Enhanced Connectors Synthesis to address Functional, Performance, and Dependability aspects

Nicola Nostro[a], Romina Spalazzese[c],
Felicita Di Giandomenico[d], Paola Inverardi[b]

[a] *University of Florence - Italy, nicola.nostro@unifi.it*
[b] *University of L'Aquila - Italy, paola.inverardi@univaq.it*
[c] *Malmö University - Sweden, romina.spalazzese@mah.se*
[d] *CNR of Pisa - Italy, f.digiandomenico@isti.cnr.it*

Abstract

Our everyday life is pervaded by the use of a number of heterogeneous systems that are continuously and dynamically available to interoperate in the networked environment to achieve some goal. The evolving nature of this environment with no a-priori knowledge of the systems, requires automated solutions as means to achieve interoperability with the needed level of flexibility. We already investigated and proposed an approach to the automated synthesis of CONNECTors (or mediators) between heterogeneous Networked Systems (NSs) for their *functional interoperability* at application layer.

In this paper we propose (i) an *approach* to *enhance* the CONNECT*ors* taking into account performance and dependability aspects spanning pre-deployment time and run-time and (ii) a CONNECT*or adaptation process*, related to the performance and dependability aspects and (iii) a stochastic model-based implementation of the performance and dependability analysis. By reasoning on systems' specification, during the *pre-deployment phase* the approach produces a mediator that satisfies the functional, performance and dependability requirements. At *run-time*, if a performance or dependability violation occurs, an adaptation is triggered: by reasoning on the new specification, the approach identifies the proper mechanism to solve the problem and updates the CONNECTor accordingly. In addition, we implemented, analysed, and critically disussed a case study.

Keywords: Connector synthesis, Dependability, Performance, Interoperability

1. Introduction

An always increasing number of heterogeneous Networked Systems (NSs) pervades our everyday life. Nowadays, we use more and more systems that are dynamically available in the networked environment and that, by interoperating with other systems, allow us to reach some goal. The goal can be about both functional and/or non functional aspects and has to be satisfied in order for two systems to interoperate. Abstractly, some of these heterogeneous applications could interact, since they have compatible functionalities and similar interaction protocols. Nevertheless, their ability to seamlessly interoperate may be undermined by mismatches in their protocols (e.g., interactions order or input/output data formats) and non functional requirements. Solving such mismatches and meeting the non functional requirements, asks for applications' adaptation through a CONNECTor. Further, in this evolving environment, there is no a-priori knowledge of the systems until they are discovered in the network and possibly learned, and automated solutions appear to be the most effective way to enable composition and interoperability of applications with the needed level of flexibility.

We already investigated and proposed an approach to the automated synthesis of CONNECTors between heterogeneous NSs for their *functional interoperability* at application layer [1, 2], i.e., referring to functional properties and aiming at allowing NSs to communicate and correctly coordinate. However, effective interoperability also requires that the CONNECT*ed system*, i.e., networked systems and CONNECTor, provides *non functional interoperability*, i.e., the required non functional properties, during their interoperation. In particular, in this work with non functional properties we refer to issues arising from the execution environment due to the initial uncertainties about the knowledge of the environment itself and its unpredictable evolution.
Thus, to support both functional and non functional interoperability, a suitable adaptive framework is required that provides a solution to both.

In this paper, we present a new approach for the synthesis of connectors addressing functional and (some) non functional interoperability in conjunction. In summary, the main contributions of this paper are the following: *(i)* an approach to *enhance* the functional CONNECT*ors* taking into account performance and dependability aspects spanning pre-deployment time and run-time; *(ii)* a CONNECT*or adaptation process* to preserve the CONNECTor adequacy with respect to non functional requirements along the system lifetime; *(iii)* a stochastic model-based implementation of the performance and

dependability analysis. These main contributions are shown through the implementation, analysis, and critical disussion of a case study.

By reasoning on systems' specification, during the *pre-deployment phase* the approach produces a mediator that satisfies the functional, performance and dependability requirements. At *run-time*, when a performance or dependability violation occurs, an adaptation is triggered: the approach, by reasoning on the new specification, identifies the proper mechanism to solve the problem and properly update the CONNECTor.

The rest of the paper is organized as follows. Section 2 presents the considered context, the process followed by our approach, and background notions. Section 3 depicts a case study used for explanation and experimentation purposes. Section 4 provides the description of the proposed approach for an automated procedure to provide dependability and performance stochastic model-based analysis as a support for the synthesis of dependable CONNECTor, both a pre-deployment time and run-time. Section 5 and 6 provide a critical discussion and related work respectively. Finally Section 7 concludes the paper.

2. Setting the Context

A number of heterogeneous networked systems, e.g., tablet, desktop, smartphone, laptop, and robots, are dynamically available in the networked environment. NSs heterogeneity spans many aspects and we focus on *application layer heterogeneity*. In the following we describe an example of heterogeneous applications that will be detailed and extended in Section 3.

The laptop in Figure 1 runs a *client application* represented by the gray icon *Appl1* that is suited to directly interoperate, with some performance and dependability requirements, with a server application represented by the gray icon *Appl1s* owned by the Unmanned Ground Vehicle (UGV). Instead, *Appl1* is not able to directly interact with the *server application* represented by the gray icon *Appl2*, owned by and running on the Unmanned Aerial Vehicle (UAV). *Appl1* and *Appl2*, despite some protocol discrepancies and non functional concerns, might be compatible through a CONNECTor C mediating their differences.

We consider that the *NSs and their applications are black box* and that, for each application, NSs expose in their interface: the description of the interaction behavior and non functional properties, and possibly the non functional requirements on potential interactions with other NSs. Thus, *the*

CONNECTor *is the only locus where we can act to make the* CONNECTed *system satisfy both functional and non functional interoperability concerns.*

From a functional standpoint, we build a CONNECTor such that it makes the NSs behaviors compatible through a proper interaction with them. For instance, if a NS sends data with a finer granularity with respect to another NS, the mediator has to collect all the data information and then to send them properly to the other.

From a non functional perspective, to make the CONNECTed system satisfy the requirements, we suitably enrich the CONNECTor behavior. For instance, in the described scenario, a performance concern arises because *Appl1* wants to interoperate with *Appl2* in a way such that the latency between the sending of a command (by *Appl1*) and the reception of the corresponding acknowledgment sent by *Appl2*, must be under a given threshold (performance requirement).

In the following we overview our *approach* to synthesize CONNECTors, developed as part of the CONNECT project [3], that meet both functional and non functional concerns, including also the description of our *adaptation process* (which are detailed in Section 4).

2.1. Our Approach

Our approach takes place both at *pre-deployment time* and *run-time*. During the *pre-deployment time*, it takes as input applications' specification (see ① in Figure 1). By reasoning on all of them, a mediator is automatically synthesized solving discrepancies and enabling the functional interoperation

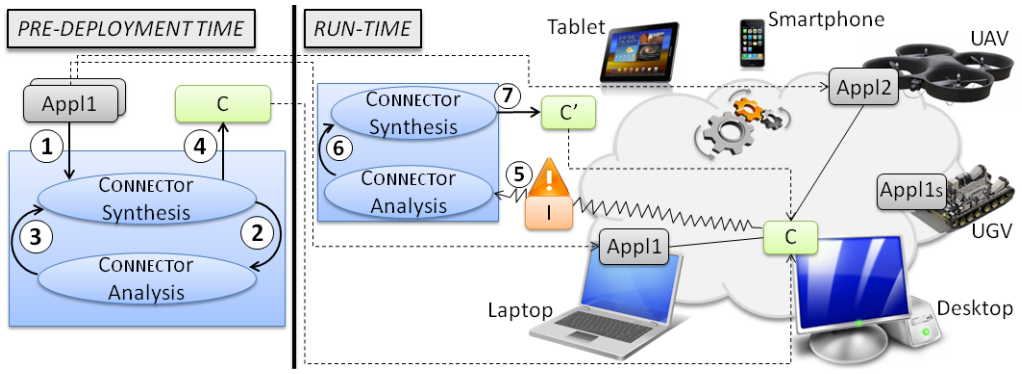

Figure 1: Our approach (from ① to ⑦) and adaptation process (cycle ⑤-⑥-⑦)

among them (CONNECTor Synthesis module). By taking as input the synthesized mediator (see ② in Figure 1), and the non functional requirements, stochastic model-based analysis assesses the desired non functional properties, and feedback is provided to the CONNECTor Synthesis module (see ③ in Figure 1) about how the system is expected to operate and how to possibly enrich the behavior of the previously synthesized mediator if the non functional requirements are not satisfied (CONNECTor Analysis module). In order to solve, if possible, the problem/violation arising from the execution environment, the CONNECTor Analysis module identifies a proper mechanism to improve performance and/or dependability choosing among *retry, majority voting, probing, error correction* [4]. We assume that the NSs populating the environment are able to manage the mechanisms mentioned above that we use to enrich the CONNECTor. The output of this collaborative computation at pre-deployment time is a CONNECTor C satisfying functional, performance, and dependability requirements (see ④ in Figure 1).

At *run-time* the applications and the synthesized mediator, also equipped with probes to monitor the CONNECTed system [5], are deployed and running on some devices. When a performance or dependability violation occurs (see ⑤ in Figure 1), due to problems arising from the execution environment, it is identified by the probes and the **adaptation process** is triggered. By reasoning on the new systems specification, that is the input systems specification modified through their run-time observation, the CONNECTor Analysis module acts similarly to what it does at pre-deployment time. It identifies a proper mechanism to improve performance/dependability in order to solve, if possible, the problem/violation choosing among *retry, majority voting, probing, error correction.* Subsequently, the CONNECTor Analysis module triggers the CONNECTor Synthesis module by providing the needed information (see ⑥ in Figure 1) to properly *enrich* the behavior of the previously synthesized CONNECTor with the identified mechanism. The output of this collaborative computation at run-time, given some violation occurrence, is a new CONNECTor satisfying the functional, performance, and dependability requirements (see ⑦ in Figure 1). This concludes one cycle of the adaptation process (⑤, ⑥, and ⑦ in Figure 1). It is worth to notice that the run-time cycle of our approach is repeated each time a new violation is detected, while the pre-deployment time activities are done only once.

2.2. Background Model

In the following, we recall notations to describe the NSs.

2.2.1. Networked Systems' applications specification

We use Labeled Transition System (LTS) to model applications' protocol and refer to ontologies to conceptualize their actions and input/output data, and to reason on them.

Specifically, we consider what we call *enhanced Labeled Transition Systems (eLTS)* that is a quintuple (S, L, D, F, s_0) where: S is a finite non-empty set of states; L is a finite set of labels describing actions with data; $D \subseteq S \times L \times S$ is a transition relation; $F \subseteq S$ is the set of final states; $s_0 \in S$ is the initial state. The eLTS' labels are of the form $<op, In, Out>$ where: op is an observable action referring to an ontology concept or is an internal action denoted by τ; an action can have output/input direction denoted by an overbar on the action, e.g., $\overline{act}$ or act. In, Out are the sets of input/output data that can be produced/expected, whose elements refer to ontology elements. We are able to describe the following actions with data: (1) *output action with outgoing parameters and incoming return data* $<\overline{op}, In, Out>$, where In is produced, while Out is expected; (2) *input action with incoming parameters and outgoing return data* $<op, In, Out>$, where In is expected, while Out is produced. One at a time In or Out might be empty because no input/output data is expected/produced. This leads to 4 variants of the actions, two for (1) and two for (2)[1].

Between protocols, we assume synchronous communications on complementary actions. Actions $<op_1, In_1, Out_1>$ and $<op_2, In_2, Out_2>$ are complementary iff $op_1 = \overline{act}$ and $op_2 = act$ and $In_2 \subseteq In_1$ and $Out_1 \subseteq Out_2$ (and similarly with exchanged roles of op_1 and op_2). Moreover, we consider finite traces by assuming a bound on the number of cycles execution.

Ontologies describe domain-specific knowledge through concepts and relations, e.g., the subsumption: a concept C *is subsumed by* a concept D in a given ontology $\mathcal{O}$, noted by $C \sqsubseteq D$, if in every model of $\mathcal{O}$ the set denoted by C is a subset of the set denoted by D [6]. We assume that each NS action and datum refer to some concept of an existing domain ontology so that we can reason on them in order to find a common language between protocols.

[1]Note that (1) ($<\overline{op}, In, Out>$) can be equivalently described by the two following actions: $<\overline{op}, In, ->$ and $<\overline{op}, -, Out>$. This applies similarly to (2) ($<op, In, Out>$ can be described as $<op, In, ->$ and $<op, -, Out>$).

2.2.2. Non functional concerns: properties, requirements, mechanisms

The NSs dependability and performance model and required properties are expressed as metrics and guarantees. *Metrics* are arithmetic expressions that describe how to obtain a quantitative assessment of the properties of interest of the CONNECTed system. They are expressed in terms of states and transitions of the eLTS of the NSs. *Guarantees* are boolean expressions that are required to be satisfied on the metrics.

In order to enrich the CONNECTor, the four *mechanisms* that we can leverage on and apply singly or in combination are: *retry, majority voting, error correction, and probing* that have been presented in [4]. The Retry, Majority Voting and Error Correction mechanisms are specifically applied if the metric to be analysed is related to dependability aspects. These mechanisms can be applied when the metric under analysis is a function of failure probabilities of all the communications/actions between the CONNECTor and the NSs and there are soft or no timing constraints on the application. The Probing mechanism is applied when the metric to be analyzed is related to performance aspects under generic constraints, in particular related to timing aspects. In general this mechanism allows to exploit the most performable communication channels available for transmission, selecting for the use the most efficient one. In the following we briefly describe the application of the four mechanisms to the CONNECTor.

Retry mechanism. The CONNECTor sends again its request n times if a confirmation/ACK is not received back from a NS. The sent messages must have a sequence ID in order to identify them.

Majority voting mechanism. A portion of the CONNECTor needs to run on each NS. On the sender side, the CONNECTor needs to access the data in order to send it on several channels. On the receiver side, the CONNECTor has to choose the correct data to be passed to the NS based on a voting policy.

Error correction mechanism. A portion of the CONNECTor needs to run on each NS. On the sender side, the CONNECTor needs to access the data to be sent and to know extra information necessary for the error detection and reconstruction in order to send them over two channels. On the receiver side, the CONNECTor has to receive the data and the extra information from two channels, in case of error a correction is applied to deliver correct data to the NS.

Probing mechanism. A portion of the CONNECTor needs to run on each

NS. On the sender side, the CONNECTor checks the performance of redundant channels in order to send the data over the better one. On the receiver side, the CONNECTor has to receive from either channels and pass the data to the NS.

2.2.3. Synthesis and DEPER *Enablers*

An Enabler is intuitively a networked entity that includes some intelligence and logic[2]. To enable a required connection it is needed the collaboration of several enablers.

-*Synthesis*- In [1] and [2] is proposed an approach for the automated synthesis of mediators that overcomes interoperability issues between two heterogeneous emerging protocols, given the applications model, the ontology describing the domain-specific knowledge and a bound on the number of executions of cycles making the traces finite. Figure 2 shows our synthesis methodology. Our emergent mediator is automatically elicited and synthesized. It makes the communication and correct coordination between heterogeneous protocols possible despite a set of mismatches that we characterized in [7] together with their related mediating connector patterns. The approach consists of three phases: identification of the common language, behavioral matching, and mediator synthesis. The *Identification of the Common Language* (*Abstraction*) (① in Figure 2) takes as input the applications models and the

[2]CONNECT EU project- `http://connect-forever.eu/`

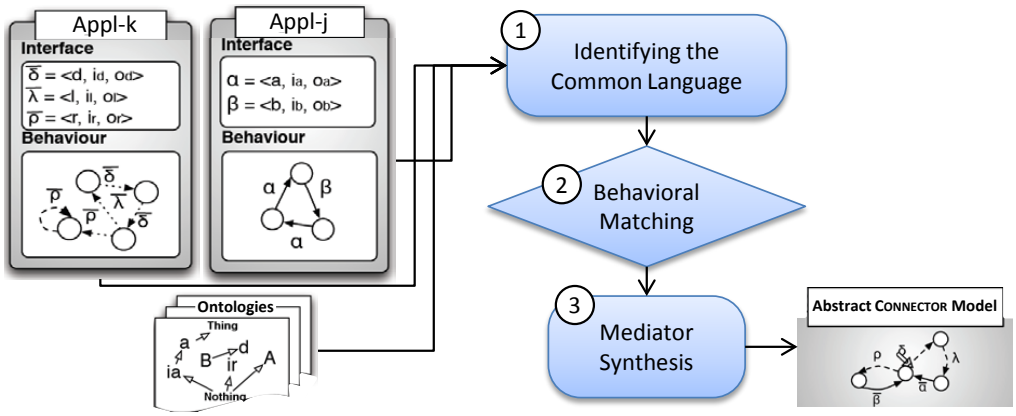

Figure 2: Synthesis Approach

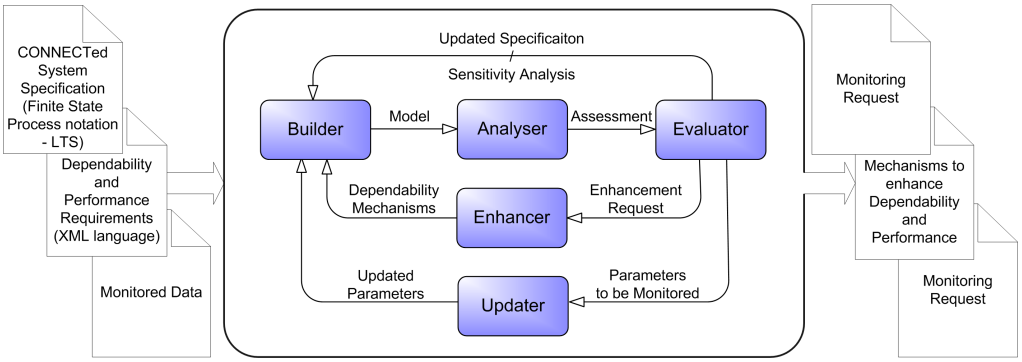

Figure 3: DePer Architecture

subset of the domain ontology they refer to, and identifies the applications common language through the ontologies. The common language makes applications behavior comparable to reason on them. The *Behavioral Matching* (*Matching*) (② in Figure 2) checks the applications behavioral compatibility, e.g., two applications can synchronize at least on one trace reaching one of their respective final state. This step identifies possible mismatches to be reconciled [8]. Finally, the *Mediator Synthesis* (*Synthesis*) (③ in Figure 2) produces a (intermediary) mediator that addresses the identified mismatches between the two applications.

-DePer- This enabler performs dependability and performance analysis through a stochastic model-based approach. DePer is composed by five modules: Builder, Analyser, Evaluator, Enhancer and Updater [9]. Figure 3 shows its architecture.

The *Builder* module takes as input the specification of the CONNECTed system. This specification is given as eLTS annotated with non-functional information necessary to build the dependability and performance model of the CONNECTed system. The model is built up using the Stochastic Activity Networks (SAN) formalism [10]. The *Analyser* module extends the model developed by the Builder with reward functions suitable to quantitative assessment of the dependability and performance properties required by the NSs. For the assessment purpose, this module exploits the Möbius tool [11]. The *Evaluator* module is in charge of checking whether the analysis results match with the properties required by the networking systems willing to communicate, or not. The *Enhancer* module is activated when the dependability and/or performance requirements are not satisfied, both at

pre-deployment and at run-time, and tries to enhance the CONNECTor with dependability mechanisms in order to satisfy the requirements. The issue of how to select both the proper dependability mechanism, and the elements of the CONNECTor where to include the mechanism, have been tackled in [12], where a method has been described, which is out of the scope of this paper. Finally, the *Updater* module is in charge to update the values of model parameters used in the analysis [13], on the basis of the data related to real executions of the CONNECTed system as gathered through a monitoring infrastructure. Keeping the analysis model updated is very important in the CONNECT context, to cope with evolution of NSs and possible inaccurate information known at pre-deployment time. Based on the monitored data, this module may trigger a new analysis by reactivating the cycle on the Builder, Analyser and Evaluator.

3. GMES Case Study: the Forest-fire Emergency

In order to show how Synthesis and DEPER work in an integrated way, this section describes our scenario based on the Global Monitoring for Environment and Security (GMES) European Programme[3]. The GMES emergency management service covers several catastrophic events, e.g., floods, earthquakes, fires.

Scenario. We concentrate on the management of a forest-fire emergency situation [14] close to a border village and a factory between Country A and Country B.

The scenario is shown in Figure 4. Country A's Command and Control center (C2-A) is in charge of forest monitoring and forest fire management. During a forest fire, the fire goes wider and there is a real threat to the village and the factory. Thus, Country A asks Country B to provide support. Country B supplies reinforcement resources to be used by the C2-A of Country A. Such resources *have the same aim* with respect to similar resources of Country A, (e.g., to provide high quality images or weather information of the area of the fire), *but use different application protocols*. Thus, it is needed to synthesize a mediator to allow Country A to exploit them during the emergency. The applications we consider in the scenario are: *i)* the **C2-A application** of Country A -*Appl1*; *ii)* an **Unmanned Aerial Vehicles** (UAVs) application of Country B -*Appl2*- equipped with various Video

[3]http://www.gmes.info

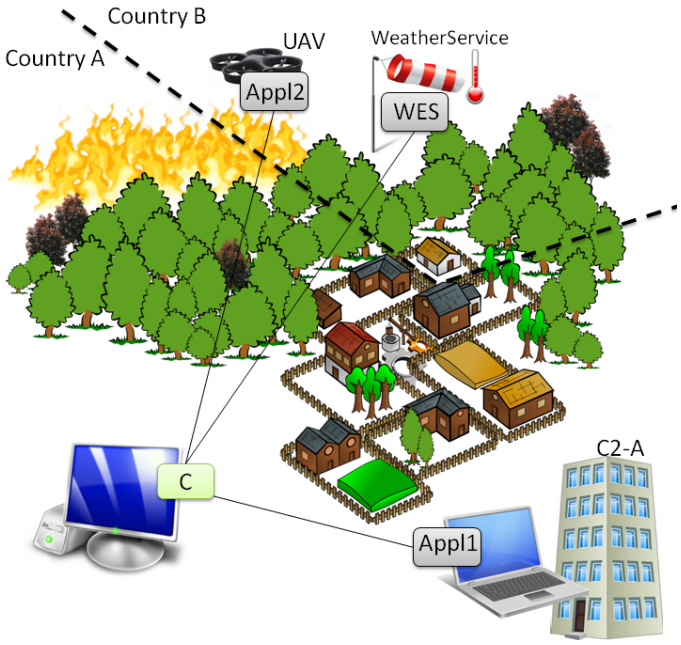

Figure 4: The Forest-fire Emergency case study

Cameras to get a better view of the fire front close to the village; *iii)* the **Weather Service** application of Country B - *WES*- in order to continuously get information about temperature, humidity and wind of the affected area.

Country B provides to Country A applications UAVs and WES that in principle are compatible with some applications that Country A already owns. However, due to some protocol discrepancies and non functional concerns, the applications of Country A cannot seamlessly interoperate with the ones of Country B and a CONNECTor that mediates their differences is needed in between.

In the described scenario, a *performance* concern arises because when C2-A sends an order to move to a robot (e.g., UAV) it requires to receive an acknowledgment within a certain time (*latency*). The latency requirement, specified by C2-A, is *to receive an acknowledgment within 5 time units.* A *dependability* issue arises when considering the Weather Service and concerns the percentage of weather data that C2-A correctly receives from the Weather Service (*coverage*). The coverage requirement specified by C2-A is *to correctly receive at least the 90% of the required weather data.* In order

to meet the above mentioned non functional requirements, a proper CONNECTor is needed for the CONNECTed system. In the following we describe the behavior of NSs involved in our scenario.

3.1. Command and Control Center of Country A

Figure 5 shows the eLTS of the Command and Control Center (C2-A) including actions to interact with both the WES and the UAV.

To get weather information of an area, C2-A simply sends a *getWeather* message with the relative *zipCode*, and receives back *weatherInfo* from the Weather Station including: temperature, humidity, and wind condition. C2-A can also get a weather forecast of the area in a specific period through a *getForecast* message. To access the resources operating in the field and equipped with one or more video cameras (e.g., the UGVs or UAVs), C2-A first needs to authenticate with the resource by sending a *getToken* message and receiving back the *token*. Then, it can instruct the resource to move *forward*, *backward*, *left* or *right*, by sending the proper message and receiving back a *response* message about the movement. Then, C2-A can choose a

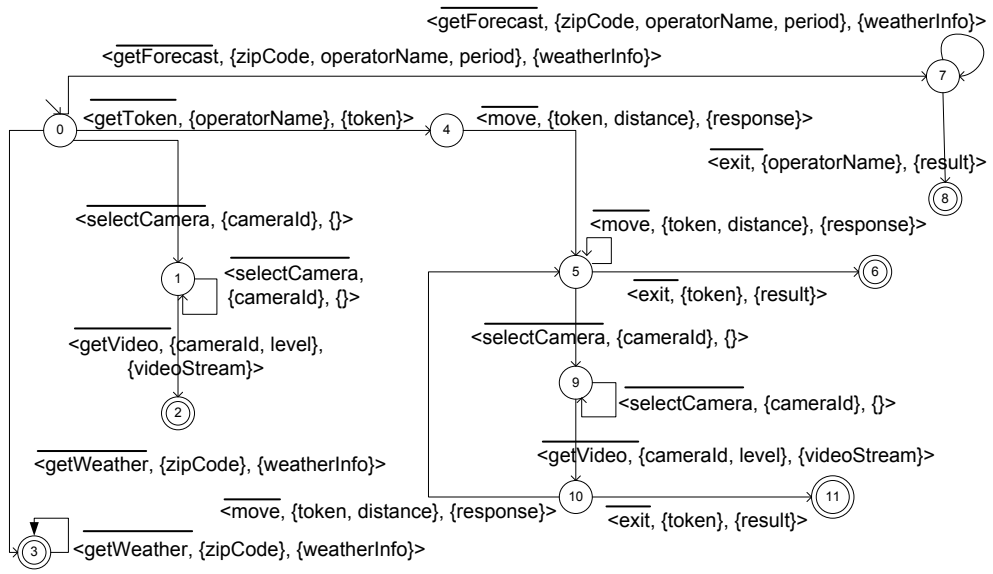

Figure 5: eLTS of the Command and Control Center where the action *move* (<$\overline{move}$, {*token, distance*}, {*response*}>) represents four alternative commands to move: *backward*, *forward*, *left*, and *right* respectively.

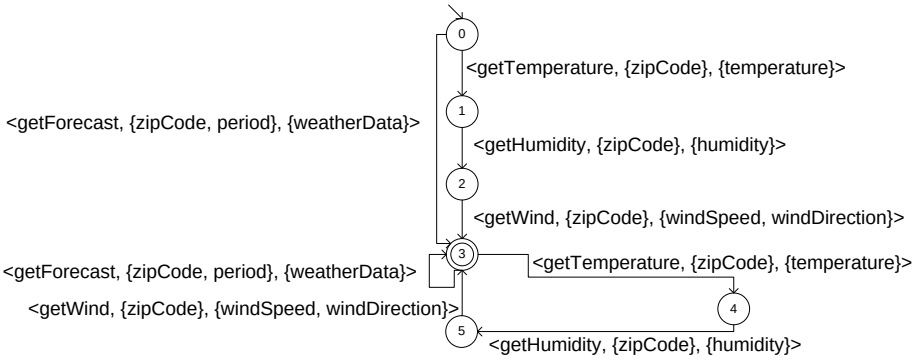

Figure 6: Behavior of the Weather Service

camera (*selectCamera*) installed on the resource and receive the video stream (*getVideo*) with a specified zoom *level*.

3.2. Weather Service of Country B

Figure 6 shows the eLTS of the Weather Service (WES). The WES expects to receive either a weather forecast request (*getForecast*) about a specified area of interest (*zipCode*) in a certain *period* of time, or the current weather information (*getTemperature*, *getHumidity*, *getWind*). In the case study the user is interested to know the current weather information.

3.3. UAV and integrated Video Cameras of Country B

Figure 7 shows the eLTS representing the UAV and its integrated Video Cameras. The UAV first receives an identification request (*getIdentifier*) for which it gives back an *identifier* followed by a *takeOff* request for which a *response* is sent back. After the *takeOff*, the UAV expects to receive either a request to *land* and then to *quit* or to *move* (left, right, forward, backward, up or down). For each movement order the UAV sends back a *response* message. Moreover, during the flight, the UAV can receive a *chooseCamera* request and it sends back to the requester the real time video stream taken from the chosen video camera. The *chooseCamera* can be followed by any number of *zoomIn* and/or *zoomOut* requests until the reception of a request to land and then to quit.

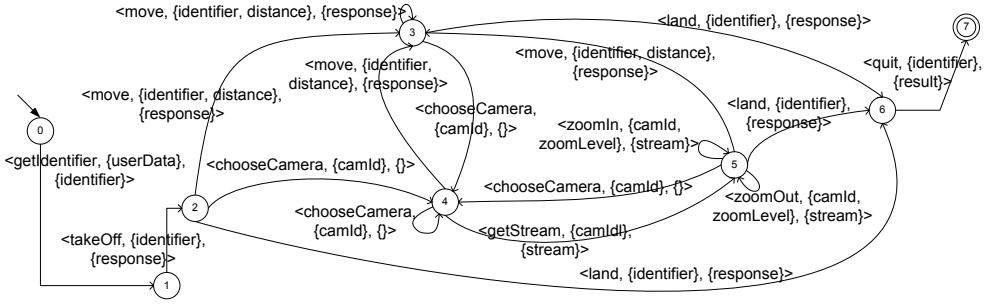

Figure 7: Behavior of the UAV and its integrated Video Cameras, where the action *move (<move, {identifier, distance}, {response}>)* represents four alternative commands to move: *backward*, *forward*, *left*, *right* respectively).

4. Our Approach to the Enhanced Connector Synthesis

In the following we detail our approach for the synthesis of enhanced CONNECTors to address functional, performance, and dependability aspects leveraging on the forest fire case study. We first investigate a scenario considering black box NSs and applications. Then, by relaxing this assumption, we consider another scenario where NSs trust and authorize the CONNECTor to access them in order to apply some mechanisms that enhance the synthesized CONNECTor.

Our preliminary study conducted in [5], shows the identified collaboration among the Synthesis, DEPER and Monitor Enablers in order to provide on-the-fly dependable mediation between heterogeneous NSs. In the following we focus on the cooperation between DEPER and Synthesis Enablers and provide a detailed description of an approach that is part of the high level vision provided in [5].

4.1. Our Approach @ Pre-deployment Time

The approach (illustrated by Figure 1) takes as input systems' specification, i.e., Command and Control Center, UAV with Video Cameras, and Weather Service, including: the applications behavior (as enhanced Labeled Transition System) in terms of actions and input/output data, the application domain ontology and sub-ontologies of the applications describing the meaning of their actions and data, a bound on the number of executions of cycles in the applications behavior.

Appl 1 (C2-A)	Appl2 (UAV)	WES (Weather Service)
<getToken, {operatorName}, {token}>	<getIdentifier, {userData}, {identifier}>	---
<right, {token, distance}, {response}>	<moveRight, {identifier, distance}, {response}>	---
<selectCamera, {cameraId}, {}>	<chooseCamera, {camId}, {}>	---
---	<land, {identifier}, {response}>	---
---	<takeOff, {identifier}, {response}>	---
<getWeather, {zipCode}, {weatherInfo}>	---	<getTemperature, {zipCode}, {temperature}> <getHumidity, {zipCode}, {humidity}> <getWind, {zipCode}, {windSpeed, windDirection}>
<getForecast, {zipCode, operatorName, period}, {weatherInfo}>	---	<getForecast, {zipCode, period}, {weatherData}>

Figure 8: Some Ontological Correspondences

The first computation is done by the Synthesis module [1] by reasoning on the input and following the three phases described in Section 2.2.3, i.e., identification of the common language, behavioral matching, and mediator synthesis. First, the common language among the actions and data of *Appl1*, *Appl2*, and *WES* is identified by classifying their respective sub-ontologies into the GMES application domain ontology through an ontology reasoner revealing the correspondences (see Figure 8 for some examples). Then, the behavioral matching checks the behavioral compatibility on the eLTSs of *Appl1*, *Appl2*, and *WES*, i.e., it checks that the applications can synchronize at least on one trace reaching one of their respective final states by properly reconciling possible mismatches through a mediator. Finally, the mediator synthesis automatically produces a CONNECTor that addresses the identified mismatches/discrepancies between the applications thus enabling their functional interoperation.

Considering our case study, this step takes as input: the eLTSs of C2-A (Figure 5), Weather Service (Figure 6), UAV with its integrated Video Camera (Figure 7); the GMES domain ontology and its sub-ontologies describing the applications actions and data to identify their common language (Figure 8). The output of this step is the mediator shown in Figure 9.

After, DEPER automatically builds the dependability and performance model of the CONNECTed system, through the Builder module, as described in Section 2.2.3. It takes as input: the eLTS of the intermediate mediator synthesized during the previous stage, the annotated non functional data of the NSs (i.e., the time to complete, the firing and failure probability of each labeled transition), and the dependability and performance requirements in terms of metrics and guarantees. DEPER automatically builds the CONNECTed system model in terms of SAN models [10], as shown in Figure 10.

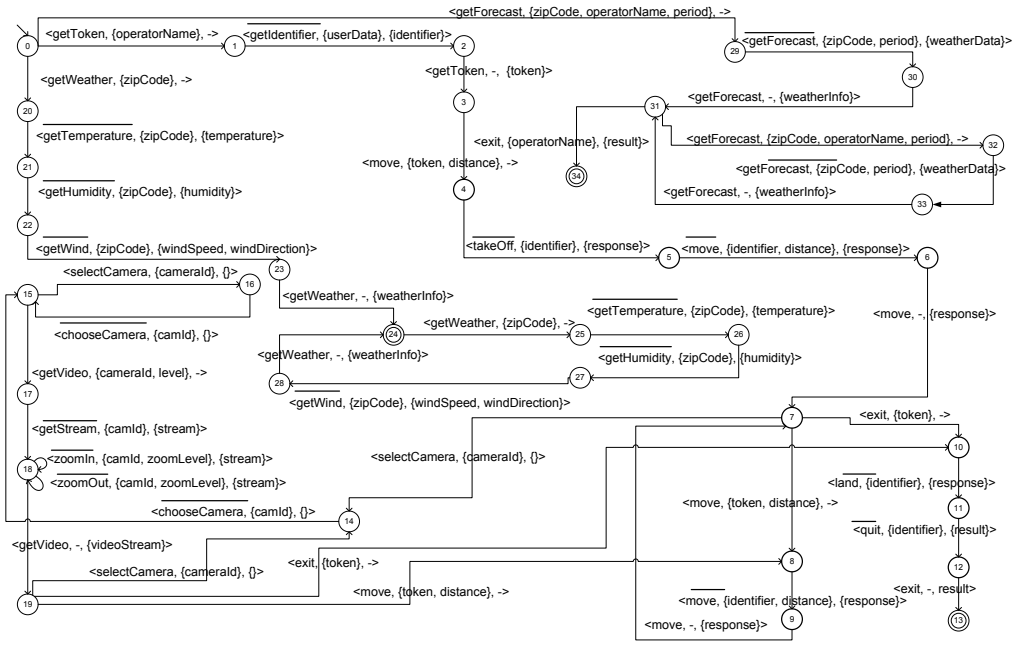

Figure 9: Behavior of the Mediator, where the action *move (<move,* {*identifier, distance*}, {*response*}>*)* represents four alternative commands to move: *backward*, *forward*, *left*, *right (<backward,* {*identifier, distance*}, {*response*}>, *<forward,* {*identifier, distance*}, {*response*}>, *<left,* {*identifier, distance*}, {*response*}>, *<right,* {*identifier, distance*}, {*response*}> respectively).

Regarding the forest fire scenario, we recall that the input eLTSs are: the intermediate mediator of Figure 9, the C2-A in Figure 5, the UAV of Figure 7, and the Weather Service in Figure 6. Further, we recall that the non functional requirements are: *latency* as a performance indicator, measured from when C2-A sends one of the possible movement order ($\texttt{<}\overline{move}\texttt{,}\ \{token, distance\}\texttt{,}\ \texttt{->}$) to when it receives an acknowledgment ($\texttt{<}\overline{move}\texttt{,}\ \texttt{-,}\ \{response\}\ \texttt{>}$); *coverage* as a dependability indicator, given by the percentage of weather data that C2-A correctly receives from the Weather Service.

In order to assess the desired non functional properties, DePer performs a stochastic model-based analysis through the Analyser module. The results are then verified, by the Evaluator module, to check whether they meet or not the NSs requirements of the ConnecTed system. At this point, two different events may occur: (i) the dependability and performance requirements are satisfied or (ii) the analysis results do not meet the requirements. In both

cases DePer provides to the Synthesis module feedback about the outcome of the analyses. In case of unsatisfied requirements, the Enhancer module is in charge of selecting one of the existing mechanisms and of proposing how to enhance the Connector.

Considering our case study, the analysis, performed by DePer, returns as output a positive feedback on the performance requirement (the Connected system satisfies it); while it returns a negative feedback on the dependability requirement that is not satisfied. Hence, an enhancement is needed to properly enrich the mediator to let the Connected system satisfy also the dependability requirement (if possible).

The output of this collaborative computation between Synthesis and DePer at pre-deployment time, given the initial knowledge of the systems, is an enhanced Connector satisfying functional, performance, and dependability requirements. Regarding the case study, the output of this stage is the enhanced Connector including the Majority Voting mechanism (see Figure 11) as indicated by DePer through the analysis of the Connected system, as detailed in the next subsection. Figure 12 shows the eLTS of the Connector enhanced with the majority voting mechanism on the portions highlighted by the small ellipses.

The big ellipse shows the portion of eLTS representing the majority voting mechanism enhancement that describes all the possible interleavings of a message sent over three channels (X_1 that is message X sent over channel 1, X_2 over channel 2, and X_3 over channel 3). This behavior is contained in each small ellipse where X_i, the generic message X over channel i, is instantiated with the proper message. From the initial state of the eLTS, the

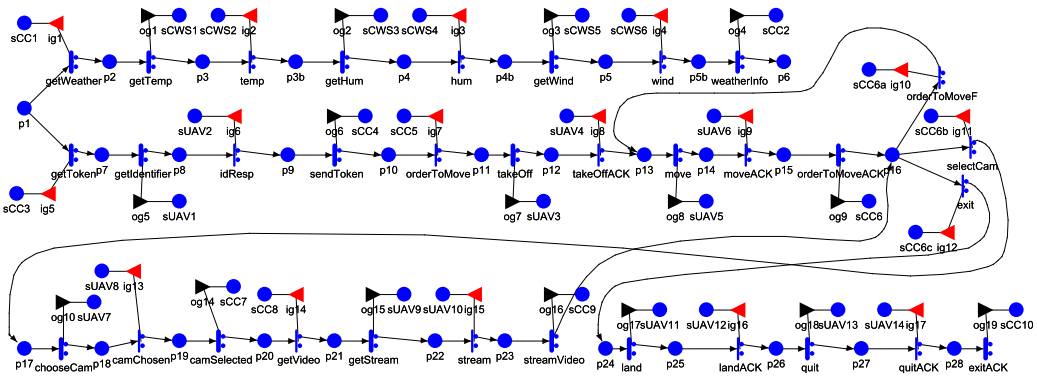

Figure 10: SAN model of the Connector

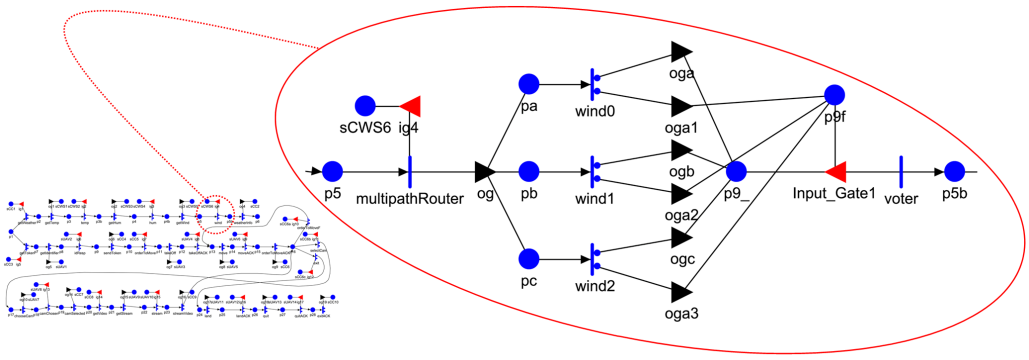

Figure 11: SAN CONNECTor model highlighting the portion where to integrate the Majority Voting mechanism (represented inside the big ellipse)

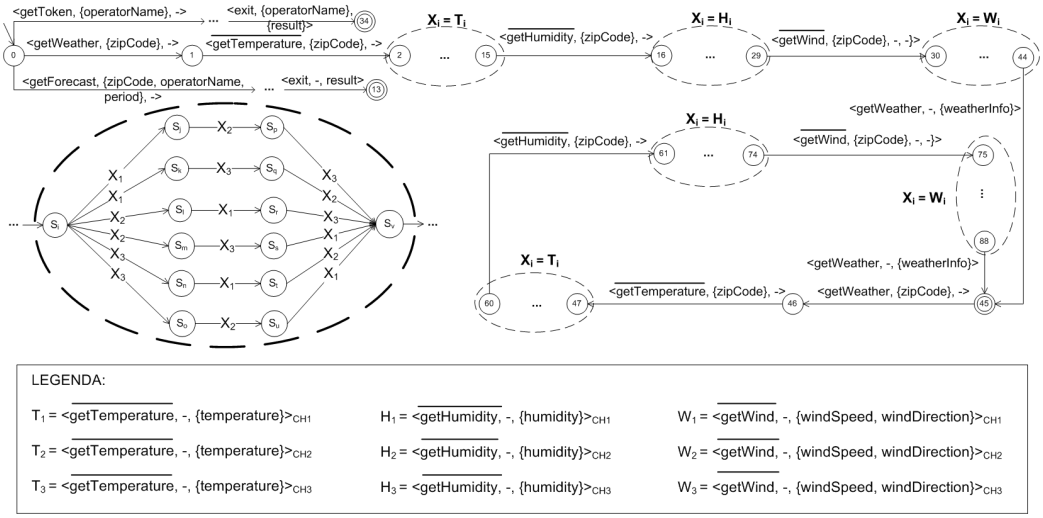

Figure 12: Enhanced CONNECTor model with majority voting mechanism. The enhancements are highlighted by the ellipses

mechanism is applied to: *(i)* $< \overline{getTemperature}, -, \{temperature\} >$ where $X_i = T_i$ meaning that $X_1 = T_1 = < \overline{getTemperature}, -, \{temperature\} >_{CH1}$, $X_2 = T_2 = < \overline{getTemperature}, -, \{temperature\} >_{CH2}$, and $X_3 = T_3 = < \overline{getTemperature}, -, \{temperature\} >_{CH3}$; *(ii)* $< \overline{getHumidity}, -, \{humidity\} >$, and *(iii)* $< \overline{getWind}, -, \{windSpeed, windDirection\} >$.

Figure 13 shows an example of mediated interaction between C2-A and UAV.

Analysis on the Forest Fire Case Study. Based on the above described scenario, we performed through Möbius [11] the analyses on the CONNECTed system in order to check if the synthesized CONNECTor satisfies the dependability and performance requirements (namely coverage and latency).

The first analysis is related to dependability issues and is intended to assess the measure of coverage, i.e., the percentage of data that the C2-A correctly receives from the Weather Service. The results obtained from the pre-deployment analysis are shown in Figure 14, where the trend of the coverage is plotted (on the y axis) at increasing values of failure probability of the communication channel (on the x axis), and the coverage threshold is shown at the value of 0.90 (i.e., the 90% of all the sent data is correctly received), as specified in the C2-A requirement. By observing the results in the figure, we note that the synthesized CONNECTor does not satisfy the dependability requirement for any value of failure probability. An enhancement is performed on the intermediate CONNECTor by properly including a *Majority Voting mechanism* [4] (as shown in Figure 11) and a new analysis is performed. The results obtained on the *enhanced model* are also shown

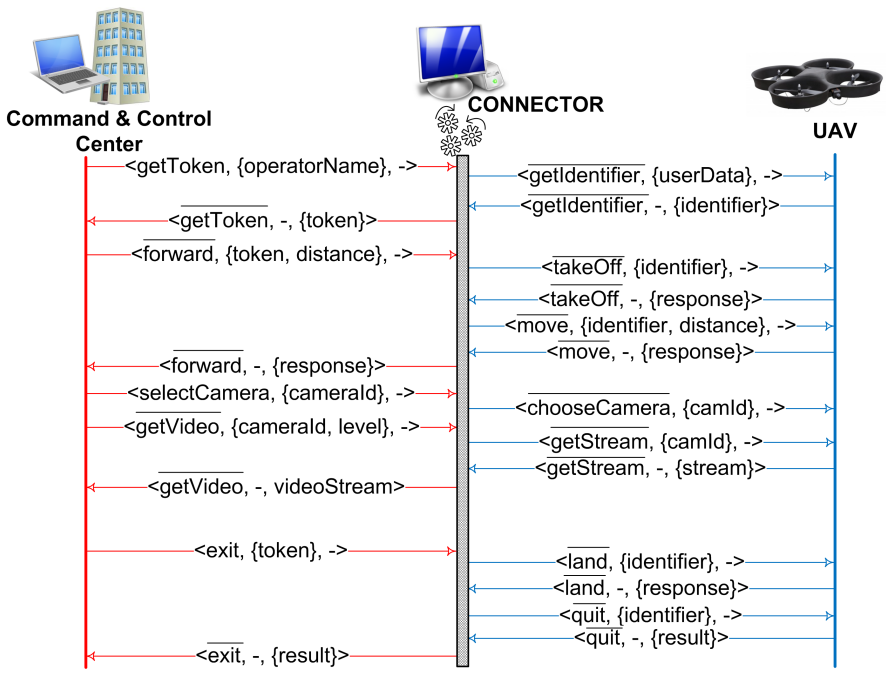

Figure 13: An interaction between C2-A and UAV

in Figure 14. Specifically, for values greater than 0 and less or equal than 0.2, the synthesized CONNECTor allows the CONNECTed system to satisfy the dependability requirement being over the threshold. For values of failure probability greater than 0.2 the CONNECTor does not satisfy the coverage requirement. Note that the use of this dependability mechanism allows to satisfy the coverage requirement when the failure probability is less than 0.2.

4.2. Our Approach @ Run-time

At *run-time* the applications and the synthesized mediator, also equipped with probes to monitor the CONNECTed system [5], are deployed and run on some devices. The probes, encoded into the CONNECTor, monitor the messages exchanged among the NSs involved into the communication and related to the metrics of interest. Figure 15 shows our approach at run-time on the case study.

The dynamism, evolvability, and poor knowledge characterizing the context we are considering, might cause a violation of the performance and dependability requirements at run-time. In our scenario, the Updater module of DEPER identifies a latency (performance) violation through the probes. By reasoning on the new systems specification, according to some internal predefined policies [12], DEPER selects the Probing mechanism. Allow-

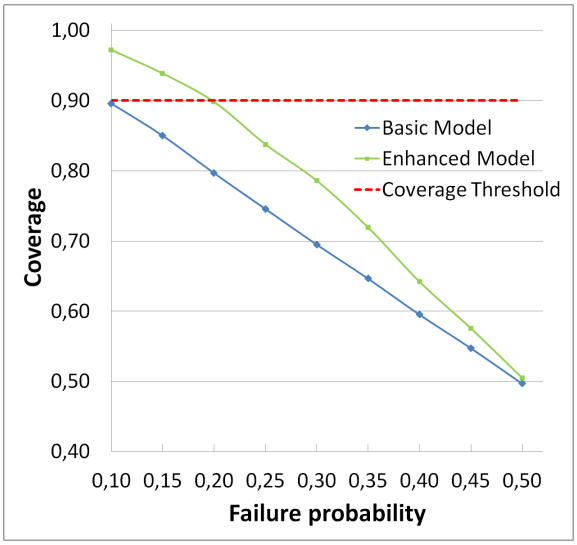

Figure 14: Coverage assessment as a function of failure probability of the communication channel

ing the CONNECTor to continue operating, the Builder module updates the model by including the selected enhancing mechanism and new analyses are triggered by the Analyser module to verify through new analysis on the enhanced CONNECTor whether it fulfills the dependability and performance requirements. When the employed mechanism is able to contrast the problem/violation, DEPER triggers the Synthesis by providing it the needed information to properly *enrich* the previously synthesized CONNECTor with the suggested mechanism, thus producing a new CONNECTor. It is important to observe that the time required for the analysis depends on the specific system under analysis, thus varying according to the extent of the problem. Details on timing with respect to the specific case study adopted are provide in Section 5. Interesting issues to be investigated as future work are the compositional synthesis of the CONNECTor enhancements and the run-time management of the system enhancement, i.e., how to safely pause and restart the CONNECTed system execution, while the overall system is running, in a way such that the intervention is as less intrusive as possible with respect to the ongoing interaction.

Analysis on the Forest Fire Case Study. Considering our case study, the analysis, performed by DEPER, returns feedback about the latency requirement that is not satisfied, and hence an enhancement is needed also at this stage. We recall that the measure of latency we are focusing on, is related

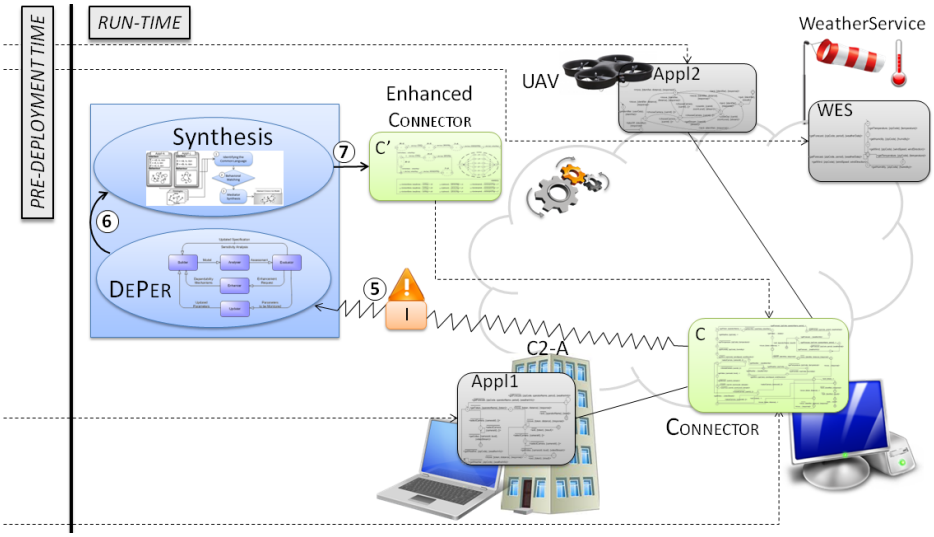

Figure 15: Our Approach at Run-time

to the communication efficiency of the channels between the CONNECTor and the networked systems.

Specifically, based on the given definition of latency, and on the characteristics of the scenario, we have to highlight that the first movement order of C2-A is performed by the UAV after the *takeOff* operations, thus increasing the latency.

Once the Updater module updated the model with real value of the CONNECTed system executions, as gathered through the monitoring system, a new analysis is needed at run-time. Figure 16 shows the results obtained through the analyses on latency at varying the transmission rate of the communication channel between 0.1 and 1 (on the x axis).

The analysis results show that the requirement, set to the value 5 *time units*, is never satisfied during the take off phase, while during the flight phase it is not met for values of distribution rate less than 0.5. Through a generic strategy for automating the selection of an appropriate dependability and performance mechanism, which is fully detailed in [12], we identify that candidate mechanism to improve the CONNECTed system with respect to the timing constraints, is the *Probing* mechanism (briefly described in Section 2).

The results obtained when employing the mechanism are also shown in

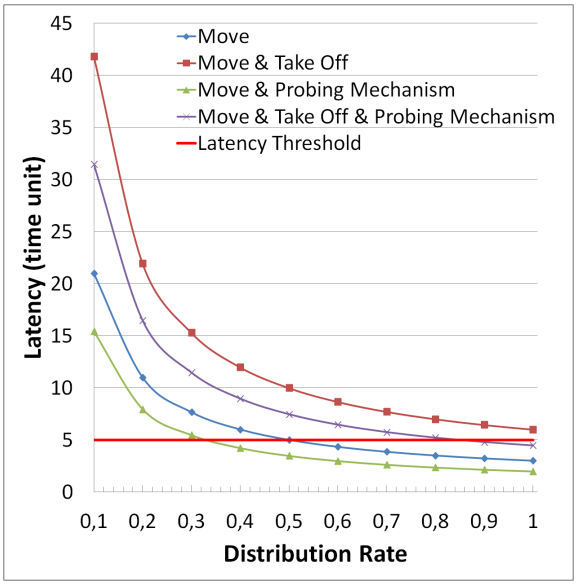

Figure 16: Latency assessment at varying of the rate of the distribution

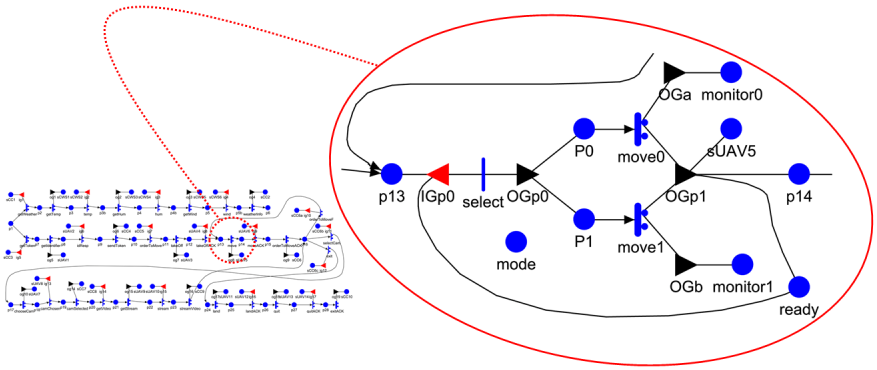

Figure 17: SAN CONNECTor model highlighting the portion where to integrate the Probing mechanism (represented inside the big ellipse)

Figure 16. We note that, for values of distribution rate greater than 0.8, the CONNECTed system is able to satisfy the requirement also during the take off phase. Figure 17 shows the SAN model of the CONNECTor with the zoom where *Probing* has been employed. Similarly, the eLTS of the CONNECTor is enhanced by Synthesis through the mechanism indicated by DEPER, as already shown for the pre-deployment case in Section 4.1.

5. Discussion

The architectural structure described in this paper constitutes an important step towards the definition of an automated procedure to provide dependability and performance analysis, as a support for the synthesis of dependable connectors, and to implement the feedbacks obtained from the analysis into an enhanced synthesized connector meeting functional and non functional requirements. The adaptation process, realized through the synergic cooperation between Synthesis and DEPER, that allows continuous adaptation of the synthesized connector during its lifetime, has been defined and detailed through a case study in the GMES context. However, there are various aspects that still need to be investigated to fully reach the ambitious goal of automated cooperation between connector's synthesis and analysis, especially at run-time.

Among the open issues that need further investigations we mention: analysis optimization, compositionality of the proposed solutions, and scalability aspects.

1. How to optimise the analysis in order to respect time constraints. The definition and solution of dependability and performance models may become a significant time consuming process depending on the complexity of the NSs and applications involved. For the case study involving the UAV and integrated video camera, the analyses performed by DePer took around 25 seconds on a Quad core processor like Intel Core i7-3770T, and even less for the Weather service case, since the resulting model was smaller. Given that the synthesis of the Connector is an on-the-fly activity, which is performed once the interoperability willingness has been manifested by one or more NSs, efficient methods need to be put in place to make the Connect support infrastructure valuable and applicable in practice. Approaches relying on compositional modeling as well as compositional solution methods would therefore be highly desirable. Currently, the DePer prototype is not equipped with such optimization features and, when at run-time violation of non functional properties is revealed, a new analysis is performed by Analyser with the updated model parameters as re-determined through the on-line observations. An improvement, which seems to be easily accommodable in the current implementation, would be to have parametric timing and reliability expressions attached to each dependability mechanism model, to be directly used in the Connected model as compact parameter values instead of enriching the Connector by including the whole mechanism model. Specularly, also optimisation of the synthesis process would be highly recommended.

2. Compositional solution methods for both the dependability and performance model and the synthesis process, when the Connector is derived as specialisation of an already synthesized and analyzed Connector. This would be beneficial for the whole chain between DePer and Synthesis, by properly reusing and adapting template models, thus gaining on efficiency of all the service chain supporting heterogeneous and dynamically evolving interoperability.

3. Scalability aspects. Both synthesis and analysis do not have limitations to address Connected systems made up of a potentially huge number of NSs, e.g., in the order of hundreds or thousands, unless those limitations characterizing the model-based stochastic analysis and synthesis domains (e.g., due to explosion of the states until resolution is no more affordable). Of course, scaling the number of components involved is paid by a higher complexity of the process and time to produce a suitable Connector. Improving on optimization features surely goes in the direction of favoring scalability

at a more affordable price.

Given the extremely high challenges we had to face, we have approached our research in steps of increasing degree of complexity, and the studies performed so far were primarily directed to consolidate a feasibility study of the proposed approach, without focusing deeply on efficiency aspects. This is a strong future exploration area. Towards this direction, we will finalize the implementation of the overall approach, including the actions towards all the actors involved in the CONNECTed system in execution. The point is that, when a violation on non functional properties is detected at run-time, actions need to be taken about the ongoing executions of the CONNECTed system. As already discussed in the previous section describing our approach at run-time, DEPER gathers on-line information to update and refine the analyses performed at pre-deployment time, to cope with degrees of uncertainties about, and evolution of, the NSs behaviors. During the period of data acquisition from monitoring and internal processing of the gathered data, possibly triggering a new analysis, executions of CONNECTed system instances involving the CONNECTor under observation are carried on. As soon as conditions change leading to violation of required properties and a new analysis establishes that adaptation of the currently used CONNECTor is required through its enhancement by a dependability mechanism, or that no enhancement is possible through the available options, the CONNECTor in place needs to be stopped and adjustments at synthesis level is required (a complete new synthesis could be necessary). After deployment, the adapted (or newly synthesized) CONNECTor becomes operative again. From this description, interaction and integration with other units of the CONNECT environment need to be addressed, as well as implementation details to realize the involved steps.

6. Related Work

A big effort has been devoted in the literature to the investigation of the interoperability problem in the form of supervisory control synthesis [15], discrete controller synthesis [16], component adaptors [17], protocol conversion [18], [19], [20], converter synthesis [21] to mention few.

The theory of mediator, on which we build upon, is closely related to the seminal paper by Yellin and Strom on protocol adaptor synthesis [22]. They propose an adaptor theory to characterize and solve the interoperability problem of augmented interfaces of applications. The authors formally

define the checks of applications compatibility and the concept of adapters. Furthermore, they provide a theory for the automated generation of adapters based on interface mapping rules, which is related to our common language of protocols found through the domain ontology.

In more recent years an increasing attention has been paid in the Web Service area where many works are related to our synthesis of mediators. Among them, papers [23, 24] propose a formal model to describe services and adapters and to automatically generate adapters. The work [25] presents an approach to specify and synthesize adapters based on domain-specific transformation rules and by using existing controller synthesis algorithms implemented in the Marlene tool[4]. The paper [26] on behavioral adaptation proposes a matching approach based on heuristic algorithms to match services for the adapter generation taking into account both the interfaces and the behavioral descriptions. Moreover, the Web services community has been also investigating how to actually support service substitution to enable interoperability with different implementations of a service (e.g., due to evolution or provision by different vendors). Based on the assumption that providing semantic annotations through the use of ontologies can facilitate the utilization of services more accurately, the authors in [27], aim at highlighting the negotiation process between agents and Web services by introducing a middleware based approach along with the ontologies to describe the semantic annotations thus resulting in the automation of service discovery to service invocation. Our mediator synthesis work relates, for instance, to [28] by sharing the exploitation of ontology to reason about interface mapping and the synthesis according to such mapping. Their approach can be seen as an instance of ours.

In recent years, the CONNECT EU project[5] aimed to allow seamless interoperability between heterogeneous protocols at various levels. The project adopted an approach for the on the fly synthesis of *emergent* CONNECTors [1] via which Networked Systems (NSs) communicate. The emergent CONNECTors (or mediators) are system entities synthesized according to the behavioral semantics of protocols run by the interacting parties at application and middleware layers. The synthesis process is based on a formal foundation for CONNECTors, which allows learning of NSs, reasoning about NSs and

[4] `http://service-technology.org/tools/marlene`

[5] CONNECT Web Site - `http://connect-forever.eu/`

adapting the interaction behavior of NSs at run-time through CONNECTors.

We conducted preliminary studies about functional and non functional interoperability in conjunction and preliminary results are presented in [29], [5], and [30].

In particular, papers [29] and [5] mainly focus on understanding the needed collaborations among different systems - in order to properly realise that. Specifically: the work in [29] briefly sketches an embryonic idea of combining functional and non functional interoperability; this idea is realized in this paper while [29] focuses on the collaboration between the synthesis approach and a monitoring system.
The work in [5] presents first thoughts on the interplay among the synthesis approach, a monitoring system, and the dependability and performance analysis system. This paper deeply investigates the interplay between the synthesis approach and the dependability analysis and proposes concrete solutions.
Paper [30], instead, proposes an automated solution for the synthesis of self-adaptive connectors. Specifically: the work in [30] focuses on user performance requirements: the synthesised connector is self-adaptive with respect to run-time changes in the user performance requirements. This paper is significantly different from the one presented in [30]: the work in [30] prunes the connector in order to satisfy user performance requirement changes, while this paper takes into account completely different non functional issues and enhances the *functional connector* with proper mechanisms.

Several works have been done in recent years that relate to our connector adaptation process with respect to performance and dependability aspects proposed in this paper. For instance, paper [31] describes a method for the specification of self-adaptive software systems using a UML based modeling language where at design time is possible to check properties like adaptation rule set stability and deadlock freedom.

Concerning combined approaches taking into account both functional and non functional issues, we can mention papers [32], [33], and [34]. This latter proposes an approach to automatically derive adaptors in order to assemble correct by construction real-time systems from COTS. The approach takes into account interaction protocols, timing information, and QoS constraints to prevent deadlocks and unbounded buffers. The synthesized adaptor is then a component that mediates the interaction between the components it supervises, in order to harmonize their communication. The purpose of our approach is similar to that in [34] since we both aim at synthesizing a me-

diator reconciling protocols, but our setting is quite different with respect to their. Indeed, our focus is mainly on solving protocols discrepancies to allow protocols synchronization also satisfying performance and dependability requirements, while they focus more on timing and deadlock issues while composing COTS real-time components.

Spitznagel and Garlan in their work [33] present an approach to formally specify connector wrappers as protocol transformations, modularizing them, and reasoning about their properties, with the aim to resolve component mismatches. In their vision a wrapper is new code interposed between component interfaces and communication mechanisms and its intended effect is to moderate the behavior of the component in a way that is transparent to the component or the interaction mechanism. Instead, our connector is a wrapper that addresses issues related to communication and compatibility including things such as changing the way data is represented during communication, the protocols of interaction, the number of parties that participate in the interaction, and the kind of communication support that is offered for monitoring, error handling, security, and so on. Their approach is to formally specify connector wrappers by means of a process algebra as a set of parallel process (one for each connector's interface and one for the glue). Protocol transformations may include redirecting, recording and replaying, inserting, replacing, and discarding particular events yielding benefits like composability and reusability.

Moreover Spitznagel in her Ph.D. thesis [35] illustrates a set of patterns of basic connector's transformations, i.e., enhancements. She also shows how these patterns can be compositionally applied to simple connectors in order to produce a number of more complex connectors. In particular she shows how to select and to apply such transformations in the domain of dependability and proposes a prototypal tool to semi-automatically derive new connectors as enhancements.

Another architectural approach is presented in [36] that proposes an abstraction based on exception handling for structuring fault-tolerant software systems. The aim of the work is to tolerate, during run time, architectural mismatches deriving from the integration of untrusted software components that were not originally designed to interact with each other.

Stochastic model-based approaches for quantitative analysis of performance and dependability aspects have been largely developed along the last decades and documented in a large literature review on this relevant issue. A survey of the most popular ones can be find in [37, 38]. Commonly, the

choice of the most appropriate model, to be used for this purpose, depends on several factors including the complexity of the system to be analyzed, the measures, the attributes and the measures to be evaluated, the accuracy required, and the resources available for the study. In this work the performance and dependability model of the CONNECTed system is specified with Stochastic Activity Networks (SANs), a generalization of Stochastic Petri Nets introduced and define in [10, 39].

In the last decade, a large number of studies addressed the problem of automated dependability analysis through the transformation of models. Automatic/automated methods from system specification languages to modeling languages amenable to perform dependability analysis has been recognised as an important support for improving the quality of systems. The paper [40] explores the state-of-the-art in engineering self-adaptive systems and identify potential improvements in the design process, pointing out that in designing self-adaptive systems, the feedback loops that control self-adaptation must become first-class entities. In [12] the authors introduce an approach for automating the identification of system's elements to be reinforced, the selection of dependability and performance mechanisms, and how to apply them on a model of the system, in order to meet given non-functional requirements. In [41] the authors present a development of an integrated environment to support the early phases of system design, where design tools based on the UML are augmented with transformation-based validation and analysis techniques. The work presented in [42] propose a method that provides an automatic and compositional means for predicting reliability of a service oriented application based on its architecture specification. The authors in [43] use a multi-formalism approach to model complex systems, by composing and integrating sub-models defined by different modeling formalisms. The work presents a particular workflow to improve the efficiency of the solution process by automatizing the process to be enforced to solve a complex model. The authors of [44] propose a workflow for the automatic assembly of large stochastic models, based on template model libraries and composition patterns. The approach is based on a Template Models Description Language (TMDL) useful to define in formal way libraries of template models and then to apply them to different system configurations and generate the related analysis model.

A modeling framework allowing the generation of dependability-oriented analytical models from AADL (Architecture Analysis and Design Language) models is described in [45].

Several tools have been developed to support the definition of model-based transformations. The Viatra tool [46] automatically checks consistency, completeness, and dependability requirements of systems designed using UML. The Genet tool [47] allows the derivation of a general Petri net from a state-based representation of a system. The ADAPT Tool supports model transformations from AADL Architectural Models to Stochastic Petri Nets [48]. However, in terms of enhancing the model-transformation environment with template models of basic fault tolerance mechanisms to allow automated assessment of enhanced, fault tolerant designs, it seems a rather new research direction.

7. Conclusions

Diversity characterizing heterogeneous systems dynamically available in the networked environment is a richness. Being able to gain from it, requires to be able to cope with interoperability problems without a-priori knowledge of the systems, and with a degree of flexibility. We already proposed as solution an approach to the automated synthesis of CONNECTors, or mediators, between heterogeneous Networked Systems for their *functional interoperability* at application layer.

In this paper, we presented a new approach for the synthesis of connectors addressing functional and (some) non-functional interoperability in conjunction. The proposed solution includes the following contributions: an approach to *enhance* the *functional* CONNECT*ors* taking into account performance and dependability aspects, a CONNECT*or adaptation process*, related to the performance and dependability mechanisms, spanning pre-deployment time and run-time, and a stochastic model-based *implementation* of the performance and dependability analysis. In addition, we implemented, analysed, and critically disussed a case study.

Future investigations we plan to do concern: the study of techniques to apply on the system at run-time when a violation is detected; a more extensive validation of the approach on a number of case studies to precisely outline the typology of systems and problems we are able to manage automatically; a complete implementation of the overall approach.

[1] P. Inverardi, V. Issarny, R. Spalazzese, A theory of mediators for eternal connectors, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification, and Validation, Vol. 6416 of Lecture

Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 236–250.

[2] P. Inverardi, R. Spalazzese, M. Tivoli, Application-Layer Connector Synthesis, in: Formal Methods for Eternal Networked Software Systems, Vol. 6659, 2011, pp. 148–190.

[3] CONNECT Consortium, Deliverable 5.4 – Finalised dependability framework and evaluation results, 2012.

[4] P. Masci, N. Nostro, F. Di Giandomenico, On enabling dependability assurance in heterogeneous networks through automated model-based analysis, in: Proceedings of the 3rd International Workshop SERENE, Lecture Notes in Computer Science, Springer Berlin Heidelberg, Geneva, CH, 2011, pp. 78–92.

[5] A. Bertolino, A. Calabrò, F. Di Giandomenico, N. Nostro, P. Inverardi, R. Spalazzese, On-the-fly dependable mediation between heterogeneous networked systems, in: M. Escalona, J. Cordeiro, B. Shishkov (Eds.), Software and Data Technologies, Vol. 303 of Communications in Computer and Information Science, ICSOFT 2011, Springer Berlin Heidelberg, 2013, pp. 20–37.

[6] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The Description Logic Handbook, Cambridge University Press, 2003.

[7] R. Spalazzese, P. Inverardi, Mediating connector patterns for components interoperability, in: Proceedings of ECSA 2010, LNCS 6285, 2010, pp. 335–343. doi:10.1007/978-3-642-15114-9_26.

[8] R. Spalazzese, P. Inverardi, Mediating connector patterns for components interoperability, in: M. Babar, I. Gorton (Eds.), Software Architecture, Vol. 6285 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 335–343.

[9] A. Bertolino, A. Calabrò, F. Di Giandomenico, N. Nostro, Dependability and performance assessment of dynamic connected systems, in: Formal Methods for Eternal Networked Software Systems, Vol. 6659, 2011, pp. 350 – 392.

[10] W. Sanders, J. Meyer, Stochastic activity networks: Formal definitions and concepts, in: E. Brinksma, H. Hermanns, J.-P. Katoen (Eds.), Lectures on Formal Methods and PerformanceAnalysis, Vol. 2090 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 315–343.

[11] D. Daly, D. Deavours, J. Doyle, P. Webster, W. Sanders, Möbius: An extensible tool for performance and dependability modeling, in: 11th International. Conference TOOLS'00, Vol. 1786 of LNCS, Springer Verlag, 2000, pp. 332–336.

[12] F. Di Giandomenico, M. L. Itria, P. M. Masci, N. Nostro, Automated synthesis of dependable mediators for heterogeneous interoperable systems, Reliability Engineering & System Safety 132 (0) (2014) 220 – 232.

[13] F. D. Giandomenico, A. Bertolino, A. Calabrò, N. Nostro, An approach to adaptive dependability assessment in dynamic and evolving connected systems, IJARAS 4 (1) (2013) 1–25.

[14] CONNECT Consortium, Deliverable 6.3– Experiment Scenarios (2012).

[15] B. Brandin, W. Wonham, Supervisory control of timed discrete-event systems, IEEE Transactions on Automatic Control 39 (2) (1994) 329–342.

[16] P. J. Ramadge, W. M. Wonham, Supervisory Control of a Class of Discrete Event Processes, SIAM Journal on Control and Optimization 25 (1) (1987) 206–230.

[17] A. Bracciali, A. Brogi, C. Canal, A Formal Approach to Component Adaptation, Journal of Systems and Software 74 (1) (2005) 45–54.

[18] K. L. Calvert, S. S. Lam, Formal Methods for Protocol Conversion, IEEE Journal on Selected Areas in Communications 8 (1) (1990) 127–142.

[19] S. S. Lam, Correction to 'Protocol Conversion', IEEE Transaction on Software Engineering 14 (9) (1988) 1376–1376.

[20] K. Okumura, A Formal Protocol Conversion Method, in: Proceedings of the ACM SIGCOMM Conference on Communications Architectures & Protocols, SIGCOMM '86, ACM, New York, NY, USA, 1986, pp. 30–37.

[21] R. Passerone, L. de Alfaro, T. Henzinger, A. Sangiovanni-Vincentelli, Convertibility verification and converter synthesis: two faces of the same coin, ICCAD'02, 2002, pp. 132–139.

[22] D. M. Yellin, R. E. Strom, Protocol Specifications and Component Adaptors, ACM Transaction on Programming Languages and Systems (TOPLAS) 19 (2) (1997) 292–333.

[23] J. min Jiang, S. Zhang, P. Gong, Z. Hong, Message dependency-based adaptation of services, in: Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific, 2011, pp. 442–449.

[24] J. min Jiang, S. Zhang, P. Gong, Z. Hong, Service Adaptation at Message Level, in: 2011 IEEE World Congress on Services (SERVICES), 2011, pp. 87–88.

[25] C. Gierds, A. Mooij, K. Wolf, Reducing Adapter Synthesis to Controller Synthesis, IEEE Transactions on Services Computing 5 (1) (2012) 72–85.

[26] H. R. M. Nezhad, G. Y. Xu, B. Benatallah, Protocol-aware matching of web service interfaces for adapter development, in: Proc. of the 19th International conference on World Wide Web, WWW'10, pp. 731–740.

[27] M. Pasha, F. Oquendo, H. Ahmad, Negotiation in heterogeneous environments, in: Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, 2010, pp. 290–295. doi:10.1109/ITNG.2010.225.

[28] L. Cavallaro, E. Di Nitto, M. Pradella, An automatic approach to enable replacement of conversational services, in: L. Baresi, C.-H. Chi, J. Suzuki (Eds.), Service-Oriented Computing, Vol. 5900 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 159–174.

[29] A. Bertolino, P. Inverardi, V. Issarny, A. Sabetta, R. Spalazzese, On-the-fly interoperability through automated mediator synthesis and monitoring, in: ISoLA 2010, Part II, LNCS 6416,, Springer, Heidelberg, 2010, pp. 251–262.

[30] A. Di Marco, P. Inverardi, R. Spalazzese, Synthesizing self-adaptive connectors meeting functional and performance concerns, in: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 133–142.
URL http://dl.acm.org/citation.cfm?id=2487336.2487358

[31] M. Luckey, G. Engels, High-quality specification of self-adaptive software systems, in: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 143–152.
URL http://dl.acm.org/citation.cfm?id=2487336.2487359

[32] Z. J. Oster, G. R. Santhanam, S. Basu, Identifying Optimal Composite Services by Decomposing the Service Composition Problem, in: Proceedings of the 2011 IEEE International Conference on Web Services, ICWS '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 267–274.

[33] B. Spitznagel, D. Garlan, A compositional formalization of connector wrappers, in: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 374–384.

[34] M. Tivoli, P. Fradet, A. Girault, G. Goessler, Adaptor synthesis for real-time components, in: O. Grumberg, M. Huth (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Vol. 4424 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 185–200.

[35] B. Spitznagel, Compositional transformation of software connectors, Ph.D. thesis, Carnegie Mellon University (May 2004).

[36] P. Brito, R. de Lemos, C. Rubira, E. Martins, Architecting fault tolerance with exception handling: Verification and validation, J. Comput. Sci. Technol. 24 (2) (2009) 212–237.

[37] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, Model-based evaluation as a support to the design of dependable systems, in: Dependable

Computing Systems: Paradigms, Performance Issues, and Applications, Wiley, 2005, pp. 57–86.

[38] D. M. Nicol, W. H. Sanders, K. S. Trivedi, Model-based evaluation: from dependability to security, IEEE Transaction on Dependable and Secure Computing 1 (1) (2004) 48–65.

[39] A. Movaghar, J. Meyer, Performability modelling with stochastic activity networks, in: Real-Time Systems Symposium, Austin, TX, 1984, pp. 215–224.

[40] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, M. Shaw, Engineering self-adaptive systems through feedback loops, in: B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee (Eds.), Software Engineering for Self-Adaptive Systems, Vol. 5525 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 48–70.

[41] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, G. Savoia, Dependability Analysis in the Early Phases of UML Based System Design, Journal of Computer Systems Science and Engineering 16 (5) (2001) 265–275.

[42] R. Mirandola, P. Potena, E. Riccobene, P. Scandurra, A reliability model for service component architectures, Journal of Systems and Software 89 (0) (2014) 109 – 127.

[43] F. Moscato, V. Vittorini, F. Amato, A. Mazzeo, N. Mazzocca, Solution Workflows for Model-Based Analysis of Complex Systems, IEEE Transactions on Automation Science and Engineering 9 (1) (2012) 83–95.

[44] L. Montecchi, P. Lollini, A. Bondavalli, A DSL-Supported Workflow for the Automated Assembly of Large Stochastic Models, in: 2014 Tenth European Dependable Computing Conference (EDCC), 2014, pp. 82–93.

[45] A. Rugina, K. Kanoun, M. Kaniche, A system dependability modeling framework using aadl and gspns, in: Architecting Dependable Systems IV, Vol. 4615 of LNCS, Springer Berlin Heidelberg, 2007, pp. 14–38.

[46] G. Csertan, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza, D. Varro, VIATRA - Visual Automated Transformations for Formal Verification and Validation of UML Models, in: ASE'02, 2002, pp. 267–270.

[47] J. Carmona, J. Cortadella, M. Kishinevsky, Genet: A tool for the synthesis and mining of petri nets, in: ACSD '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 181–185.

[48] A. Rugina, K. Kanoun, M. Kaâniche, The ADAPT tool: From AADL architectural models to stochastic petri nets through model transformation, in: EDCC-7 2008, Kaunas, Lithuania, 7-9 May 2008, IEEE Computer Society, 2008, pp. 85–90.