in the cloud is indeed happening as described and certifications can support this evidence-based confidence, by providing an interesting channel between end-users and service providers, allowing users internal observations of cloud service operations [179][181]. Concerning the trust evaluation models that have been developed, Zhou and Hwang have implemented a PowerTrust model to calculate peer reputation, based on trust scores and on a trust overlay network, where peers in the network rate each other after transactions as their local trust [173]. The PowerTrust model evolved into a GossipTrust model that addressed the limitations of the previous work, and is based on a gossip method to aggregate peers' global reputation [174]. A4Cloud combines risk analysis risk analysis, policy enforcement, monitoring auditing and compliance auditing, providing methods and tools to enable transparency and legal, regulatory and socio-economic policy enforcement, to make stakeholders accountable for the privacy and confidentiality of information held in the cloud [171][175]. Additionally, a Trust Management Framework for differentiating trustworthy cloud services and for protecting cloud consumers, has been introduced by Cloud Armor. This framework relies on decentralized architecture and rates the credibility of cloud services, by using the cloud consumer's perspective and feedback collections and by offering recommendations and feedback to cloud consumer about cloud services [171][176][177]. Moreover, there are certain existing schemes that support self-certification, such as the Cloud Industry Forum (CIF), IT-Grundschutz, Leet and NIST SP800-144 and others that support third-party assessment, such as EuroCloud, Federal Information Security Management Act (FISMA), ISO, Leet, Service Organisation Control (SOC) 1-2-3, McAfee, and TUV, and others that support both self-certification and third-party assessment, such as CSA [178][179]. It is worth mentioning that self-certification fails to provide objective assessment-based trust judgements, as first-time service consumers usually do not trust the information about the attributes of a cloud service, casting doubts on the sources of the attribute assessment [178].

Trustworthy sources of attribute assessment include cloud auditors, accreditors or certifiers [179]. NIST has identified a cloud auditor as a party that can conduct independent assessment of cloud services, information system operations, performance, and security of a cloud implementation, evaluating the services provided by a cloud provider in terms of security controls, privacy impact, performance, but, for certain end-users, even the trustworthiness of a cloud auditor, accreditor and certifier needs to be evaluated by looking into their own attributes and/or policies, requiring a certification authority to accredit and monitor the certifiers [179][180]. These transparency

mechanisms provide channels for cloud users to “observe” how cloud service providers operate and help to establish trust by making the cloud services more “visible”, resulting in increased confidence in services and providers’ competency, integrity and goodwill [179]. Monitoring SLAs and the quality of standards can play an important role to trust verification, as it provides additional evidence about the trustee’s competency, integrity, and goodwill, while testing can provide tools and continuous testing processes for compliance with the scheme requirements [179].

My research and contribution in hybrid cloud services certification focuses on providing enhanced security when certifying services in the cloud, and not on increasing the trustworthiness of cloud services. The confidence level of trust is relative and changes over time. Assessing the trustworthiness of cloud resources is hard because of the complexity involved in managing the individual resources and in determining trust relationships [182]. However, it can be implied that increased security can lead to increased trustworthiness in service certification. Nevertheless, as trustworthiness is based on several subjective factors, such as the trust of the cloud consumer to the services offered by the cloud provider, increased security does not always lead to increased trustworthiness.

3. The Hybrid Certification Approach

3.1 Research methods and evaluation

The objective of this chapter is to present the hybrid certification approach that I propose, covering the gaps that arise from the use of existing certification approaches. For that reason, firstly, I reviewed the existing framework that is used in the CUMULUS project and I analysed its strengths and limitations. This gave me the opportunity to identify the existing gaps of that framework. More specifically, the gaps in the testing and monitoring certification model schemas for realising the cloud properties led me to focus on developing a powerful hybrid certification model schema that will realise these properties and will be able to bridge the aforementioned gaps. So, a hybrid certification language was formed to cover the gaps arising from testing and monitoring certification. Certain elements of the framework of CUMULUS project that fit the hybrid purposes have been modified and redesigned, whereas other elements have been designed from scratch and are newly introduced in this chapter. After the description of the hybrid approach, I present the architecture of the certification infrastructure that is an extension of the CUMULUS architecture and I outline a number of examples that demonstrate the hybrid certification of cloud security properties. Thus, I selected certain security properties that fail to be certified by other certification approaches, in order to show how my hybrid solution certifies security properties that other approaches fail to certify. The biggest strength of this approach is that the existing gaps of the CUMULUS framework are identified and an advanced framework is proposed. However, the biggest limitation of this method is the fact that my approach is focused on addressing the needs of the CUMULUS EU FP7 Project, and fails to support multi-layer hybrid certification. Admittedly, this could be an area of investigation for future work.

3.2 Overview

Hybrid certification combines evidence gathered through testing and monitoring to collect the elements that are required to certify a security property. The hybrid approach that I present overcomes the limitations of each on of the individual certification processes (i.e. testing or monitoring in isolation). Continuous monitoring allows the non-stop observation of the system without interfering with it, but can insufficient in certain cases, as it can make the verification

infeasible. For example, monitoring requires testing agents to verify signatures and inject traffic. On the other hand, testing can be powerful in pre-production environments, but on live systems it can interfere with business operations. For these reason, I decided to introduce an approach that will be based on the combination of testing and monitoring.

My work on hybrid security certification is part of the CUMULUS EU FP7 Project, which focuses on continuous, multiple evidence and multi-layer cloud service security certification. In CUMULUS, certification is a process that is carried out according to a certification model [165]. This model defines: (i) the security property to be certified, (ii) the cloud service that this property applies to (aka target of certification (TOC)), (iii) the evidence that should be used to assess the property, (iv) the conditions that determine the sufficiency of evidence for issuing a certificate for the property, and (v) ways to treat conflicts in the evidence.

In my approach, the certification process is a function that takes as input certain assertions about a security property and a target of certification and produces as an output a machine-readable certificate that contains evidence that prove the security property. The evidence is collected through testing and monitoring. Testing can be offline or online. In offline testing, the collected evidence show that a test has been performed on the cloud software and that a specific result has been produced. In online testing, only when a specific service is invoked, then the tests are carried out. Monitoring is continuous, so the monitoring evidence required for assessing the security properties is acquired through continuous monitoring.

The certification process starts when a service provider makes a request to the certification authority. This certification process is driven by certification models that define the security property to be certified, the evidence that needs to be collected, the way the different bodies of evidence will be correlated, the lifecycle of the certificates and, finally, the involved agents that have the responsibility for this process. When enough evidence for verifying a security property is collected, then a certificate is issued. The certificate is an instance of this security property. More specifically, the certification model schema defines the security property that needs to be certified, the types and the extent of evidence that need to be collected for the certification of the property, the ways the consistency of the evidence is checked, the way further exploration will take place in the event of inconsistencies between the gathered evidence and, how the certification decisions will be made. It also provides an assessment scheme that is responsible for defining the evidence that will be used for the assessment of the security property and for specifying the evidence consistency conditions. When the evidence is collected, then my

proposed model will compare the different types of evidence and will perform the cross-checking. If the results gathered from testing and monitoring comply with each other, and if enough evidence for issuing a certificate is collected (sufficiency conditions satisfied), then a hybrid certificate is issued. Certificates need to be updated, as the operational conditions of the cloud service might change.

The definition of hybrid certificates is realised through an XML schema for specifying the elements needed for generating the hybrid certificates. The XML schema supports the definition of executable certification models that can be used to drive the certification process and generate certificates.

3.3 Background

CUMULUS offers a monitoring infrastructure for realising monitoring based certification models based on EVEREST [37]. EVEREST enables the monitoring of runtime events produced by distributed systems based on rules and assumptions expressed in an Event Calculus based language, called EC-Assertion [37]. Rules express conditions that must be satisfied at all times by runtime events, whilst assumptions express the ways of deducing information from such events (e.g. the state of the monitored system). Both rules and assumptions are defined in terms of events and fluents. An event is something that occurs at a specific instance of time and has instantaneous duration. Fluents represent system states and are initiated and terminated by events. The basic predicates used by EC-Assertion are:

- *Happens(e,t,[L,U])* – This predicate denotes that an event *e* of instantaneous duration occurs at some time point *t* within the time range *[L,U]*. An event *e* is specified as *e(_id,_snd,_rcv,TP,_sig,_src)* where *_id* is its unique id of it, *_snd* is its sender, *_rcv* is its receiver, *_sig* is its signature, and *_src* is the source where *e* was captured from. *TP* is the event's type. EC-Assertion offers three built-in event types: (a) captured operation calls (REQ), (b) captured operation responses (RES) and (c) forced operation execution events (EXC), i.e., operation executions triggered by the monitor itself.
- *Initiates(e,f,t)* – This predicate denotes that a fluent *f* is initiated by an event *e* at time *t*.
- *Terminates(e,f,t)* – This predicate denotes that a fluent *f* is terminated by an event *e* at time *t*.

- *HoldsAt(f,t)* – This is a derived predicate denoting that a fluent *f* holds at time *t*. *HoldsAt(f,t)* is true if *f* has been initiated by some event at some time point *t'* before *t* and has not been terminated by any event within *[t',t]*.

Concerning the testing mechanisms, CUMULUS offers a distinction between *offline*/*static* testing when test cases are performed in a pre-production cloud environment which is a replication of the in production cloud and *online/dynamic* testing when test cases are executed in production environments periodically when a specific event occurs or when specific conditions are met.

3.4 Description of Hybrid Certification Models (dependent-mode & independent-mode)

The key concept underpinning a hybrid certification model is the need to have both testing and monitoring types of evidence, as none of them would be sufficient in isolation for certifying a given property. One very important aspect is the cross-checking of evidence regarding the assessment of a security property that have been gathered from testing and monitoring and, provided that there is no conflict within the evidence, to combine it providing assurance for properties. Consider, for example, a scenario where the property to be certified is cloud service availability. If availability is measured as the percentage of the calls to service operations for which a response was produced with a given time period d, a monitoring check should verify that responses are produced within the required time limit. However, the trace of service calls that has been examined by the monitoring process might not cover all the operations in the service interface. In such cases, before issuing a certificate for service availability, it would be necessary to test any of the above service usage conditions that have not been covered yet.

The combination of monitoring and testing can be attempted in two basic modes:

(1) The **dependent mode** – In this mode, a security property is assessed for a TOC by a primary form of assessment (monitoring or testing) which triggers the other (subordinate) form in order to confirm and/or complete the evidence required for the assessment. More specifically, when not enough evidence is gathered by one form of assessment, or when we just want to confirm the evidence gathered, then the certification mode will trigger the collection of evidence from the other form

of assessment so that enough evidence will be acquired to proceed with the certification process.

For example, let us suppose that we want to assess *data integrity-at-rest* security property. As defined in [160], this property expresses the ability to detect and report any alteration of stored data in a target of certification (TOC). With monitoring we can capture requests for an update operation and the relevant responses to them, along with the existence of events that show the call of an operation to notify the receipt and execution of the update request. However, we cannot capture updates of data that might have been carried out without using the update interface assumed. For this reason, the hybrid model will be based on periodic testing to detect if stored data have been modified and monitor the periods between the tests that revealed data modifications to check if appropriate notifications have also been sent. In this case, both types of evidence will be combined. Data modifications could be detected by obtaining the hash value of the relevant data file in the TOC periodically.

(2) The **independent mode** – In this mode, a security property is assessed for a TOC by both monitoring and testing independently without any of these assessments being triggered by outcomes of the other. Then at specific points defined by the evidence sufficiency conditions of the certification model the two bodies of evidence are correlated and cross-checked to complete the hybrid assessment. The evidence is collected through independent tests and through monitoring that take place in parallel.

Let us suppose that we are going to assess the *availability* security property. As defined in [160], this property expresses the percentage of successful requests processed by the TOC over the total number of submitted requests and is measured by the percentage of calls that satisfy this condition over an assessment period. An independent hybrid model for the certification of TOC availability could be based on collecting evidence regarding the availability of a TOC through monitoring and testing independently (i.e., without any of these activities being triggered by outcomes of the other) and then correlating and cross-checking the evidence.

A non-hybrid certification model includes all the elements required for the certification of a security property and is used to specify the security property that needs to be certified, it provides the formal definition of the security property (assertion) and describes the elements that refer to the monitoring and test-based certificates. In addition to what a non-hybrid CUMULUS certification model defines, a hybrid certification model will define: (a) the mode of hybrid certification; (b) the way of correlating and analysing monitoring and testing evidence; (c) conditions for characterising these types of evidence as conflicting, and (d) the way in which a final overall assessment of the property can be generated based on both types of evidence.

3.5 Hybrid certification model schema description

In the following section, I will provide a more detailed description of the hybrid certification model schema. The schema is based on collecting evidence to certify the availability of a TOC through monitoring and testing, and then, on correlating and cross-checking the evidence collected. The hybrid certification model schema will be based on the test-based certification model schema and on the monitoring-based certification model schema that have been developed for CUMULUS and are defined in [166][167].

For that reason, the test-based and the monitoring-based certification model schemas will be presented, and the elements of these schemas that have been used or refined in the hybrid schema will be explained. Moreover, the new elements that have been added to the hybrid schema will be introduced and described thoroughly. Finally, a table summarising all the elements of the hybrid certification model will be provided, and then, a detailed description of the hybrid certification model schema will follow. In this table, I explain which elements are new and which already-existing elements have been refined to fit the hybrid purposes.

Test-based Certification Model Schema

The Test-based Certification Model Schema presented below shows the fundamental artefacts deriving from the model for Static/Offline and Dynamic/Online Test-based Certification (Figure 1). This model is part of the CUMULUS project.

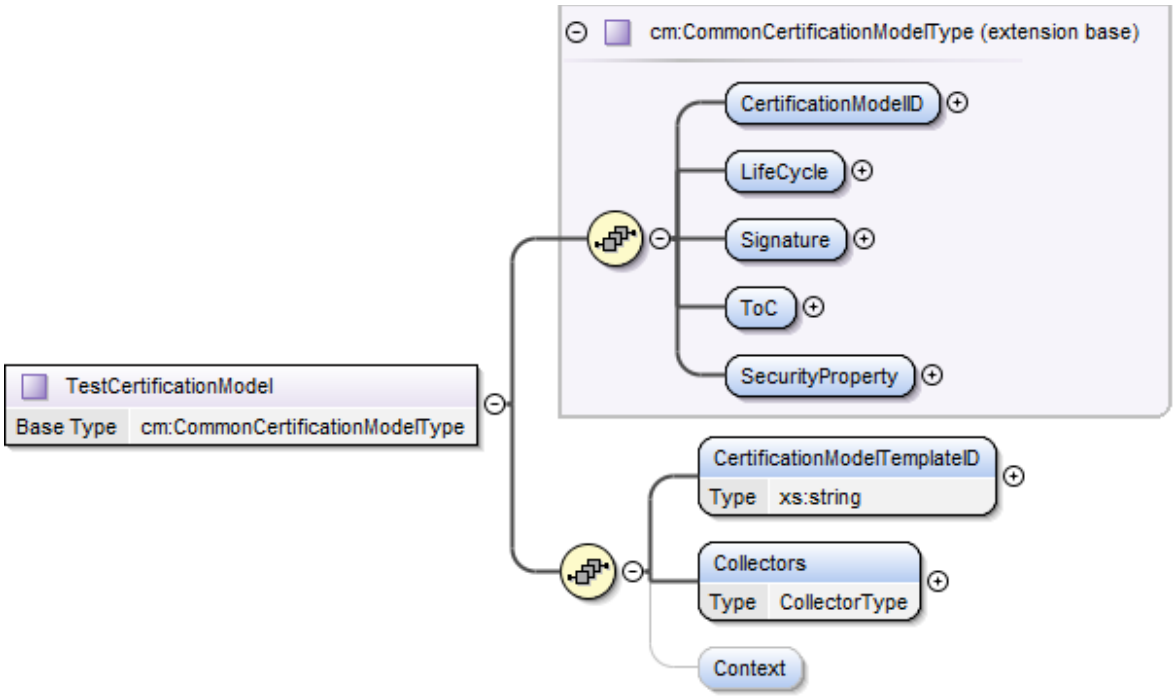

Figure 1 - Test-based Certification Model schema

Monitoring-based Certification Model Schema

The Monitoring-based Certification Model Schema is presented below (Figure 2). This model is part of the CUMULUS project.

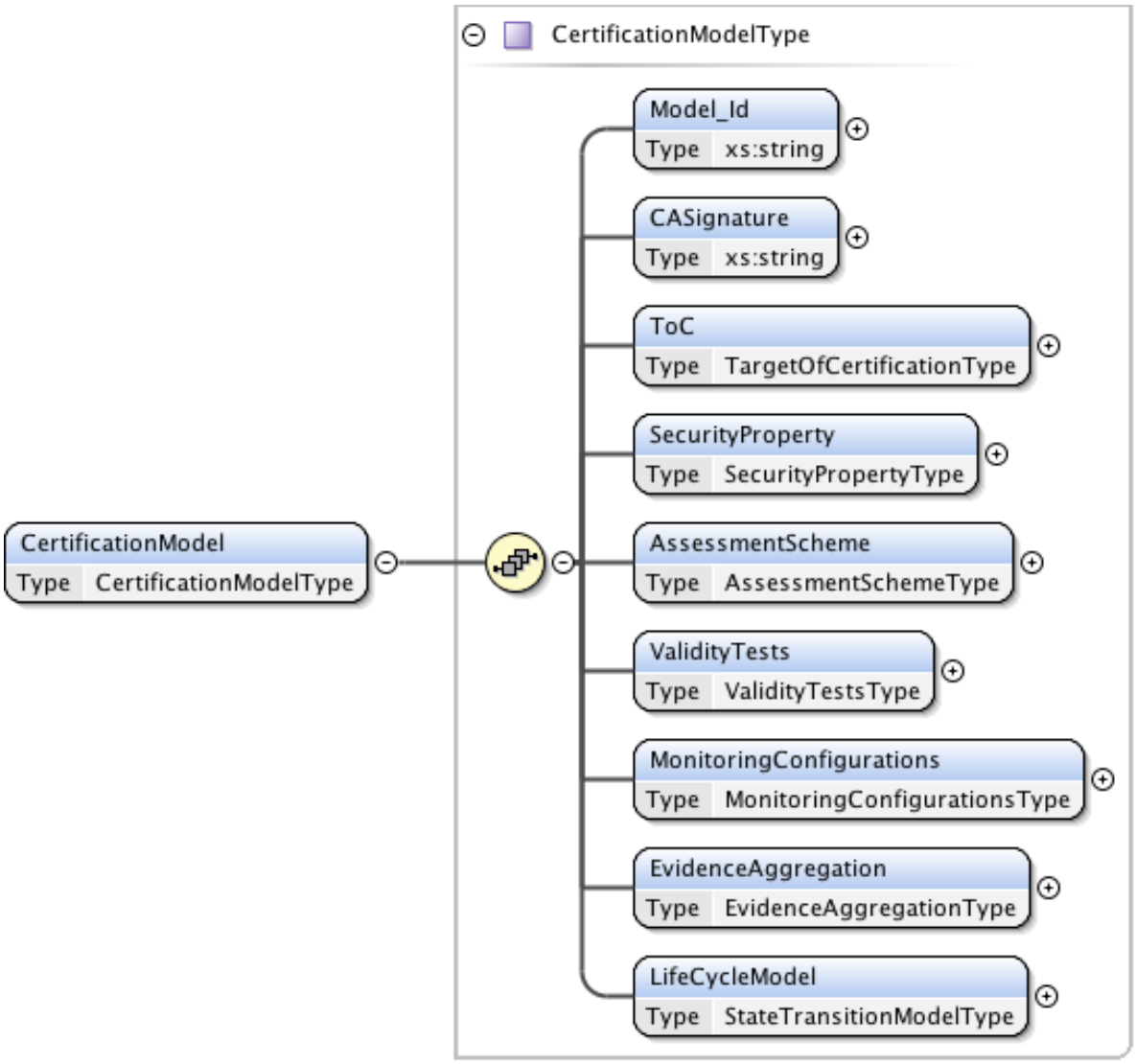

Figure 2 - Monitoring-based Certification Model Schema

The testing and monitoring certification model schemas developed for the CUMULUS project have certain limitations. More specifically, the test-based certification model schema does not include an element to specify explicitly the interfaces that are provided or required for the assessment of the TOC. Also, the test-based certification model schema does not include an element concerning the expiration time of the evidence collected for the assessment of the security properties and does not define a state transition model that can drive the mechanisms behind the lifecycle element that are required when the certificate moves from one state to another. On the other hand, the monitoring-based certification model schema defines different states than the ones defined in the lifecycle element in the test-based certification model schema and uses a different way of defining the TOC and the security properties to be certified. To overcome the above problems and limitations, a new certification model had to be defined that can be applied when certifying security properties for cloud services and evidence are collected through monitoring and testing. So, elements from both certification model schemas were used in order to have a common way of certifying security properties when monitoring and testing are performed. At this point, I will present the top layer of the hybrid certification model schema that I proposed for hybrid certification is presented below (Figure 3). The elements of the hybrid certification model schema will be thoroughly described below.

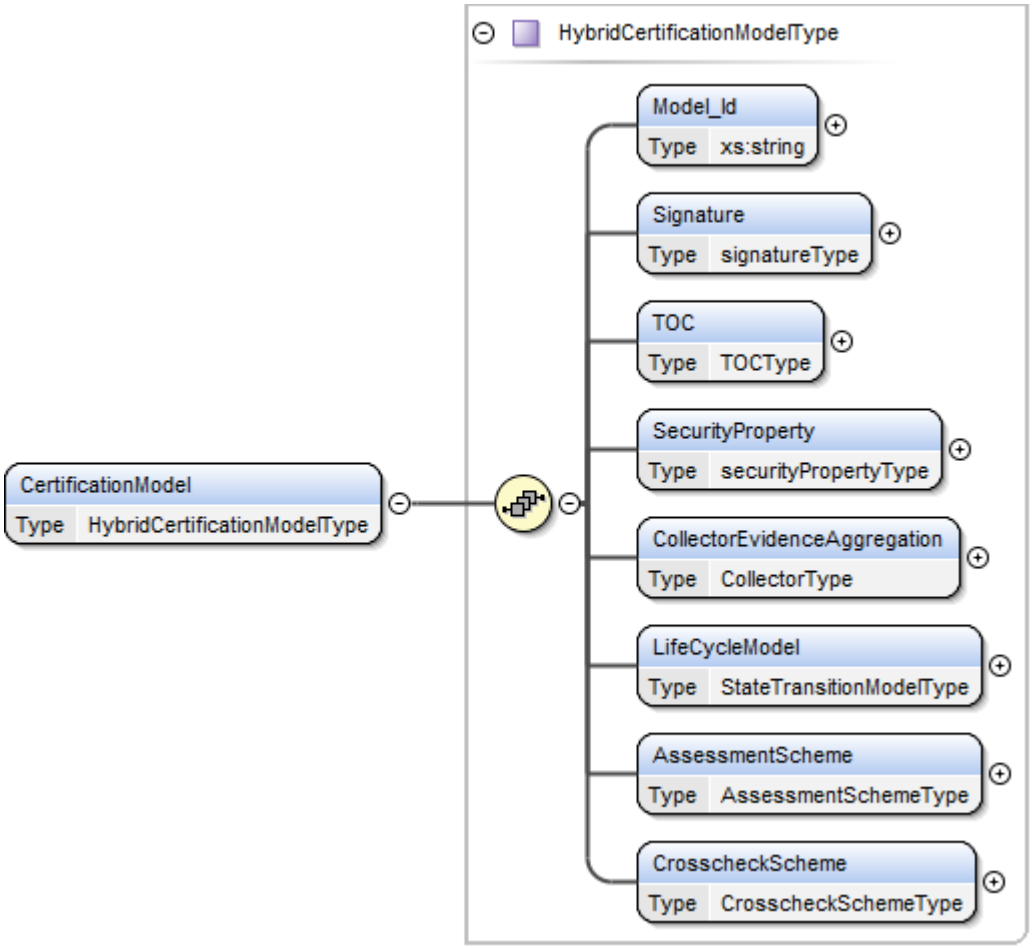

Figure 3 - Hybrid Certification Model Schema

The following table summarises the elements of the hybrid certification model schema and the correspondences between the test-based and the monitoring-based certification model schemas already developed. Furthermore, the following table demonstrates which elements of the CUMULUS framework have been used and modified to fit the hybrid purposes and which elements are newly introduced.

Hybrid based Certification model	**Description**
Model Id	Represents the unique identifier of the certification model instance. This element was included in the Monitoring-based certification model schema and has been used for my approach, as it fits the hybrid purposes.
Signature	Represents the digital signature of the certification authority that has defined or advocated the certification model. This element was included in the Monitoring-based certification model schema and has been used for my approach, as it fits the hybrid purposes.
TOC	Describes the cloud service to be certified by the particular instance of the certification model. For the hybrid schema, elements from the Test-based and the Monitoring-based certification model schemas have been used and refined to form this element.
Security Property	Defines the Security Property, which has to be certified by the Certification Model instance. Security properties are specified through one or more formal assertions. For the hybrid certification model schema, I have used the *SecurityProperty* element from the Monitoring-based certification model schema as it fits the hybrid purposes.
Collectors Evidence Aggregation	Defines how evidence will be collected and aggregated. This element contains sub-elements of the *Collectors* element from the Test-based certification model schema and sub-elements of the *MonitoringConfigurations* element and *EvidenceAggregation* elements of the Monitoring-based certification model schema, which have been refined to fit the hybrid purposes.
Lifecycle Model	Defines the basic stages in the generation and management of the certificates. The *LifecycleModel* element from the Monitoring-based certification model schema has been used, but refined to fit the hybrid purposes.
Assessment Scheme	The core metrics and conditions regarding the assessment of the property to be certified are formally expressed through the assertion(s) included in the definition of this property. The *AssessmentScheme* element specifies the evidence sufficiency conditions, the expiration conditions and the *TriggerDecisionCondition* element. The *EvidenceSufficiencyConditions* and *ExpirationConditions* elements from the Monitoring-based certification model schema have been used and refined to fit the hybrid purposes. The *TriggerDecisionCondition* element is new.
Cross-check	Defines the conditions for cross-checking evidence collected from testing and

Hybrid based Certification model	Description
Scheme	monitoring. This element also contains the deviation conditions and further exploration conditions when more evidence is required for issuing a certificate. This element is new.

Table 1- Elements of the Hybrid Certification Model Schema and correspondence with previous Schemas

3.5.1 Model Id element

The *Model_Id* element, as shown in Figure 4, is the element in the schema that represents the unique identifier of the certification model instance. *Model_Id* is an element of type string. This element was included in the Monitoring-based certification model schema and has been used in the Hybrid Certification model schema as it fits the hybrid purposes. An example of this element in XML is shown below.

```
<Model_Id>cumulus:cm:id:hybrid:00001</Model_Id>
```

The identifier of the certification model is different from the identifier of an instance of the model that is used when the model is applied, in order to certify a given property of a particular TOC.

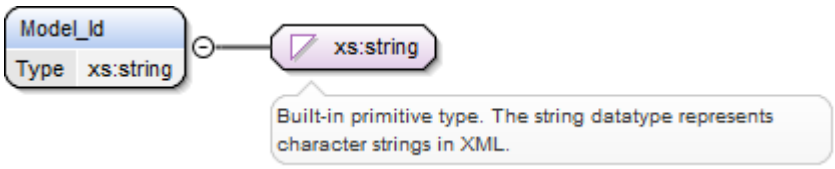

Figure 4 - Model Id element

3.5.2 Signature element

The *Signature* element, as shown in Figure 5, is the element in the schema that represents the digital signature of the certification authority and is an element of type string. This element was included in the Monitoring-based certification model schema and has been used in the Hybrid

Certification model schema, as it fits the Hybrid purposes. An example of the *Signature* element in XML is shown below.

```
<Signature>
      <Name>CUMULUS_City</Name>
      <Role>Certifier</Role>
<Signature>
```

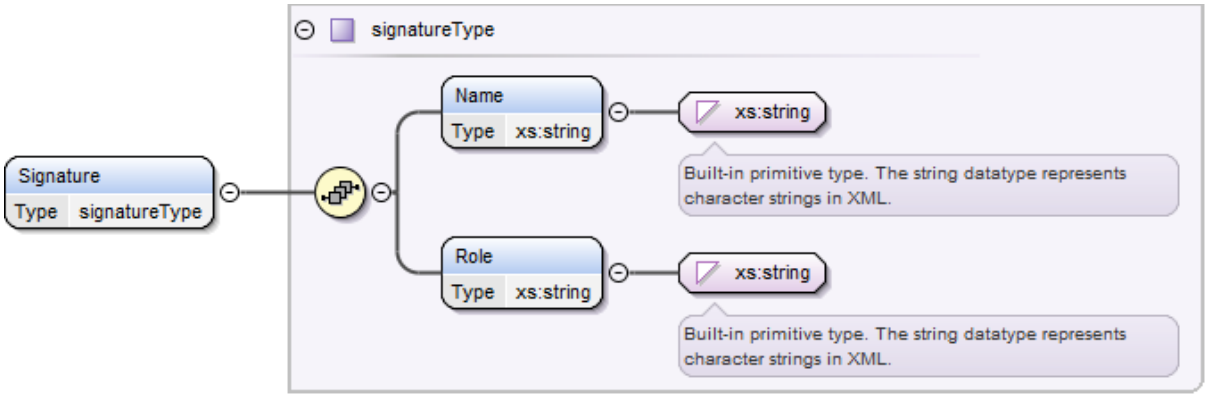

Figure 5 - Signature element

3.5.3 TOC element

The *TOC (TargetOfCertification)* element describes the cloud service to be certified by a particular instance of the certification model, as shown in Figure 6. The *TOC* element allows the definition of the interfaces provided by the cloud stack that need to be further analysed. For the hybrid schema, elements from both testing and monitoring schemas have been used. The *TOC* is an element of type *TOCType*. This element includes an attribute, called "Id", which represents the unique identifier of the TOC and is an attribute generated by the framework, in order to guarantee the alignment for Test-based and Monitoring-based Certification Model (CM) instances. The other sub-elements are: the *ConcreteToc* sub-element that contains a description about the specific concrete instance to be certified, the *Scope* sub-element that describes the generic cloud stack layer (i.e. SaaS, PaaS or IaaS), the *TocDescription* sub-element that contains a textual description about the TOC, the *TocURI* sub-element that contains a URI identifier (i.e. the reference to the services to be certificated) and the *TocInterfaces* sub-element that specify the operations whose execution and results are needed for the certification process.

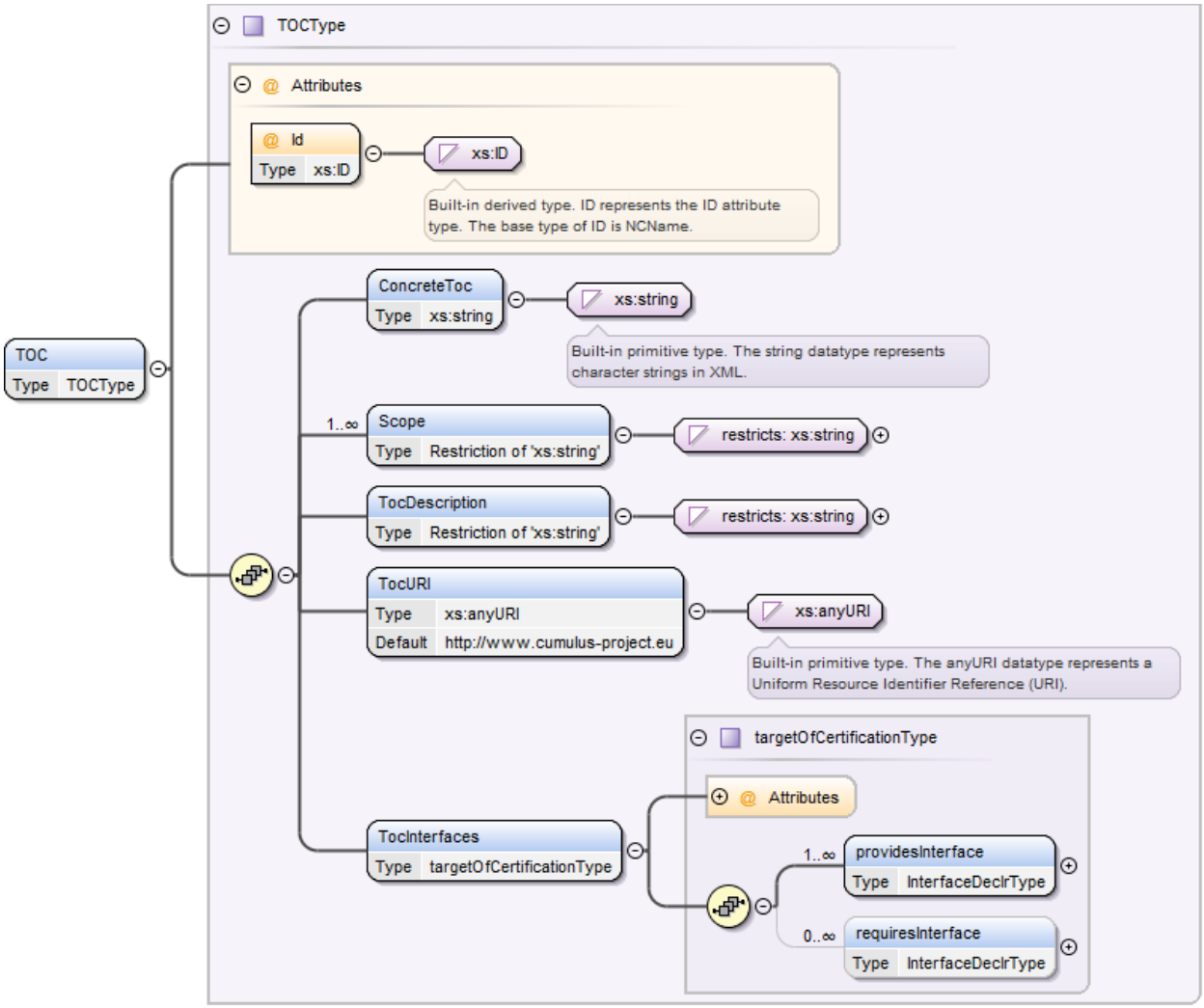

Figure 6 - TOC element

An example is given below:

```
<TOC Id="ID001">
    <ConcreteToc>server</ConcreteToc>
    <Scope>Saas</Scope>
    <TocDescription>Application</TocDescription>
    <TocURI>10.0.0.155</TocURI>
    <TocInterfaces> [.....] </TocInterfaces>
</TOC>
```

The *TocInterfaces* is an element of type *targetOfCertificationType*. This element includes an attribute, called "id", which represents the unique identifier of the TOC, and is mandatory. Also,

the *TocInterfaces* includes a sequence of *providesInterface* and *requiresInterface* sub-elements. These elements specify sets of operations whose execution and results will need to be monitored or tested during the certification process. The *providesInterface* element specifies the interfaces that the target of certification offers itself. The *requiresInterface* element specifies the interfaces that the target of certification expects an external entity to have.

```
<TocInterfaces id="id1">
        <providesInterface>
            [...]
        </providesInterface>
   <requiresInterface>
            […]
        </requiresInterface>
</TocInterfaces >
```

The sub-elements of the *TocInterfaces* element specify how interfaces are described in the XML schema. This element is of *TargetOfCertificationType* and has an *ID* attribute specifying the unique identifier of the TocInterfaces element generated by the framework, and a sequence of *providesInterface* and *requiresInterface* elements, that are of *InterfaceDeclType.*

- The *providersInterface* element as shown in Figure 7 is mandatory, includes a sequence of IDs, provider references (*ProviderRef*), zero or more *Endpoints* that indicate where the service will be invoked and *Interfaces* that define the interface that a TOC itself realises.

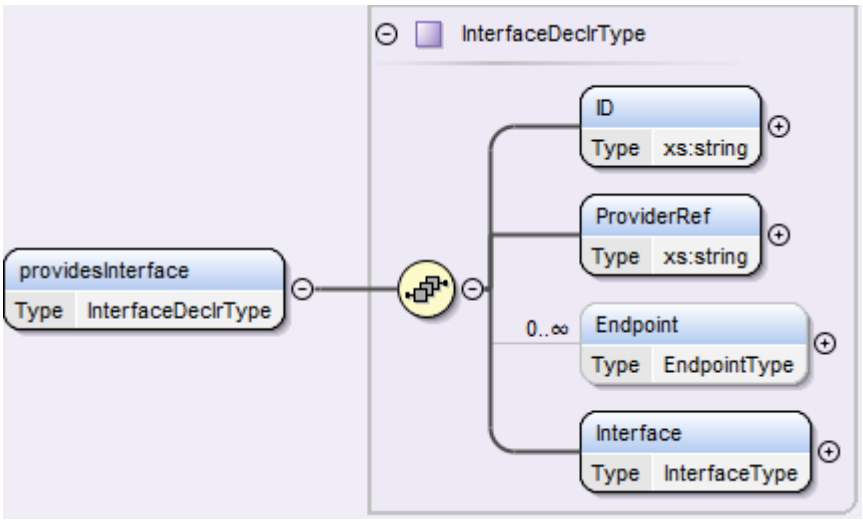

Figure 7 - Provides Interface sub-element

- The *requirersInterface* element (Figure 8) is optional and includes a sequence of *IDs*, one or more provider references (*ProviderRef*) and zero or more *Endpoints* that indicate where the service will be invoked and *Interfaces* that the TOC requires from external entities.

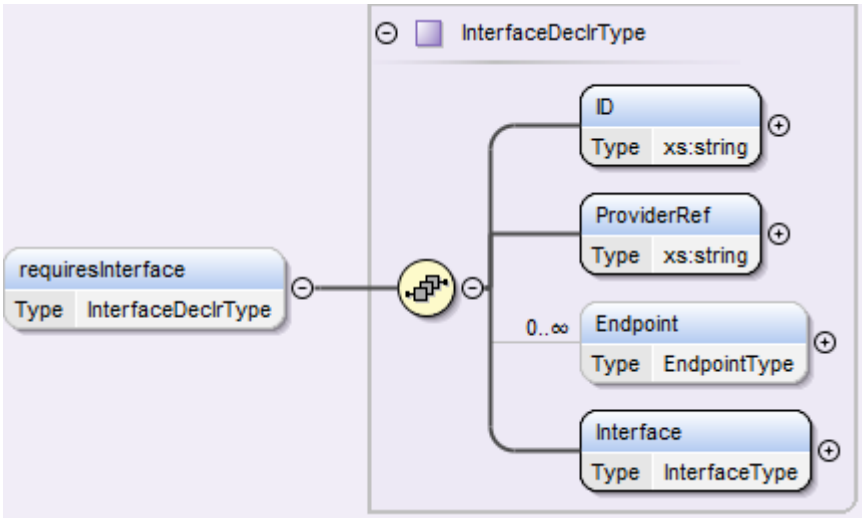

Figure 8 - Requires Interface sub-element

Below is an example of the *providesInterface* and the *requiresInterface* elements in XML. In the following example, the security property that is certified is the *data integrity-at-rest* security property. According to that example, the interface that is provided by this entity is a set of operations for updating data by the service consumer to the TOC. The TOC requires one external interface from the certification authority concerning the authorisation of the service consumer who requested the update operation.

```
<providesInterface>
    <ID>interface::update::a::1/</ID>
    <ProviderRef>Cumulus::provider::a::1</ProviderRef>
    <Endpoint>
        <ID>a111</ID>
        <Location>http://www.cumulus-project.eu</Location>
        <Protocol>SOAP</Protocol>
    </Endpoint>
    <Interface>
        <InterfaceSpec>
            <Name>av::update::a::1</Name>
            <Operation>
                <InterfaceId>update::data::a::1</InterfaceId>
                <OperationId>update::operation::a::1</OperationId>
                <OperationName>update</OperationName >
            </Operation>
        </InterfaceSpec>
    </Interface>
</providesInterface>

<requiresInterface>
    <ID>interface::auth::ca::1</ID>
    <ProviderRef></ProviderRef>
    <Endpoint>
        <ID>b111</ID>
        <Location>http://www.cumulus-project.eu</Location>
        <Protocol>SOAP</Protocol>
    </Endpoint>
    <Interface>
        <InterfaceRef>
            <InterfaceLocation> http://www.cumulus-project.eu </InterfaceLocation>
        </InterfaceRef>
    </Interface>
</requiresInterface>
```

3.5.4 Security property element

For the hybrid certification model schema, I have used the *SecurityProperty* element from the Monitoring-based certification model schema, as it fits the hybrid purposes. A description of the Security Property element will be presented.

The SecurityProperty element, as shown in Figure 9, defines the security property, which has to be certified by the Certification Model instance and is of *securityPropertyType*.

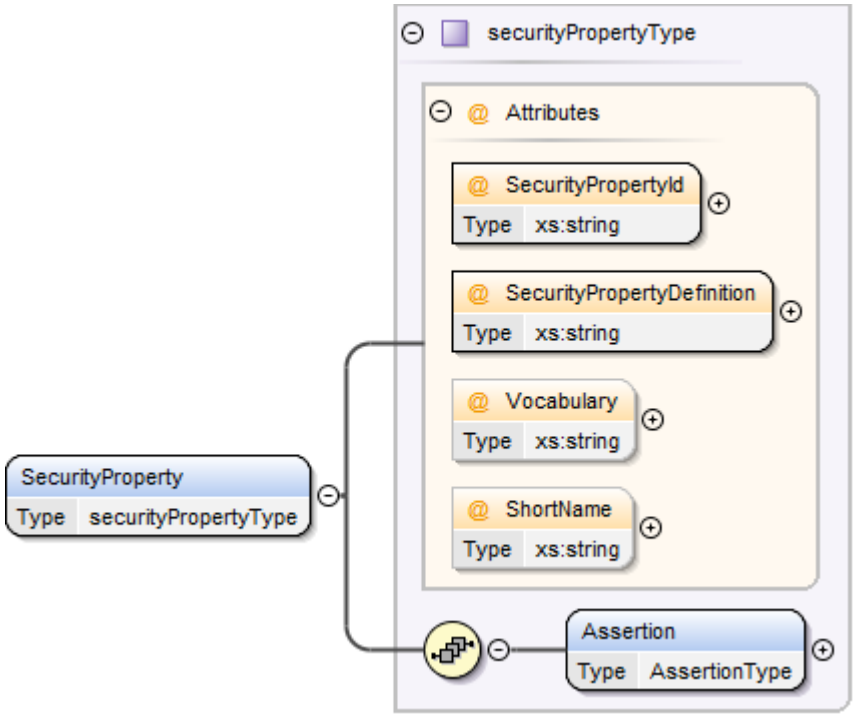

Figure 9 - Security Property element

The *SecurityProperty* element includes a *SecurityPropertyId*, a *SecurityPropertyDefinition*, a *Vocabulary* and a *ShortName* attribute as defined in (CSA, D2.1 Development of security properties specification scheme and security dependency models, 2013). Additionally, it contains an *Assertion* element, which is of *AssertionType*.

An example of the Security Property element in XML is given below:

```
<SecurityProperty SecurityPropertyId=" PercentageOfProcessedRequests">
SecurityPropertyDefinition="percentage of successful requests processed by the TOC over the total number
of submitted requests" Vocabulary=" BCR:availability:percentage-of-processed-requests" ShortName="BCR01"
<Assertion ID="1100"> … </Assertion>
```

The *Assertion* sub-element, as shown in Figure 10, is used to provide the definition of the security property to be certified. *Assertion* is an element of complex type *AssertionType*. According to this type, an assertion consists of the interface declaration that specifies the interfaces that must be certified, the variable declaration that specify conditions within assertions that specify security properties and the guaranteed terms, which define the conditions that must be monitored and tested to guarantee that a security property is satisfied.

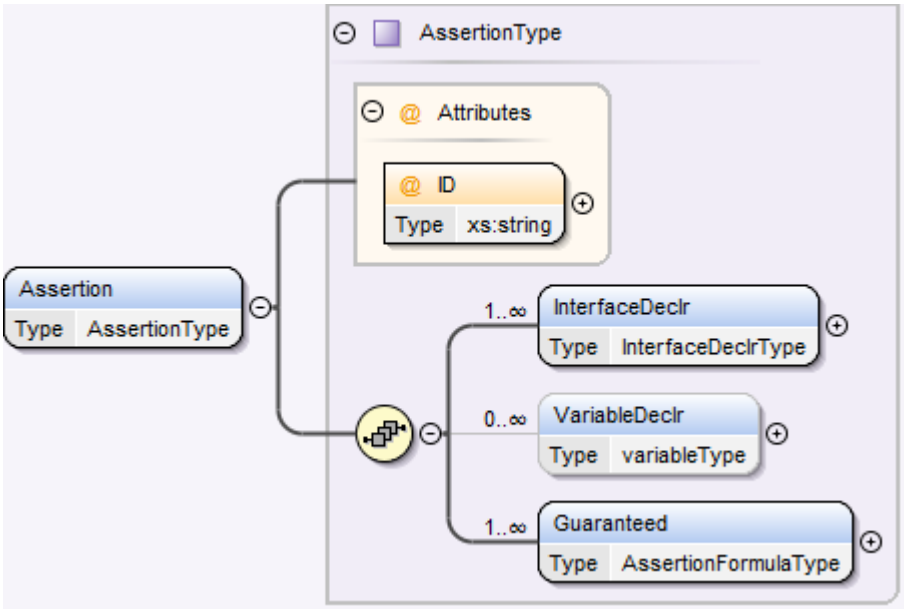

Figure 10 - Assertion sub-element

The element *InterfaceDeclr,* as shown in Figure 11, defines the interface declarations and provides the means to represent the details of functional interface specifications. It includes the mandatory sub-element *ID* that identifies the interface declarations, the mandatory sub-element *ProviderRef* that identifies the party that is provides the interface, the optional *Endpoint* sub-element that specifies the endpoints that implement the specific interface and where the interface operations can be invoked, and, finally, the mandatory sub-element *Interface* that defines the required interface.

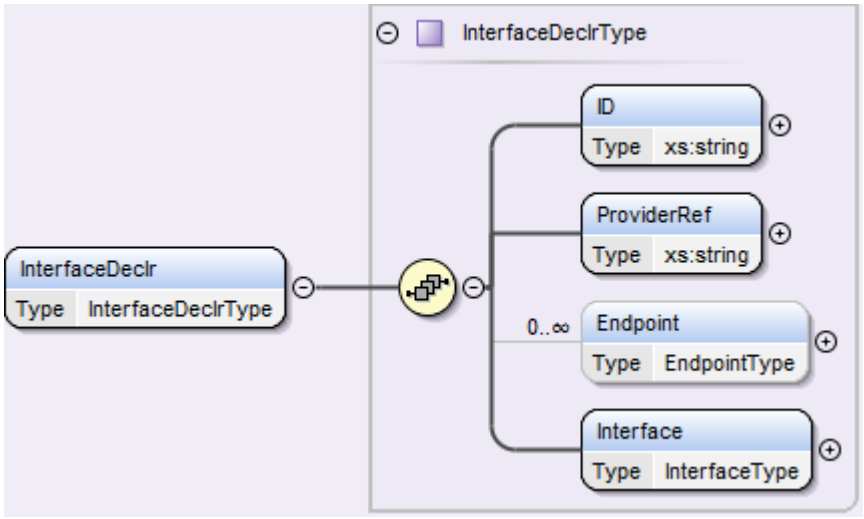

Figure 11 - InterfaceDecl sub-element

The element *VariableDeclr*, as shown in Figure 12, is of type *variableType* and is used to define variables to describe the conditions within assertions that specify formally security properties. This element has two attributes:

i) *persistent,* that indicates whether the value of the variable is the same throughout all instances and

ii) *forMatching,* that distinguishes between internal and external variables

and consists of the following elements:

- *varName,* that is of type *String* signifying the name of the variable;
- either a *varType* of type *String* and *value* element of type *String* or a value element of object type; OR
- an *array* element of type *arrayType* with elements that describe the array structure

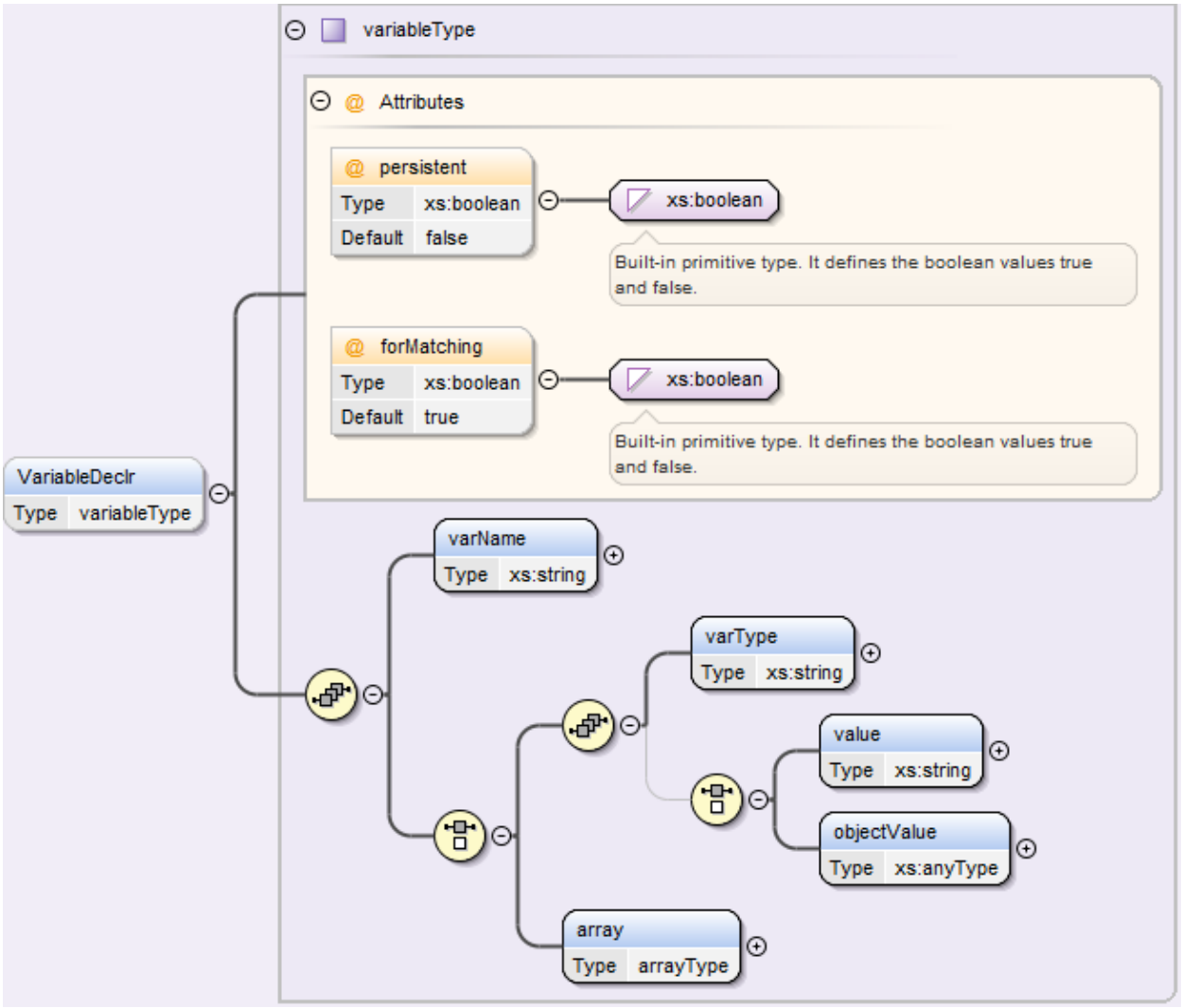

Figure 12 - VariableDeclr element

The element *Guaranteed,* as shown in Figure 13, is of complex type *AssertionFormulaType* and represents a guarantee that a certain state of affairs will hold. The *AssertionFormulaType* has two attributes:

i) the attribute *ID* that is the unique id of the formula, and

ii) the attribute *type* that signifies if the guarantee is an assertion that needs to be checked against information that will arise after the events that will trigger the check of the assertion have occurred (i.e., a "future" assertion), or an assertion that should be checked against information that exists at the time point when the events that trigger the check of it have occurred (i.e., a "past" assertion)

and contains the following sub-elements:

i) a list of *quantification* elements that define the quantifiers for the variables and time variables, used to specify conditions of the assertion

ii) an optional *precondition* element that determines the conditions under which the assertion should be checked (i.e., the conditions which if become true should trigger the checking of the assertion)

iii) a *postcondition* element that and determines the conditions that are guaranteed to hold (i.e. they become true if the preconditions are true).

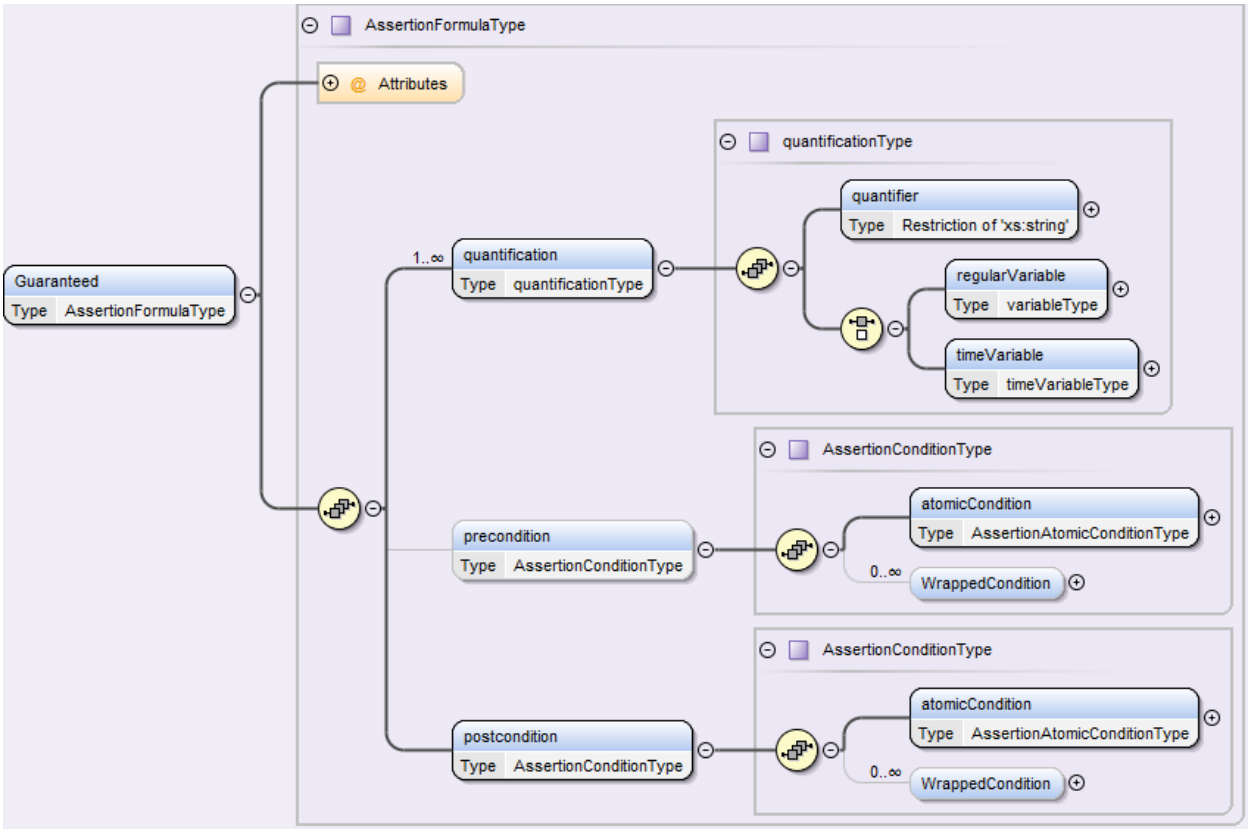

Figure 13 - Guaranteed sub-element

An atomic condition can be of three different types:

- an event condition (i.e., an element of *eventConditionType*) ,
- a state condition (i.e., an element of type *stateConditionType*) or
- a relational condition (i.e., an element of type *relationalConditionType*).

Event conditions are conditions regarding the occurrence of events related to the TOC that the assertion refers to (e.g., the occurrence of an invocation (call) of an operation in one of the TOC's interfaces or a response to such a call). A state condition is a condition about the state of the system that is being monitored at a given time point (e.g., a condition stating a certain user has already logged in to it). A relational condition is a condition about the value of a variable used in an assertion (e.g., a condition requiring a variable to have a certain value or a condition requiring two variables to have the same value).

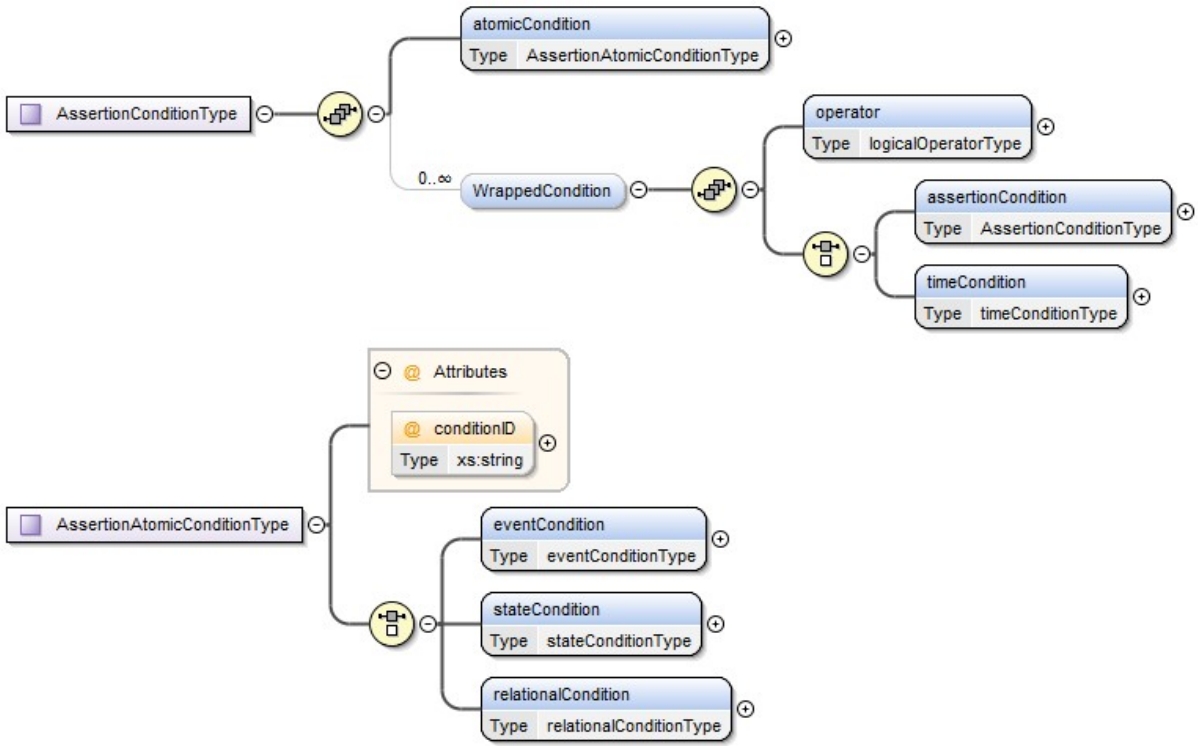

Figure 14 - Condition Types

An event condition is a condition regarding the occurrence of a call of an operation (i.e., a *call* event), a response to a call of an operation (i.e., a *reply* event), or an execution of an operation that must be invoked (i.e., an *execute* event). An event condition includes by the following sub-elements:

(i) an *eventID* that identifies the event of the condition

(ii) an optional *correlatedEventID* which refers to another event that may be related to this event. This is particularly important for my hybrid model as an event

representing the execution of a test can be correlated with an event representing the call

(iii) a *call*, *reply* or *execute* element that specifies if the event of the condition is a call, reply or execute event.

(iv) an element *tVar* that defines the timepoint at which the event is expected to occur. A *tVar* element can be defined as either *timeVar* element that is useful when an event is expected to occur at a specific time instance or a *timePeriod* element that defines the time period. A *timeVar* element consists of a *varName* element of type string for specifying the name of the variable, a *varType* element of type string, which has a fixed value, and, a *value* element of type string for specifying the time instance when the relevant event occurs. A time period element consists of a time *period* and a time period *unit*. The *TimePeriod* element is used when there is an event that is associated with is an execution event (i.e., operation call) that must be executed periodically.

(v) a time range within which the event must occur and is defined by an upper and a lower time boundary. The lower boundary of the time range of an event is defined by the element *fromTime*. The upper boundary of the time range of an event is defined by the element *toTime*.

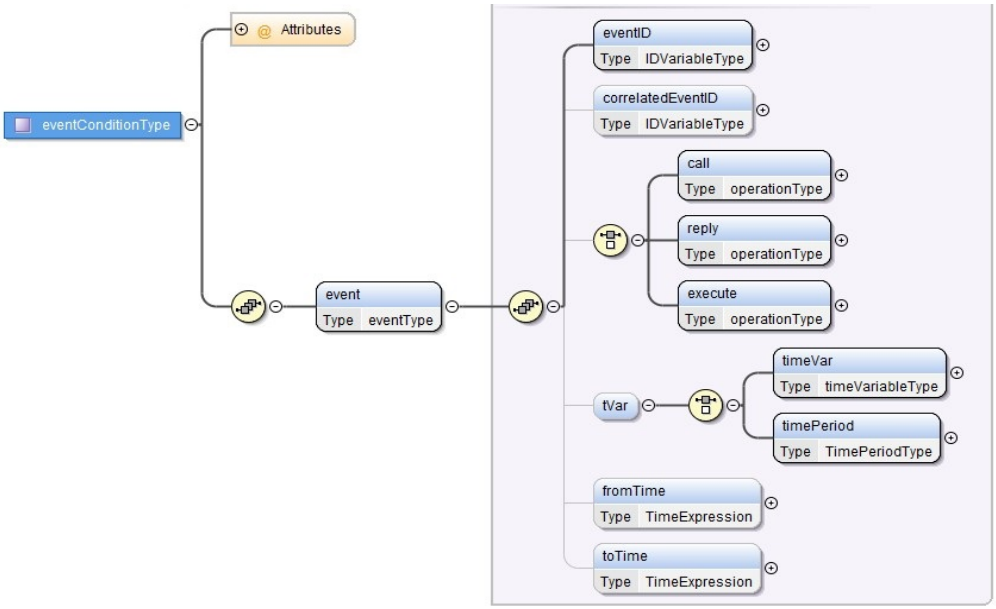

Figure 15 - Event Condition Type

The *call, reply* and *execute* elements are of *operationType*, as shown in Figure 16, and consist of the following elements:

- an *interfaceId* element that specifies the interface that the operation belongs to
- an *operationId* element that specifies the unique id of the operation within the interface
- an *operationName* element that specifies the name of the operation
- zero or more *inputVariable* elements that specify the input variables of the operation.
- zero or more *outputVariable* elements that specify the output variables of the operation.

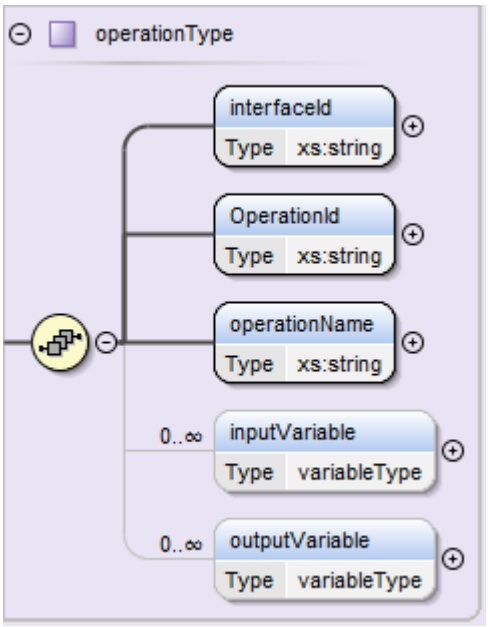

Figure 16 - Operation Type

The second type of atomic conditions is the state conditions, as shown in Figure 17. State conditions refer to the state of the system. These relations can be set up at the beginning of the operation of a system or initiated by events that occur at specific time. These relations describe also the termination of events by other events. When a state condition is initiated by an event, it holds until the time it is terminated by another event.

A *StateCondition* element consists of the following elements:

- an *initiates* element that expresses the initialisation of a state by an event at some time point
- a *terminates* element that expresses the termination of a state by an event
- a *holdsAt* element that represents a system state which it holds at specific time point
- an *initially* element that represents a state value at the beginning of a monitored period of system operation.

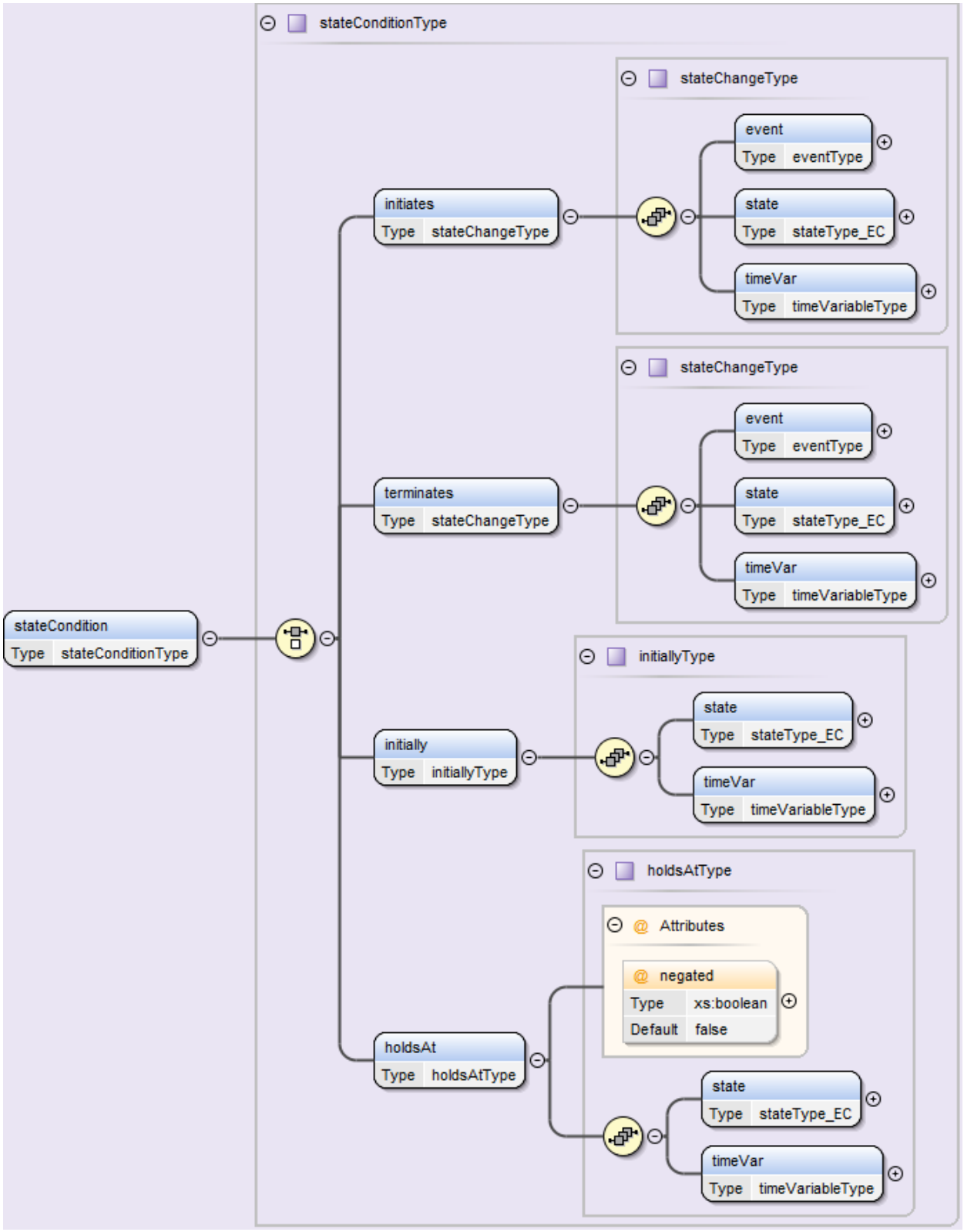

Figure 17 - stateCondition sub-element

The third type of atomic conditions is the relational conditions that enable the specification conditions concerning the values of variables used in the assertions (e.g. a condition requiring that one of the input variables of an operation that has been invoked by a call event has a particular value at the time of the invocation), as shown in Figure 18. The *relationalCondition* element can be one of the following conditions:

- *equalto (operand1, operand2)*
- *notEqualTo (operand1, operand2)*
- *lessThan (operand1, operand2)*
- *greaterThan (operand1, operand2)*
- *lessThanEqualTo (operand1, operand2)*
- *greaterThanEqualTo (operand1, operand2)*

Finally, a relational condition element has also a *timeVar* element that expresses the time point at which the relational condition should hold.

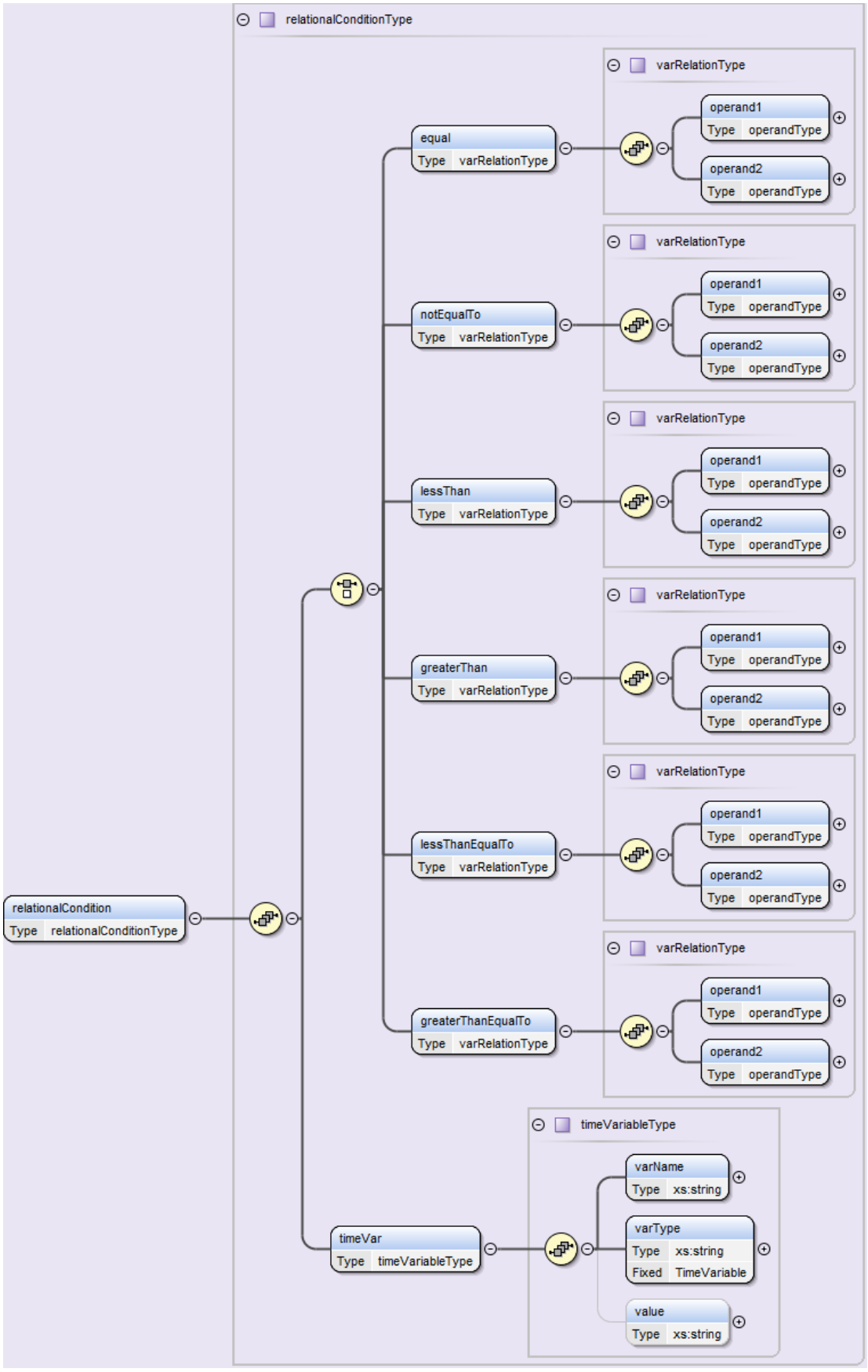

Figure 18 - relationalCondition sub-element

The operands are of *OperandType* and can be variables, operation calls, expressions, event series expressions or constants, as shown in Figure 19.

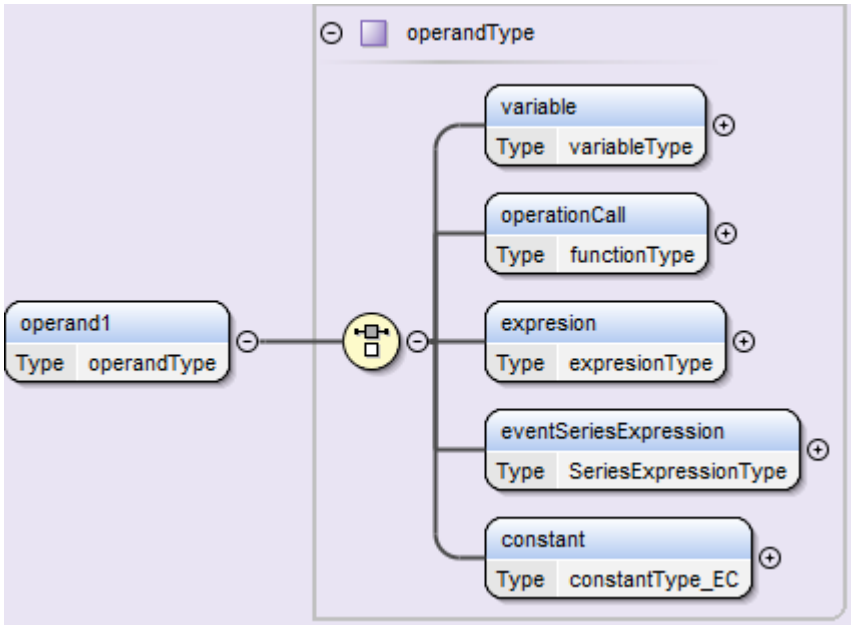

Figure 19 - operandType

An *operationCall* element is of type *functionType,* as shown in Figure 20, and includes the following elements:

- a *name* element that specifies the name of the function;
- a *partner* element that signifies the service providing the function and
- zero or more *argument* elements which may be of one of the following types: an *eventSeriesVariable,* a *variable* element, a *constant* element or a *function* element of *functionType*. An operation call element defines a function that needs to be executed. When a function operand is encountered during the evaluation of a relational condition, the values of its arguments are established first and then the function is executed.

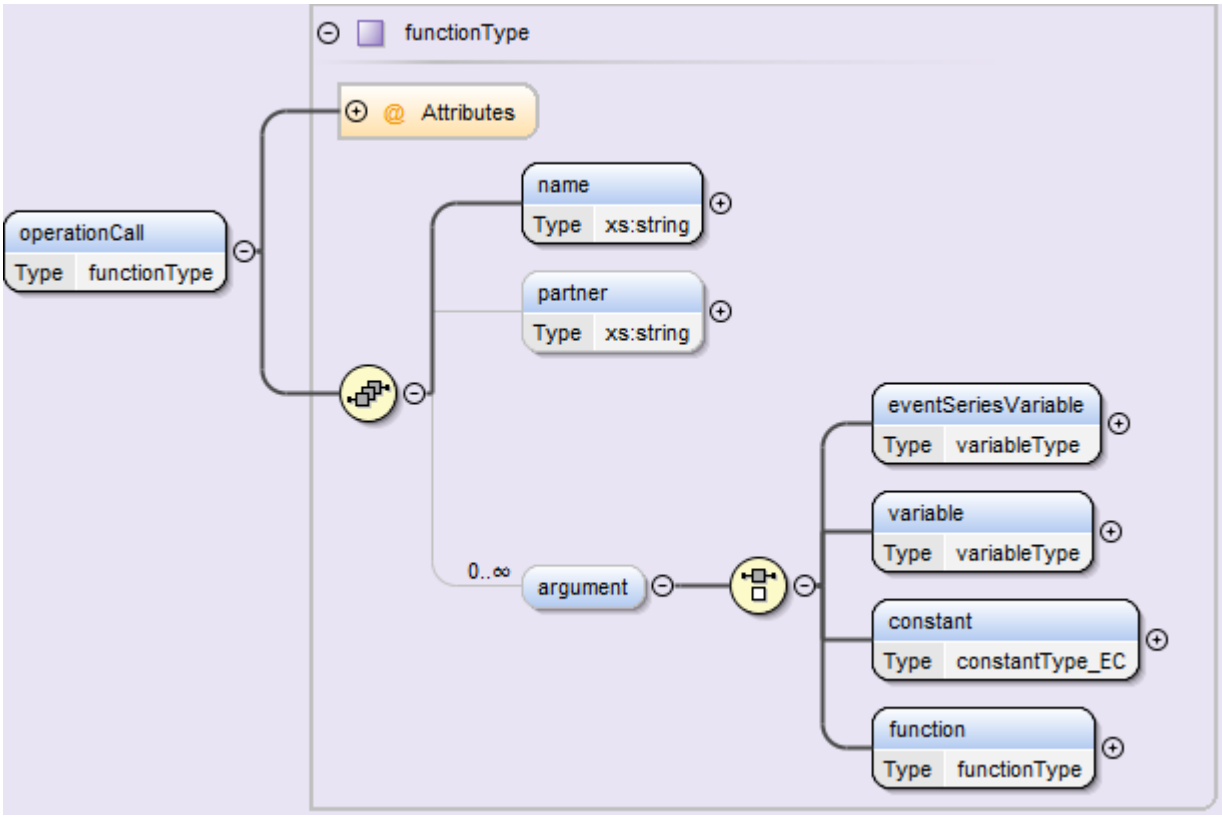

Figure 20 - operationCall sub-element

The *eventSeriesExpression,* as shown in Figure 21, specifies the computation over a series of values of some arguments. This element consists of the following elements: (a) an *eventSeriesCondition* element that signifies the conditions that produces the series values and (b) a *computation* element which signifies the computation that should be performed over the series values. The *computation* element contains either an *execute* element or a *function* element. The *eventSeriesExpression* element supports the cases where the execution of tests is required.

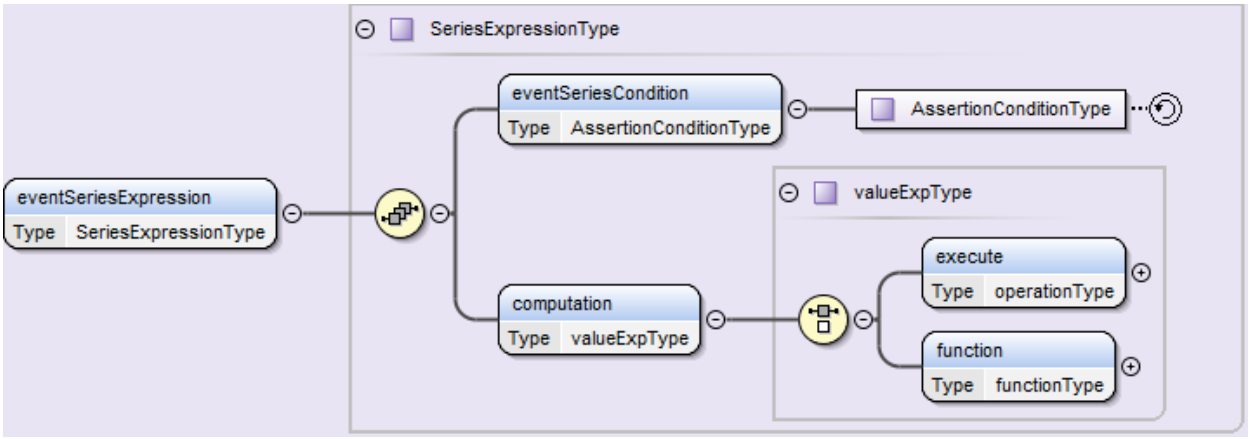

Figure 21 - eventSeriesExpression sub-element

An example of the *Guaranteed* sub-element is given below expressed in XML. I am providing a rule for the *data integrity-at-rest* security property where requests for updates of _TOC data are monitored, and, for every such request that is granted by the TOC (there is a verification code with the value "granted") within a specific time frame of 100 msec, the rule requests the execution of a test to check if the entity that requested the update had indeed the authorisation to update data.

```
<Assertion ID="AS001">
   <InterfaceDeclr>
        <ID>0001</ID>
        <ProviderRef>proRef1</ProviderRef>
        <Endpoint>
                <ID>eop1</ID>
                <Location>http://</Location>
                <Protocol>SOAP</Protocol>
        </Endpoint>
        <Interface>
                <InterfaceSpec>
                                <Name>update</Name>
                                <Operation>
                                <interfaceId>updateData</interfaceId>
                                <OperationId>id0001</OperationId>
                                <operationName>updateOp</operationName>
                                <inputVariable forMatching="true" persistent="false">
                                        <varName>cred</varName>
                                        <varType>string</varType>
                                </inputVariable>
                                <inputVariable forMatching="true" persistent="false">
```

```
                                <varName>data</varName>
                                <varType>string</varType>
                        </inputVariable>
                        <outputVariable>
                                <varName>auth</varName>
                                <varType>string</varType>
                        </outputVariable>
                        <outputVariable>
                                <varName>cred</varName>
                                <varType>string</varType>
                        </outputVariable>
                        <outputVariable>
                                <varName>data</varName>
                                <varType>string</varType>
                        </outputVariable>
                       <outputVariable>
                                <varName>vCode</varName>
                                <varType>string</varType>
                        </outputVariable>
                        </Operation>
            </InterfaceSpec>
    </Interface>
</InterfaceDeclr>
<InterfaceDeclr>
    <ID>002</ID>
    <ProviderRef>proRef2</ProviderRef>
    <Endpoint>
            <ID>eop2</ID>
            <Location>http://</Location>
            <Protocol>SOAP</Protocol>
    </Endpoint>
    <Interface>
            <InterfaceSpec>
                        <Name>authorise</Name>
                        <Operation>
                        <interfaceId>authoriseagent</interfaceId>
                        <OperationId>id0002</OperationId>
                        <operationName>authop</operationName>
                        <inputVariable forMatching="true" persistent="false">
                                <varName>cred</varName>
                                <varType>string</varType>
                        </inputVariable>
                        <inputVariable forMatching="true" persistent="false">
                                <varName>data</varName>
                                <varType>string</varType>
                        </inputVariable>
                        <outputVariable>
                                <varName>auth</varName>
                                <varType>string</varType>
                        </outputVariable>
```

```
                              </Operation>
               </InterfaceSpec>
        </Interface>
</InterfaceDeclr>
<Guaranteed ID="av1" type="Future_Formula">
             <quantification>
                <quantifier>forall</quantifier>
                      <timeVariable>
                        <varName>t1</varName>
                        <varType>TimeVariable</varType>
                      </timeVariable>
                </quantification>
                <quantification>
                      <quantifier>existential</quantifier>
                           <timeVariable>
                        <varName>t2</varName>
                        <varType>TimeVariable</varType>
                           </timeVariable>
                       </quantification>
                <precondition>
                      <atomicCondition conditionID="ac0">
                          <eventCondition unconstrained="true">
                        <event>
                             <eventID forMatching="true" persistent="false">
                                   <varName>VID0</varName>
                             </eventID>
                             <call>
                                 <interfaceId> interface::update::a::1/</interfaceId>
                                  <OperationId>1</OperationId>
                                  <operationName>upload</operationName>
                                   <inputVariable forMatching="true" persistent="false">
                                      <varName>status1</varName>
                                      <varType>OpStatus</varType>
                                      <value>REQ-B</value>
                                   </inputVariable>
                                   <inputVariable forMatching="true" persistent="false">
                                      <varName>sender</varName>
                                      <varType>Entity</varType>
                                       <value></value>
                                  </inputVariable>
                                  <inputVariable forMatching="true" persistent="false">
                                      <varName>receiver</varName>
                                      <varType>Entity</varType>
                                      <value></value>
                                  </inputVariable>
                                  <inputVariable forMatching="true" persistent="false">
                                      <varName>source1</varName>
                                      <varType>Entity</varType>
                                      <value></value>
                                  </inputVariable>
```

```
                    <inputVariable forMatching="true" persistent="false">
                        <varName>serviceId</varName>
                        <varType>string</varType>
                        <value></value>
                     </inputVariable>
                     <inputVariable forMatching="true" persistent="false">
                        <varName>cred</varName>
                        <varType>string</varType>
                     </inputVariable>
                      <inputVariable forMatching="true" persistent="false">
                           <varName>data</varName>
                           <varType>string</varType>
                     </inputVariable>
                     <inputVariable forMatching="true" persistent="false">
                           <varName>auth</varName>
                           <varType>string</varType>
                     </inputVariable>
          </call>
             <tVar>
                <timeVar>
                <varName>t1</varName>
                <varType>TimeVariable</varType>
                </timeVar>
             </tVar>
             <fromTime>
                   <time>
                <varName>t1</varName>
                <varType>TimeVariable</varType>
                   </time>
              </fromTime>
              <toTime>
                    <time>
                <varName>t1</varName>
                <varType>TimeVariable</varType>
                     </time>
              </toTime>
        </event>
            </eventCondition>
      </atomicCondition>
      <WrappedCondition>
        <operator>and</operator>
                <assertionCondition>
                <atomicCondition>
                      <relationalCondition>
                        <equal>
                             <operand1>
                                    <variable>
                                     <varName>vCode</varName>
                                     <varType>string</varType>
                                    </variable>
```

```
                                        </operand1>
                                    <operand2>
                                            <constant>
                                            <name>vcode</name>
                                              <value>granted</value>
                                           </constant>
                                    </operand2>
                                </equal>
                                <timeVar>
                                    <varName>t2</varName>
                                    <varType>TimeVariable</varType>
                                </timeVar>
                            </relationalCondition>
                        </atomicCondition>
                    </assertionCondition>
                </WrappedCondition>
</precondition>
<postcondition>
      <atomicCondition conditionID="ac1">
        <eventCondition unconstrained="true">
              <event>
                  <eventID forMatching="true" persistent="false">
                <varName>VID1</varName>
                  </eventID>
                <execute>
                      <interfaceId> interface::auth::ca::1</interfaceId>
                      <OperationId>2</OperationId>
                      <operationName>authorise</operationName>
                      <outputVariable forMatching="true" persistent="false">
                        <varName>status2</varName>
                        <varType>OpStatus</varType>
                        <value>RES-B</value>
                     </outputVariable>
                     <outputVariable forMatching="true" persistent="false">
                        <varName>receiver1</varName>
                        <varType>Entity</varType>
                     </outputVariable>
                     <outputVariable forMatching="true" persistent="false">
                        <varName>sender1</varName>
                        <varType>Entity</varType>
                      </outputVariable>
                       <outputVariable forMatching="true" persistent="false">
                        <varName>source1</varName>
                        <varType>Entity</varType>
                     </outputVariable>
                     <outputVariable forMatching="true" persistent="false">
                        <varName>serviceId</varName>
                        <varType>string</varType>
                     </outputVariable>
                     <outputVariable forMatching="true" persistent="false">
```

```
                                    <varName>cred</varName>
                                    <varType>string</varType>
                                 </outputVariable>
                                 <outputVariable forMatching="true" persistent="false">
                                    <varName>auth</varName>
                                    <varType>string</varType>
                                 </outputVariable>
                                 <outputVariable forMatching="true" persistent="false">
                                    <varName>vCode2</varName>
                                    <varType>string</varType>
                                 </outputVariable>
                              </execute>
                              <tVar>
                                 <timeVar>
                                    <varName>t3</varName>
                                    <varType>TimeVariable</varType>
                                 </timeVar>
                              </tVar>
                              <fromTime>
                                 <time>
                                    <varName>t2</varName>
                                    <varType>TimeVariable</varType>
                                 </time>
                              </fromTime>
                              <toTime>
                                 <time>
                                    <varName>t3</varName>
                                    <varType>TimeVariable</varType>
                                 </time>
                                 <Expression>
                                   <plus>100</plus>
                                 </Expression>
                              </toTime>
                           </event>
                        </eventCondition>
                     </atomicCondition>
                  </postcondition>
               </Guaranteed>
```

3.5.5 Collector Evidence Aggregation element

The *Collector Evidence Aggregation* element, as shown in Figure 22, defines how evidence will be collected and aggregated. This element contains the *TestingCollector* element that describes the type of testing that needs to be performed along with the frequency of periodic testing, the *MonitoringConfigurations* element that specifies the list of the monitoring configurations that

have been used to collect the evidence for generating certificates, the *MonitoringAggregatedResultsInfo* element that defines how often the monitoring evidence will be checked to create a new certificate, the *EventSummary* elemene that contains information about the number of violations detected in monitoring and testing, the *FunctionalAggregatorId* that defines what type of aggregation should be done in the events (e.g. min value, max value, standard deviation, average value), and, the *IntermediateResults* element which specifies if there is a need to aggregate evidence between two predefined aggregation periods, to check the validity of the certificate. This element contains elements from the Test-based certification model schema and the Monitoring-based certification model schema, but has been refined to fit the hybrid purposes.

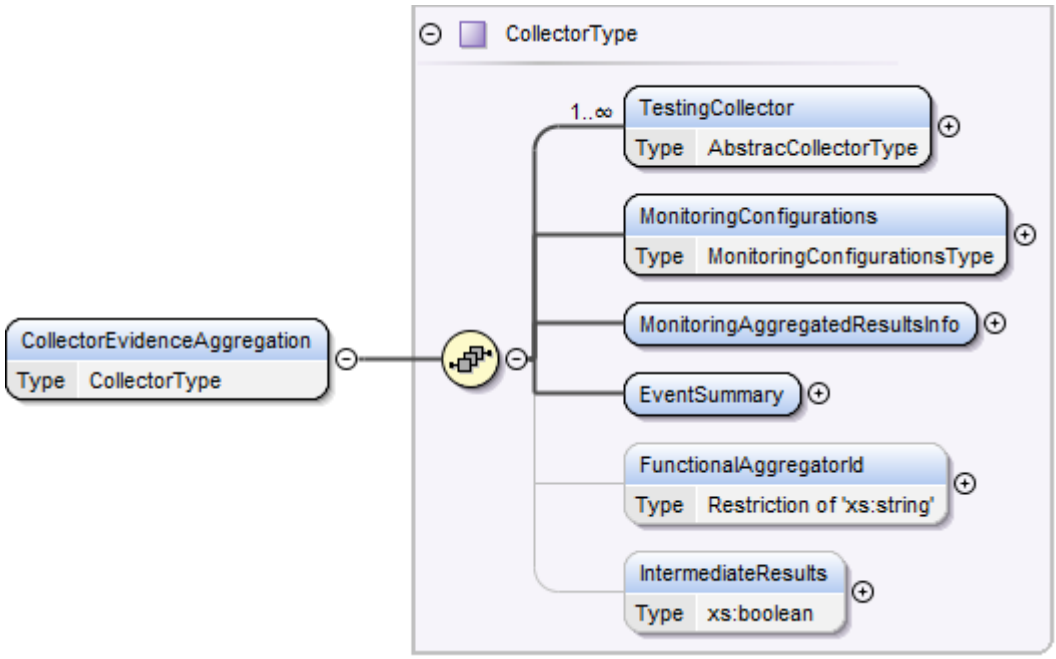

Figure 22 - CollectorsEvidenceAggregation element

The *TestingCollector,* as shown in Figure 23, has a unique identifier attribute of the collector element and an attribute that defines if the testing required will be static or dynamic, and consists of the following elements:

- a *TestCategory* element, that defines if the type of testing required
- a *TestDescription* element that specifies the testing mechanism, describing what needs to be tested
- a *TestFrequencyUnit* element that defines the unit concerning the frequency of testing that is required

- a *TestFrequencyValue* that defines the value of the frequency of testing that is required

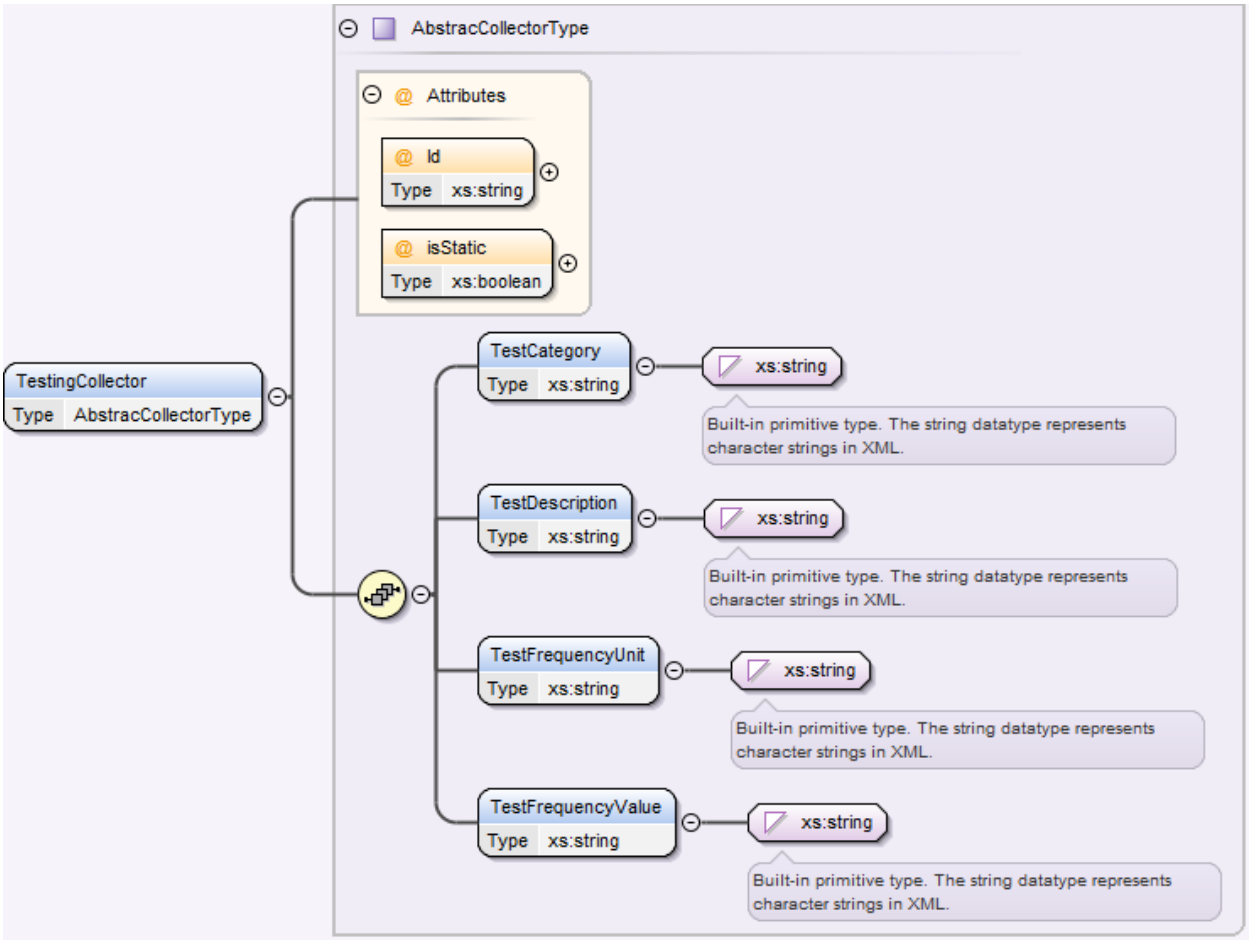

Figure 23 - TestingCollector sub-element

An example of the *TestingCollector* element for the certification of the *data integrity-at-rest* security property is shown below. According to this example, in order to issue a certificate we need to perform periodic online integration testing and to send *authcodes* to perform the authorisation operation.

```
<TestingCollector Id="Id0" isStatic="false">
            <TestCategory>integration</TestCategory>
            <TestDescription> Send authcodes (with malformed pass) to perform authorisation
                 operation</TestDescription>
            <TestFrequencyUnit>sec</TestFrequencyUnit>
            <TestFrequencyValue>3</TestFrequencyValue>
</TestingCollector>
```

The *MonitoringConfigurations* element, as shown in Figure 24, specifies the list of the monitoring configurations that have been used to collect the evidence for generating certificates and includes a unique *Id*, a list of *components* of the monitoring environment that can be *sensors,* which are components capable of capturing and transmitting primitive monitoring events or *reasoners,* which are the monitors that are capable of analysing events and checking whether monitoring conditions are satisfied.

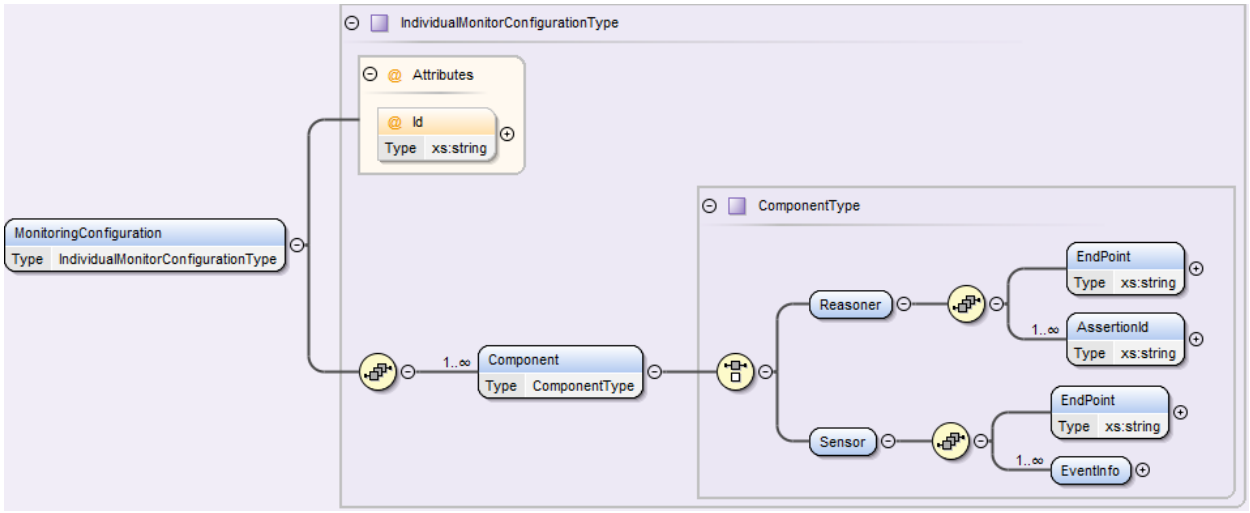

Figure 24 - MonitoringConfigurations sub-element

In the following example, the framework is configured with a reasoner component, whose endpoint is: https://192.168.43.23:8888/CumulusService.wsdl.

```
<MonitoringConfigurations>
            <MonitoringConfiguration>
                <Component>
                    <Reasoner>
                        <EndPoint>https://192.168.43.23:8888/CumulusService.wsdl</EndPoint>
                        <AssertionId>av::1</AssertionId>
                    </Reasoner>
                </Component>
</MonitoringConfiguration>
```

The *MonitoringAggregatorResultsInfo* element, as shown in Figure 25, defines how often the monitoring evidence will be checked to create a new certificate, with new aggregated evidence. This element includes the following attributes:

- the *StartDate* of the first aggregation
- the *Timestamp* of the monitoring event
- the *NumberOfMonitoringEvents,* which is the number of the primitive monitoring events being aggregated
- the *intervalsTime,* which specifies how often should the evidence being aggregated
- the intervalU*nit*, which declares the unit used to specify the intervals.

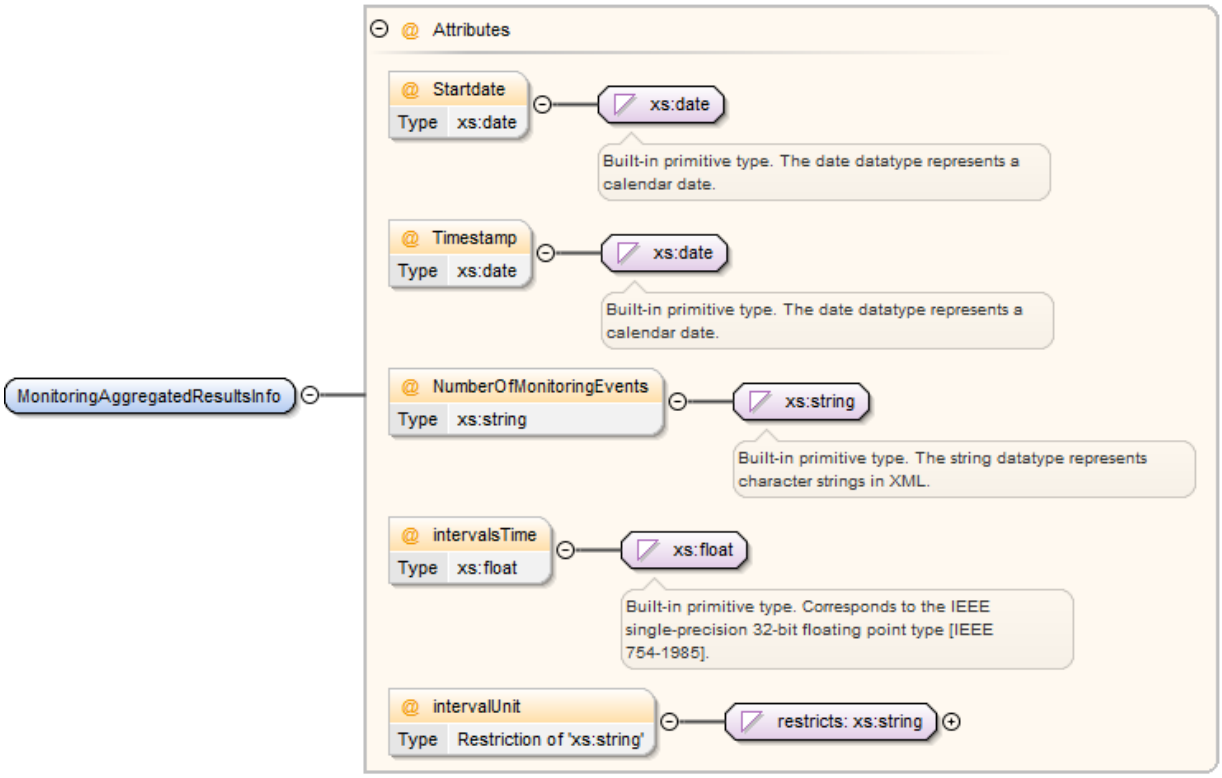

Figure 25 - MonitoringAggregatedResultsInfo sub-element

The *EventSummary* element, as shown in Figure 26, defines the minimum number of satisfactions and the number of violations required for issuing a certificate. The evidence used to detect the violations and the satisfactions come from monitoring and testing evidence. This sub-element

includes the *maxNumberOfMonitoringTestingEvidenceViolations* and the *minNumberOfMonitoringTestingEvidenceSatisfactions* attributes, and describes the maximum acceptable number of violations and the minimum acceptable number of satisfactions, for a certificate to be issued.

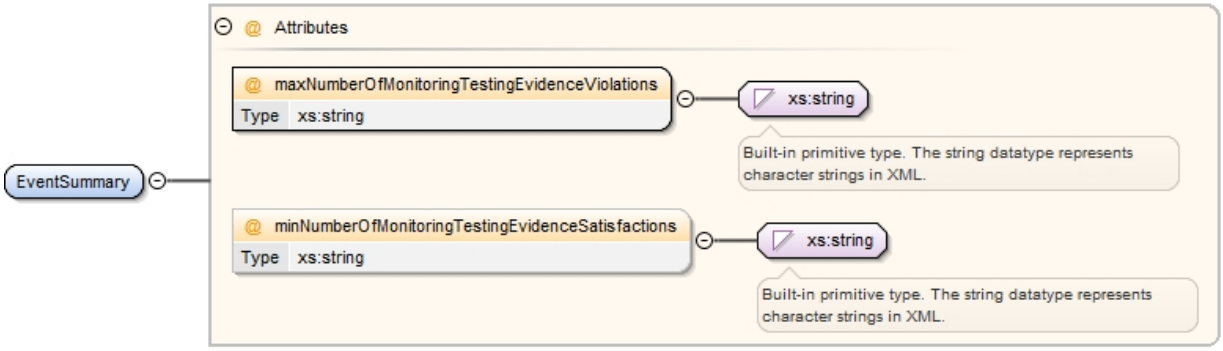

Figure 26 - EventSummary sub-element

The *FunctionalAggregatorId*, as shown in Figure 27, is an optional element and defines what type of aggregation should be done in the events (e.g. min value, max value, standard deviation, average value).

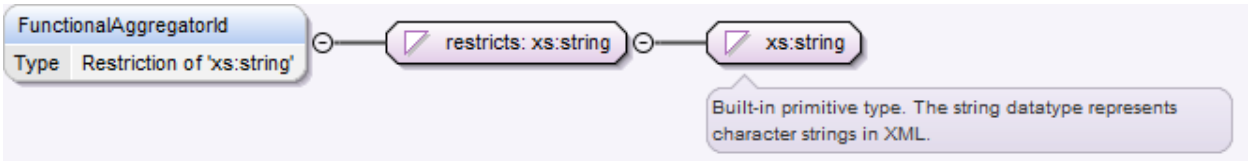

Figure 27 - FunctionalAggregationId sub-element

Finally, the *IntermediateResults,* as shown in Figure 28, is an optional element and specifies if there is a need to aggregate evidence between two predefined aggregation periods, to check the validity of the certificate.

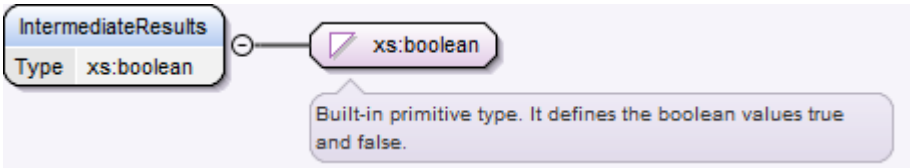

Figure 28 - IntermediateResults sub-element

A full example of the *CollectorEvidenceAggregation* element in XML is given below. In this example, I examine the *data integrity-at-rest* security property. With periodic online testing *authcodes* are sent to perform the authorisation operation. The framework is configured with a reasoner component with a predefined endpoint. The monitoring aggregation will start on date "2015-01-01" and the aggregation of detailed evidence should be carried out at intervals of 840 hours. The number of monitoring events than need to be collected is 520. The maximum number of violations between testing and monitoring evidence is 100. The average value that will be applied for the property will be a Boolean.

```
<CollectorEvidenceAggregation>
        <TestingCollector Id="Id0" isStatic="false">
            <TestCategory>integration</TestCategory>
            <TestDescription> Send authcodes (with malformed pass) to perform authorisation
                 operation</TestDescription>
            <TestFrequencyUnit>sec</TestFrequencyUnit>
            <TestFrequencyValue>3</TestFrequencyValue>
        </TestingCollector>
        <MonitoringConfigurations>
            <MonitoringConfiguration>
                <Component>
                    <Reasoner>
                        <EndPoint>https://192.168.43.23:8888/CumulusService.wsdl</EndPoint>
                        <AssertionId>av::1</AssertionId>
                    </Reasoner>
                </Component>
        </MonitoringConfiguration>
        <MonitoringAggregatedResultsInfo StartDate="2014-01-01" Timestamp"2014-01-01"
                 NumberOfMonitoringEvents="520" IntervalsTime="840" IntervalUnit="hours" />
        <EventSummary minNumberOfMonitoringTestingEvidenceViolations="100">
        <FunctionalAggregatorId>Average</FunctionalAggregatorId>
        <IntermediateResults>False</IntermediateResults>
</CollectorEvidenceAggregation>
```

3.5.6 Lifecycle element

The *LifecycleModel* element, as shown in Figure 29, defines the basic stages in the generation and management of the certificates. The *LifecycleModel* element from the Monitoring-based certification model schema has been used, but has been refined to fit the hybrid purposes. This element consists of an optional *InitialState*, an optional *FinalState*, a sequence of atomic *states*, a *transitions* element that describes the transition between the different states, zero or more provided interfaces and zero or more required interfaces.

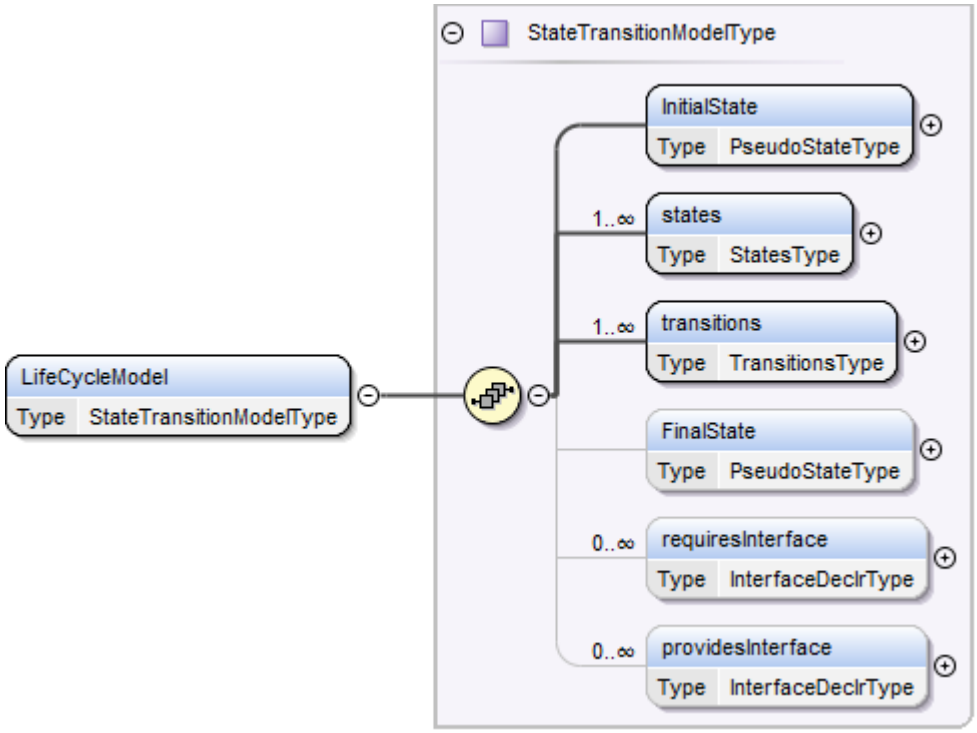

Figure 29 - LifeCycleModel element

The *InitialState* element has a *stateId* attribute that identifies the initial state uniquely in the state transitional model and a *name* attribute, as shown in Figure 30.

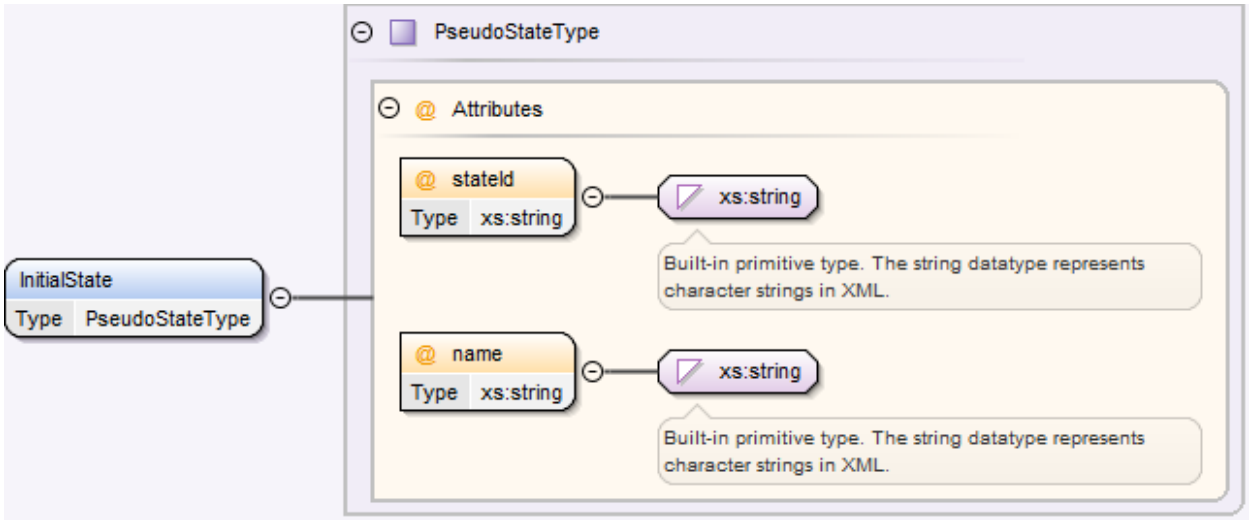

Figure 30 - InitialState sub-element

An example of the *InitialState* sub-element is given below in XML:

```
<InitialState stateId="stateId0" name="Activated"/>
```

The *states* element is of type *StatesType*, which is defined as a sequence of *state* elements of type *StateType* as shown in Figure 31. The possible atomic states of a certificate are “Activated” (Initial State), “Issued”, “Expired”, “SuspendCertificate”, “Revoke”, “Ended”, “Continuous MonitoringandTestExecution” and “Issuing”. A *state* element consists of a sequence of one or more atomic states that are of *AtomicStateType.* The definition is given below:

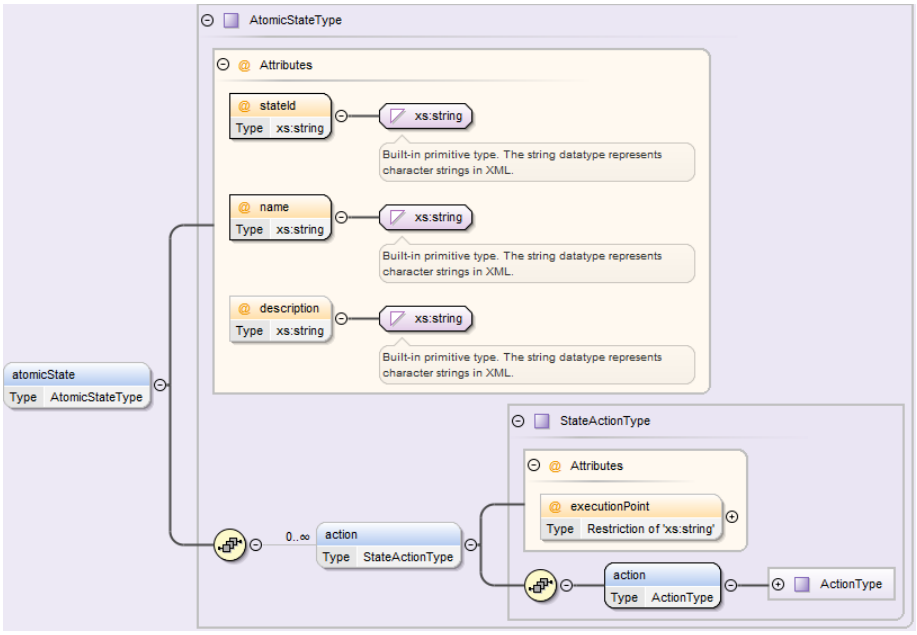

Figure 31 - StateType sub-element

An atomic state element is described by the attributes includes a *stateId* attribute which uniquely identifies the state within a state transition model, a *name* attribute which provides the name of the state, a *description* attribute which can be used to provide a description about this state and a sequence of actions that must be executed, such as the execution of tests, the updating of internal variables, the invocation of operations or the assignment of variables. An action has an *executionPoint* attribute that defines whether the action is executed when the entity enters or exits the state and contains a sequence of actions, as shown in Figure 32.

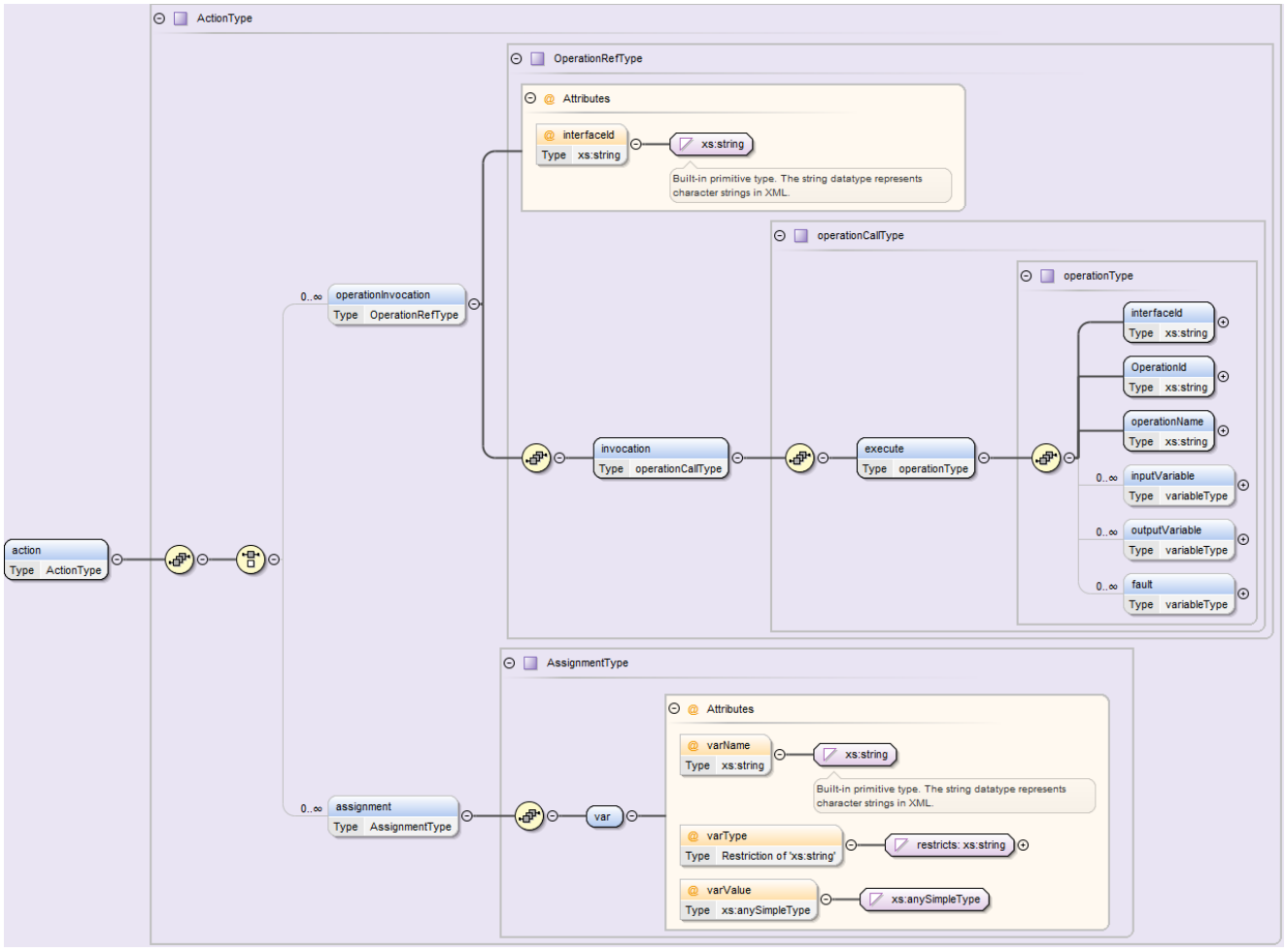

Figure 32 - action sub-element

An example of the *states* element is given below. This example refers to the *data integrity-at-rest* security property that has been introduced earlier. According to this example, the "ContinuousMonitoringAndTestExecution" state requires the execution of a test to check if the agent who requested the update of data is authorised.

```
<states>
    <state>
       <atomicState stateId="stateId1" name="ContinuousMonitoringAndTestExecution" description="collection
of monitoring and testing evidence">
             <action executionPoint="authorisation interface">
                   <action>
                       <operationInvocation interfaceId="interface::auth::ca::1">
                              <invocation>
                                   <execute>
                                        <interfaceId>interface::auth::ca::1</interfaceId>
                                        <OperationId>id0002</OperationId>
                                        <operationName>authop</operationName>
                                   </execute>
                              </invocation>
                       </operationInvocation>
                   </action>
             </action>
        </atomicState>
     </state>
</states>
```

The transitions in a LifeCycle Model are described by a *Transitions* element, which is of type *TransitionsType* and is shown in Figure 33. This element has one or more sub-elements, which are of type *IndividualTransitionType*.

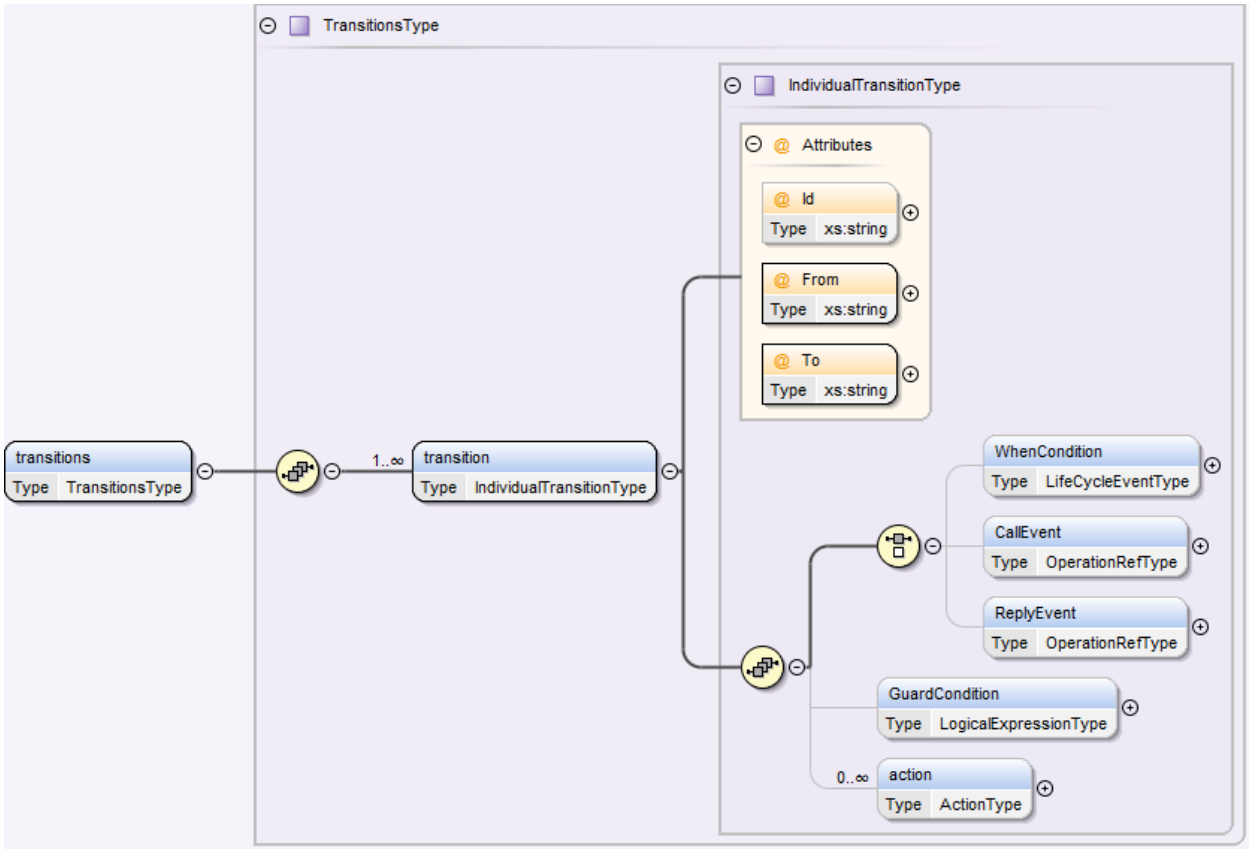

Figure 33 - transition sub-element

Each *IndividualTransitionType* element includes:

- an *Id* attribute of type string, which refers to the identifier of the transition
- a *From* attribute of type string, which refers to the identifier of the state that the transition starts from
- a *To* attribute of type string, which refers to the identifier of the state that the transition ends up

Transitions can be triggered by the following types of events:

- a system condition whose value is changed (such conditions are described by the sub-element *WhenCondition*), or
- a call event sent to the entity
- a reply event sent to the entity

- an optional *GuardCondition* element expressing a condition which must be true when an event that can trigger the transition occurs for the transition to be taken.

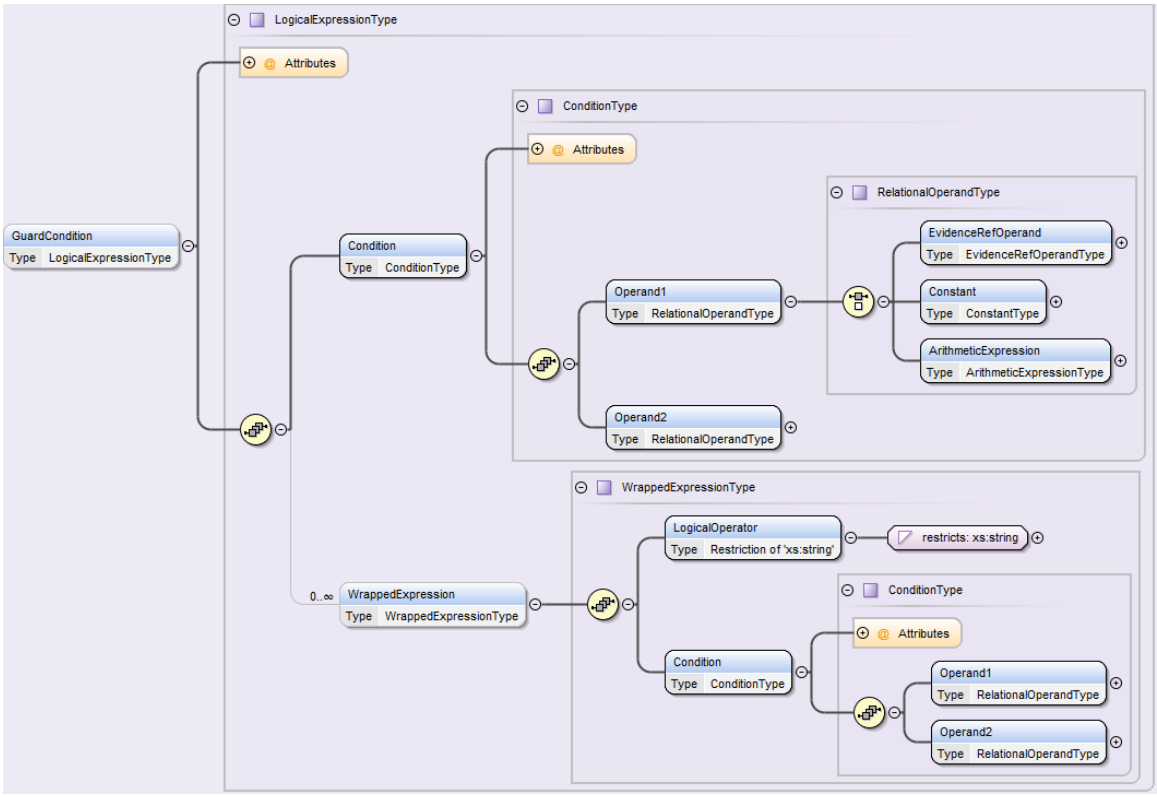

Figure 34 - Guard Conditions

- *Action* elements defining the actions that must be executed and completed before the transition takes place and the before the entity the destination state of the transition. The tests that need to be executed in order to collect evidence are part of the assertion language. Therefore, when additional tests are required, they are defined through the Action element.

An example is given below. In this example, I describe the transition from "Activated" state to the "ContinuousMonitoringAndTestExecution" state. This transition is activated when a reply of an update operation in a TOC is detected at some time point t1. The *GuardCondition* element describes the authorisation condition that must be true for the transition to take place. According to this condition the *vCode* needs to have a value *"granted"*. The *action* element defines the execution of tests that need to take place in order to collect the testing evidence required.

```
<transitions>
       <transition Id="1011" From="ContinuousMonitoringAndTestExecution" To="Issued">
                <ReplyEvent interfaceId="interfaceId4">
                    <invocation>
                        <execute>
                            <interfaceId>interface::auth::ca::1</interfaceId>
                            <OperationId>id0002</OperationId>
                            <operationName>authop</operationName>
                        </execute>
                    </invocation>
                </ReplyEvent>
                <GuardCondition negated="false">
                    <Condition negated="false" relation="EQUAL-TO">
                        <Operand1>
                            <ArithmeticExpression>
                                <ArithmeticOperand>
                                    <evidenceRefOperand referencePath== "//VariableDeclr/Var/[text()=
                                    'vCode']"/>
                                </ArithmeticOperand>
                            </ArithmeticExpression>
                        </Operand1>
                        <Operand2>
                            <ArithmeticExpression>
                                <ArithmeticOperand>
                                    <Constant type="granted">Constant0</Constant>
                                </ArithmeticOperand>
                            </ArithmeticExpression>
                        </Operand2>
                    </Condition>
                </GuardCondition>
                <action>
                    <operationInvocation interfaceId="interfaceId6">
                        <invocation>
                            <execute>
                                <interfaceId> interface::auth::ca::1</interfaceId>
                                <OperationId>id002</OperationId>
                                <operationName>authop</operationName>
                            </execute>
                        </invocation>
                    </operationInvocation>
                </action>
       </transition>
</transitions>
```

The *FinalState* element has a *stateId* attribute that identifies the final state uniquely in the state transitional model and a *name* attribute, as shown in Figure 35.

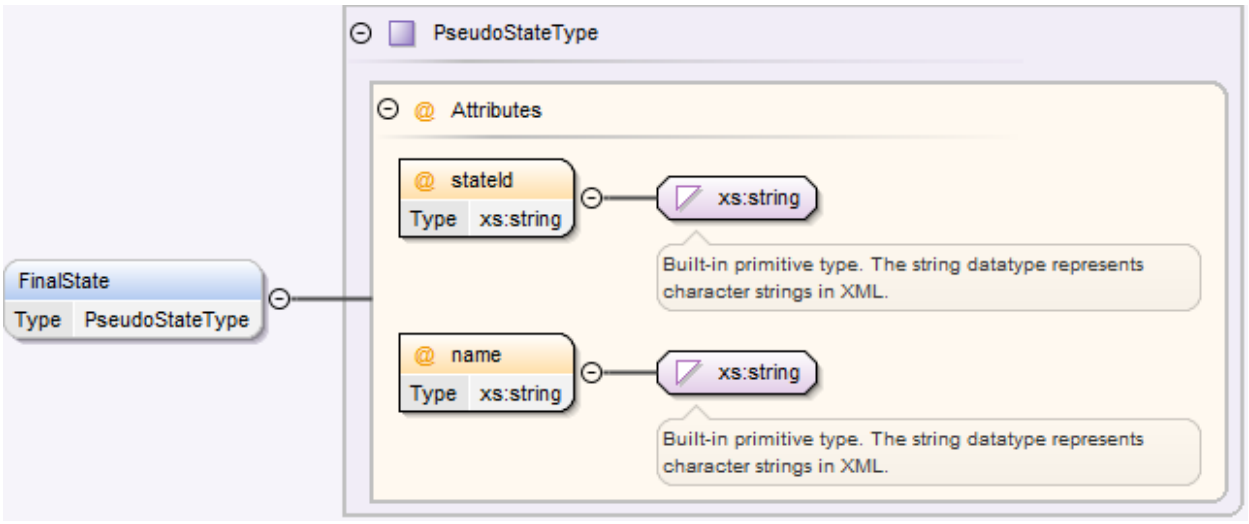

Figure 35 - Final State

An example of the *FinalState* sub-element is given below in XML:

```
<FinalState stateId="stateId5" name="Ended"/>
```

The *providesInterfaces* element includes a set of interfaces, which are realised by the entity whose behaviour is described by the model, while the *requiredInterfaces* element includes a set of operations which the entity whose behaviour is described by the model expects other external interacting entities to have. Both these elements are of *InterfaceDeclrType* (Figure 36).

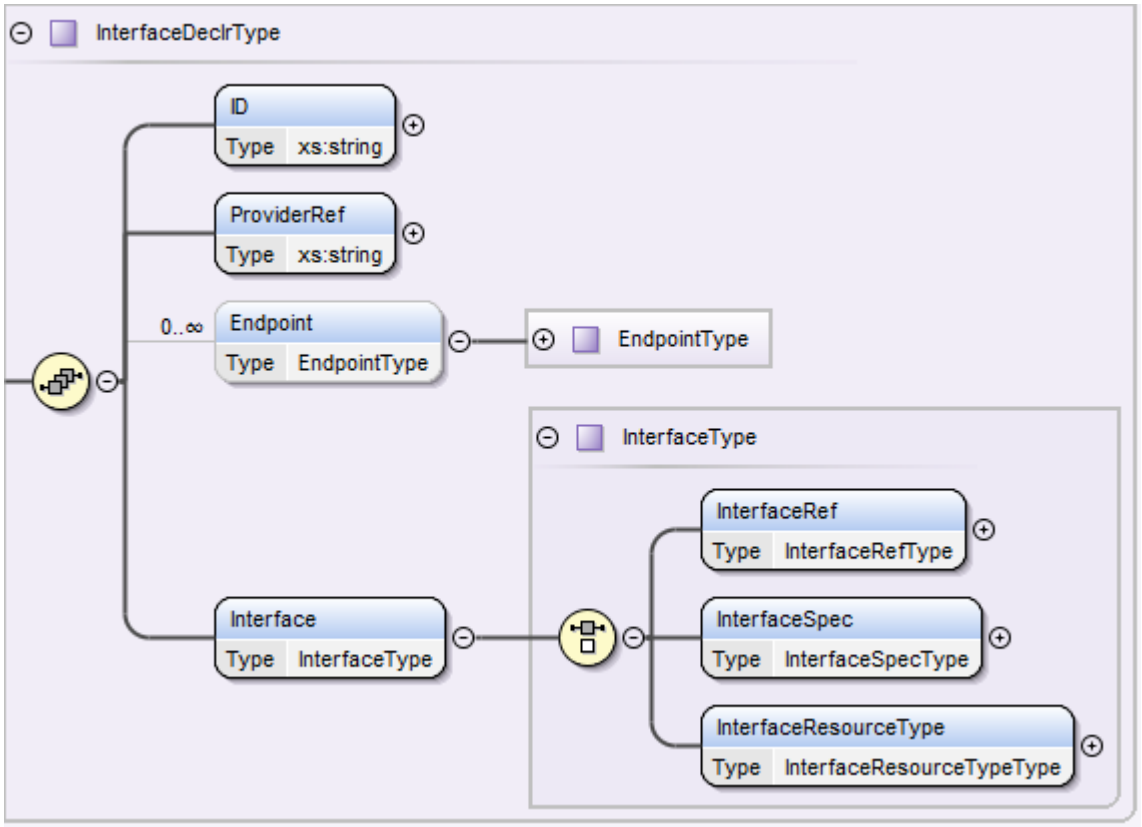

Figure 36 - InterfaceDeclrType

An example of a hybrid Lifecycle for a certificate is presented below. Initially, after the activation, the certificate enters the "ContinuousMonitoringAndTestExecution" state and evidence is gathered through monitoring and testing. When enough evidence is collected, according to the Sufficiency Conditions, then the certificate status is updated to "Issued". Then, if the expiration conditions are reached, the status of the certificate is updated to "Expired", otherwise evidence collected from monitoring and testing are cross-checked. If the evidence is contradictory then the status of the certificate is updated to "Suspend Certificate", otherwise the certificate will enter the "Issuing" state. In this case, if violations are not resolved the status will be updated to "Revoke", otherwise the certificate will enter the "Issuing" state.

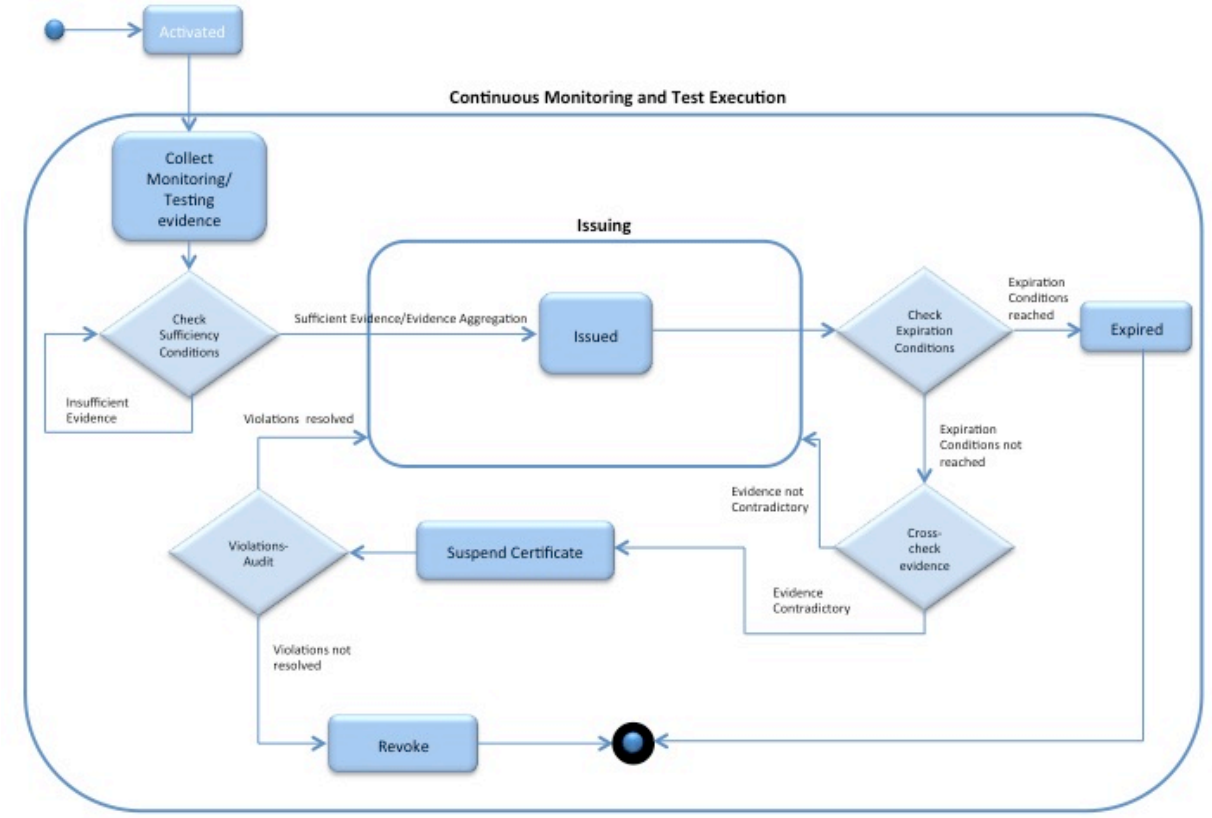

Figure 37 - LifeCycle Model

An XML example of the lifecycle model describing the transition from the "ContinuousMonitoringAndTestExecution" to the "Issued" state is given below.

```
<LifeCycleModel>
   <InitialState stateId="stateId0" name="Activated"/>
       <states>
         <state>
           <atomicState stateId="stateId1" name="ContinuousMonitoringAndTestExecution"
```

```
        description="collection of monitoring and testing evidence">
          <action executionPoint="authorisation interface">
            <action>
              <operationInvocation interfaceId="interface::auth::ca::1">
                    <invocation>
                         <execute>
                              <interfaceId>interface::auth::ca::1</interfaceId>
                              <OperationId>id0002</OperationId>
                              <operationName>authop</operationName>
                         </execute>
                    </invocation>
              </operationInvocation>
            </action>
          </action>
      </atomicState>
    </state>
  </states>
  <transitions>
      <transition Id="1011" From="ContinuousMonitoringAndTestExecution" To="Issued">
          <ReplyEvent interfaceId="interfaceId4">
              <invocation>
                  <execute>
                      <interfaceId>interface::auth::ca::1</interfaceId>
                      <OperationId>id0002</OperationId>
                      <operationName>authop</operationName>
                  </execute>
              </invocation>
          </ReplyEvent>
          <GuardCondition negated="false">
              <Condition negated="false" relation="EQUAL-TO">
                  <Operand1>
                      <ArithmeticExpression>
                          <ArithmeticOperand>
                              <evidenceRefOperand referencePath== "//VariableDeclr/Var/[text()=
                              'vCode']"/>
                          </ArithmeticOperand>
                      </ArithmeticExpression>
                  </Operand1>
                  <Operand2>
                      <ArithmeticExpression>
                          <ArithmeticOperand>
                              <Constant type="granted">Constant0</Constant>
                          </ArithmeticOperand>
                      </ArithmeticExpression>
                  </Operand2>
              </Condition>
          </GuardCondition>
```

```
                <action>
                    <operationInvocation interfaceId="interfaceId6">
                        <invocation>
                            <execute>
                                <interfaceId> interface::auth::ca::1</interfaceId>
                                <OperationId>id002</OperationId>
                                <operationName>authop</operationName>
                            </execute>
                        </invocation>
                    </operationInvocation>
                </action>
            </transition>
        </transitions>
        <FinalState stateId="stateId2" name="Issued"/>
                <providesInterface>
              <ID>interface::update::a::1/</ID>
              <ProviderRef>Cumulus::provider::a::1</ProviderRef>
              <Endpoint>
                  <ID>a111</ID>
                  <Location>http://www.cumulus-project.eu</Location>
                  <Protocol>SOAP</Protocol>
              </Endpoint>
              <Interface>
                  <InterfaceSpec>
                      <Name>av::update::a::1</Name>
                      <Operation>
                          <InterfaceId>update::data::a::1</InterfaceId>
                          <OperationId>update::operation::a::1</OperationId>
                          <OperationName>update</OperationName >
                      </Operation>
                  </InterfaceSpec>
              </Interface>
          </providesInterface>
          <requiresInterface>
              <ID>interface::auth::ca::1</ID>
              <ProviderRef></ProviderRef>
              <Endpoint>
                  <ID>b111</ID>
                  <Location>http://www.cumulus-project.eu</Location>
                  <Protocol>SOAP</Protocol>
              </Endpoint>
              <Interface>
                  <InterfaceRef>
                      <InterfaceLocation> http://www.cumulus-project.eu </InterfaceLocation>
                  </InterfaceRef>
              </Interface>
          </requiresInterface>
</LifeCycle Model>
```

3.5.7 AssessmentScheme element

The *AssessmentScheme,* as shown in Figure 38, element specifies how the evidence collected from testing and monitoring will be assessed in order to check if a security property is satisfied. This element will check if the evidence sufficiency and expiration conditions are satisfied. Additionally, it includes the extra conditions that will define if there are events whose execution is forced as part of the assertion conditions (hybrid dependent-mode certification where monitoring triggers testing and vice-versa), otherwise the hybrid certification will be of independent mode. The *EvidenceSufficiencyConditions* and *ExpirationConditions* elements of the Monitoring-based certification model schema have been refined and amended for my hybrid approach.

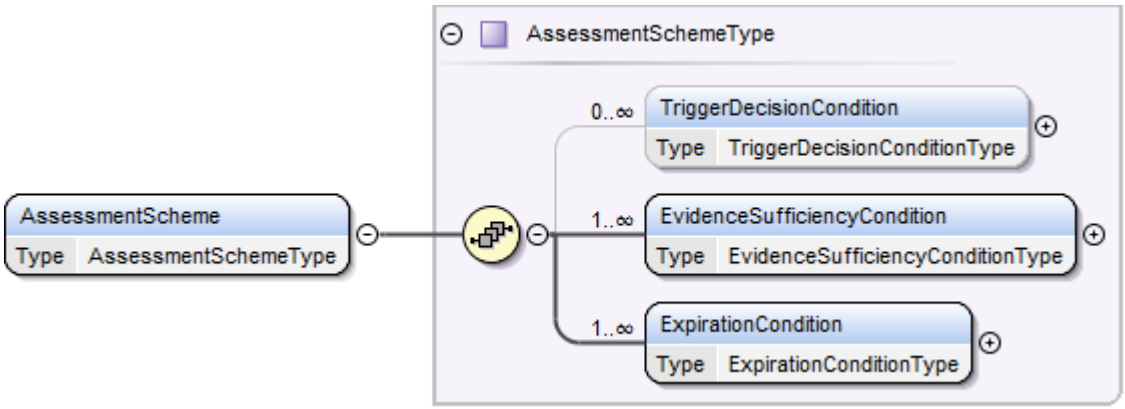

Figure 38 - Assessment Scheme element

The *TriggerDecisionCondition* element, as shown in Figure 39, is an optional element of *TriggerDecisionConditionType* and defines the type of hybrid dependent-mode certification, describing the conditions that trigger the collection of additional events when there are events whose execution are forced as part of the assertion conditions. This element includes a choice between the *MonitoringTriggeredConditions* sub-element that defines the triggering conditions when Monitoring is triggered by Testing and the TestingTriggeredConditions sub-element that defines the triggering conditions when Testing is triggered by Monitoring.

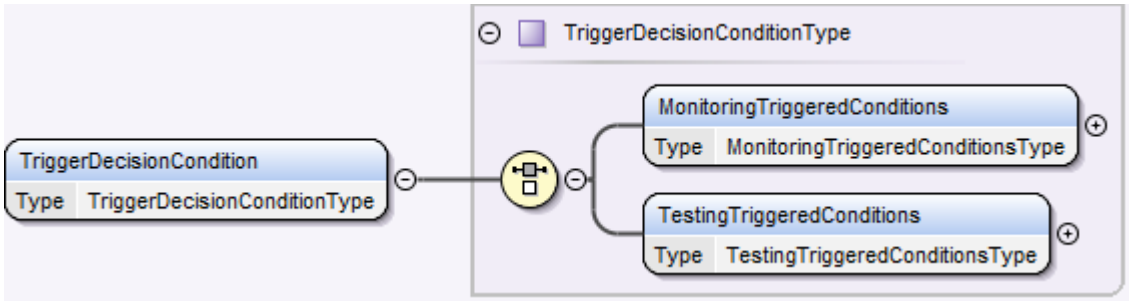

Figure 39 - TriggerDecisionCondition sub-element

The *MonitoringTriggeredConditions* element, as shown in Figure 40, includes a sequence of the following sub-elements:

- an *EventId* sub-element, which is of type string and specifies the Id of the testing event which will trigger monitoring
- a *ConditionsDescription* sub-element of string type, which describes why monitoring is triggered by the execution of test cases
- a *TimestampOfEventFromTime* sub-element, which is of *timeVariabletype* and specifies when the testing event could have started
- a *TimestampOfEventToTime* sub-element, which is of *timeVariabletype* and specifies when the testing event could have terminated
- a *TimestampOfTriggeredEventFromTime* sub-element, which is of *timeVariabletype* and specifies when the monitoring event could have started
- a *TimestampOfTriggeredEventToTime* sub-element, which is of *timeVariabletype* and specifies when the monitoring event could have terminated

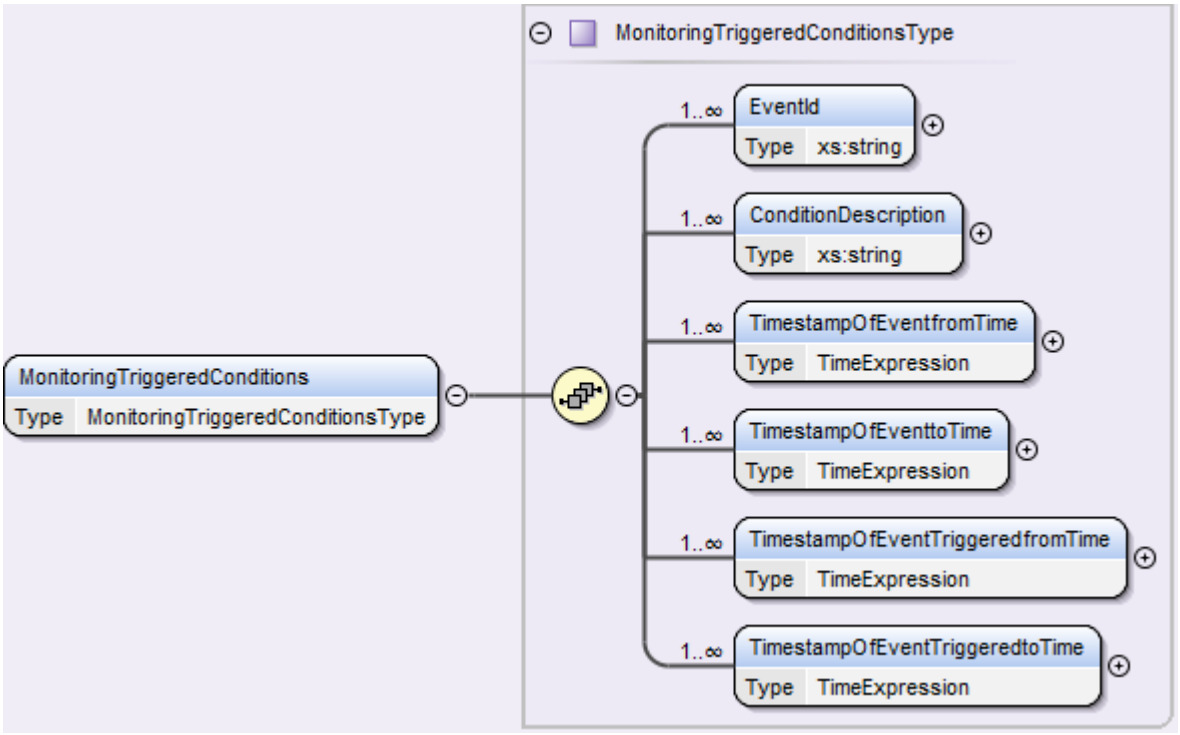

Figure 40 - MonitoringTriggeredConditionsType

An example of the *MonitoringTriggeredConditions* sub-element is following. In this example, I want to certify the *data integrity-at-rest* security property, which expresses the ability to detect and report any alteration of stored data in a target of certification (TOC). I will use a hybrid model based on periodic testing that will detect if stored data have been modified and will monitor the periods between the tests that revealed data modifications to check if appropriate notifications have also been sent. Data modifications could be detected by obtaining the hash value of the relevant data file in the TOC periodically. Then, if across the execution of two consecutive tests, the last retrieved hash value of the file is different from the previous hash value, a data modification action can be deduced. In parallel with the execution of this periodic test, the hybrid model will also monitor the execution of notification operations. Hence, when a data modification action is detected by two consecutive tests, the hybrid model could also check whether a correlated notification operation has been executed within the period between the tests. This example in XML is given below:

```
<TriggerDecisionCondition>
        <MonitoringTriggeredConditions>
            <EventId>Id1010</EventId>
            <ConditionDescription>monitor period between tests</ConditionDescription>
            <TimestampOfEventfromTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
            </TimestampOfEventfromTime>
            <TimestampOfEventtoTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>2</plus>
                    </Expression>
            </TimestampOfEventtoTime>
            <TimestampOfEventTriggeredfromTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>2</plus>
                    </Expression>
            </TimestampOfEventTriggeredfromTime>
            <TimestampOfEventTriggeredtoTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>6</plus>
                    </Expression>
            </TimestampOfEventTriggeredtoTime>
         </MonitoringTriggeredConditions>
</TriggerDecisionCondition>
```

The *TestingTriggeredConditions* element as shown in Figure 41 includes a sequence of the following elements:

- an *EventId* sub-element, which is of type string and specifies the Id of the monitoring event which triggers testing

- a *ConditionsDescription* sub-element, which describes why testing is triggered by monitoring
- a *TimestampOfEventFromTime* sub-element, which is of *timeVariabletype* and specifies when the monitoring event could have started
- a *TimestampOfEventToTime* sub-element, which is of *timeVariabletype* and specifies when the monitoring event could have terminated
- a *TimestampOfTriggeredEventFromTime* sub-element, which is of *timeVariabletype* and specifies when the testing event could have started
- a *TimestampOfTriggeredEventToTime* sub-element, which is of *timeVariabletype* and specifies when the testing event could have terminated
- an *AdditionalCoverageRequired* sub-element which is of type string and describes why monitoring triggers testing.
- a *Periodicity* sub-element, which specifies the periodicity of online testing.

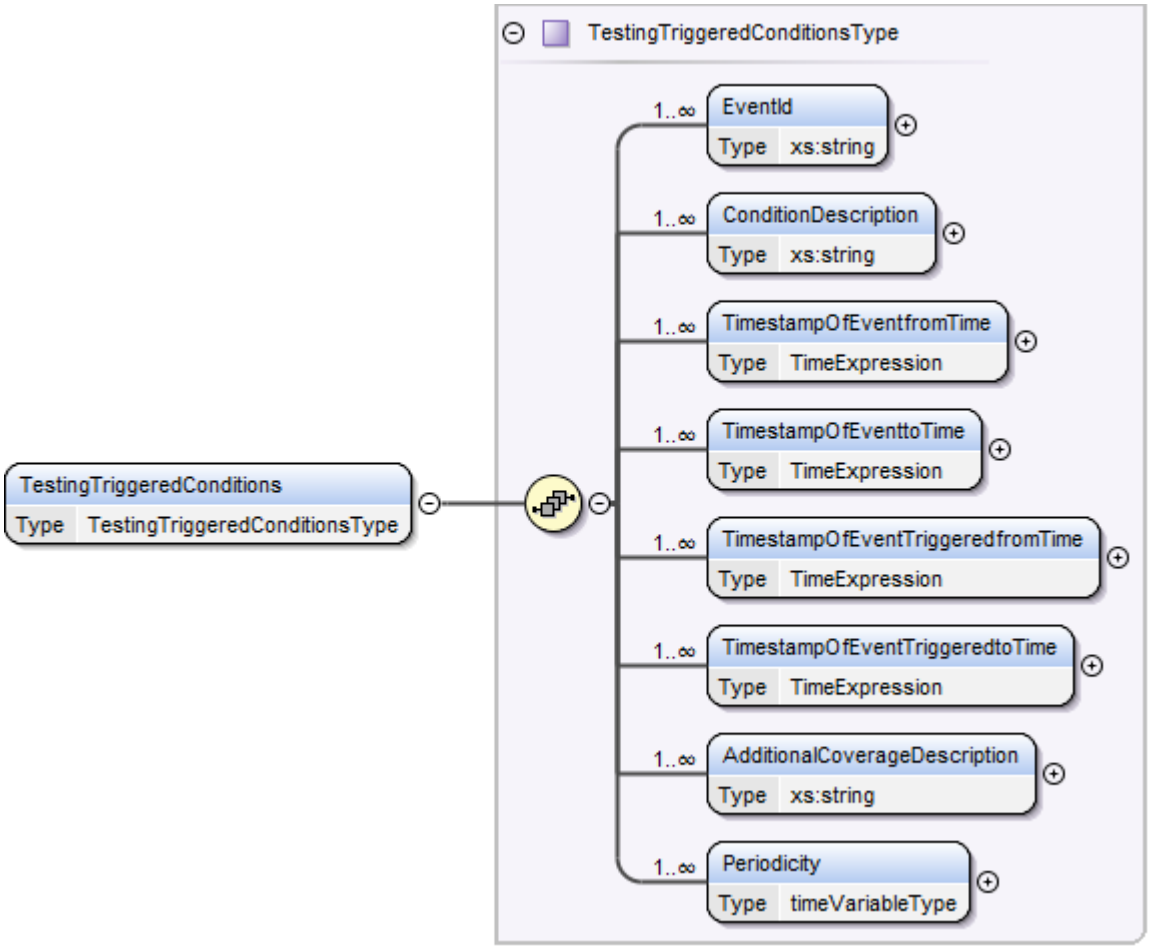

Figure 41 - TestingTriggeredConditionsType

An example of the *TestingTriggeredConditions* sub-element is following. This time, I will use another hybrid model to certify the same security property, that will rely on testing to ensure that every time that an agent that requests a data alteration, it has the authorisation right to do the requested alteration. I will monitor requests for updates of the TOC data through the normal updating interface. However, for every such request that is granted by the TOC, the model will request the execution of a test to check if the entity that requested the update had indeed the authorisation to update data. This example in XML is given below:

```
<TriggerDecisionCondition>
        <TestingTriggeredConditions>
            <EventId>Id1011</EventId>
            <ConditionDescription>compare hash values to check authorisation rights</ConditionDescription>
            <TimestampOfEventfromTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
            </TimestampOfEventfromTime>
            <TimestampOfEventtoTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>2</plus>
                    </Expression>
            </TimestampOfEventtoTime>
            <TimestampOfEventTriggeredfromTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>2</plus>
                    </Expression>
            </TimestampOfEventTriggeredfromTime>
            <TimestampOfEventTriggeredtoTime>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                    </time>
                    <Expression>
                        <plus>6</plus>
                    </Expression>
            </TimestampOfEventTriggeredtoTime>
```

```
            <Periodicity>
                    <time>
                        <varName>seconds</varName>
                        <varType>TimeVariable</varType>
                        <value>1<value/>
                    </time>
            <Periodicity>
        </TestingTriggeredConditions>
</TriggerDecisionCondition>
```

The evidence sufficiency conditions are conditions regarding the minimum extent and the profile of the monitoring and testing events. The *EvidenceSufficiencyCondition* element, as shown in Figure 42, has a unique identifier (*Id*) and defines the conditions of the sufficiency conditions that must apply to monitoring and testing evidence in order to issue a certificate.

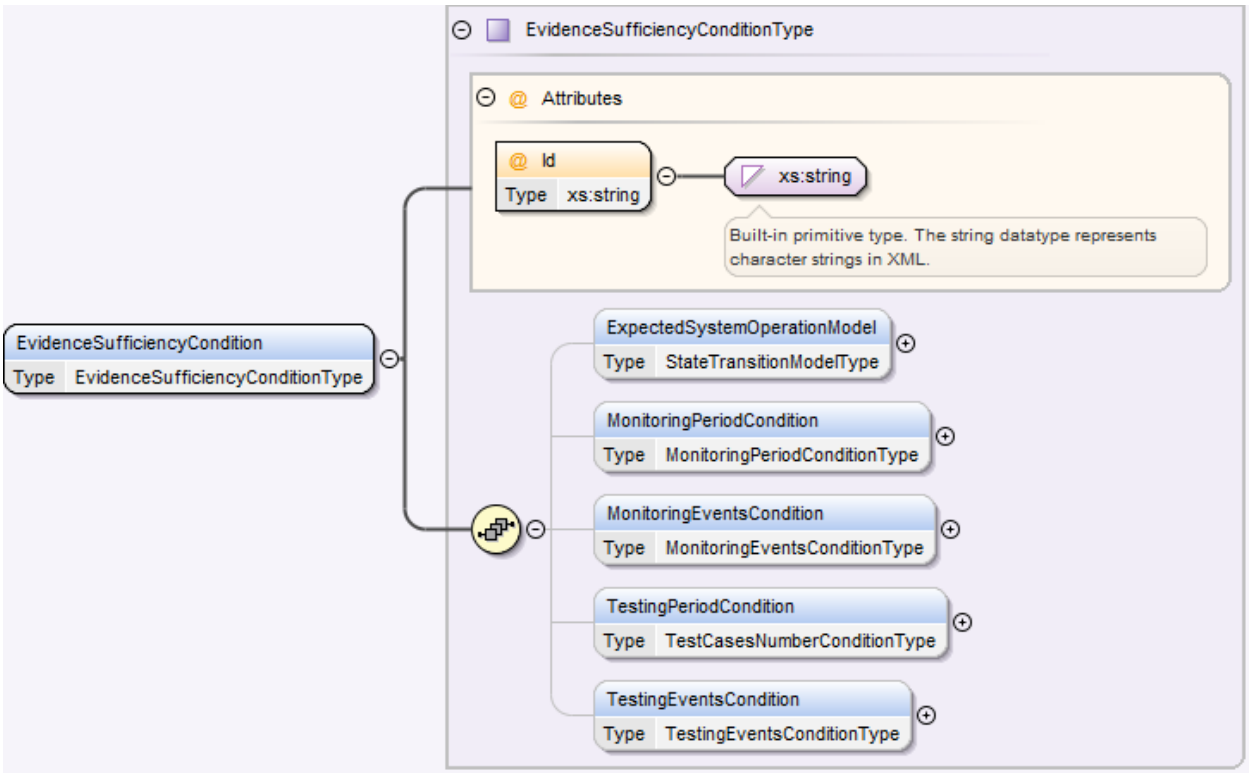

Figure 42 - Evidence Sufficiency Conditions sub-element

The sub-elements are presented below:

- *ExpectedSystemOperationModelCondition* (Figure 43)*:* A condition of this type is used to define an expected operation model of TOC. When the *StateTransitionModel* is

defined, then the gathered evidence will be deemed sufficient for issuing a certificate only if the actual operation of TOC does not deviate from this model.

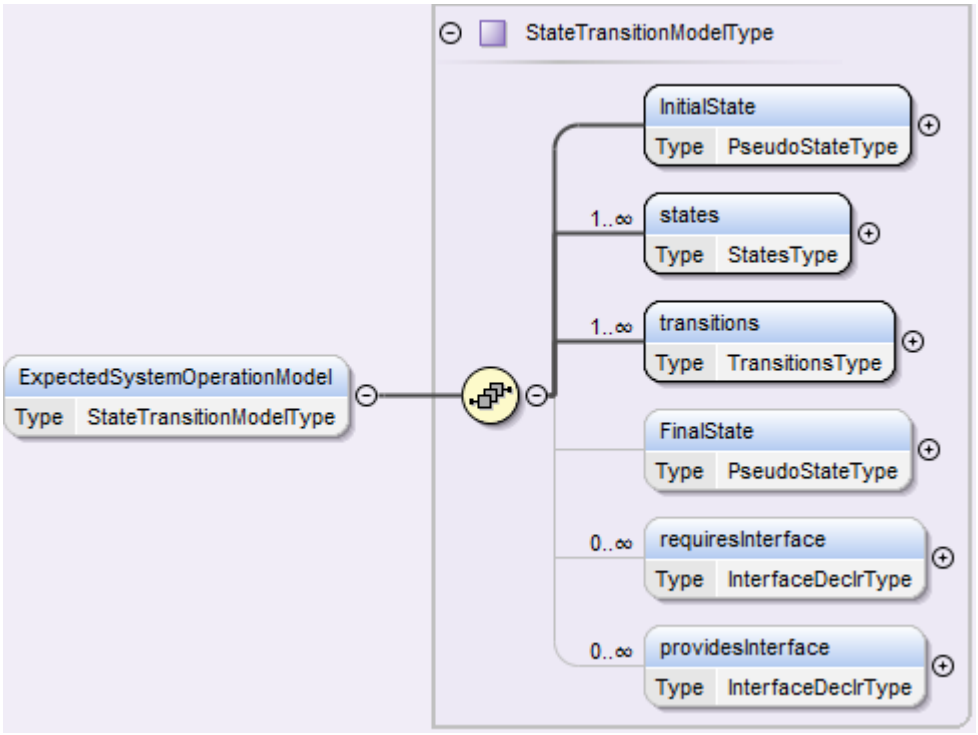

Figure 43 - ExpectedSystemOperationModel sub-element

- *MonitoringPeriodCondition* (Figure 44): A condition of this type can be used to define for how long the TOC should be monitored before the evidence is considered sufficient and a certificate can be issued.

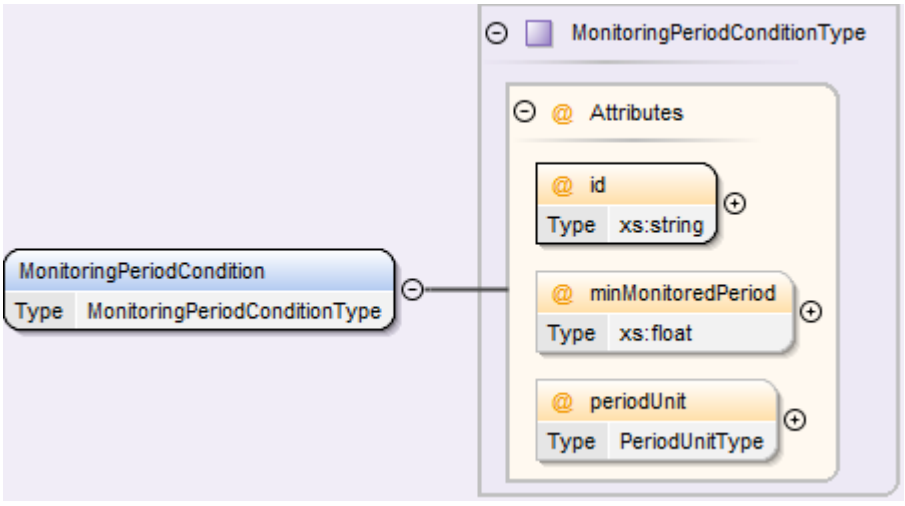

Figure 44 - MonitoringPeiodCondition sub-element

- *MonitoringEventsCondition* (Figure 45): A condition of this type can be used to define the minimum number of monitoring events that should be gathered before a certificate can be issued.

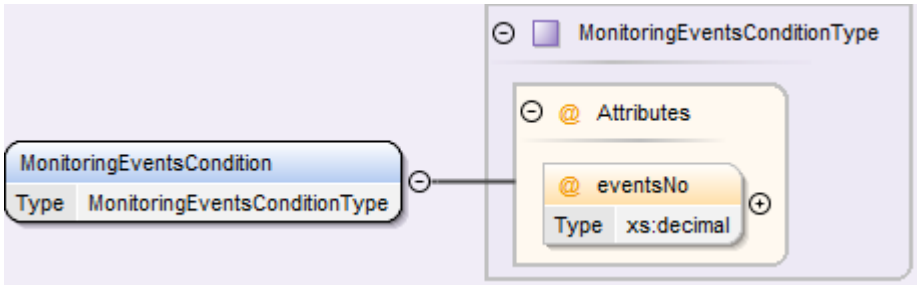

Figure 45 - MonitoringEventsCondition sub-element

- *TestingPeriodCondition* (Figure 46): A condition of this type can be used to define the minimum testing period required for a certificate to be issued.

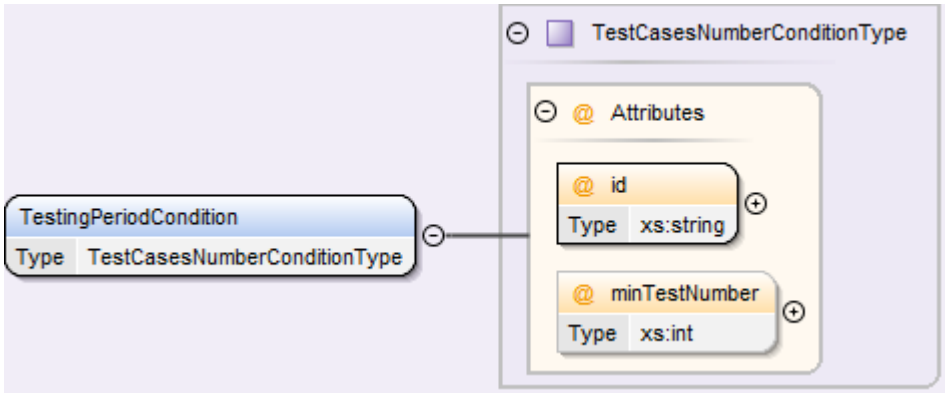

Figure 46 - TestingPeriodCondition sub-element

- *TestingEventsCondition* (Figure 47): A condition of this type can be used to define the minimum number of test events that should be gathered before a certificate can be issued.

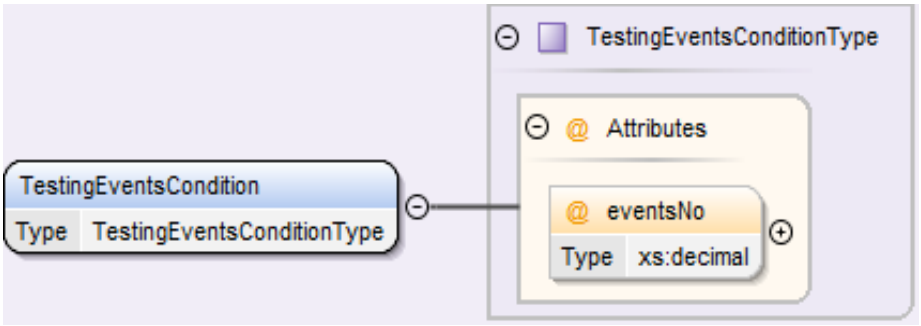

Figure 47 - TestingEventsCondition sub-element

Several examples of that cover all the cases of the *EvidenceSufficiencyCondition* elements are given below. The first example refers to an expected operation model of the TOC and describes the expected states and transitions.

```
<EvidenceSufficiencyCondition Id="1011">
    <ExpectedSystemOperationModel>
       <InitialState stateId="state1" name="Activated"/>
       <states>
            <state>
                <atomicState stateId="state2" name="Issued"/>
            </state>
            <state>
                <atomicState stateId="state3" name="Expired"/>
            </state>
            <state>
                <atomicState stateId="state4" name="Suspend Certificate"/>
            </state>
            <state>
                <atomicState stateId="state5" name="Revoke"/>
            </state>
            <state>
                <atomicState stateId="state6" name="End"/>
            </state>
        </states>
        <transitions>
             <transition Id="1011" From="state1" To="state2">
                <WhenCondition>
                  <event sufficiencyConditionSatisfied />
                 </WhenCondition>
             </transition>
             <transition Id="1100" From="state2" To="state3">
                 <WhenCondition ExpirationReached>
                   <event expirationReached/>
                 </WhenCondition>
```

```
            </transition>
            <transition Id="1001" From="state2" To="state4">
                <WhenCondition>
                  <event violationDetected/>
                </WhenCondition>
            </transition>
            <transition Id="1010" From="state4" To="state5">
               <GuardCondition>
                   <Condition negated="True" relation="NOT-EQUAL-TO">
                     <Operand1><Operand1/>
                     <Operand2><Operand2/>
               </GuardCondition>
            </transition>
            <transition Id="1000" From="state4" To="state6">
                <WhenCondition>
                 <Condition>
                  <event violationResolved/>
                </Condition>
                </WhenCondition>
            </transition>
           </transitions>
</EvidenceSufficiencyCondition>
```

The second example refers to a monitoring period condition and states that the cloud service needs to be monitored for a minimum of 500 hours.

```
<EvidenceSufficiencyCondition Id="1011">
            <MonitoringPeriodCondition id="1011" minMonitoredPeriod="500" periodUnit="hours"/>
</EvidenceSufficiencyCondition>
```

The third example refers to a monitoring events condition and states that a minimum of 1000 events need to be monitored before issuing a certificate.

```
<EvidenceSufficiencyCondition Id="1001">
            <MonitoringPeriodCondition eventsNo="1000"/>
</EvidenceSufficiencyCondition>
```

The fourth example refers to a testing period condition and states that the cloud service needs to be tested for a minimum of 100 hours.

```
<EvidenceSufficiencyCondition Id="1011">
            <TestingPeriodCondition id="1010" minTestPeriod="100" periodUnit="hours"/>
</EvidenceSufficiencyCondition>
```

The fifth example refers to a testing events condition and states that a minimum of 500 events need to be tested before issuing a certificate.

```
<EvidenceSufficiencyCondition Id="1001">
            <TestingEventsCondition eventsNo="500"/>
</EvidenceSufficiencyCondition>
```

The *ExpirationConditions* element as shown in Figure 48 defines when an issued certificate, which has been generated according to the given certification model, should expire and a new one could be issued by considering further evidence.

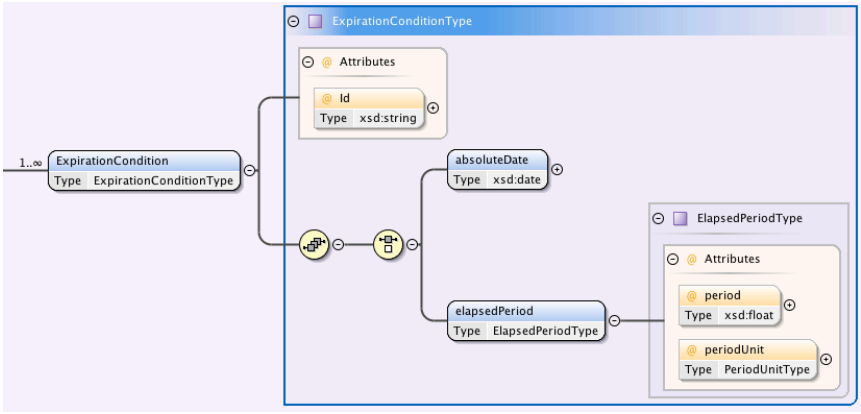

Figure 48 - Expiration Conditions sub-element

An expiration condition has a unique identifier (*Id*) and a choice of two different ways to define the expiration date:

- The first way is by an *absoluteDate*, which is an element of type *date*, or
- The second way is by an *elapsedPeriod*, which is an element of type *ElapsedPeriodType.* An *ElapsedPeriod* element can be used when a certificate needs to expire at the end of a specific period of time from the date that it was issued. An *ElapsedPeriod* element expresses this by defining a period of time, as the number of time units that should elapse following the creation of the certificate.

An example of an expiration condition is given below. According to this example, the certificate will expire 5 months after the issuing date.

```
<ExpirationCondition Id="486">
       <elapsedPeriod period="5" periodUnit="months"/>
</ExpirationCondition>
```

3.5.8 CrosscheckScheme element

This element defines the conditions for cross-checking evidence collected from testing and monitoring. This element also contains further exploration conditions. In case there are discrepancies between the evidence gathered from testing and monitoring, certain data exploration techniques will be applied in order to analyse further the evidence to be correlated and further assessed. This element is new.

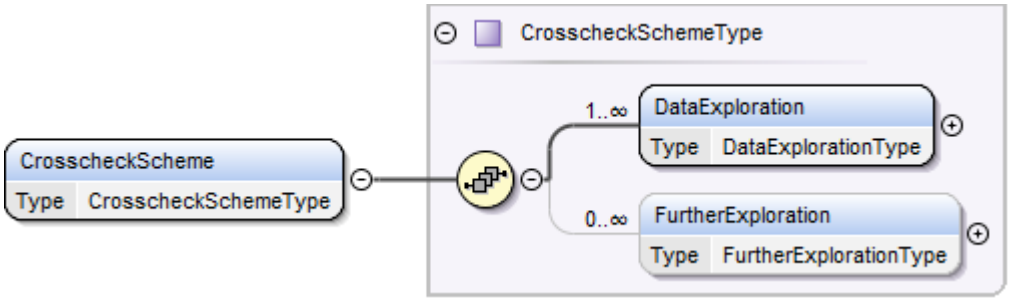

Figure 49 - CrosscheckScheme element

The *CrosscheckScheme* element, as shown in Figure 49, includes a *DataExploration* sub-element and an optional *FurtherExploration* sub-element. The *DataExploration* sub-element is mandatory and includes the conditions for cross-checking evidence collected through monitoring and testing. The *FurtherExploration* sub-element contains the conditions for exploring further testing and monitoring cases when violations persist, enabling the expression of sub-periods. These sub-elements are analysed below.

The *DataExploration* element includes the following elements:

- the *CrossCheckDescription* element (Figure 50) which is of string type and includes a general description of the evidence that needs to be cross-checked.

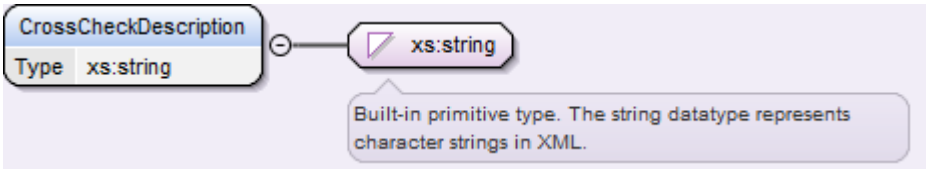

Figure 50 - CrossCheckDescription sub-element

For example, if I want to certify the availability security property (i.e. the ability of a TOC to produce a non-faulty response within a certain period of time) I will measure the percentage of calls that satisfy this condition over an assessment period. An independent hybrid model for the certification of TOC availability could be based on collecting evidence regarding the availability of a TOC through monitoring and testing independently (i.e., without any of these activities being triggered by outcomes of the other). This example in XML is given below:

```
<CrossCheckDescription> compare the percentage of calls where a non-faulty response is produced throughmonitoring and testing </CrossCheckDescription>
```

- the *ComparePerformanceValues* element (Figure 51), which is of *ComparePerformanceValuesType* and allows the comparison of the absolute difference between the monitoring and the testing performance values and a value that indicates the maximum accepted difference between the testing and monitoring evidence, for which the evidence will be considered valid.

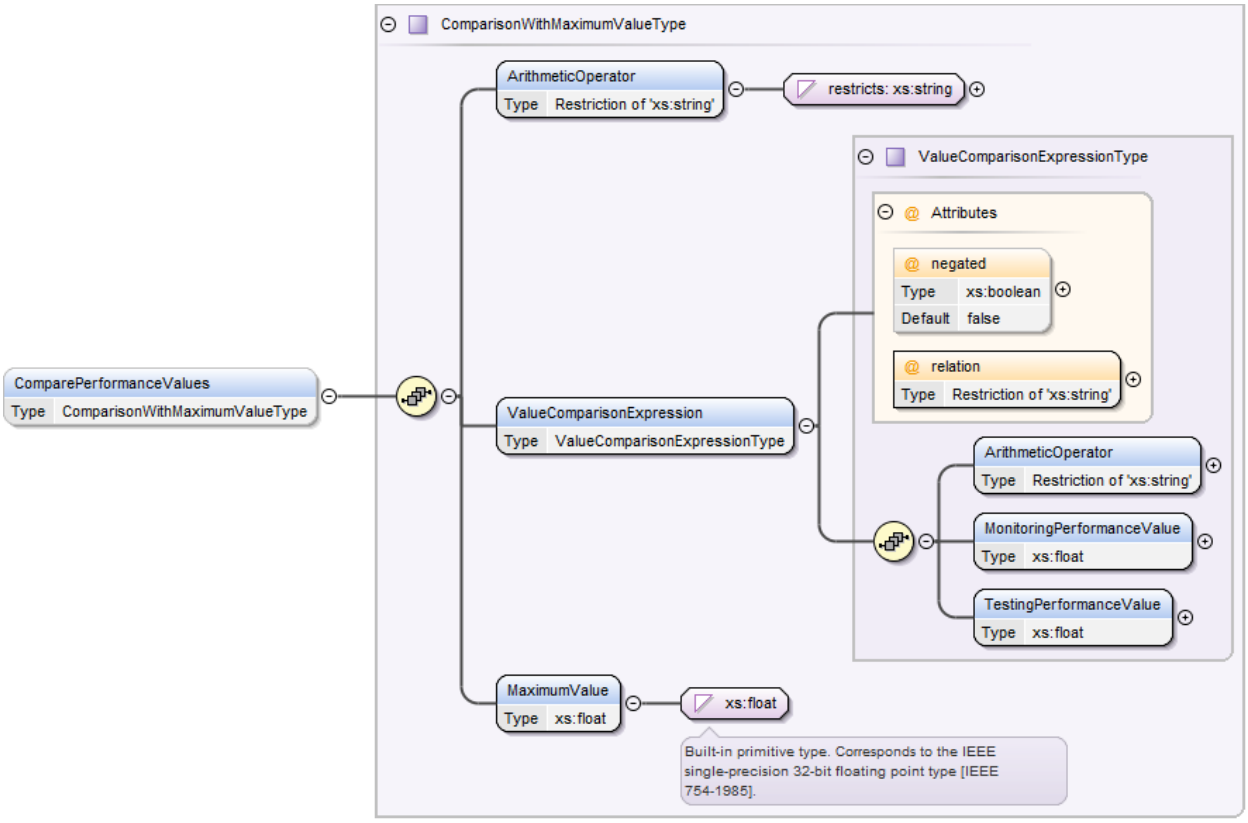

Figure 51 - ComparePerformanceValues sub-element

When I want to certify the availability security property and I compare the percentage of calls where a non-faulty response is produced through monitoring and testing, an availability measure based on testing (or monitoring) evidence will be used for issuing a certificate only if the availability measure based on monitoring (or testing) evidence over the same period is no more than 1% different from it.

```
<ComparePerformanceValues>
      <ArithmeticOperator>LESS-THAN-EQUAL-TO</ArithmeticOperator>
           <ValueComparisonExpression relation="ABSOLUTE-VALUE">
                 <ArithmeticOperator>MINUS</ArithmeticOperator>
                 <MonitoringPerformanceValue> </MonitoringPerformanceValue>
                 <TestingPerformanceValue> </TestingPerformanceValue>
            </ValueComparisonExpression>
      <MaximumValue>0.01</MaximumValue>
</ComparePerformanceValues>
```

- the *SufficientMutualObservedPeriodsSatisfaction* sub-element (Figure 52), which specifies the period where testing and monitoring are performed in parallel, and

compares it with a specific value. It is of *SufficientMutualObservedPeriodsSatisfactionType* and the definition is given below.

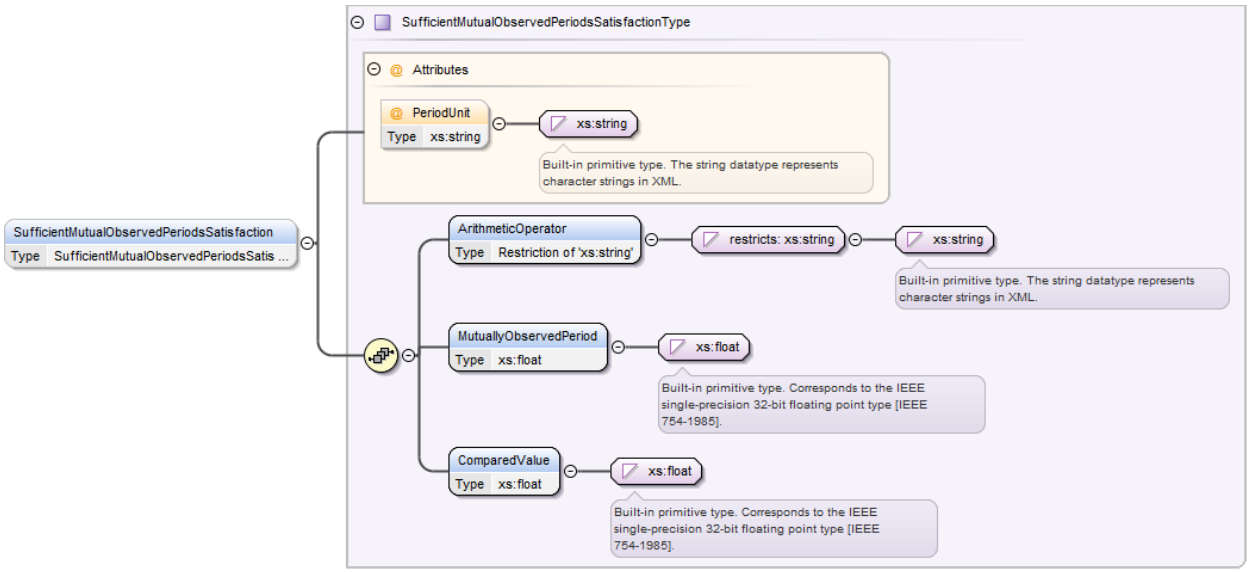

Figure 52 - SufficientMutualObservedPeriodsSatisfaction element

An example of the sub-element above is described here. When I certify the availability security property and I cross-check monitoring and testing evidence, one way of correlating evidence is to consider the testing events as valid evidence of TOC availability only if they were performed during the range [t_{mon}, t_{mon}] and the mutual observed time where testing and monitoring took place in parallel was more than 5200 seconds.

```
<SufficientMutualObservedPeriodsSatisfaction PeriodUnit="seconds">
        <ArithmeticOperator>GREATER-THAN-EQUAL-TO</ArithmeticOperator>
        <MutuallyObservedPeriod> </MutuallyObservedPeriod>
        <ComparedValue>5200.00</ComparedValue>
</SufficientMutualObservedPeriodsSatisfaction>
```

- the *CrossCheckStatus* sub-element (Figure 53), which describes if the cross-checking is successful or if evidence is violated and more testing and monitoring cases are required. The values of this element can be "CROSS-CHECKING-SUCCESSFUL" if the conditions above are satisfied or "EVIDENCE-VIOLATED" if the conditions above are not satisfied.

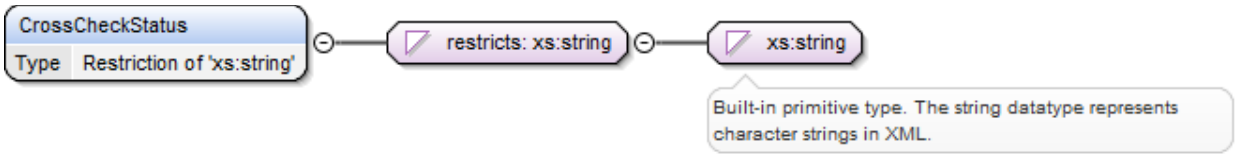

Figure 53 - CrossCheckStatus sub-element

The *FurtherExploration* sub-element (Figure 54) contains the *AssessmentTimeExploration* sub-element and describes the expression of sub-periods. This specific element is particularly important to check if there are time patterns that underpin violations and inconsistencies between testing and monitoring evidence. This element contains the *AssessmentTimeExploration* sub-element, as shown below.

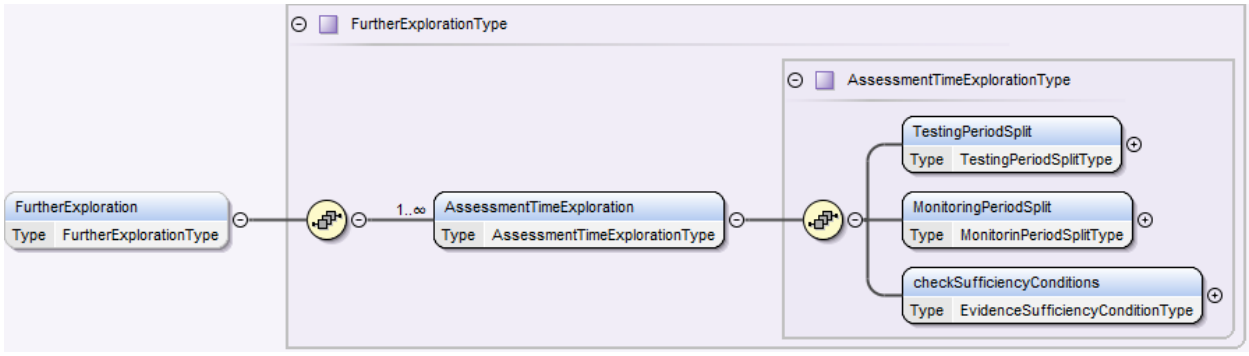

Figure 54 - FurtherExploration element

The *AssessmentTimeExploration* sub-element (Figure 55) contains the *TestingPeriodSplit* sub-element that helps us specify the division of the testing period into smaller sub-periods, the *MonitoringPeriodSplit* sub-element that helps us specify the division of the monitoring period into smaller sub-periods and the *checkSufficiencyConditions* sub-element. The *AssessmentTimeExploration* sub-element allows different splits of the monitoring and testing periods. After splitting the assessment periods into smaller sub-periods, I will check the sufficiency conditions of the evidence in order to explore if there are time patterns underpinning the deviations.

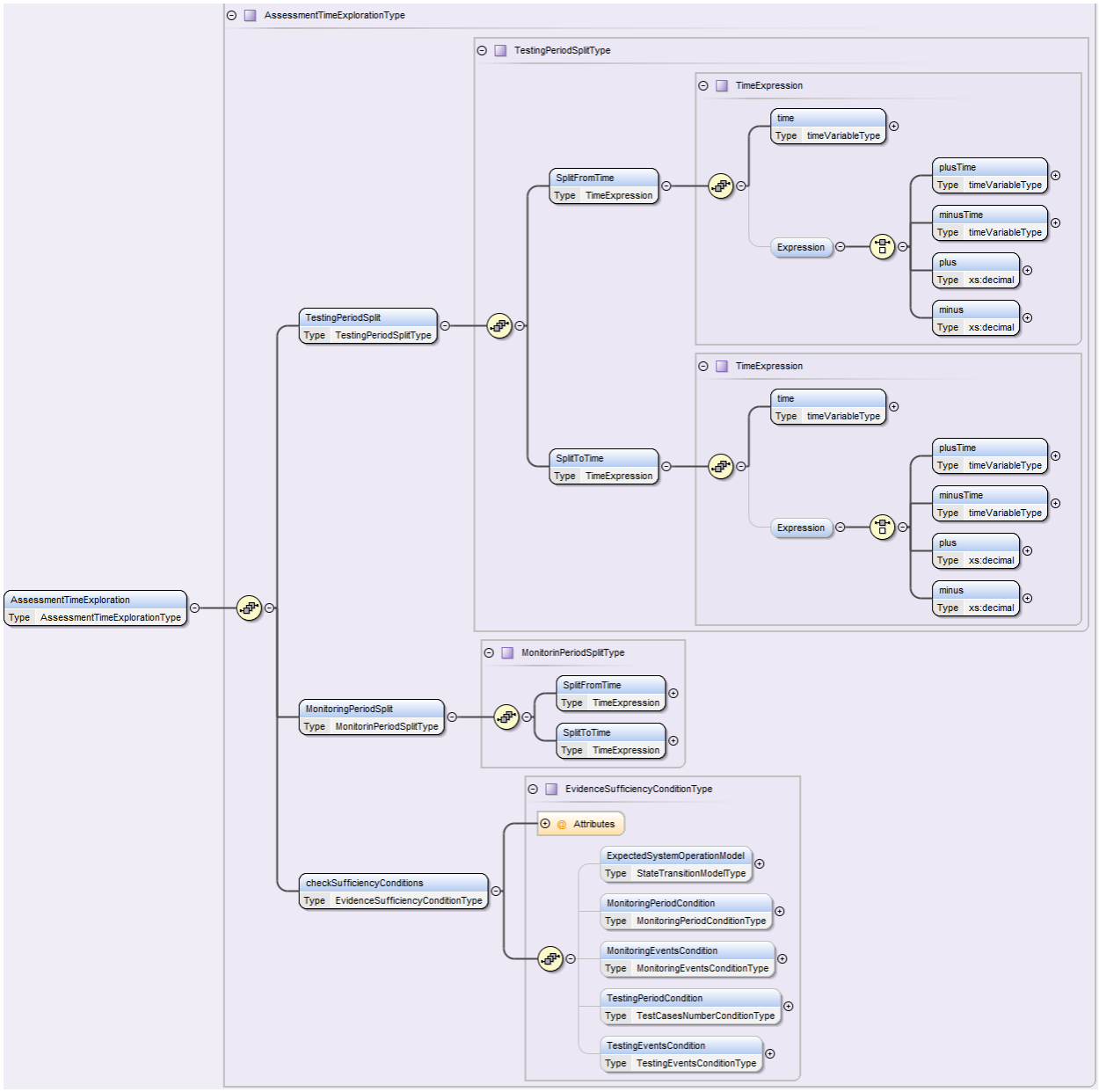

Figure 55 - AssessmentTimeExploration element

An example of the sub-element above is provided at this point. If I want to certify the *availability* security property and the ability of a TOC to produce a non-faulty response within a certain period of time and there are deviations between the percentage of calls that satisfy this condition over an assessment period, concerning the evidence collected through monitoring and testing, I can shorten the testing period by x time units, and the monitoring period by y time units, to check if the deviations will persist. This action will allow the discovery of time patterns.

This can be very useful when, for instance, I observe that the availability is very low at specific periods of time. So, in this case, by splitting the time periods I can make further explorations. It is very important the fact that the model allows the expression of different sub-periods for monitoring and testing, as at a specific time period, it might be possible to collect enough evidence of one mode of assessment, but the collection of evidence for the sub-ordinate form of assessment may be impossible. If, for example, I split the monitoring and testing period evenly, but with testing I collect sufficient evidence and with monitoring I do not manage to collect the sufficient number of monitoring evidence, then I can further split the monitoring period to check time patterns. According to the following example, I shorten the testing period by 5 second and the monitoring period by 10 seconds and then I check if the evidence sufficiency conditions are satisfied.

```
<AssessmentTimeExploration>
                <TestingPeriodSplit>
                    <SplitFromTime>
                        <time>
                            <varName>seconds</varName>
                            <varType>TimeVariable</varType>
                        </time>
                    </SplitFromTime>
                    <SplitToTime>
                        <time>
                            <varName>seconds</varName>
                            <varType>TimeVariable</varType>
                        </time>
                        <Expression>
                            <minus>5</minus>
                        </Expression>
                    </SplitToTime>
                </TestingPeriodSplit>
                <MonitoringPeriodSplit>
                    <SplitFromTime>
                        <time>
                            <varName>seconds</varName>
                            <varType>TimeVariable</varType>
                        </time>
                    </SplitFromTime>
                    <SplitToTime>
                        <time>
                            <varName>seconds</varName>
                            <varType>TimeVariable</varType>
                        </time>
```

```
                        <Expression>
                            <minus>10</minus>
                        </Expression>
                    </SplitToTime>
                </MonitoringPeriodSplit>
                <checkSufficiencyConditions Id="Id14">…</checkSufficiencyConditions>
</AssessmentTimeExploration>
```

3.6 Architecture of the proposed framework for implementation

The architecture for hybrid certification will be an extension of the existing architecture that has been developed for the CUMULUS project and is defined in [168]. Below, I will present an overview of the architecture and I will describe the main components that are required for the realisation of hybrid certification. The interfaces for the Certification Manager, Certification Communicator, Monitoring Manager, Monitoring Module, Testing Manager and Testing Module are already implemented for the CUMULUS project, so these interfaces will not be presented here. The components that I introduce are the Hybrid Manager, the Hybrid Certification Generator Attestation, the Hybrid Lifecycle Manager, the Hybrid Aggregator, the Hybrid Evidences database, the Hybrid Certification Models database and the Hybrid Certificates database. These components represent my contribution concerning the architecture of the CUMULUS framework. The architecture for hybrid certification is shown in Figure 56, and is based on the CUMULUS Framework Prototype Architecture Figure 56 - Architecture for Hybrid Certification[168]-Figure 1.

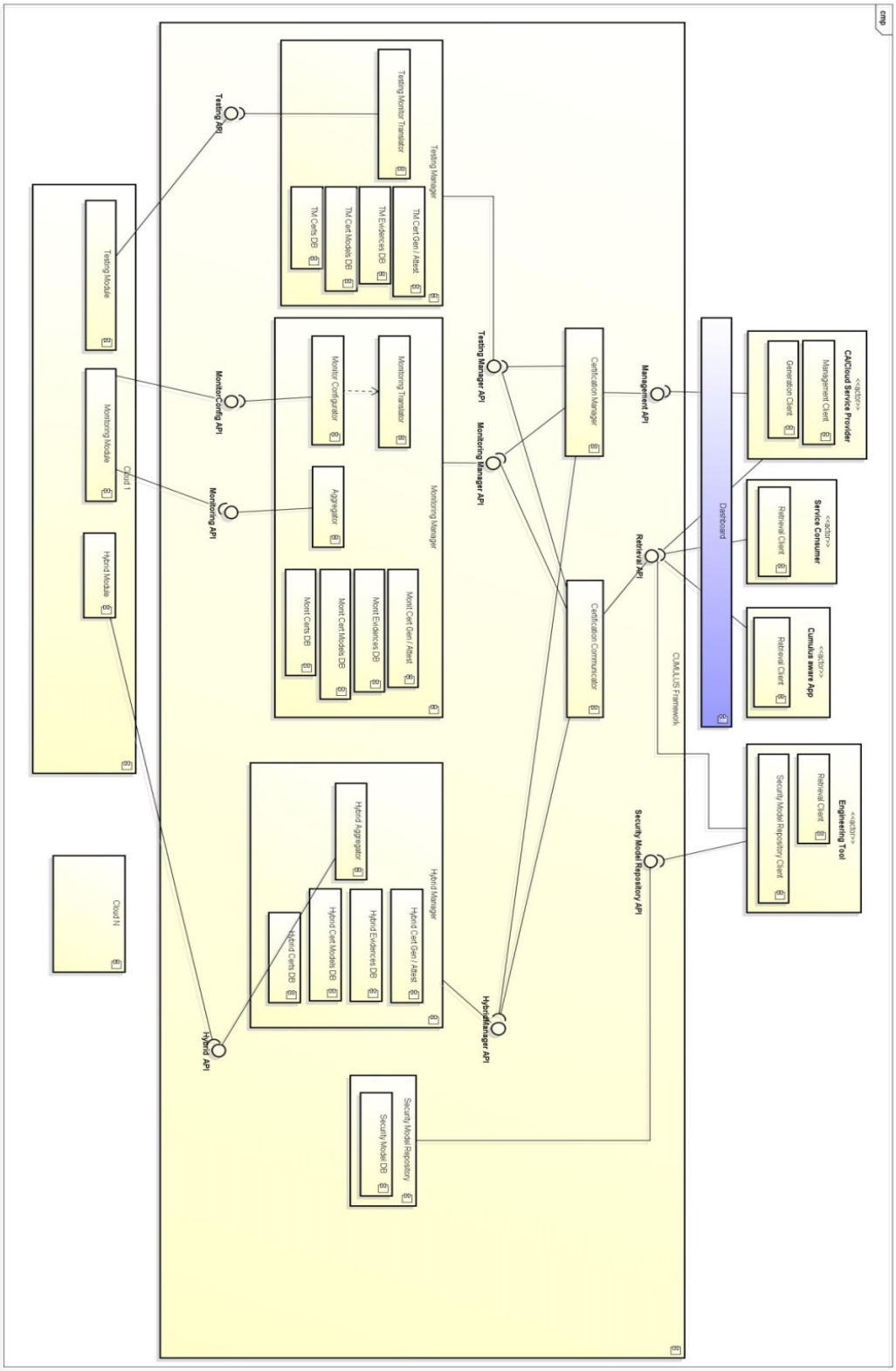

Figure 56 - Architecture for Hybrid Certification

- ***Certification Manager***

This component is responsible for communicating with the actors to create or modify the certification models and will start the certification process by delegating the appropriate managers.

- ***Certificate Communicator***

This component is responsible for communicating with the actors who requested a certificate, for sending them the certificate and for notifying them if there is an update about changes in the certificate.

- ***Monitoring Manager***

This component is responsible for detecting the availability of the monitoring infrastructure, for assembling and configuring the infrastructure, for initiating the monitoring process, for collecting the monitoring evidence and for aggregating them.

- ***Monitoring Module***

This component is responsible for monitoring the events for a specific security property and Target of Certification (TOC).

- ***Testing Manager***

This component is responsible for receiving inputs from various actors, starting the certification process and delegating the Testing Module to perform the tests.

- ***Testing Module***

This component is responsible for performing tests on the TOC for a specific property and receiving the set of results.

- ***Hybrid Manager***

This component is responsible for collecting all the evidence for hybrid certification and communicates with the Testing Manager and the Monitoring Manager through the Certification Communicator. The Hybrid Manager will decide how the different types of evidence will be combined and how, based on the information that is defined in the Certification Model. The specifications and the operations provided by this component are presented below.

Hybrid Manager API		
Operation		**Description**
submitCertificationModel (String model)		This method allows submitting a certification model to the Hybrid Manager
Parametres		
model	String	String representation of the XML certification model

Table 2 - Hybrid Manager API

- ***Hybrid Certification Generator Attestation (Hybrid Generator Attestation)***

This component is responsible for generating a certificate when enough data are collected through monitoring and testing. More specifically, this component receives the certification model from the Certification Manager through the Generation API and creates the certificate if the sufficiency conditions are fulfilled and if the expiration and violation conditions are not reached. Additionally, this component stores the hybrid certificates issued in the Hybrid Certificates DB. The specifications and the operations provided by this component are presented below.

Generation API	
Operation	**Description**
submitCertificationModel (String model)	This method allows submitting a certification model to the Certificate Generator

Generation API		
Operation		**Description**
Parametres		
model	String	String representation of the XML certification model

Table 3 - Generation API

The Hybrid Certification Generator Attestation includes the Hybrid Lifecycle Manager that is described below.

- ***Hybrid Lifecycle Manager***

This component is responsible for defining all the states and the transitions in the certification model. Additionally, it contains information about the conditions for lifecycle transitions, indicating under what conditions the states will be reached and how they will be altered. This Lifecycle model is different from the one used in Monitoring-based certification and the one used in Test-based certification as presented in the CUMULUS project, since I have introduced the elements for cross-checking and analysing evidence.

- ***Hybrid Aggregator (Hybrid Aggregation Manager)***

This component is responsible for aggregating the hybrid results collected in order to issue a certificate. This component aggregates monitoring and testing results and stores them in the "Hybrid Evidences DB" of the framework, as required by the given certification model. The aggregation manager polls the detailed evidence at regular intervals.

- ***Dashboard***

This component provides the user a web-based frond-end interface for requesting the generation of a certificate or for retrieving a specific certificate from the framework.

- ***Hybrid Evidences DB***

This component records the aggregated evidence according to the Hybrid Aggregator and is inserted in the certificates. This component is new and is based on the common structure of the Monitoring and Testing evidences databases.

Hybrid Evidences DB	
Element	**Description**
AggregatedEvents_Id	Id of the aggregated evidence
CM_Instance_Id	Id of the Certification model instance
Creation_Time	The time the aggregated evidence was created
Start_Time	The time the aggregation period started
End_Time	The time the aggregation period ended
Assertion_Id	The Id of the assertion of the security property
Result	The result that expresses violation or satisfaction
Evidence_XML	Aggregated evidence as XML
Evidence_JAVA	Aggregated evidence as JAVA object

Table 4 - Hybrid Evidences Database

- ***Hybrid Cert Models DB***

This component keeps a record of all certification models created. This component is new and is based on the common structure of the Monitoring and Testing certification models databases.

Hybrid Cert Models DB	
Element	**Description**
CM_Id	Id of the certification model
CM_Instance_Id	Id of the Certification model instance
CA_Signature	The signature of the certification authority

Hybrid Cert Models DB	
Element	**Description**
Security property	The definition of the security property
Assertion_Id	The Id of the assertion of the security property
Assertion	The specification of the security property
TOC_Name	The name of the TOC
Toc_Id	The ID of the TOC
Evidence_XML	Aggregated evidence as XML
Evidence_JAVA	Aggregated evidence as JAVA object

Table 5 - Hybrid Certification Models Database

- ***Hybrid Certs DB***

This component keeps a record of all generated certificates. This component is new and is based on the common structure of the Monitoring and Testing certificates databases.

Hybrid Certs DB	
Element	**Description**
certId	Id of the certificate
Cert_SerialNo	Serial Number of the certificate
Security property	The definition of the security property
TOC_Name	The name of the TOC
Valid_From	Date when the certificate starts to be valid
Valid_Until	Date when the certificate stops to be valid
certString	The certificate as a sting represented in XML
certObject	The certificate as a JAVA object

Table 6 - Hybrid Certificates Database

3.7 Flow of Action

At this section I will present the sequence diagram for issuing a certificate.

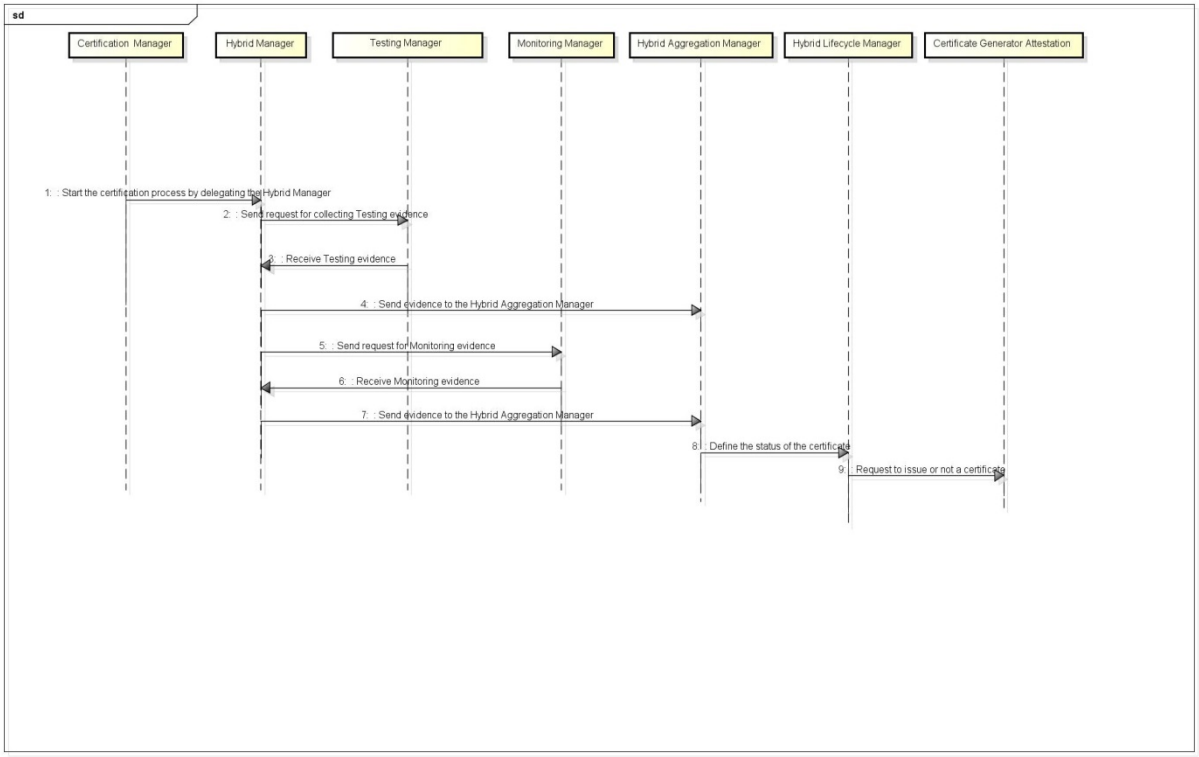

Figure 57 - Sequence Diagram for Issuing a Hybrid Certificate

When the user of the tool requests a certificate, then the Certificate Manager starts the certification process and delegates the Hybrid Manager to communicate with the Testing Manager and the Monitoring Manager in order to collect all the information for hybrid certification. The Hybrid Manager sends a request to the Testing Manager to collect the testing evidence. When it receives the testing evidence, the Hybrid Manager sends a request to the Hybrid Aggregation Manager to store the testing evidence and also sends a request to the Monitoring Manager to collect the monitoring evidence. When the Monitoring Manager responds to the Hybrid Manager, then, the Hybrid Manager sends a request to the Hybrid Aggregation Manager to store the monitoring evidence. The Hybrid Aggregation Manager will send a request to the Hybrid Lifecycle Manager, which defines the status of the certificate. Finally, if a certificate can be issued, the Certificate Generator Attestation will issue a certificate.

3.8 Hybrid, independent mode model (using EC-Assertion)

The following example shows the use of a hybrid approach in certifying cloud service availability using EC-Assertion. As defined in [160], this property expresses the ability of a TOC to produce a non-faulty response within a certain period of time and is measured by the percentage of calls that satisfy this condition over an assessment period. An independent hybrid model for the certification of TOC availability could be based on collecting evidence regarding the availability of a TOC through monitoring and testing independently (i.e., without any of these activities being triggered by outcomes of the other) and then correlating and cross-checking the collected pools of evidence to produce a hybrid assessment of the property. More specifically, the hybrid model could include monitoring formulas to record instances of invocation of TOC operators where TOC produced a response within the acceptable time limit and the instances where it did not, and keep a record of counters of these instances from which an overall availability measure could be drawn. The formulas that could be used to collect this monitoring evidence are as follows:

Assumption A1 (monitoring evidence):

Happens(e(_e1,_CA,_TOC,REQ,_OP(_data),_TOC),t1,[t1,t1])^

Happens(e(_e2,_TOC,_CA,RES,_OP(_data),_TOC), t2,[t1,t1+t_{av}]) ^

HoldsAt(MCounterA(_TOC,_MCA),t2)

⇒

Terminates(_e1,MCounterA(_TOC, _MCA), t2) ^

Initiates(_e1,MCounterA(_TOC, _MCA+1), t2)^

Initiates(_e1,MAvail(_TOC,_OP(_data),t2-t1), t2)

Assumption A2 (monitoring evidence):

Happens(e(_e1,_CA,_TOC,REQ,_OP(_data),_TOC),t1,[t1,t1])^

¬ **Happens**(e(_e2,_TOC,_CA,RES,_OP(_data),_TOC),t2,[t1,t1+t_{av}]))^

HoldsAt(MCounterU(_TOC,_MCU), t2)

⇒

Terminates(_e1,MCounterU(_TOC,_MCU), t2) ^

Initiates(_e1,MCounterU(_TOC, _MCU+1), t2) ^

Initiates(_e1,MUnav(_TOC,_OP(_data),t2−t1),t2)

The first of the above monitoring formulas (i.e., assumption A1) monitors calls to any operation in a _TOC and the responses to them (see events Happens(e(_e1, _CA, _TOC, REQ, _OP(_data),_TOC), t1, R(t1,t1)) and Happens(e(_e2, _CA, _TOC, RES, _OP(_data), _TOC), t2,[t1,t1+t_{av}])) and if a response is within the required period (t_{av}), it updates the counter of instances where _TOC was available and records the related call (in fluents MCounterA(_TOC,_MCA,t2) and MAvail(_TOC,_OP(_data),t2−t1), respectively). The second formula (i.e., assumption A2) monitors calls to _TOC operations that did not produce a response within the required time, and keeps an overall counter of unavailability and the related calls in fluents MCounterU(_TOC,_MCU,t2) and MUnav(_TOC,_OP(_data),t2−t1).

The hybrid model for the certification of availability could also incorporate a test-based availability assessment sub-model. This sub-model can execute a randomly selected operation in the interface of _TOC periodically to check its availability, and keep a record of instances of test-triggered invocations of operations of the TOC in which a response was produced within the required time period, and instances of test-triggered invocations where it was not.

This sub-model is expressed by the following formulas for collecting testing evidence:

Assumption A3 (testing evidence):

Happens(e(_e1,_CA,_TOC, EXC(T_{per}), _x=random(interface(_TOC)),_TOC),t1,[t1,t1])
^

Happens(e(_e2,_TOC, _CA RES,_x,_TOC),t2,[t1,t1+t_{av}])^

HoldsAt(TCounterA(_TOC,_TCA),t2)

⇒

Terminates(_e1,TCounterA(_TOC,_TCA),t2) ^

Initiates(_e1,TCounterA(_TOC,_TCA+1),t2)^

Initiates(_e1,TAvail(_TOC,_x,t2-t1),t2)

Assumption A4 (testing evidence):

Happens(e(_e1,_CA,_TOC, EXC(T_{per}), _x=random(interface(_TOC)),_TOC),t1,[t1,t1])
^

¬ **Happens**(e(_e2, _TOC, _CA, RES,_x,_TOC),t2, [t1,t1+t_{av}])^

HoldsAt(TCounterU(_TOC,_TCU),t2)

⇒

Terminates(_e1,TCounterU(_TOC,_TCU),t2) ^

Initiates(_e1,TCounterU(_TOC,_TCU+1),t2) ^

Initiates(_e1,TUnav(_TOC,_x, t2-t1),t2)

A3 and A4 are similar to assumptions A1 and A2 respectively except that, instead of monitoring real operation calls, they execute a randomly selected operation in the interface of _TOC periodically (see the event Happens (e (_e1, _CA, _TOC, EXC(T_{per}), _x=random (interface(_TOC)), _TOC),t1,[t1,t1])) to check its availability, and update fluents recording the overall counters of availability and unavailability of _TOC and the test executions that revealed them.

In the hybrid model, the assumption pairs (A1, A2), and (A3, A4) are used to collect evidence independently without any monitoring events triggering tests or vice versa. However, it might still be desirable to correlate the testing and monitoring evidence, according to the hybrid certification model schema I have introduced. The overall availability measure may be computed on the basis of combining both test and monitoring evidence as A = (_MCA + _TCA)/(_MCA+ _TCA+_MCU+ _TCU), so an aggregate assessment is formulated from both monitoring and testing evidence. Concerning the evidence sufficiency conditions, I will check if the monitoring period is greater than the minimum monitoring period required and if the testing period is greater than the minimum testing period required. Additionally, I will check if the number of the monitoring and testing events is greater than the minimum number of monitoring and testing

events. Concerning the cross-check conditions, I will compare the monitoring and testing assessment period against the minimum assessment period required for this property and I will check if the absolute value of the difference between the _MCA and _TCA is greater than a minimum value defined for this property. If the above conditions are not satisfied then I will revoke the certificate. If violations persist, I will consider extending the assessment period and collecting more testing and monitoring events. Furthermore, I can split the monitoring and testing periods into smaller sub-periods and I can check again the sufficiency conditions of the evidence in order to understand if there are time patterns underpinning the deviations.

It becomes clear that in this example of independent hybrid certification, where testing and monitoring take place independently, without having one form of assessment triggered by the outcomes of the other form, I will capture more cases than performing only testing or only monitoring (Figure 58). Consequently, with testing I can discover cases that might have escaped monitoring and vice versa.

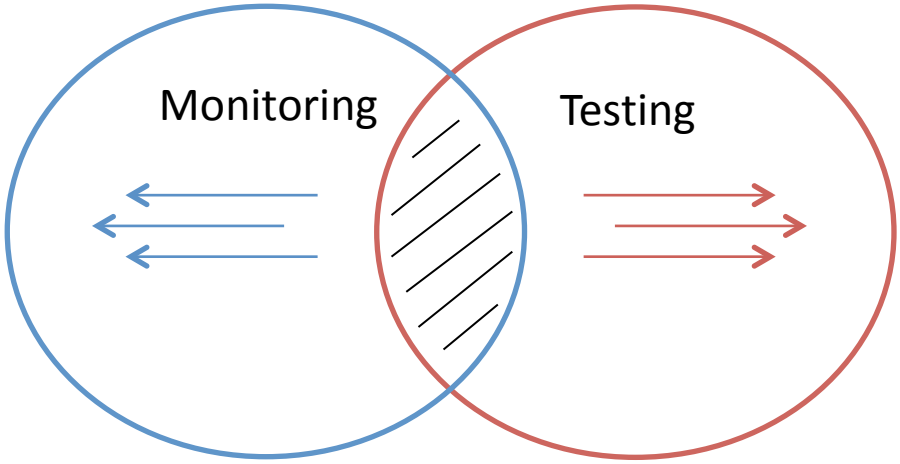

Figure 58 - Cases covered by Independent-mode Certification Models

Examples of Hybrid, independent-mode model cases

Below I will present the hybrid independent-mode certification models of certain security properties as defined in [160]. These hybrid models will verify either the completeness or the trustworthiness of the assessment.

HTTP to HTTPS redirection

This security property corresponds to the *AIS:authentication:network-authenticated-server-access* security property in [160]. The TOC provides a communication channel between a client and itself, which offers the following guarantees:

(1) The client receives assurance that he is communicating with the TOC.

(2) The integrity of the exchange of data between the client and the TOC is protected with guarantees between exchanged messages.

The property is considered as verified if the client communicates with a webserver (TOC) using SSL/TLS and verifies the authenticity of the webserver with a certificate, providing adequate encryption. This security property guarantees that HTTP requests are redirected to HTTPS and that the network channel is confidential, verifying the authenticity of the user and the web server. Every time there is a request call from a client to a webserver (HTTPserver) and the request for data transmission is to port 80 which indicates an unsecure communication, then the monitor will check for the existence of another event that will contain the response of the web server back to the client, notifying the execution of the redirection to the HTTPS port (port 443). The "location" attribute contains the location header which is a string that indicates if there is redirection to the HTTPS port, the "HTTPStatus" attribute which includes the status code of the response and the "serverAddress" attribute which indicates the number of the port.

HTTPS (HTTP over SSL or HTTP Secure) is the use of Secure Socket Layer (SSL) or Transport Layer Security (TLS) as a sublayer under regular HTTP application layering. HTTPS encrypts and decrypts user page requests as well as the pages that are returned by the web server. The use of HTTPS protects against eavesdropping and man-in-the-middle attacks. HTTPS and SSL support the use of X.509 digital certificates from the server so that, if necessary, a user can authenticate the sender. Unless a different port is specified, HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP. The security of HTTPS is therefore that of the underlying TLS, which uses long-term public and secret keys to exchange a short-term session key to encrypt the data flow between client and server [169].

The monitoring rules below will demonstrate that the TOC redirects every HTTP request to HTTPS every time I see a response from the webserver to the client where `(_location2 = locationHTTPS) and (_HTTPStatus2 = RedirectStatus).` Also, the rule demonstrates that the TOC provides a secure communication (SSL/TLS connection) between the client and the server

of the communication. The SSL/TLS encryption validation will verify the presence of SSL/TSL channel encryption.

Rule R1 (monitoring evidence):

Happens (e(_e1, _client, _webserver, REQ, _httpCall(location, _serverAddress, _HTTPStatus), _webserver),t1, [t1,t1]) ^

(_serverAddress = port80)

⇒

Happens (e(_e2, _webserver, _client, RES, _httpCall(_location2, _serverAddress2, _HTTPStatus2), _webserver),t2, [t1,t1+t_{av}]) ^

(_location2 = locationHTTPS) ^

(_HTTPStatus2 = RedirectStatus) ^

(_serverAddress = port443)

With testing, I will verify that every time I force data transmission to a webserver through port 80, the TOC provides secure communication by redirecting to the HTTPS port (port 443) and by verifying the presence and the robustness of the SSL/TSL channel encryption (by executing the `_dataTransmit` operation).

Rule R2 (testing evidence):

Happens (e(_e1, _AI, _webserver, EXC(Tper), _dataTransmit(_encrCode, _serverAddress), _webserver) ,t1, [t1,t1]) ^

(_serverAddress = port80)

⇒

Happens (e(_e2, _webserver, _AI, RES, _dataTransmit(_encrCodeVerified, _serverAddress2), _webserver),t2, [t1,t1+t_{av}]) ^

(_encrCodeVerified ≠Nil) ^

(_serverAddress2 = port443)

If with testing I cannot confirm the redirection to the HTTPS port (port 443), then the hybrid certificate will be revoked and I will proceed to data exploration. If violations persist, I will proceed to further exploration. This means that I can extend the assessment period or I can split the monitoring and testing periods into smaller sub-periods to check if there are time patterns underpinning these deviations.

Percentage of timely incident reports

This security property corresponds to the *SEF:incident-management-quality:percentage-of-timely-incident-reports* security property in [160]. This security property describes the percentage of incidents that are reported within a predefined time limit after their discovery, over the total number of incidents discovered. The independent hybrid model will prove the trustworthiness of outcomes from testing and monitoring. The attribute "`maxtime`" indicates the maximum reporting time limit. This security property is based on the Protection Profile (PP) developed by Common Criteria and applies to encrypted personal storage devices used for temporary storage of data whilst the data is in transit between two trusted host computers [164][170]. The test-based sub-model will execute periodically the `_uploadData` operation, and keep a record of instances of test-triggered invocations of operations of TOE in which a response was produced within the required time period, and instances of test-triggered invocations where it was not.

Assumption A5 (testing evidence):

Happens (e(_e1,_CA,_TOE, EXC(Tper), _uploadData(_username,_usersecret,_data, _startTime, _endTime),_TOE), t1, [t1,t1]) ^

Happens (e(_e2,_TOE,_CA, RES, _uploadData(_vusername,_vusersecret,_data, _startTime, _endTime),_TOE), t2, [t1,t2]) ^

```
( t2-t1 <=_maxtime) ^

HoldsAt (TCounterTimelyInc(_TOE,_TCTI),t2)

⇒

Terminates (_e1, TCounterTimelyInc(_TOE,_TCTI),t2) ^

Initiates (_e1, TCounterTimelyInc(_TOE,_TCTI+1),t2)
```

and

```
Assumption A6 (testing evidence):

Happens (e(_e1,_CA,_TOE, EXC(Tper), _uploadData(_username,_usersecret,_data,
_startTime, _endTime),_TOE), t1, [t1,t1])  ^

Happens (e(_e2,_TOE,_CA, RES, _uploadData(_vusername,_vusersecret,_data,
_startTime, _endTime),_TOE), t2, [t1,t2])  ^

( t2-t1 >_maxtime) ^

HoldsAt (TCounterNotTimelyInc(_TOE,_TCNTI),t2)

⇒

Terminates (_e1, TCounterNotTimelyInc(_TOE,_TCNTI),t2) ^

Initiates (_e1, TCounterNotTimelyInc(_TOE,_TCNTI+1),t2)
```

The percentage of timely incidents using testing evidence will be calculated using the following formula:

Percentage of Timely Incidents (Testing) = TCTI/(TCTI +TCNTI)

The hybrid model will include monitoring formulas to record instances of invocation of TOE operators where TOE produced a response within the acceptable time limit and the instances where it did not, and will keep a record of counters of these instances from which an overall availability measure could be drawn.

Assumption A7 (monitoring evidence):

Happens (e(_e1,_sc,_TOE,REQ, _uploadData _username, _usersecret, _verCode1, _data, _startTime, _endTime),_TOE),t1,[t1,t1]) ^

Happens (e(_e2,_TOE,_sc, RES, _uploadData(_vusername, _vusersecret, _verCode2, _data, _startTime, _endTime),_TOE), t2,[t1,t2]) ^

(t2-t1 <=_maxtime) ^

HoldsAt (MCounterTimelyInc(_TOE,_MCTI),t2)

⇒

Terminates (_e1,MCounterTimelyInc(_TOE, _MCTI), t2) ^

Initiates (_e1,MCounterTimelyInc(_TOE, _MCTI+1), t2)

And

Assumption A8 (monitoring evidence):

Happens (e(_e1,_sc,_TOC,REQ, _uploadData(_username, _usersecret, _verCode1, _data, _startTime, _endTime),_TOE),t1,[t1,t1]) ^

Happens (e(_e2,_TOC,_sc, RES, _uploadData(_vusername, _vusersecret, _verCode2, _data, _startTime, _endTime),_TOE), t2,[t1,t2]) ^

(t2-t1 >_maxtime) ^

HoldsAt (MCounterNotTimelyInc(_TOE,_MCNTI),t2)

⇒

Terminates (_e1,MCounterNotTimelyInc(_TOE, _MCNTI), t2) ^

Initiates (_e1,MCounterNotTimelyInc(_TOE, _MCNTI+1), t2)

The percentage of timely incidents using monitoring evidence will be calculated using the following formula:

Percentage of Timely Incidents (Monitoring) = MCTI/(MCTI +MCNTI)

After calculating the percentage of timely incident reports based on monitoring data and the percentage of timely incident reports based on testing data, I will use the two different percentages and I will compare the absolute value of their difference with a predefined performance value for this security property. Additionally, the monitoring period along with the testing period will be compared with the minimum assessment period required. If the following conditions are not satisfied or if the ratio of violations to satisfactions of testing or monitoring data is bigger than a specific percentage, then I will revoke the certificate.

It is worth mentioning that in the event of conflicts, the tool provides the option of retrieving the events that raised the conflicts so that the user can decide if they want to proceed with conflicts resolution when these can be resolved. After the evidence polling, the user can analyse the evidence from monitoring and testing to check which ones are conflicting. The overall percentage of timely incidents can be computed on the basis of both test and monitoring evidence as **Hybrid Percentage** = [Percentage of Timely Incidents (Monitoring) + Percentage of Timely Incidents (Testing)] / 2, so aggregate assessments based on each type of evidence may be validated against an aggregate assessment based on the other type before issuing a certificate. A measure based on testing evidence can be used for issuing a certificate only if the availability measure based on monitoring evidence over the same period is no more than 1% different from it. If violations persist, I will consider extending the assessment period and collecting more testing and monitoring events. Furthermore, I can split the monitoring and testing periods into smaller sub-periods and I can check again the sufficiency conditions of the evidence in order to check if there are time patterns underpinning the deviations.

The above hybrid formulas guarantee incident management quality by enabling the calculation of the percentage of timely incidents for a given target of evaluation. In this example the contribution of the independent-mode hybrid approach to the Protection Profile is of great importance, as the results of testing will confirm and complement the results from monitoring.

3.9 Hybrid, dependent mode model (using EC-Assertion)

The following example shows the use of a hybrid approach in certifying *data integrity-at-rest* using EC-Assertion. As defined in [160], this property expresses the ability to detect and report any alteration of stored data in a target of certification (TOC).

To demonstrate the difference between monitoring and hybrid certification models, I first present the monitoring certification model for data integrity-at-rest, expressed by the EC_Assertion monitoring rule R3 that is listed below. The specification of this rule as well as all models in the paper, assumes the following agents and variables denoting them: service consumers (_sc), target of certification (_TOC), authentication infrastructure (_AI), certification authority (_CA).

Rule R3:

Happens (e(_e1,_sc,_TOC,REQ, _updOp(_cred,_data,_auth),_TOC),t1,[t1,t1]) ^

Happens (e(_e2,_TOC,_AI,RES, _updOp(_cred,_data,_vCode),_TOC),t2,[t1,t1+d1]) ^

(_vCode ≠ Nil)

⇒

Happens (e(_e3,_TOC,_AI,REQ, _notifO(_cred,_data,_auth,_h),_TOC),t3,[t2,t2+d2])

According to R3 when a call of an update operation in a _TOC is detected at some time point t1 (see event `Happens(e(_e1,_sc,_TOC,REQ, _updOp (_cred,_data,_auth), _TOC),t1,[t1,t1])`) and a response to this call occurs after it (see event `Happens(e(_e2,_TOC,_AI,RES, _updOp(_cred,_data,_vCode), _TOC), t2, [t1,t1+d1])`) indicating that the request has been granted (see condition `(_vCode ≠ Nil)` in the rule), the monitor should also check for the existence of another event showing the call of an operation in some authorisation agent _A to notify the receipt and execution of the update request (see `Happens(e(_e3,_TOC,_AI,REQ, _notifO(_cred,_data,_auth,_h),_TOC),t3,[t2,t2+d2])`).

The above model has two limitations in providing assurance for the integrity-at-rest property: (1) it cannot capture updates of data that might have been carried out without using the update interface assumed of _TOC (i.e., `_updOp(_cred,_data,_vCode)`), and (2) it cannot check that the operation `_updOp` has checked authorisation rights before updating data.

A hybrid model could be used in this case to overcome partially the first of these limitations. More specifically, a hybrid model in this case could be based on periodic testing to detect if stored data have been modified and monitor the periods between the tests that revealed data modifications to check if appropriate notifications have also been sent. Data modifications could be detected by obtaining the hash value of the relevant data file in the TOC periodically. Then, if across the execution of two consecutive tests, the last retrieved hash value of the file is different from the previous hash value, a data modification action can be deduced. In parallel with the execution of this periodic test, the hybrid model will also monitor the execution of notification operations. Hence, when a data modification action is detected by two consecutive tests, the hybrid model could also check whether a correlated notification operation has been executed within the period between the tests.

Testing triggers Monitoring

This hybrid model for this instantaneous property that will cover the trustworthiness of outcomes can be expressed using the following monitoring rule and assumption:

Rule R4:

Happens (e(_e1,_CA,_TOC,EXC(T_{per}), _getHash(_TOC,_file,_h1),_CA), t1, [t1,t1]) ^

HoldsAt (LastHash(_file,_h2,t2),t1) ^

(_h1 ≠ _h2)

⇒

Happens (e(_e3,_TOC,_CA,REQ, _notifO(_cred,_data, _auth,_h1),_TOC),t3,[t2,t1])

Assumption A9:

Happens (e(_e1,_CA,_TOC,REQ, _getHash(_TOC,_file,_h1),_TOC),t1,[t1,t1]) ^

HoldsAt (LastHash(_file,_h2,t2),t1) ^

(_h1 ≠ _h2)

⇒

Terminates (_e1,LastHash(_file,_h2,t2),t1) ^

Initiates (_e1,LastHash(_file,_h1,t1),t1)

Rule R4 is “hybrid” as it includes normal monitoring events (i.e., REQ and RES events) and events that trigger the execution of tests (i.e., EXC events). R6 expresses a hybrid dependent mode model where evidence arising from testing triggers the acquisition of monitoring evidence. Hence, testing is the primary form of assessment. In particular, R4 forces the execution of the event `Happens (e (_e1, _CA , _TOC, EXC (`T_{per}`), _getHash (_TOC, _file, _h1), _CA), t1, [t1,t1])` periodically every T_{per} time units to invoke the operation `_getHash` in the testing interface of `_TOC` and obtain the current hash value (`_h1`) of the data file (`_file`) of `_TOC`. If this value is different from the hash value recorded by a previous test at some t2 (i.e., the value recorded in the fluent `LastHash(_file,_h2,t2),t1)`, rule R6 checks if an update notification has also occurred between t2 and t1, as expressed by the monitoring event `Happens(e (_e3, _TOC, _CA,REQ, _notifO (_cred, _data, _auth, _h1), _TOC), t3, [t2,t1])`. The hybrid model uses also a monitoring assumption (i.e., A9). This assumption is used in the model to update the hash value recorded in the fluent `LastHash`, if a test retrieves a hash value that is different from the last recorded one.

Although the above model can capture data updates that have taken place without the invocation of the file updating interface, it cannot guarantee that it can capture all possible updates that might have taken place. In particular, it won't be able to detect if more than one updates have taken place between two consecutive executions of the periodic test. Hence, it addresses the first of the limitations of the monitoring problem (i.e., limitation (1)) only partially.

Monitoring triggers Testing

To address the second limitation of the monitoring model (i.e. it cannot check that the operation `_updOp` has checked authorisation rights before updating data), it is possible to construct a different hybrid model. This model could rely on testing to ensure that every time that an agent that requests a data alteration, it has the authorisation right to do the requested alteration. This model can be expressed by the monitoring rule below:

Rule R5:

```
Happens (e(_e1,_sc,_TOC,REQ, _updOp(_cred,_data, _auth),_TOC),t1,[t1,t1])  ^
Happens (e(_e2,_TOC,_AI,RES,_updOp(_cred,_data,_vCode1),_TOC),t2,[t1,t1+d1]) ^
(_vCode1 ≠ Nil)
```

⇒

Happens (e(_e3,_CA,_AI,EXC,_authorO(_cred,_auth,_vCode2),_TOC),t3,[t2,t2+d2])^

(_vCode2≠Nil)

Rule R5 monitors requests for updates of _TOC data through its normal updating interface. However, for every such request that is granted by _TOC, it requests the execution of a test to check if the entity that requested the update had indeed the authorisation to update data. This is expressed by the EXC event Happens(e (_e3, _CA, _AI, EXC, _authorO (_cred, _auth, _verCode2), _TOC) ,t3, [t2,t2+d2])) and the condition (_verCode2 ≠ Nil). In R3, the monitoring evidence triggers the execution of tests. Hence, the rule expresses a dependent hybrid model where monitoring is the primary form of assessment. Rules R4 and R5 are examples of general time correlation structures that may arise in dependent hybrid certification model and which are shown in Figure 58.

Part (a) of the figure shows dependent hybrid certification models where testing is the dominant form of assessment. In such models, test plans each consisting of a series of tests (i.e., $\{Test_{n1},\ldots,Test_{nL}\}$) are executed according to some periodic schedule. Assuming that the execution of a test plan starts at $t_s^{(n)}$ and ends at $t_e^{(n)}$, the hybrid model may also check for monitoring events that occurred within the interval $[t_s^{(n)}–d_1,\ t_e^{(n)}+d_2]$ in order to provide an assessment of the security property of interest. Note that the length of the execution of each test plan and the monitoring events found within $[t_s^{(n)}–d_1,\ t_e^{(n)}+d_2]$ may vary.

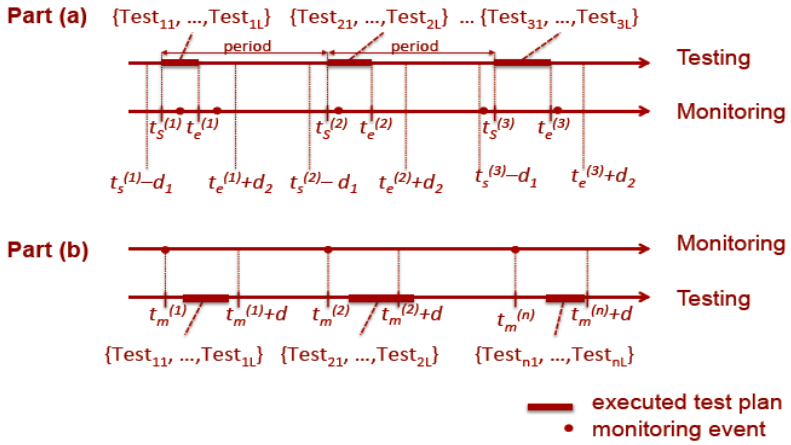

Figure 59 - Dependent mode hybrid certification models

Part (b) of the figure shows the timelines of evidence collection in dependent hybrid certification models where monitoring is the dominant form of assessment. In such models following the collection of monitoring evidence (events), tests plans are executed to cross-check/complete it. The execution of these plans starts within the range [$t_{m(n)}$, $t_{m(n)}$ +d] where $t_{m(n)}$ is the time of occurrence of the last event in a pattern of events that should trigger the execution of the plan and d is a period set by the model. The length of the execution of each test plan may vary.

Examples of Hybrid, dependent mode model cases - Monitoring Triggers Testing

Below, I will present the hybrid dependent-mode certification models of certain security properties from [160] where monitoring triggers testing. The hybrid models that will be used to assess the security properties will confirm the completeness of data collected or the trustworthiness of assessment.

Percentage of timely incident reports

This security property corresponds to *SEF:incident-management-quality:percentage-of-timely-incident-reports* as defined in [160] and describes the percentage of incidents that are reported within a predefined time limit, over the total number of incidents discovered. In my example, I will monitor calls to upload data in the TOC and I will check which of the responses are timely, keeping a counter of the responses. The dependent hybrid model will prove the trustworthiness of outcomes from testing and monitoring. More specifically, testing will complement monitoring as it will check the authorisation rights of the agent who requested the *uploadData* operation. This security property has been also covered above by an independent hybrid certification model, but at this point I will show how it can be assessed with a hybrid dependent-mode model where monitoring is the dominant form of assessment. This security property is based on the Protection Profile (PP) developed by Common Criteria and applies to encrypted personal storage devices used for temporary storage of data whilst the data is in transit between two trusted host computers [164][170].

Rule R6:

```
Happens (e(_e1, _sc, _TOC, REQ, _uploadData (_username, _usersecret, _data,
_auth), _TOC), t1, [t1,t1]) ^
```

Happens (e(_e2, _TOC, _AI, RES, _uploadData (_vusername, _vusersecret, _data, _vCode), _TOC), t2,[t1,t2]) ^

(_vCode ≠ Nil) ^

(t2-t1 <=_maxtime) ^

HoldsAt (MCounterTimelyInc(_TOC,_MCTI),t2)

⇒

Terminates (_e1,MCounterTimelyInc(_TOC, _MCTI), t2) ^

Initiates (_e1,MCounterTimelyInc(_TOE, _MCTI+1), t2) ^

Happens (e(_e3, _AI, _TOC, EXC (Tper), _checkAuthor (_username, _usersecret, _auth, _vCode2), _TOE), t3,[t2,t2+d1]) ^

(_vCode2 ≠ Nil)

and

Rule R7:

Happens (e(_e1, _sc, _TOC, REQ, _uploadData (_username, _usersecret, _data, _auth), _TOC), t1, [t1,t1]) ^

Happens (e(_e2, _TOC, _AI, RES, _uploadData (_vusername, _vusersecret, _data, _vCode), _TOC), t2,[t1,t2]) ^

(_vCode ≠ Nil) ^

(t2-t1 >_maxtime) ^

HoldsAt (MCounterNotTimelyInc(_TOE,_MCNTI),t2)

⇒

Terminates (_e1,MCounterNotTimelyInc(_TOE, _MCNTI), t2) ^

Initiates (_e1,MCounterNotTimelyInc(_TOE, _MCNTI+1), t2) ^

Happens (e(_e3, _AI, _TOC, EXC (Tper), _checkAuthor (_username, _usersecret, _authcode, _vCode2), _TOE), t3,[t2,t2+d1]) ^

(_vCode2 ≠ Nil)

Rule R6 monitors calls to the `_uploadData` operation in the TOC and the responses to them, and if the time required for a response is less that the `maxtime` and the fluent that keeps a counter of the responses that were produced on time holds, then it updates the counter of instances where the responses were timely and executes periodic testing to check that the agent who quested the data upload has the authorisation rights to do the operation. The second rule (Rule R7) keeps an overall counter of the non-timely incidents and executes periodic testing to check if the login operation has checked authorisation rights. The percentage of timely incidents using monitoring evidence will be calculated using the following formula:

Percentage of Timely Incidents = MCTI/(MCTI +MCNTI)

The above hybrid formulas guarantee incident management quality by allowing the calculation of the hybrid percentage of the timely incidents for a given target of evaluation. In this example the contribution of the dependent-mode hybrid approach to the Protection Profile is of great importance, as the hybrid rules enable the authorisation of the agent who requested the `_uploadData` operation, providing a higher level of security.

Authentication of data origin

This security property corresponds to the *AIS:authentication:authentication-of-data-origin* security property as defined in [160] and describes a data authentication mechanism which assures that data produced by the TOE (Target of Evaluation) can be verified to originate from the TOE. The dependent hybrid model will prove the completeness of data from testing and monitoring. The property will be considered "verified" if the data produced by the TOC is associated with an Authentication Code `(_authcode)`. If (`(_authcode)`≠ Nil) I can assume that data originates from the TOE. The TOE will be the "upgrade Web Interface". The following rule R9, for every request to upgrade the TOE data (request granted by TOE), through the normal upgrade interface, it requests the execution of a test to check if the entity that requested the update had indeed the authorisation to upgrade data and that the data that originate from the TOE is associated with an `_authcode` whose value is not Nil. This security property is based on the Protection Profile (PP) developed by Common Criteria and applies to encrypted personal storage devices used for temporary storage of data whilst the data is in transit between two trusted host computers [164][170]. The hybrid formulas will certify the authentication of the TOE by using both testing and monitoring evidence. More specifically, evidence gathered from testing will complete the monitoring evidence gathered.

Rule R9:

```
Happens (e(_e1,_sc,_TOE,REQ, _upgradeOp(_username, _usersecret, _data,
vCode),_TOE),t1,[t1,t1])  ^

Happens (e(_e2,_TOE,_sc, RES, _upgradeOp(_vusername, _vusersecret, _data,
_authcode),_TOE),t2,[t1,t1+d1]) ^

 (_authcode ≠ Nil)

⇒

Happens (e(_e3,_CA,_AI,EXC, _authenticateOp(_username,_usersecret, _vCode,
_authcode2),_TOE),t3,[t2,t2+d2])^

(_authcode2≠Nil)
```

In this dependent hybrid certification model, monitoring is the dominant form of assessment and the test plans are executed to check the authorisation rights of the entity who requested the upgrade. The execution of these plans starts within the range [t2,t2+d2] where t2 is the time of occurrence of the last event in a pattern of events that should trigger the execution of the plan and d2 is a period set by the model.

Example of Hybrid, dependent mode model cases - Testing Triggers Monitoring

Below, I will present the hybrid dependent-mode certification model for the *Storage freshness* security property from [160]. In this model testing is the primary form of assessment and triggers monitoring.

Storage freshness

This security property corresponds to the *DSI:durability:storage-freshness* security property as defined in [160] and assures that any attempt to retrieve a data object from storage in the TOC is guaranteed to return the updated version of the data object, reporting any alteration of the data object. The dependent hybrid model will prove the trustworthiness of data. First of all, I will execute the update operation in the interface of the TOC and if a response is produced within a required time period, then the counter of instances where the data in the TOC were altered, is updated. Additionally, the fluent concerning the test executions that revealed the data alteration in the TOC is recorded.

Assumption A10:

Happens (e(_e1,_CA,_TOC, EXC(Tper), _update(_TOC, _object, _h1) , _TOC), t1, [t1,t1]) ^

Happens (e(_e2,_TOC, _CA RES, _update(_TOC,_object, _h2),_TOC),t2,[t1,t1+tav])^

HoldsAt (TCounterUpdate(_TOC,_TCU),t2)

⇒

Terminates (_e1,TCounterUpdate(_TOC,_TCU),t2) ^

Initiates (_e1,TCounterUpdqate(_TOC,_TCU+1),t2)^

Initiates (_e1,TUpdate(_TOC,_x,t2-t1),t2)

Also, with testing I will detect if data stored in the TOC have been modified and I will monitor the periods between the tests that revealed data modifications to check if appropriate notifications have also been sent. Data modifications could be detected by obtaining the hash value of the relevant data file in the TOC periodically. Then, if across the execution of two consecutive tests, the last retrieved hash value of the file is different from the previous hash value, a data modification action can be deduced. In parallel with the execution of this periodic test, the hybrid model will also monitor the execution of notification operations.

Rule R10:

Happens (e(_e1,_CA,_TOC, EXC(Tper), _update(_TOC,_object,_h1),_CA), t1, [t1,t1]) ^

HoldsAt (LastHash(_object,_h2,t2),t1) ^

(_h1 ≠ _h2) ^

(t2 <= t1)

⇒

Happens (e(_e3,_TOC,_CA,REQ, _notifUpdate(_cred,_data,_auth,_h1), _TOC), t3, [t2,t1])

Rule R10 expresses a hybrid dependent mode model where evidence arising from testing triggers the acquisition of monitoring evidence. Hence, testing is the primary form of assessment. R10 forces the execution of the event `Happens(e(_e1,_CA,_TOC,EXC(Tper), _update(_TOC,_object,_h1),_CA), t1, [t1,t1])` periodically to invoke the operation `_update(_TOC,_object,_h1)` in the testing interface of _TOC and obtain the current hash value `(_h1)` of the data object of the _TOC. If this value is different from the hash value recorded by a previous test at some t2 (i.e., the value recorded in the `fluent LastHash(_object,_h2,t2),t1)`, rule R11 checks if an update notification has also occurred between t2 and t1, as expressed by the monitoring event `Happens(e(_e3,_TOC,_CA,REQ, _notifUpdate(_cred,_data,_auth,_h1), _TOC), t3, [t2,t1]).`

4. Evaluation

4.1 Research methods and evaluation

The objective of this chapter is to evaluate critically the hybrid approach that I have introduced and to provide the reader with conclusions about the hybrid certification. The qualitative evaluation of my approach is based on comparing my hybrid model against other approaches that have extended industrial applicability, in terms of certain criteria that I am defining. The biggest strength of this method is the fact that my approach is compared against other popular approaches, so the advantages and the limitations of my hybrid solution are easily identified and revealed. However, this method is limited to a finite number of comparisons and, thus, entails a degree of subjectivity.

4.2 Overview and evaluation assessment

At this point, I will provide an evaluation of the hybrid certification approach that has been introduced. For the purposes of the evaluation, I will firstly present the hybrid certification process, along with the certification processes of other approaches and, then, I will proceed with comparing the hybrid approach with the Common Criteria approach, which is based on third-party certification. Common Criteria (CC) is recognised as one of the most popular and distinguished approaches for third party certification, representing a good real-world example of third party certification approaches. Also, I will compare my approach with the ISO 27001 approach, which is based on third-party audit certification and with the CSA Open Certification Framework, which is based on self-attestation and third-party audit certification. The comparison that follows is based on the certification process of each of the previous approaches. Finally, I will outline how my approach for certifying cloud services proves to be more effective than the aforementioned certification processes.

4.2.1 Hybrid Certification Approach overview

First of all, concerning my hybrid certification approach, it is worth mentioning that the entities involved in the certification process include a cloud service provider who provides the cloud service to be certified and request its certification for a specific security property, a certification authority who issues and manages the certificate for the given security property and cloud

service, an issuer of the certification model instance who issues and manages the certification model instance for the defined certification process, and the executor of the instance who is responsible for collecting the evidence. The certification model instance executor and the issuer are covered by an accredited Lab. The inputs of the hybrid certification process include the cloud service that is requested to be certified and a certification model instance that includes information about the TOC, the security property, the conditions for the collection of evidence and for the lifecycle of the certificate, while the output of the hybrid certification process is the hybrid certificate.

4.2.2 Comparison between the hybrid certification approach and Common Criteria

Concerning the Common Criteria (CC) approach, it has to be mentioned that the certification process is applied to a product and generates certificates that are recognised within the Common Criteria Recognition Arrangement. After the product application, the CC approach checks that the product implements all the security measures claimed to the level claimed, using penetration and functional testing. The entities involved in the certification process include a developer who produces the object to be certified, the evaluation laboratory who performs the evaluation activities to the entities provided by the developer and the certification body who issues the certificates. The inputs of the CC certification process include the product that needs to be certified (TOE or Target of Evaluation), the security target which is a document uniquely associated to the TOE containing information about the TOE, its functionalities and the evaluation activities required, and, finally, additional inputs that involve further documentation (development documentation, guidance documentation, testing documentation). The output of the CC process is a CC certificate and a certification report that is associated to the CC certificate.

The hybrid certification approach is suitable for certifying services in cloud environments, as it provides machine-readable certificates and focuses on automated activities. On the contrary, the CC approach is not suitable for certifying services in the cloud, as it is based on costly manual actions and activities, such as analysing documentation provided by the developer, thus having a human-centric approach. The CC aims at certifying that the product implements a specific set of technical security measures. Additionally, the hybrid certification model introduced is based on collecting evidence from the service operational environment through testing and monitoring,

thus, covering cases where one form of assessment does not provide the sufficient evidence required, and also, cases where I just want to confirm the results from one form of assessment with the results from the other form of assessment, while the CC approach is based on collecting evidence through functional and penetration testing of the interfaces of the object in a controlled testing environment.

My hybrid independent approach supports continuous monitoring, which means that evidence is continuously collected through the service lifecycle, providing a continuous assurance, while CC approach does not consider new potential vulnerabilities, as testing is performed at regular intervals and assurance is restored at fixed moments specified by the certification body. For that reason, the lifecycle of the certificate in the hybrid certification model is more complex, but proves to be more dynamic, addressing more cases than the ones that can be addressed by the CC approach. Finally, my hybrid approach supports the provision of digital signatures, enabling the dynamic and automatic processing of the certificates.

Additionally, the hybrid model enables the certification of security properties from the Protection Profiles of the CC that could not be certified without it (such as the authorisation of agents who request update of data), as evidence that comes from both testing and monitoring is combined. In the previous sections, it became obvious how the hybrid formulas can be used to certify the authentication and the incident management quality in an Encrypted Storage Device. The CC evaluation process is abstract and the approach is not designed to support automated security certification, targeting static, monolithic systems and requiring a large investment of resources and time. Consequently, the hybrid model can extend the number of security properties that can be certified, and the number of cases that are covered by the Common Criteria approach.

Summarising, the hybrid model introduced can produce availability assessments of higher confidence as the monitoring and testing evidence can be cross-checked before being used in an assessment (and certificate). Also, hybrid models offer an extended pool of evidence and are more customisable than traditional certification models, since they offer the choice of deciding how test and monitoring evidence should be correlated, cross-checked and used in assessments. Finally, in terms of efficiency, effectiveness and cost, it is obvious that the hybrid approach I introduced is a better solution concerning the certification of cloud services.

4.2.3 Comparison between the hybrid certification approach and ISO 27001

The ISO 27001 approach is a third-party audit certification approach and includes technical, physical and procedural security measures, providing a wide variety of control and implementation of these controls. Concerning the certification process, the ISO 27001 approach includes an auditing process where an auditor checks if all security measures in an organisation are implemented correctly. The entities involved in the certification process include the organisation who requests to be certified and is responsible for providing the inputs of the certification process, the certified auditor who is accredited by the Certification Body (CB) and performs the auditing and the CB that certifies the organisation as compliant with ISO 27001 and generates the output of the certification process. The final outcome of the ISO 27001 approach is either a certification or a failure. Additionally, the auditing process takes place twice, as firstly, a pre-certification auditing is performed and then, the organisation acts on the recommendations and the final auditing is performed. The ISO 27001 certification lasts for three years, before the renewal process begins.

In the ISO 27001 approach the target of certification focuses on the Information Security Management System of an organisation and not on a specific cloud service as my hybrid approach does. That means that the ISO 27001 approach audits all the procedures for the services and does not perform actions directly on cloud services and their underlying cloud stack, as my hybrid approach does. Moreover, the ISO 27001 approach is focused on analysing reports and documentation provided by the service provider, has a human-centric nature and cannot be automated easily, as opposed to the hybrid certification approach I introduced that produces machine-readable certificates and supports automation. Concerning the lifecycle of the certificate, both my hybrid approach and ISO 2007 are capable of covering occurring changes without requiring re-certification from scratch. Finally, in terms of efficiency, effectiveness and cost, it is obvious that the hybrid approach I introduced is a better solution concerning the certification of cloud services, since my approach consists of an automatic approach without requiring a certified auditor to certify the whole certification process.

4.2.4 Comparison between the hybrid certification approach and CSA's OCF

The Open Certification Framework (OCF) developed by CSA is based on self-attestation and third-party audit certification, aiming to certify the use of the best practice information security management for the cloud. The OCF focuses on the organisational dimension of security and translates it to technical measures. The OCF includes the Cloud Control matrix (CCM), which comprises of a list of 133 controls that are grouped in 16 domains. Additionally, it allows cloud providers to self-assess their adherence to the CCM in two ways. The first way is directly, and the second one is through the Consensus Assessment Initiative Questionnaire. The OCF supports real-time monitoring, by collecting events and other auditing evince, but this functionality is under development.

While my hybrid approach supports the automatic certification of security properties of the cloud services providing machine-readable certificates, the OCF certifies information systems as whole entities and includes risk, governance and compliance processes, usually requiring human auditing intervention. The OCF can certify qualitative controls that the hybrid approach introduced cannot. One example is the certification of the control concerning the compliance obligations of an organisation to maintain and update regularly the infrastructure network and system components based on the business needs. However, it cannot certify security properties that require testing, such as the provision of digital signatures. Concerning the continuous monitoring supported by the OCF, it is difficult to evaluate it against my hybrid approach, as this component is still under development. Finally, in terms of efficiency, effectiveness and cost, it is obvious that the hybrid approach I introduced is a better solution concerning the certification of cloud services, as it dos not require the intervention of an external certified auditor.

4.2.5 Evaluation Summary

The following table summarises the findings from the qualitative evaluation, of the comparison of the certification process between my hybrid approach and the CC approach, the ISO 27001 approach and the OCF approach. The evaluation criteria used for the comparison are the nature of the approach, the focus of the approach, the validity of the certification outcome, the automation of the certification, the need for human auditing intervention, the ability to cover potential changes without having to recertify from scratch and the cost efficiency.

Criteria	Hybrid Approach	Common Criteria	ISO 27001	OCF
Nature of the approach	Machine-centric approach	Human-centric approach	Human-centric approach	Human-centric approach
Focus of the approach	Cloud Services	Software products	Organisations	Information Systems
Validity of certificate	Dynamic validity	Assurance is restored at fixed moments	Needs to be renewed every 3 years	Not known (as not fully developed yet)
Automation of certification	Enabled	Not enabled	Not enabled	Not enabled
Need for human auditing intervention	Not Applicable	Applicable	Applicable	Applicable
Ability to cover potential changes	Enabled	Not enabled	Enabled	Will be enabled
Cost efficiency	Does not require a large investment of time and money	Requires a sufficient investment of time and money	Requires a sufficient investment of time and money	Not known (as not fully developed yet)

Table 7 - Evaluation Summary Table

Concerning the nature of each certification approach, it needs to be said that my hybrid approach is the only certification approach of the ones analysed and examined, that provides machine-readable certificates and has a machine-centric approach. This means that it does not rely on human evaluation, auditing and analysis of documents and reports. The CC approach, along with the ISO 27001 approach and the OCF approach have a human-centric nature and provide certificates that are not machine-readable. Additionally, the hybrid approach is the only approach that is automated. This is a strong advantage of the hybrid approach that I developed, as it can have broader applicability. At this point, it is worth mentioning that the hybrid approach for the certification of cloud services does not require human auditing intervention, as opposed to the other certification approaches, and, as a result, it is free of any subjectivity that could be introduced with human intervention. Concerning the ability to cover potential changes that may occur, the hybrid certification approach, along with the ISO 27001 approach, do not require re-

certification from scratch, as opposed to the Common Criteria approach that is unable to cover changes without repeating the certification process again. The OCF approach enables the ability to cover potential changes, but it has not been fully implemented yet. In terms of the validity of the certificates produced, the hybrid approach is the most dynamic approach of the ones analysed, because of its ability to be automated and to cover potential changes without the need of human intervention. The assurance in the CC approach is restored at specific points of time, making the approach quite flexible and, thus, satisfactory, in terms of validity of the certificates. The duration of the OCF approach is not known yet, as it is under development. Finally, when it comes to cost efficiency, the hybrid approach is the only approach that does not require a large investment of money and time and does not require any external auditing entities.

At this point, I will present the major drawbacks of the certification approaches analysed above. First of all, the CC approach is not suitable for certifying services in the cloud and uses only testing to assess and certify security properties, providing abstract certificates. Moreover, as it was demonstrated earlier, it fails to certify the percentage of timely incidents reports security property and the authentication of data origin, failing to certify the authentication and the incident management quality in an Encrypted Storage Device and, thus, proving to be quite limited. The ISO 27001 approach is focused on the Information Security Management System of an organisation and not on a specific cloud service and requires pre-certification. The OCF approach is not fully developed and certifies information systems as whole entities focusing on risk, governance and compliance processes. Additionally, the above approaches are not fully customisable, as opposed to my hybrid approach for certification, where the user can select the elements that will consist the certification model schema for realising the security properties. For these reasons, it can be concluded that the hybrid certification approach is the most ideal approach for cloud service certification.

4.3 Conclusions and Future work

Hybrid certification combines evidence gathered through testing and monitoring to collect the elements that are required to certify a security property. The hybrid approach that I introduced, overcomes the limitations of each on of the individual certification processes (i.e. testing or monitoring in isolation). Continuous monitoring allows the non-stop observation of the system without interfering with it. However, monitoring is insufficient in certain cases, as it can make the verification infeasible. More specifically, monitoring requires testing agents to verify signatures and inject traffic. On the other hand, testing can be powerful in pre-production environments, but on live systems it can interfere with business operations. According to the hybrid approach for certifying services in the cloud, monitoring and testing can take place in parallel, so that testing can complement monitoring and vice-versa. Monitoring can force the execution of testing in order receive the evidence required for assessing security properties, and, testing can force the collection of monitoring evidence to compete the assessment. This is particularly useful when we want to certify security properties, such as *data-integrity-at-rest.* Additionally, monitoring and testing can take place independently from each other, and the evidence from each form of assessment can help to verify the evidence collected from the other form of assessment. This is obvious when assessing security properties, such as *availability.* In this way, hybrid security certification provides high confidence in certifying security properties, compared to certification schemes based solely on testing or monitoring. The testing and monitoring certification model schemas developed for the CUMULUS project include certain limitations. More specifically, the test-based certification model schema does not include an element for specifying explicitly the interfaces that are provided or required for the certification of the TOC. Also, the test-based certification model schema does not include an element concerning the expiration period of the evidence collected for the assessment of security properties and does not define a state transition model that can drive the mechanisms behind the lifecycle element. On the other hand, the monitoring-based certification model schema defines different states than the ones defined in the lifecycle element in the test-based certification model schema, and, uses a different way of defining the TOC and the security properties to be certified. That means, that there were a lot of inconsistencies and incompatibilities between these two certification model schemas.

To overcome the above problems and limitations, a new certification model had to be defined that can be applied when certifying security properties for cloud services and evidence are

collected through monitoring and testing. So, elements from both certification model schemas were used in order to have a universal way of certifying security properties when monitoring and testing are performed. Moreover, the hybrid certification model schema was developed taking into consideration the gaps of the previous schemas and their incapability of certifying certain security properties. As a result, I came up with the idea of a hybrid concept that can lead to the certification of a bigger number of security properties and that can provide higher security. My hybrid approach was compared with popular third-party certification and auditing approaches. After a comparison with the CC approach, the Open Certification Framework and the ISO 27001 approach, it is clearly obvious that the hybrid approach introduced is the most cost-effective, automated and dynamic cloud certification approach, being able, at the same time, to cover potential changes and provide dynamically valid certificates. Moreover, it can certify security properties that the other approaches fail, such as the authentication and the incident management quality in an Encrypted Storage Device, offering an extensive pool of evidence.

The hybrid approach I introduced, successfully meets the aims and the objectives of my research, as it enables the automated and continuous certification of security properties of cloud services with high level of assurance, overcoming the limitations of assessments based on testing and monitoring, and, reduces the uncertainty arising from reliance on any of the testing and monitoring evidence in isolation. The certification of security properties is based on the dynamic collection of monitoring and testing evidence. Monitoring is continuous, so that it can cover changes at any layer of the cloud that might affect properties already certified, without requiring recertification from scratch. Also, a schema has been defined to support the definition of executable certification models. This hybrid schema can be used to drive the certification process and generate the certificates. Additionally, the certification infrastructure required for hybrid certification is presented in my work. For my research, I developed the tools and mechanisms that support the management of certificates and the analysis of testing and monitoring data that are gathered and are used as evidence in order to generate and issue the hybrid certificates, based on hybrid certification models. The examples of hybrid certification of security properties provided, demonstrate how the hybrid approach certifies security properties that other existing certification models fail to do so.

The evaluation of my approach proves that the hybrid approach presented can confirm the reliability, security and potential applicability of the proposed model to the industry. The proposed model provides advanced security in cloud service certification and not

trustworthiness, as trustworthiness is a generic concept that includes a high degree of subjectivity, as already described in Chapter 2.

The biggest strength of my work is that a hybrid certification model was formed to cover the gaps arising from testing and monitoring certification. My approach bridges the gaps arising from the use of the CUMULUS framework and provides an advanced framework to overcome the existing limitations. Additionally, the examples I provided show a potential industrial use and applicability of the hybrid certification approach. The biggest limitation of the hybrid approach is the fact that it was mostly focused on addressing the needs of the CUMULUS EU FP7 Project. Additionally, the certificate authority needs to specify how to correlate the retrieved evidence on the basis of a hybrid lifecycle, where the certificate status changes depending on conditions triggered by combination of evidence of different types, making the hybrid approach complex and less flexible. Another limitation of the hybrid approach is the fact that it cannot certify qualitative controls. One example is the certification of the control concerning the compliance obligations of an organisation to maintain and update regularly the infrastructure network and system components based on the business needs.

Concerning the future work of my research, I have identified certain areas of investigation for future improvements. First of all, it will be helpful for the hybrid approach to be extended so that it can support multi-layer certification and can certify cloud services that reside in more than one layers of the cloud stack. As a result, in that way, it can become more applicable in the industry. In addition, the hybrid certification model defined, could potentially become simpler in order to enable higher flexibility and reusability, so that the certification authority does not have to specify how the retrieved evidence will be correlated on the basis of a hybrid lifecycle.

4.4 Publication of my Research

A part of my research on Hybrid Certification Models has been presented in a conference paper under the title "Towards Hybrid Cloud Service Certification Models" at the 11th IEEE International Conference on Services Computing (IEEE SCC 2014) in Anchorage, Alaska [38]. This paper has been published at the CPS Online system for SCC2014 and at the IEEE digital library. This publication includes an early version of the hybrid approach for certifying security properties of cloud services that combines monitoring and testing data and argues about the need for hybrid certification, examining some basic characteristics of hybrid certification models.

References

[1] G. Spanoudakis, E. Damiani, A. Mana, “Certifying Services in Cloud: The Case for a Hybrid, Incremental and Multi-layer Approach,” IEEE 14th Int. Symp. On High-Assurance Systems Engineering, 2012.

[2] F.Etro, “The Economic Impact of Cloud Computing on Business Creation, Employment and Output in Europe”, Review of Business and Economics, 2009.

[3] Industry Recommendations to Vice President Neelie Kroes on the orientation of a European Cloud Computing Startegy, November 2011

[4] S. Ibrahim, B. He, H. Jin, “Towards Pay-As-You-Consume Cloud Computing”, 2011 IEEE International Conference on Services Computing

[5] A. Bisong, S. M. Rahman, “An Ovewrview of the security concerns in Enterprise Cloud Computing”

[6] S. Subashini, V.Kavitha, “A survey on security issues in service delivery models of cloud computing”, 2011

[7] R. Smith Computing in the cloud. “Research Technology Management”, ABI/INFORM Global., 2010

[8] M. Al Morsy, J. Grundy, I. Mülle, “An Analysis of The Cloud Computing Security Problem”, 2010

[9] D. Catteddu, G. Hogben, “Cloud Computing: Benefits, Risks and Recommendations for Information Security,” European Network and Information Security Agency (ENISA), 2009

[10] P. Meller, T. Grance “The NIST Definition of Cloud Computing”, 2011

[11] U. Somani, K. Lakhani, M. Mundra, “Implementing Digital Signature with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing”. 2010

[12] T. Ristenpart, E. Tromer, H. Shacham, S. Savage, “Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds”, 2009

[13] IDC, http://www.idc.com/getdoc.jsp?containerId=prUS24977214

[14] Gartner, http://www.gartner.com/newsroom/id/2352816

[15] D. S. Herrmann, Using the Common Criteria for IT Security Evaluation, CRC Press, Inc., Boca Raton, FL, USA, 2002

[16] I. Windhorst, A. Sunyaev, “Dynamic Certification of Cloud Services”, Eighth International Conference on Availability, Reliability and Security (ARES) 2013, pp.412,417, 2-6 Sept. 2013

[17] B. Jr. Kaliski and W. Pauley, Toward Risk Assessment as a Service in Cloud Environments. Proceeding HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010

[18] S. Ramgovind, M. Eloff, E. Smith, “The Management of Security in Cloud Computing”, 2010

[19] P. Mell, T. Grance, “The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technology”. Special Publication 800-145, September 2011.

[20] T. Singleton, IT-Audits of Cloud and SaaS. ISACA Journal(Volume 3), 2010.

[21] Wikibon, Audit and the cloud. Wikibon Blog, 26.02.2010.

[22] D. Goodin, “Web Host Hack Wipes Out Data for 100,000 Sites.” http://www.theregister.co.uk/2009/06/08/webhost_attack/.

[23] J. Brodkin, “Megaupload data in Europe wiped out by hosting company.” http://arstechnica.com/tech-policy/2013/06/kim-dotcom-megaupload-data-in-europe-wiped-out-by-hosting-company/.

[24] S. Cimato, E.Damiani, F.Zavatarelli, R.Menicocci, "Towards the Certification of Cloud Services," Services (SERVICES), 2013 IEEE Ninth World Congress on , vol., no., pp.92,97, June 28 2013-July 3 2013

[25] NIST, Guide for Applying the Risk Management Framework to Federal Information Systems: A Security Life Cycle Approach, (NIST Special Publication 800-37), Feb. 2010

[26] M. Krotsiani, G.Spanoudakis, K. Mahbub, “Incremental Certification of Cloud Services”. In SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies (pp. 72-80), August 2013

[27] J. Reavis, D.Catteddu, “Open Certification Framework. Vision Statement. Cloud Security Alliance,” August 2012.

[28] B. R. Kandukuri, R. Paturi, A., “Cloud Security Issues”, 2010

[29] A. Haeberlen. “A case for the accountable cloud.”, SIGOPS Oper. Syst. Rev. 44(2): 52-57, April 2010.

[30] L. Kaufman. “Data Security in the World of Cloud Computing,” IEEE Security and Privacy 7(4): 61-64, July 2009.

[31] M. Jensen, J. Schwenk, N. Gruschka, L.L. Iacono, "On Technical Security Issues in Cloud Computing," Cloud Computing, 2009. CLOUD '09. IEEE International Conference on , vol., no., pp.109,116, 21-25 Sept. 2009

[32] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing v2.1," available from: http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf

[33] H. Foster, G. Spanoudakis, K. Mahbub, "Formal Certification and Compliance for Runtime Service Environments", Proc. of 9th IEEE Int. Conf. on Service Computing, 2012

[34] J.C. Pazzaglia, et al., Advanced Security Service cERTificate for SOA: Certified Services go Digital!, Proc. of Information Security Solutions for Europe, 2011

[35] CESG - UK IT Security Evaluation and Certification body, "CC - Common Criteria Certification in the UK - UK IT security evaluation & certification scheme"

[36] M. Montenegro, A. Mana. "Improving Interoperability of Digital Certificates for Software & Services". in Proc. of the IEEE 2013 International Workshop on Services Discovery and Composition (SDC 2013). 2013.

[37] G. Spanoudakis, C.Kloukinas, K. Mahbub, "The SERENITY Runtime Monitoring Framework", in Spyros Kokolakis; Antonio Maña Gómez & George Spanoudakis, ed., 'Security and Dependability for Ambient Intelligence' , Springer, , pp. 213-237, 2009 .

[38] S. Katopodis, G. Spanoudakis, K. Mahbub, "Towards Hybrid Cloud Service Certification Models," Services Computing (SCC), 2014 IEEE International Conference on , vol., no., pp.394,399, June 27 2014-July 2 2014

[39] Anisetti M.; Ardagna C.A.;.Damiani A.; Saonara A.,"A test-based security certification scheme for web services." ACM Trans. Web 7, 2, Article 5, May 2013.

[40] K. Dempsey et al., Information Security Continuous Monitoring (ISCM) for Federal Systems and Organisations, NIST 800-137, 2011, available from: http://csrc.nist.gov/publications/nistpubs/800-137/SP800-137-Final.pdf

[41] https://resilience.enisa.europa.eu/cloud-computing-certification

[42] S. Sakr, A. Liu, "SLA-Based and Consumer-centric Dynamic Provisioning for Cloud Databases," Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on , vol., no., pp.360,367, 24-29 June 2012

[43] CUMULUS consortium, "cumulus Framework Architecture v1," Deliverable D5.1.

[44] M. Anisetti, C.A. Ardagna, and E. Damiani. "Fine-grained modeling of web services for test-based security certification". In Proc.of the 8th International Conference on Service Computing (SCC 2011), Washington, DC, USA, July 2011.

[45] E. Damiani, and A. Mana, "Toward WS-Certificate", In Proc. of the ACM Workshop on Secure Web Services (SWS 2009), 2009

[46] E. Denney and B. Fischer, "Software Certification and Software Certificate Management Systems", NASA, USA, 2005.

[47] D.N. Christodoulakis, C. Tsalidis, C.J.M. van Gogh, and V.W. Stinesen, "Towards an automated tool for software certification", IEEE Software, 1989.

[48] A. Alhussein, H. Zedan; H. Janicke; O. Alshathry, "Software Certification through Quality Profiling," New Trends in Information and Service Science, 2009. NISS '09. International Conference on , vol., no., pp.444,446, June 30 2009-July 2 2009

[49] A. Munoz, A. Mana, "Bridging the GAP between Software Certification and Trusted Computing for Securing Cloud Computing," Services (SERVICES), 2013 IEEE Ninth World Congress on , vol., no., pp.103,110, June 28 2013-July 3 2013

[50] E. Damiani, C.A. Ardagna, and N. El Ioini, "Open Source Security Certification". Springer, December 2008.

[51] C. Jahl. "The information technology security evaluation criteria". In Proccedings of the 13th International Conference on Software Engineering, Austin, TX, USA, May 1991, pp. 306 – 312

[52] D.S. Herrmann. Using the Common Criteria for IT security evaluation. Auerbach Publications, 2002.

[53] K. Beckers; D. Hatebur, M. Heisel, "A Problem-Based Threat Analysis in Compliance with Common Criteria," Availability, Reliability and Security (ARES), 2013 Eighth International Conference on , vol., no., pp.111,120, 2-6 Sept. 2013

[54] M. Razzazi1, M. Jafari, S. Moradi2, H.Sharifipanah, M.Damanafshan, K. Fayazbakhsh2, A. Nickabadi2, "Common Criteria Security Evaluation: A Time and Cost Effective Approach", 2006

[55] S.B. Seidman, "Software Engineering Certification Schemes," Computer , vol.41, no.5, pp.87,89, May 2008

[56] G.S. Robinson, "ANSI's role in standards development," Micro, IEEE , vol.17, no.6, pp.84,85, Nov/Dec 1997

[57] http://www.iso.org/iso/about/iso_members/iso_member_body.htm?member_id=1511

[58] https://www.cen.eu/about/Pages/default.aspx

[59] http://www.iso.org/iso/home/about.htm

[60] http://www.iec.ch/about/?ref=menu

[61] http://www.itu.int/en/Pages/default.aspx

[62] M. Anisetti, C. A. Ardagna and E. Damiani. "A Low-Cost Security Certification Scheme for Evolving Services". in Proc. of the 19th IEEE International Conference on Web Services (ICWS 2012). 2012.

[63] “A security certification scheme for SOA and web services,” Note del Polo - Ricerca 135, Universit`a degli Studi di Milano, Polo didattico e di ricerca di Crema, Tech. Rep., January 2012, http://www.crema.unimi.it/Biblioteca/Note pdf/163.pdf.

[64] “Securing Web services for army SOA,” http://www.sei.cmu.edu/solutions/softwaredev/securingweb-services.cfm.

[65] M. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui, “A QoS broker based architecture for efficient web services selection,” in Proc. of ICWS 2005, Orlando, FL, USA, July 2005.

[66] S. Ryu, F. Casati, H. Skogsrud, B. Betanallah, and R. Saint- Paul, “Supporting the dynamic evolution of Web service protocols in service-oriented architectures,” ACM Transactions on the Web, vol. 2, no. 2, April 2008.

[67] M. Papazoglou, V. Andrikopoulos, and S. Benbernou, “Managing evolving services,” IEEE Software, vol. 28, no. 3, pp. 49–55, May-June 2011.

[68] N. Parimala, A. Saini, "Web service with criteria: Extending WSDL," Digital Information Management (ICDIM), 2011 Sixth International Conference on , vol., no., pp.205,210, 26-28 Sept. 2011

[69] H. Roth, J. Schiefer, A. Schatten, "Probing and Monitoring of WSBPEL Processes with Web Services," E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on , vol., no., pp.30,30, 26-29 June 2006

[70] H. Foster, G. Spanoudakis, “Taming the cloud: Safety, certification and compliance for software services” - Keynote at the Workshop on Engineering Service-Oriented Applications (WESOA) 2011. In: Lecture Notes in Computer Science. (pp. 3-8). Springer. ISBN 9783642318757, 2011

[71] SERENITY Project. Serenity, system engineering for security & dependability. www.serenity–project.org, 2006.

[72] A. Alvaro, E.S. de Almeida, S.L. Meira, "Component Quality Assurance: Towards a Software Component Certification Process," Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on , vol., no., pp.134,139, 13-15 Aug. 2007

[73] V. Basili, J. Heidrich, M. Lindvall, J. Munch, M. Regardie, A. Trendowicz, "GQM^+ Strategies -- Aligning Business Strategies with Software Measurement," Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on , vol., no., pp.488,490, 20-21 Sept. 2007

[74] A. Sunyaev, S. Schneider, : Cloud Services Certification. In: Communications of the ACM (CACM), Volume 56, Number 2, pp. 33-36, 2013

[75] S.P. Kaluvuri, H. Koshutanski, F.D. Cerbo, A.Mana, “Security assurance of services through digital security certificates”, 20th IEEE International Conference on Web Services (ICWS-2013), 2013

[76] M. Anisetti, C.A. Ardagna, F. Guida, “ASSERT4SOA: Toward security certification of service-oriented applications”, OTM Workshops, 2010

[77] http://www.avantssar.eu/

[78] http://www.avispa-project.org/

[79] G. Canfora, M. Di Penta,”Service-oriented architectures testing: A survey”, Softw. Engin Int.Summer Schools 1, 78–105, 2009

[80] L. Baresi and E. Di Nitto. Test and Analysis of Web Services. Springer, New York, USA, 2007

[81] R. Chandramouli, M. Blackburn, “Automated testing of security functions using a combined model and interface-driven approach”, In Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS’04), 2004.

[82] J. Jurjens, “Model-based security testing using UMLsec: A case study”, Electron. Not. Theor. Comput.Sci. 220, 1, 93–104, 2008

[83] M. Zulkernine, M.F. Raihan, M.G.Uddin, “Towards model-based automatic testing of attack scenarios”, In Proceedings of the 28th International Conference on Computer Safety, Reliability and Security (SAFECOMP’09, 2009).

[84] G. Canfora, M. Di Penta, "Testing services and service-centric systems: challenges and opportunities," IT Professional , vol.8, no.2, pp.10,17, March-April 2006

[85] W.T. Tsai, R.Paul, W. Yamin, F.Chun, W. Domg, “Extending WSDL to facilitate web services testing”, In Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering(HASE’02), 2002

[86] E. Martin, S. Basu, T. Xie, “WebSob: A tool for robustness testing of web services”, In: 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007, pp. 65–66 (2007)

[87] E. Martin, S. Basu, T. Xie, T., “Automated testing and response analysis ofweb services”, In International Conference on web Services (ICWS 2007), Salt Lake City, Utah, USA, July 9-13, 2007, pp. 647–654 (2007)

[88] M.S. Jokhio, G. Dobbie, J. Sun, “ Towards specification based testing for semantic web services”, In Proceedings of the 20th Australian Software Engineering Conference (ASWEC’09), 2009

[89] R. Heckel, M. Lohmann, “Towards contract-based testing of web services”, In Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS’04), 2004

[90] L. Fratzen, J.Tretmans, R. De Vries “Towards model-based testing of web services”, In Proceedings of the International Workshop on Web Services - Modeling and Testing (WS-MaTe’06), 2006

[91] C.S. Keum, S.Kang, I.-Y. KO, J. Baik, Y.I. Choi, “Generating test cases for web services using extended finite state machine”. In Proceedings of the 18th IFIP International Conference on Testing Communicating Systems (TestCom’06), 2006.

[92] A.Tarhini, H. Fouchal, N. Mansour, “A simple approach for testing web service based applications”, In Proceedings of the 5th International Workshop on Innovative Internet Community Systems (IICS’05), 2005

[93] L.Bentakouk, P. Poizat, F. Zaidi, “Checking the behavioral conformance of web services with symbolic testing and an SMT solver”, In Proceedings of the 5th International Conference on Tests and Proofs (TAP’11), 2011

[94] A. T. Endo, A. SIMAO, “Model-based testing of service-oriented applications via state models”, In Proceedings of the 8th IEEE International Conference of Service Computing (SCC’11), 2011

[95] J. Tretmans, “Model-based testing and some steps towards test-based modelling”, In Proceedings of the 11th International School on Formal Methods for Eternal Networked Software Systems (SFM’11), 2011

[96] E. Damiani, N.El Ioini, A. Sillitti, G. Succi, “WS-certificate”, In Proceedings of the IEEE Congress on Services (SERVICESI’09), 2009

[97] SEI, “Securing web services for army SOA”, 2011. http://www.sei.cmu.edu/solutions/softwaredev/securingwebservices.cfm.

[98] A. Al-Moyaed, B. Hollunder, “Quality of service attributes in web services”. In Proceedings of the 5th International Conference on Software Engineering Advances (ICSEA’10), 2010

[99] Y. Hao, Y. Zhang, J. Cao, “A novel QoS model and computation framework in web service selection”,World Wide Web 15, 5-6, 663–684.

[100] C.-T. Kuo, H.-M. Ruan, C.-L. Lei, S.-J. Chen, “A mechanism on risk analysis of information security with dynamic assessment”, Third International Conference on Intelligent Networking and Collaborative Systems, p. 643-646, 2011

[101] http://www.assert4soa.eu/

[102] H.W. Gwendolyn , J. H. Poore, C. J. Trammell, “Statistical testing of software based on a usage model”. *Softw. Pract. Exper.* 25, 1 (January 1995), 97-108, 1995

[103] C. Kallepalli, J. Tian, "Measuring and modeling usage and reliability for statistical Web testing," Software Engineering, IEEE Transactions on , vol.27, no.11, pp.1023,1036, Nov 2001

[104] C. Trammell, “Quantifying the reliability of software: Statistical testing based on a usage model”, Proceedings of the 2nd IEEE Software Engineering Standards Symposium, 1995.

[105] J.H. Poore, and C.J. Trammell, "Engineering Practices for Statistical Testing", Crosstalk: The Journal of Defense Software Engineering, Vol. 11, No. 4, April 1998

[106] James A. Whittaker and Michael G. Thomason. 1994. A Markov Chain Model for Statistical Software Testing. IEEE Trans. Softw. Eng. 20, 10 (October 1994), 812-824.

[107] G. H. Walton, Jesse H. Poore, Carmen J. Trammell: Statistical Testing of Software Based on a Usage Model. Softw., Pract. Exper. 25(1): 97-108 (1995)

[108] A. Metzger, O. Sammodi, K. Pohl, M. Rzepka, “Towards pro-active adaptation with confidence: augmenting service monitoring with online testing”. In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10), 2010

[109] B. Dhiyanesh and A. Thiyagarajan, A novel third party auditability and dynamic based security in cloud computing. International Journal of Advanced Research in Technology, 2011, Vol. 1(Issue 1), pp. 29-33.

[110] Z. Chen and J. Yoon, IT Auditing to assure a secure cloud computing. 2010 IEEE 6th World Congress on Services, 2010

[111] B. Jr. Kaliski and W. Pauley, Toward Risk Assessment as a Service in Cloud Environments. Proceeding HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 2010.

[112] R. Accorsi and L. Lowis, ComCert: Automated Certification of cloud-based Business Processes. ERCIM News(83), October 2010

[113] Cloud Security Alliance, “Security Guidance for Critical Areas of Focus in Cloud Computing v2.1,” http://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf .

[114] http://www.truste.com/

[115] C.-T. Kuo, H.-M. Ruan, C.-L. Lei, and S.-J. Chen, A mechanism on risk analysis of information security with dynamic assessment. 2011 Third International Conference on Intelligent Networking and Collaborative Systems, 2011, pp. 643-646

[116] I. Windhorst, A. Sunyaev, "Dynamic Certification of Cloud Services," Availability, Reliability and Security (ARES), 2013 Eighth International Conference on , vol., no., pp.412,417, 2-6 Sept. 2013

[117] http://www.cumulus-project.eu/

[118] M. Cooray, J. Hamlyn-Harris, R. Merkel, "Dynamic Test Reconfiguration for Composite Web Services," Services Computing, IEEE Transactions on , vol.PP, no.99, pp.1,1, 2014

[119] N.Delgado, A.Q. Gates, S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools", IEEE Trans. Software Eng. 30 (2004) 859–872, 2004

[120] S. Benbernou, "State of the art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of sbas. Deliverable PO-JRA-1.2.1, S-Cube Consortium (2008), http://www.s-cube-network.eu/

[121] A. Metzger, O. Sammodi, K. Pohl, M. Rzepka, "Towards pro-active adaptation with confidence: augmenting service monitoring with online testing", In Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '10). 2010

[122] D. Peters, "Automated Testing of Real-Time Systems," technical report, Memorial Univ. of Newfoundland, Nov. 1999.

[123] K. Mahbub and G. Spanoudakis. "A framework for Requirements Monitoring of Service Based Systems", 2nd International Conference on Service Oriented Computing (ICSOC 2004), November 2004, pp 84 – 93.

[124] D. Bianculli, C. Ghezzi, "Monitoring Conversational Web Services", in IWSOSWE' 07, 2007.

[125] A. Grabner, "Challenges of Monitoring, Tracing and Profiling your Applications running in The Cloud", SYS-CON Media, Inc., 2009

[126] A. Grabner, "Proof of Concept:dynaTrace provides Cloud Service Monitoring and Root Cause Analysis for GigaSpaces", SYS-CON Media, Inc., 2009

[127] E. Novikoff, "The role of remote monitoring in Cloud Computing". Retrieved 09 25, 2013 from enki: http://www.enki.co/Enki-Blog/reliability-and-cloud-computing.html

[128] Ganglia System, Ganglia Monitoring Sourceforge page. Retrieved 09 25, 2013 from Ganglia: http://ganglia.sourceforge.net

[129] W. Gilani, "SLA-aware Service Management, Deliverable DA3.a", M38, F7 SLA@SOI Project. Retrieved 09 25, 2013 from SLA@SOI: http://sla-at-soi.eu/wp-content/uploads/2011/08/D.A3a-M38-SLA-aware-service-management.pdf

[130] G.B. Aceto, "Cloud Monitoring: definitions, issues and future directions". 1st IEEE International Conference on Cloud Networking (IEEE CloudNet'12), 2012

[131] V.C. Emeakaroha, T. Ferreto, M. Netto, I. Brandic, C. De Rose, "CASViD: Application Level Monitoring for SLA Violation Detection in Clouds". IEEE Computer Software and Applications Conference (COMPSAC 2012).

[132] V.C.Emeakaroha, "DeSVi:An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures.", 2nd International ICST Conference on Cloud Computing (CloudComp 2010).

[133] V.N. Emeakaroha, "Towards autonomic detection of SLA violations in Cloud infrastructures", Future Generation Comp. Syst. , 28(7), pp. 1017-1029, 2012

[134] Nagios. (2011). Nagios Is The Industry Standard In IT Infrastructure Monitoring. Retrieved 09 25, 2013 from Nagios: http://www.nagios.org

[135] D. Abramowski, "Monitoring Applications in the Cloud", Retrieved 09 25, 2013 from SYS-CON Media, Inc.: http://abramowski.sys-con.com/node/870088

[136] F.R.C. Doelitzscher, "An Agent Based business aware incident detection system for cloud environments", Journal of Cloud Computing: Advances, Systems and Applications , 2012

[137] F.R.C. Doelitzscher, "Incident detection for cloud environments", Third International Conference on Emerging Network Intelligence (EMERGING 2011), 2011 .

[138] CloudWatch, A. (2013). Amazon CloudWatch. Retrieved 09 25, 2013 from Amazon Web Services: http://aws.amazon.com/cloudwatch/

[139] F. Sabahi, "Secure Virtualization for Cloud Environment Using Hypervisor-based Technology", International Journal of Machine Learning and Computing , 2011

[140] J. Szefer, "Architectural Support for Hypervisor-Secure Virtualization", Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems. New York, 2012

[141] H. Jong, S. Weidong, Z. Xinwen, H. Woo, "Architectural support of multiple hypervisors over single platform for enhancing cloud computing security". Conf. Computing Frontiers, (pp. 75-84), 2012

[142] E. Keller, "Eliminating the Hypervisor Attack Surface for a More Secure Cloud". ACM Conference on Computer and Communications Security , 2011

[143] S. Jin, Architectural Support for Secure Virtualization under a Vulnerable Hypervisor. The 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44), 2011

[144] H. Foster , G. Spanoudakis, "Advanced Service Monitoring Configurations with SLA Decomposition and Selection" 26th ACM Symposium Applied Computing – Track on Service Oriented Architecture and Programming.

[145] CSA. (2013). CSA Cloud Security Alliance. Retrieved 09 25, 2013 from Introduction to CloudTrust Protocol: https://cloudsecurityalliance.org/research/ctp/

[146] https://cloudsecurityalliance.org/star/self-assessment/

[147] http://cloud-standards.org/wiki/index.php?title=Main_Page

[148] https://www.owasp.org/index.php/Main_Page

[149] https://www.ogf.org/ogf/doku.php

[150] Cloud Security Alliance, Cloud Audit, Available from: https://cloudsecurityalliance.org/research/cloudaudit/

[151] Cloud Security Alliance, Cloud Controls Matrix, Available from: https://cloudsecurityalliance.org/research/ccm/

[152] https://eurocloud-staraudit.eu/

[153] http://www.coso.org/

[154] http://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Minimum_information/SecurityRecommendationsCloudComputingProviders.pdf?__blob=publicationFile

[155] http://www.cert.org/resilience/products-services/octave/

[156] http://www.cesg.gov.uk/servicecatalogue/product-assurance/cpa/Pages/CPA.aspx

[157] https://cloudsecurityalliance.org/research/ccm/

[158] M. Montenegro, A. Mana and H. Koshutanski. "Improving Security Assurance of Services Through Certificate Profiles". in Proc. of the 3rd International Workshop on Adaptive Services for the Future Internet.

[159] M. Razzazi, M. Jafari, S. Moradi, H. Sharifipanah, M. Damanafshan, K. Fayazbakhsh, A. Nickabadi, "Common Criteria Security Evaluation: A Time and Cost Effective Approach," Information and Communication Technologies, 2006. ICTTA '06.

[160] CUMULUS consortium, "Security-aware SLA specification language and cloud security dependency model," Deliverable D2.1, Sep 2013.

[161] K.t. Kearney, F. Torelli, C. Kotsokalis, C., "SLA★: An abstract syntax for Service Level Agreements," 11th IEEE/ACM International Conference on Grid Computing (GRID), 2010, vol., no., pp.217,224, 25-28 Oct. 2010.

[162] K. Mahbub, G.E. Spanoudakis, T. Tsigkritis, "Translation of SLAs into Monitoring Specifications, In Service Level Agreements for Cloud Computing", Weider P. et al. (eds), Part 2, pp. 79-101, 2011

[163] ANTLR, http://www.antlr.org/

[164] https://www.commoncriteriaportal.org/pps/

[165] CUMULUS consortium, "Certification models", Deliverable D2.2, Sep 2013.

[166] CUMULUS consortium, "Certification Models v.2", Deliverable D2.3

[167] CUMULUS consortium, "Core Certification Mechanisms v.2", Deliverable D3.2

[168] CUMULUS consortium, "CUMULUS Infrastructure v.2", Deliverable D5.2

[169] http://searchsoftwarequality.techtarget.com/definition/HTTPS

[170] https://www.commoncriteriaportal.org/files/ppfiles/FMV-PP-ESD.pdf

[171] L. Wang, Zhengping Wu, "A Trustworthiness Evaluation Framework in Cloud Computing for Service Selection," Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on , vol., no., pp.101,106, 15-18 Dec. 2014

[172] A. Jøsang, "Trust and Reputation Systems" In A. Aldini and R. Gorrieri (Eds.), Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures. Springer, Italy, September 2007.

[173] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted P2P Computing," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 4, pp. 460-473, Apr. 2007.

[174] R. Zhou, K. Hwang and Min Cai "GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks," IEEE Trans. Knowledge and Data Engineering, vol. 20, no. 9, pp. 1282-1295, Sep. 2008.

[175] D, Nuñez., F. C. Gago., S. Pearson, M. Felici, "A Metamodel for Measuring Accountability Attributes in the Cloud," Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2013), pp. 355-362.

[176] T. H. Noor, Q. Z. Sheng, A. H. Ngu, A. Alfazi, “CloudArmor: A Platform for Credibility-based Trust Management of Cloud Services,” The 22nd ACM Conference on Information and Knowledge Management, pp. 2509-2512, October 2013.

[177] T. H. Noor, Q. Z. Sheng, S. Zeadally, J. Yu, “Trust Management for Services in Cloud Environments: Obstacles and Solutions,” ACM Computing surveys Journal, Vol 46, No 1, p12, 2013.

[178] J. Huang, D. M Nicol, “Trust mechanisms for cloud computing”, Journal of cloud computing, Springer, 2013.

[179] “Certidication Schemes for Cloud Computing”, A study prepared for the European Commission, DG Communications Networks, Content & Technology, Trilateral Research and Consulting, https://ec.europa.eu/digital-agenda/en/news/certification-schemes-cloud-computing

[180] “NIST cloud computing standards roadmap, first edition”, NIST Gaithersburg, MD, USA, 2011, http://www.nist.gov/itl/cloud/upload/NIST_SP-500-291_Jul5A.pdf

[181] R. Knode and D. Egan, “Digital trust in the cloud – A precis for the CloudTrust protocol (V2.0)”, CSC, July 2010.

[182] M. Kuehnhausen, V.S. Frost, G.J. Minden, "Framework for assessing the trustworthiness of cloud resources," Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on , vol., no., pp.142,145, 6-8 March 2012