VALIDATION AND VERIFICATION POLICIES FOR GOVERNANCE OF SERVICE CHOREOGRAPHIES

Guglielmo De Angelis[1], Antonia Bertolino[1] and Andrea Polini[2]
[1]*CNR– ISTI, Via Moruzzi 1, 56124 Pisa, Italy*
[2]*Computer Science Division, School of Science and Technologies, University of Camerino, 62032 Camerino, Italy*

Keywords: Service Choreography, SOA Governance, V&V Policies, On-line Testing, Ranking, Reputation, Service Oriented Architecture.

Abstract: The Future Internet (FI) sustains the emerging vision of a software ecosystem in which pieces of software, developed, owned and run by different organizations, can be dynamically discovered and bound to each other so to readily start to interact. Nevertheless, without suitable mechanisms, paradigms and tools, this ecosystem is at risk of tending towards chaos. Indeed the take off of FI passes through the introduction of paradigms and tools permitting to establish some discipline. Choreography specifications and Governance are two different proposals which can contribute to such a vision, by permitting to define rules and functioning agreements both at the technical level and at the social (among organizations) level. In this paper we discuss such aspects and introduce a policy framework so to support a FI ecosystem in which V&V activities are controlled and perpetually run so to contribute to the quality and trustworthiness perceived by all the involved stakeholders.

1 INTRODUCTION

The Future Internet (FI) is changing the way software is conceived, implemented and used: many research projects are undertaken from all over the world (Pan et al., 2011) to re-design the FI architecture so to address the various emerging needs for trustworthiness, security, mobility, scale, flexibility, content distribution, adaptation, and so on. As highlighted in (Pan et al., 2011), the FI cannot be achieved by "assembling different cleanslate solutions targeting different aspects", but a coordinated comprehensive approach addressing new design principles is required.

Still today, by reading the reports and roadmaps (e.g., (Pollak, 2006; Fiedler et al., 2011)) produced by such widespread research effort, it is evident that analysts and experts do not yet agree on a common vision for, or are not yet able to precisely predict, the shape of the next-coming FI. Nevertheless, from the above mentioned documents some underlying commonalities can be already identified. One of such common thinkings concerns recognizing the growing importance in the engineering of the FI architecture of *social* aspects over those merely *technical* (Fiedler et al., 2011), due to the inherent inter-organization nature of the FI. In particular, we claim that one of the biggest challenges concerns the capability of introducing some form of discipline in new forms of collaborative software development and management, so to permit the effective integration and definition of independently developed service-based applications, as foreseen in the Service-oriented Architecture (SOA) paradigm. Thus, we need to conceive methodologies and approaches that will permit the smooth integration of independently developed pieces of software in order to derive and provide users with more complex composite services, according to their changing requests.

In this line two main proposals can be identified. The first one concerns the introduction of application-level global protocols that service providers can take as a reference in the development of their own services. These specifications, typically defined and managed by third party organizations, are generally referred to as **Choreographies**. Specifically a service choreography is a description of the *peer-to-peer* externally observable interactions that cooperating services should put in place (Barker et al., 2009). Such multi-party collaboration model focuses on message exchange, and does not rely on a central coordination of the described activities. Therefore, service choreographies can be one of the right instruments to solve "technical and social" issues posed by FI, as they model a higher level of abstraction with respect

Angelis G., Bertolino A. and Polini A..
VALIDATION AND VERIFICATION POLICIES FOR GOVERNANCE OF SERVICE CHOREOGRAPHIES.
DOI: 10.5220/0003936200580070
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 58-70
ISBN: 978-989-8565-08-2

to those of services.

The second proposal is the introduction of a clear and explicit SOA governance (Woolf, 2007). Generally speaking, governance is the act of governing or administrating, and refers to all measures, rules, decision-making, information, and enforcement that ensure the proper functioning and control of the governed system. SOA governance addresses specific aspects of SOA life-cycle (Woolf, 2007). Specifically, SOA governance approaches include: SOA policy management and enforcement; registry and metadata management; data collection for statistical and Key Performance Indicators; monitoring and management; application and service life cycle management.

Choreographies and SOA governance are also needed in order to support the progressive moving of traditional Software Engineering activities from the only pre-runtime phases towards "also at runtime" (Trossen et al., 2009), i.e., during normal operating of deployed services. In this line, some authors (Greiler et al., 2010; Bertolino et al., 2012) elaborated in particular on the possibility, as well as the opportunity, of extending to on-line the SOA testing activities, also based on appropriate governance approaches for V&V.

In this paper, we continue the investigation on such topic proposing a V&V policy framework aiming at sustaining the governance of on-line V&V. Policies are at the heart of governance: they define the rules according to which systems and their owners should behave to ensure the mutual understanding, interaction and cooperation. More specifically, the concept of a policy has been introduced as representing some constraint or condition on describing, deploying, and using some service (MacKenzie et al., 2006). When such constraints or conditions are agreed between two or more parties, they become a *contract*. Indeed, SOA governance asks for the definition of policies and supporting tools to take into account the social and runtime aspect of FI above highlighted. It is worth noting that the framework we propose mainly focuses on V&V aspects, nevertheless a FI infrastructure will have to include mechanisms for governing all the other aspects related to service and choreography lifecycle. At the same time policies and mechanisms we propose should not be considered exhaustive. Instead they permit to define a "border" in which a set of rules need to be observed and monitored. It is certainly possible to imagine that in a given context the defined policies and rules to observe are constituted by a different set.

In this paper the focus is mainly on V&V policies on which governance of choreography is based, specifically to support V&V on-line activities. Of course, these policies cover only part of the complex task of choreography governance, and other policies are needed to support overall choreography management. The rest of the paper is organized as follows: Section 2 introduces some background material in particular with reference to on-line testing and SOA governance, then Section 3 introduces a classification for policies enabling V&V activities. Successively in the following sections 4, 5 and 6 we respectively illustrate policies for activation, rating and ranking both of services and choreographies. In Section 7 we introduce some policies related to Choreography enactment and in Section 8 policies for test case selection are discussed. Then the paper closes with Section 9 in which policies are illustrated with real examples and Section 10 in which we draw some conclusions and opportunities for future work.

2 BACKGROUND

The motivation behind V&V Policies for Service Choreographies relies on two main concepts: On-line testing (Section 2.1) and SOA Governance for V&V activities (Section 2.2).

2.1 Testing in the SOA Setting

Due to the high dynamism and the multi-organizational characteristics of SOA, the traditional testing process is not anymore adequate to guarantee acceptable levels of reliability and trust. Specifically, during the last years, various research works (Greiler et al., 2010; Bertolino et al., 2012; Ghezzi, 2011) argued in favor of extending testing activities from the laboratory to the field, i.e., towards run-time.

This is in line with the emerging trend of a surrounding open world (Baresi et al., 2006), in which software-intensive systems are more and more pervasive and evolve at unprecedented pace, the boundaries between software and the external world become fuzzy, and no single organization is anymore in control of a whole system. In the future, SOA development is going to be decentralized and multi-supplier, and services are going to be pervasive, dynamic, autonomic, multi-tenant. Quoting (Baresi et al., 2006), developers recognize that they *must shift from the prevailing synchronous approach to distributed programming to a fundamentally more delay-tolerant and failure resilient asynchronous programming approach. Global behaviors emerge by asynchronous combinations of individual behaviors, and bindings and compositions change dynamically*.

In such a context, where it will be the norm that

services that are "stranger" to each other dynamically connect and collaborate, how can the behavior of a SOA system be validated?

SOA on-line testing foresees the proactive launching of selected test cases that evaluate the real system within the services' real execution environment. By extending testing from the laboratory to the field during normal operation, on-line testing can help in the timely detection of unrevealed inconsistencies as well as of functional and non-functional failures. For example, as shown in (Bertolino et al., 2012), on-line testing can be used to assess an access control system's resilience. In this case, test cases consist of access requests under different roles and to different resources. If the system grants unauthorized access, the tester will report a security breach in the audit log; if the system denies a supposedly authorized access, the tester will report this as well, permitting a quick and proactive solution of problems that might affect legitimate users. An on-line testing session can occur periodically or after some relevant event, according to the established SOA test governance process.

Choreography-based applications yield distinguishing characteristics such as adaptiveness, dynamic execution environment, limited controllability on the execution environment, and late service binding. All such characteristics demand new testing strategies both for development time and for run-time. In this paper we focus on on-line testing of choreographies.

2.2 SOA Governance for V&V Activities

In the introduction we anticipated that two means to address FI challenges are choreographies and SOA governance.

SOA governance is meant as a set of best practices and policies for ensuring interoperability, adequacy, and reuse of services over several different platforms (Woolf, 2007) and among different organizations. Both design-time and run-time governance are usually considered in SOA governance solutions.

Achieving SOA governance is a complex and articulated task that involves many different and orthogonal domains (i.e., technical, legal, standards, internal procedures). In the most abstract formulation, SOA governance is realized through a cycle consisting of: policy definition, auditing and monitoring, and finally evaluation and validation.

As people would formulate decisions in terms of policies (e.g., design policies, development policies, run-time policies) that would be audited, monitored, and validated, it is necessary to clearly state the scope of their activities, and their responsibilities. With respect to V&V activities for choreographies, we identify the following main actors: the *Choreography Designer*, the *Service Provider*, the *Service Consumer*, and the *V&V Manager*.

Briefly, the Choreography Designer is the responsible of a given choreography specification. For each choreography specification, he/she may foresee a set of policy templates that actors playing within the choreography have to abide by. The Service Provider roughly represents both the developer, and the maintainer of a set of services; while the Service Consumer discovers and then uses such services by querying a repository/registry. Both the Service Consumer, and the services offered by a Service Provider would play a role in a given choreography specification.

In this scenario, the V&V Manager is responsible for the definition and the implementation of the V&V strategy within the established governance setting. For example the V&V Manager is in charge of: the selection of a suitable test strategy (e.g., driven by the choreography specification); the analysis and the management of the impact of the V&V results on both the choreography and the service life-cycle; the definition of the V&V policies that will be applied in the governance settings (e.g., how often V&V activities are performed or after which events).

In the following we present two relevant scenarios for that area of the SOA Governance regulating V&V activities. Specifically, each scenario describes both the respective responsibilities of the actors introduced above, and the their mutual interactions.

The first scenario is referred to as **Choreography Registration Use Case**. Specifically, when the design of a choreography is completed, the Choreography Designer can make it available by registering a specification of such choreography on a dedicated registry. With reference to such registration, the life-cycle at run-time of the choreography is regulated by means of a set of policies that are mainly defined by the V&V Manager. For example, a policy may concern the rules under which a choreography is enactable, another deals with the criteria and the parameters for evaluating the quality associated with of a choreography.

The second scenario concerns the **Service Registration Use Case**. We consider that the Service Providers promote their services (i.e. either develop, or adapt) as participants "fitting" one or more roles to be played in a given choreography. Specifically, this scenario is conceived to fit the case that a choreography specification is already available, and services can dynamically enter and exit the choreography, playing one of the specified roles.

From a testing perspective, these are the most challenging scenarios, as the V&V Manager would refer to them in order to check whether a service can comply to the role, by testing it for conformance against the choreography specification.

3 GOVERNANCE POLICIES ENABLING V&V ACTIVITIES

As part of the choreography V&V process would take place at run-time, thus also the V&V activities should refer to a set of rules, policies, and mechanisms governing the validation of both the choreography enactments, and the implementations of the services they refer.

The policies governing the V&V activities we are proposing in this paper extend the work on on-line testing earlier proposed in (Bertolino et al., 2006) and (Bertolino et al., 2012). These policies are meant to regulate the testing of a service implementation in order to validate if it actually behaves according to the role the Service Provider claimed at the time of its registration. Furthermore, such policies are equally important for governing "perpetual" V&V activities; in other word V&V policies can regulate when to re-test a service (i.e. either periodically, or event-driven) that has been already registered in order to guarantee that its behaviour did not change over time.

In the next sections we propose and discuss a set of policies and rules that could be adopted and implemented within a given V&V governance framework. For clarity of exposition we organize such policies according to the following classification:

1. **V&V Activation Policies** that describe rules for regulating the activation of the on-line testing sessions. As described in the following, such activation could be either periodical, driven from the events about the lifecycle of a service, or linked to some kind of quality indicator;
2. **V&V Rating Policies** that prescribe which aspects have to be considered for rating the quality of both choreographies, and services.
3. **V&V Ranking Policies** defining the rules and the metrics for computing the quality parameters expressed into the V&V Rating Policies;
4. **Choreography Enactment Policies** that prescribe rules for deciding when a service choreography could be enacted;
5. **Test Cases Selection Policies** regulating the testing strategies that can be applied at run-time.

The rest of the paper describes such policies in detail by considering a given choreography `C`.

4 V&V ACTIVATION POLICIES

The idea of V&V governance was originally proposed in (Bertolino and Polini, 2009) to support an on-line testing session when a service asks for registration within a registry. In this vision, only services passing the testing phase are logged in the registry. As a result, the registry is expected to include only "high-quality" services that passed the validation steps foreseen by a more general governance framework. In addition to the registration of a new service, the on-line validation process could be also extended to other events, like the release of a new service version.

Note that when entering a new service registration in a registry, the service provider is naturally wishful to promote the service and therefore can be explicitly willing to submit it to on-line testing. On the other hand, the notification of a service upgrade could be notified only sporadically. The governance mechanisms oriented to V&V activities could mitigate this aspect by means of specific policies and obligations that the service providers should abide by, when binding their services to a choreography.

During the life-cycle of a choreography `C`, a service that was originally registered to play a given role in `C` could be modified or become deprecated. This event might impact on the regular progress of some of the activities described in `C`. Thus, the V&V governance should support rules that verify if the registry still points at one or more active services that could play all the roles subscribed by the removed service in `C`.

Besides, the service provider might omit to notify the deprecation of a service to the service registry. For these cases, the monitoring system infrastructure of the V&V governance architecture could cooperate with the Service Registry in order to verify and notify if any registered service is not reachable anymore.

Within a choreography a service may play one or more roles. Also, the same service may be involved in several choreographies. Service provider may decide to change the roles that their services are playing in the choreographies in order to fulfill new needs, new business requirements, or the evolution of the technical solutions offered by the service. In all these scenarios, one of the V&V governance rules we propose is that any modification (i.e. activation, modification, cancellation) to the role of a service in `C` should activate a new testing session. Specifically:

Activation: when a new role `A` is added to a service `S` in `C`, execute the integration test suite for `A` in

C and run it through S. Evaluate the impact of the result of such on-line testing session on the testing scores of the choreography C (as detailed in Section 5).

Cancellation: when a role A is deleted from the roles that a service S in could play in C, verify that the service registry still points to at least one active service that could play A in C.

Modification: this step could be processed as a sequence of a cancellation and an activation of a role for a service in C.

V&V activation policies could also regulate on-line testing sessions by referring to rating management architectures (e.g. based either on trust, or reputation models). As an example, they can specify the admissible ranking rates that each service participating in a choreography must yield, and then trigger on-line testing sessions whenever the rank of a service decreases below such thresholds. We introduce ranking policies in Section 6.

Finally, in addition to the event-driven activation of the on-line testing sessions (e.g. service registration, etc), V&V policies can also regulate the "perpetual" testing of software services either periodically, or when a specified deadline is met.

5 V&V RATING POLICIES

Section 4 describes how the on-line validation of a service that is indexed by a service registry can be exploited in order to build a trustful environment. This section describes the V&V governance policies rating both a single service that is bound to a choreography C, and also C as a whole. We partition these policies in two main categories: rating policies based on objective testing scores; and rating policies based on subjective evaluation (i.e. feedbacks).

Concerning the first category, we refer to a scenario implementing some perpetual validation techniques, e.g. (Bertolino et al., 2006; Bertolino et al., 2012). The results of the testing sessions somehow allow us to quantify the service trustworthiness values according to the testing goals. Thus, we can use such results to determine what service is trustable according to an objective estimation (i.e. test passed VS. test failed). Several trust models could be associated to V&V governance aspects; in Section 6.2 we detail a metric based on testing results.

Similarly to the rating of trustworthiness for single services the V&V governance framework could refer to some objective trust models in order to estimate the potential trustworthiness score for the whole choreography. In particular, for each role A defined in C, the trust model could consider the trustworthiness based on testing score of all the services on the registry that can play as A. The trustworthiness of the whole C is a function of the testing scores computed for all the roles, and it could be interpreted as the resulting and objective potential quality state (e.g. a benchmark) of the whole choreography. A concrete metric that instantiates this trust model is described in Section 6.1.

Concerning the subjective category, we refer to service reputation. As argued in (Jøsang et al., 2007), in this context we consider service reputation as a metric of how a service is generally perceived by its users. Thus, differently from the testing-based trust systems described above, that are based on objective measures, here reputation systems are based on subjective judgments.

We consider reputation as an interesting indicator to be referred within governance policies for V&V activities. The basic idea of a reputation system is to let parties rate each other, for example after the completion of an interaction, and use the aggregated ratings about a given party to derive a reputation score, which can assist other parties in deciding whether or not to interact with that party in the future (Jøsang et al., 2007). Currently, reputation systems represent an interesting and significant trend in decision support, service provision, and service selection (e.g. the Google's +1 Button, the eBay's feedback forum, the Facebook's Thumbs Up button, the LinkedIn's Recommendations). Several and configurable reputation models could be associated to this V&V governance aspect; in particular Section 6.3 describes a model implementing this kind of reputation rule.

Finally, V&V governance policies could refer to some compositional model of the user's feedbacks in order to estimate a potential reputation score for the whole choreography. In particular, for each role A defined in C, the trust model could consider the reputation score (i.e. positive feedbacks VS. negative feedbacks) of all the services on the registry that can play as A. The reputation score of C (as a whole) is a function of the feedback scores computed for all the roles. Thus here, the reputation score of C is interpreted as a benchmark of subjective judgments for the choreography.

6 RANKING RULES FOR V&V RATING POLICIES

Rating policies rely on some metrics to evaluate the choreography as well as its participating services. In this section we propose some possible ranking rules

for this purpose. In particular, Section 6.1 describes a rule for ranking a service choreography according to both the topology of the choreography itself and the ranking of the available/known services that can play a role specified by the choreography. Then, according to the V&V Rating Policies described in Section 5, we provide two possible strategies for calculating the service ranking function. Specifically, Section 6.2 describes an objective ranking strategy that is based on the results of the testing sessions, while Section 6.3 describes a subjective ranking strategy that is based on a reputation model.

Other different criteria for service ranking could be considered in future extensions, for example an interesting way to complement our proposed numerical rankings could be to also take into account ontological matching, as proposed in (Arroyo and Sicilia, 2008).

6.1 Choreography Rank

The ranking metric for a service choreography described in the following is based on the well-known PageRank algorithm (Page et al., 1999) used by Google.

Differently from the other web search engines that have been used until year 2000, Google dominated the market due to the superior search results that its innovative PageRank algorithm delivered. As described in (Jøsang et al., 2007), the PageRank algorithm can be considered as a reputation mechanism because it ranks the web pages according to the number of other pages pointing at it. In other words, the algorithm interprets the collection of hyperlinks to a given page as public information that can be combined to derive an objective reputation score.

In the original definition (Page et al., 1999) (Kamvar et al., 2003), the PageRank algorithm considers a collection of web pages as a graph where each node is a web page, and the hyperlinks of the pages where modeled as outgoing edges of the graph. In this sense, the strategy proposed by PageRank algorithm can be applied to any problem that can be formulated in terms of a graph.

Our interpretation of the PageRank algorithm considers both the services involved in a choreography, and the graph that the choreography subsumes. Specifically, let us denote $\mathtt{S}$ as a set of services. We define :

$$\mathcal{A}_{\mathtt{C}} = \{A | A \text{ is a role in } \mathtt{C}\} \tag{1}$$

as the set of all the roles defined in $\mathtt{C}$, and:

$$\Omega_{\mathtt{C}}(A) = \{\omega \in \mathtt{S} | A \in \mathcal{A}_{\mathtt{C}}, \omega \text{ plays } A \text{ in } \mathtt{C}\} \tag{2}$$

as the set of all the services in $\mathtt{S}$ that can play the role A in $\mathtt{C}$. Also, given a relation of dependency among the roles in a choreography, for each role A in $\mathtt{C}$ we denote both the set of roles in $\mathtt{C}$ on which A depends (i.e. $N^{+}_{\mathtt{C}}(A)$), and the set of roles in $\mathtt{C}$ that depend on A (i.e. $N^{-}_{\mathtt{C}}(A)$). Specifically:

$$N^{+}_{\mathtt{C}}(A) = \{B | A \in \mathcal{A}_{\mathtt{C}}, B \in \mathcal{A}_{\mathtt{C}}, \exists \text{ dep. from } A \text{ to } B \text{ in } \mathtt{C}\} \tag{3}$$

$$N^{-}_{\mathtt{C}}(A) = \{B | A \in \mathcal{A}_{\mathtt{C}}, B \in \mathcal{A}_{\mathtt{C}}, \exists \text{ dep. from } B \text{ to } A \text{ in } \mathtt{C}\} \tag{4}$$

Note that the definitions above are given in terms of an abstract notion of dependency that can occour among the roles belonging to a choreography. Section 9.1 will present an example by using a specific dependency relation.

Let us denote R as a ranking function for a given service (see either Section 6.2, or Section 6.3), thus Equation 5 defines the ranking function of a role A in a choreography $\mathtt{C}$.

$$\mathcal{R}(A,t,\mathtt{C}) = \frac{\sum_{\omega \in \Omega_{\mathtt{C}}(A)} R(\omega,t)}{|\Omega_{\mathtt{C}}(A)|} \tag{5}$$

Specifically, Equation 5 gives rank values based on the ranking of all the services that are implementing A in $\mathtt{C}$: the more the services that can play A in $\mathtt{C}$ rank well, the better A will perform in the choreography. The effect of such impact is normalized according to the number of services playing the role A. On the other hand, the equation computes a poor ranking for A if there exist only few and poorly ranked services that can play A in $\mathtt{C}$. In other words, this means that A could be considered critical for the enactment of $\mathtt{C}$.

Nevertheless, the role ranking in Equation 5 does not take into account the context where the role is used. Equation 6 introduces a correction factor of the ranking $\mathcal{R}$ taking into account the role dependencies implied by a choreography specification.

Given a role A in choreography $\mathtt{C}$, the case $t = 0$ in Equation 6 defines the initial condition for δ, while the recursive definition for the case $t \geq 1$ is an interpretation of the PageRank algorithm (Page et al., 1999) (Kamvar et al., 2003). Specifically, such definition is composed by two terms: using the terminology adopted within the PageRank algorithm, the first term of Equation 6 is the *source-of-rank* which is a parameter influencing the final rank. In our case the *source-of-rank* is $\mathcal{R}$, giving rank values based on the evaluation of all the services that are implementing A in $\mathtt{C}$.

The second term in Equation 6 gives rank values as a function of the other roles in $\mathtt{C}$ on which A depends. In other words, we consider that if the be-

$$\delta(A,t,\mathsf{C}) = \begin{cases} \lambda\mathcal{R}(A,t,\mathsf{C}) + (1-\lambda) \sum\limits_{B \in N^{+}_{\mathsf{C}}(A)} \dfrac{\mathcal{R}(B,t-1,\mathsf{C})}{\left|N^{-}_{\mathsf{C}}(B)\right|} & \text{for } t \geq 1 \\ \varphi & \text{otherwise} \end{cases} \quad (6)$$

haviour foreseen for A in C relies on the actions performed by another role B (e.g. B is the initiator of a task with A), then the rank values scored by B should impact the ranking of A within the whole C. This metric is generally helpful, however we consider it particularly significant when B is badly ranked, for example because most of the services playing B are not reliable. Note that, as the metric in Equation 6 does not take into account the enactment status of the choreography C, the effect of how much B impacts on A is proportionally calculated by considering the number of all the roles that depend from B.

The parameter λ can be used to tune the contribution of each term in the computation of the role's ranking function; in the literature on the PageRank algorithm, typically its most cited value is 0.85 (Kamvar et al., 2003).

Finally, we denote the ranking function for the whole choreography C as:

$$\mathbb{R}(\mathsf{C},t) = \sum_{A \in \mathcal{A}_{\mathsf{C}}} \delta(A,t,\mathsf{C})\mathcal{R}(A,t,\mathsf{C}) \quad (7)$$

6.2 Testing-based Service Rank

As described in Section 5, in those scenarios including perpetual V&V activities, the analysis of the results of each testing session can be used for building a trust model for services.

The very general idea of this assumption is that data from testing results (i.e. both test passed, and test failed) represents quantitative facts that permit to determine how much a service playing a given role is trustable according to an objective estimation. For example, if the testing sessions that are executed focus on integration issues, the testing-based service rank explains how a service is behaving with respect to the scenario foreseen by a choreography.

Furthermore, as service behaviour may continuously evolve (e.g. change in the implementation, dynamic binding with other services), a trust model should consider that the closer in time a service has been tested the more reliable are the results obtained from the testing session. Thus the definition of a testing-based ranking for services should decrease over time.

The logistic function is a well-studied mathematical function that was originally proposed in (Verhulst, 1838), and is often used in ecology for modeling population growth. Specifically, in one of its most common formulation, the logistic function offers a non linear evolution of the population function P over the parameter t (e.g. the time) depending on the two parameters: the carrying capacity K, and the growth rate r.

The testing-based ranking we propose defines a function of trust basing on the logistic function, where K is interpreted as the highest admissible level of trust, while for each service w, r is the number of the tests passed over the total number of test executed (see Equation 8).

$$r_{\omega} = \frac{\#\text{passedTest}_{\omega}}{\#\text{runTest}_{\omega}} \quad (8)$$

For a given service $\omega \in \Omega_{\mathsf{C}}(A)$ that can play the role A in C, Equation 9 defines a test-based trust model based on the logistic function.

$$\mathcal{T}(t,\omega) = \frac{K v_{\text{Fade}} e^{(-r_{\omega}(t-h_{\text{Offset}}))}}{K + v_{\text{Fade}}(e^{(-r_{\omega}(t-h_{\text{Offset}}))} - 1)} \quad (9)$$

Specifically, in Equation 9, the h_{Offset}, and the v_{Fade} are configurable parameters useful for translating, and fading the values returned by the trust model. In addition, as the trust model $\mathcal{T}$ is actually an instantiation of the logistic function, the setting of the parameters for $\mathcal{T}$ must keep satisfying the stability criteria foreseen by logistic function (e.g. $K > 1$).

Finally, Equation 10 defines a testing-based ranking function for a given service ω playing a given role A in C (i.e. $\omega \in \Omega_{\mathsf{C}}(A)$). As introduced in Section 6.1, such service-level ranking function can be exploited in order to compute the testing-based rank of the role A (see Equation 5), and consequently the testing-rank of the whole choreography C (see Equation 7).

$$R(t,\omega) = \begin{cases} \mathcal{T}(t,\omega) & \text{if } t < h_{\text{Offset}} \\ 0 & \text{else} \end{cases} \quad (10)$$

According to the definition in Equation 10, the configurable parameter h_{Offset} is interpreted as the maximum observation period (e.g time, hours, days, week, etc.) after which the testing-based service rank is considered not sufficiently reliable, and then a new testing session for the service is recommended.

As an example, Figure 1 depicts the evolution of the function $R(t,\omega)$ with respect to t (i.e. time), and by considering different values of r_{ω}. Specifically the

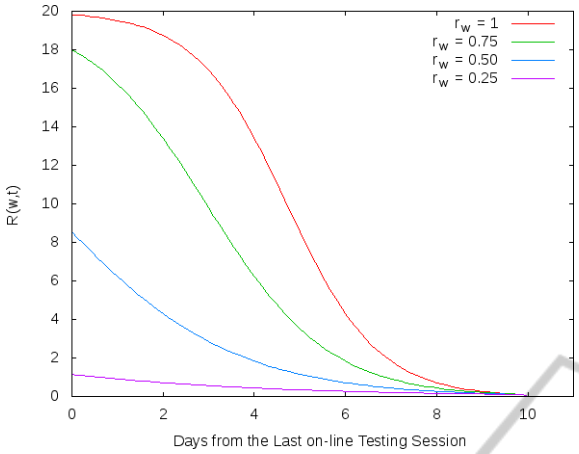

Figure 1: Examples of the evolution of the testing-based service rank function.

figure shows the case all the test cases passed (i.e. $r_\omega = 1$), the case some of the test cases passed (i.e. $r_\omega = 0.75$), the case half of the test cases passed (i.e. $r_\omega = 0.50$), and the case most of the test cases failed (i.e. $r_\omega = 0.25$).

6.3 Reputation-based Service Rank

The testing-based service rank proposed in Section 6.2 offers a quantifiable, and objective metric evaluating how a service is behaving; for example with respect to the scenario envisioned by a service choreography. Differently, reputation systems based on feedbacks provide a widely adopted solution in order to share subjective evaluation of a service after the direct experience of its users.

In the literature several ranking models have been proposed in order to combine user's feedbacks and derive reputation ratings (Jøsang et al., 2007). Among the others, in this section we will refer to a reputation model based on the Beta Density function (β), and originally proposed in (Jøsang and Ismail, 2002). Specifically, the authors in (Jøsang and Ismail, 2002) argue how reputation systems based on the β function are both flexible, and relatively simple to implement in practical applications. Furthermore, such systems have good foundation on the theory of statistics.

Let us consider a service ω playing a given role A in C (i.e. $\omega \in \Omega_{\mathsf{C}}(A)$). Than, we denote $f_\omega^+, f_\omega^- \geq 0$ as the number of positive and negative feedbacks collected by the service ω, respectively.

According with the formulation given in (Jøsang and Ismail, 2002), the β function can be written as reported in Equation 11, where Γ is the well-studied Gamma function.

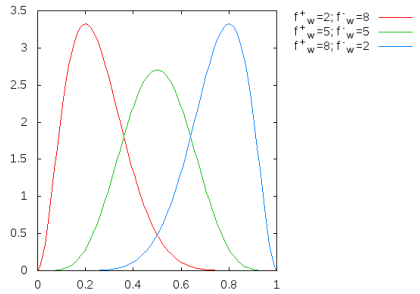

Figure 2: Examples of the β function.

$$\beta(p, f_\omega^+, f_\omega^-) = \frac{\Gamma(f_\omega^+ + f_\omega^- + 2)}{\Gamma(f_\omega^+ + 1) * \Gamma(f_\omega^- + 1)} * p^{f_\omega^+} * (1-p)^{f_\omega^-} \tag{11}$$

The interesting consideration about the β function is that its mathematical expectation is trivial to compute; it is given by the Equation 12.

$$E(\beta(p, f_\omega^+, f_\omega^-)) = \frac{f_\omega^+ + 1}{f_\omega^+ + f_\omega^- + 2} \tag{12}$$

In other words, the feedbacks that the users reported during past interactions with a service ω are interpreted by Equation 12 as the probability of collecting a positive feedback with ω on average in the future interactions. For example, if E for the service ω is 0.8 means that is still uncertain if ω will collect a positive feedback in the future (i.e. due to a positive interaction), but it is likely that this would happen.

Figure 2 depicts three instantiation of the β function: the case most of the feedback are negatives (i.e. $f_\omega^+ = 2, f_\omega^- = 8$), the case half of the feedback are positives (i.e. $f_\omega^+ = 5, f_\omega^- = 5$), and the case most of the feedback are positives (i.e. $f_\omega^+ = 8, f_\omega^- = 2$).

Finally, Equation 13 defines a reputation ranking function based on user's feedbacks for a given service ω playing a given role A in C (i.e. $\omega \in \Omega_{\mathsf{C}}(A)$).

$$R(t, \omega) = E(\beta(p, f_\omega^+, f_\omega^-)) \tag{13}$$

As introduced in Section 6.1, also such service-level ranking function can be exploited in order to compute the reputation-based rank of the role A (see Equation 5), and consequently the reputation-rank of the whole choreography C (see Equation 7).

7 CHOREOGRAPHY ENACTMENT POLICIES

When the design of a choreography is completed, it could be made available by registering its specification on a dedicated Choreography Registry. In addition to the canonical registry functionalities based

on service choreographies (i.e. store and retrieve choreography specifications), a Choreography Registry should also store meta-information; for example about the status of a choreography, i.e., if a given choreography C is enactable or not.

Thus, from the registration of a choreography specification on a Choreography Registry, the lifecycle at run-time of the choreography could be regulated by means of a set of V&V governance policies; among the others, some concerning the rules under which a choreography could be considered enactable.

Many kind of strategies could be applied in order to classify a choreography as enactable. For example, a policy defines a choreography C enactable if for any role A in C, it is possible to point to a set of services (i.e. one or more services) that can play A in C.

In addition, Section 5 describes some V&V governance policies based on rating mechanisms for both services and choreographies. Thus, enactment policies for C could be also regulated in terms of the rating scores evaluated for the choreography. For example, C is enactable if and only if it scores either a minimal trust (i.e. based on tests), or a minimal reputation (i.e. based on feedbacks) level.

8 TEST CASES SELECTION POLICIES

In literature various policies have been proposed in order to identify a proper test suite for testing a third party service. For example in (Tsai et al., 2003), the authors suggested that for testing a service implementation, integrators should directly access and use test cases provided by the same Service Provider.

In (Eler et al., 2010) and (Bartolini et al., 2011), the authors proposed testable services as a solution to provide third-party testers with structural coverage information after a test session, yet without revealing their internal details.

The main drawback of such approaches is that both integrators, and service testers, do not have enough information to improve their test set when they get a low coverage measure because they do not know which test requirements have not been covered. In (Eler et al., 2011) the authors propose an approach in which testable services are provided along with test meta-data that will help their testers to get a higher coverage.

Nevertheless, all these ideas could fit well with the specification of both governance policies, and rules enabling V&V activities. Specifically, policies could require such meta-data as complementary documentation that service providers should supply for making their service testable.

All the policies described above select test cases from the test suite provided by the Service Provider. In some cases, such approaches could not provide completely objective test suites focusing on integration aspects.

An alternative approach is the definition of test cases selection policies that enable the derivation of test cases from models provided by the Service Provider, for example during the registration of the service. Specifically, similarly to (De Angelis et al., 2010), it could be equally useful to derive test cases from the service choreography. In fact, a choreography specification defines both the coordination scenarios in which a service under registration plays a role, and the abstract behaviour expected by each role.

9 EXAMPLES

Most of the policies proposed in the previous sections are quite intuitive and we have given simple examples while introducing them. For lack of space we provide in the following more detailed examples of implementations only for ranking policies (Section 9.1) and selection policies (Section 9.2).

9.1 A Simple Example about Rankings

Section 6.1 defines the ranking function of a role in terms of an abstract notion of dependency that can occour among the roles belonging to a choreography. Specifically, for each role A in a choreography C, such dependency is referred in order to compute both the sets $N^+_{\mathsf{C}}(A)$, and $N^-_{\mathsf{C}}(A)$. In the following we describe a dependency relation that will be implemented within the V&V governance framework.

Let us consider a choreography C, and let us denote the set of tasks defined within C:

$$T^{\mathsf{C}} = \{\tau | \tau \text{ is a task defined in } \mathsf{C}\} \tag{14}$$

the set of roles participating in a given task of C:

$$Part(\tau, \mathsf{C}) = \{A | A \in \mathcal{A}_{\mathsf{C}}, \tau \in T^{\mathsf{C}}, A \text{ is involved in } \tau\} \tag{15}$$

and, for each task τ in C, $Init(\tau)$ is the role that initiates a task τ.

Thus, given A, B roles in C, the dependency definition we propose relates A with B if and only if B is the initiator of a task were also A is involved. In other words, in order to accomplish a given task in C, A requires some action from B. More formally, we denote

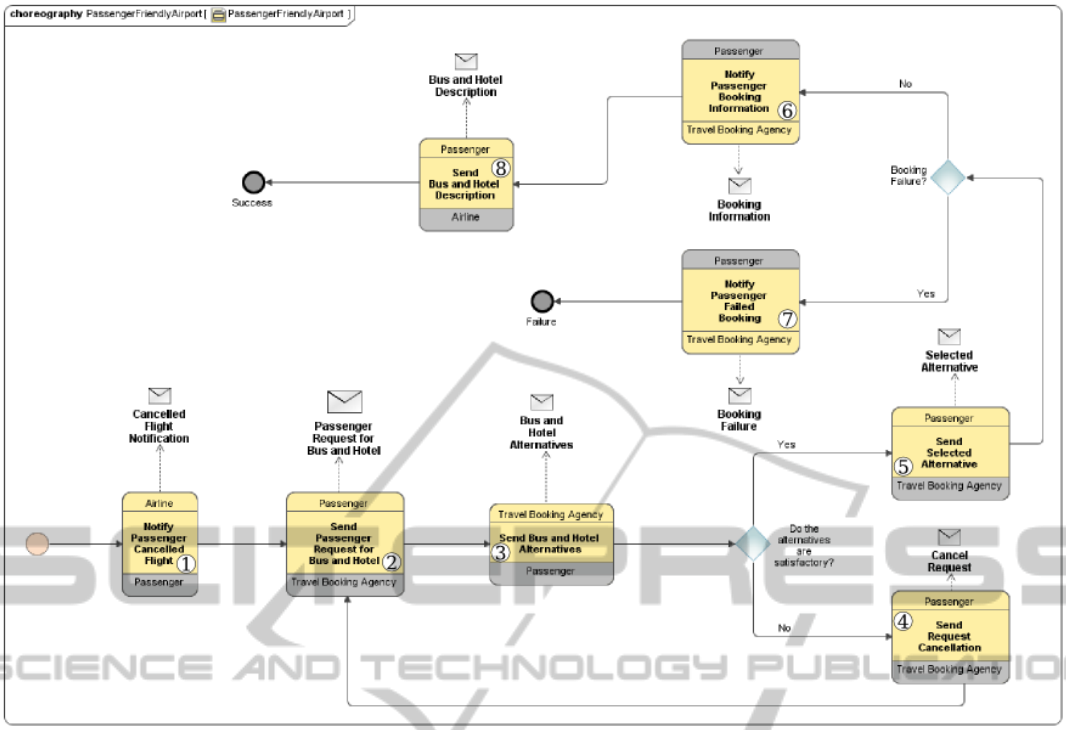

Figure 3: Example from the passenger-friendly-airport choreography.

this dependency relation on C as $\rightsquigarrow^{\mathsf{C}} \subseteq \mathcal{A}_{\mathsf{C}} \times \mathcal{A}_{\mathsf{C}}$, so that:

$$\begin{array}{c} A1 \rightsquigarrow^{\mathsf{C}} A2 \Leftrightarrow \exists \tau \in T^{\mathsf{C}}, \text{ and} \\ A1 \neq A2, \text{ and} \\ A1, A2 \in Part(t, \mathsf{C}), \text{ and} \\ A2 = Init(\tau) \end{array} \tag{16}$$

For example, Figure 3 depicts a simple service choreography modeled with the BPMN Choreography Diagram notation. With respect to this example, the sets described above are instantiated as it follows:

- $\mathcal{A}_{\mathsf{C}} = \{ Passenger, Airline, TBA \}$
- $T^{\mathsf{C}} = \{①, ②, ③, ④, ⑤, ⑥, ⑦, ⑧,\}$

In addition, for each task $\tau \in T^{\mathsf{C}}$, Table 1 reports both the set $Part(\tau, \mathsf{C})$, and $Init(\tau)$; while Figure 4 depicts the dependency graph resulting from the instantiation of the relation $\rightsquigarrow^{\mathsf{C}}$ on this simple example. Finally, according to the definitions given in both Equation 3, and Equation 4, Table 2 reports the sets $N^{+}_{\mathsf{C}}(A)$, and $N^{-}_{\mathsf{C}}(A)$ resulting form the specific instantiation of the dependency relation here adopted.

Figure 4: Dependency graph according to the relation $\rightsquigarrow^{\mathsf{C}}$.

Table 1: Instantiation of *Part*, and *Init*.

Task	*Part*	*Init*
①	*Passenger*, *Airline*	*Airline*
②	*Passenger*, *TBA*	*Passenger*
③	*Passenger*, *TBA*	*TBA*
④	*Passenger*, *TBA*	*Passenger*
⑤	*Passenger*, *TBA*	*Passenger*
⑥	*Passenger*, *TBA*	*TBA*
⑦	*Passenger*, *TBA*	*TBA*
⑧	*Passenger*, *Airline*	*Passenger*

Table 2: Examples of $N^{+}_{\mathsf{C}}(A)$, and $N^{-}_{\mathsf{C}}(A)$.

$N^{+}_{\mathsf{C}}(Passenger)$	*Airline*, *TBA*
$N^{-}_{\mathsf{C}}(Passenger)$	*Airline*, *TBA*
$N^{+}_{\mathsf{C}}(Airline)$	*Passenger*
$N^{-}_{\mathsf{C}}(Airline)$	*Passenger*
$N^{+}_{\mathsf{C}}(TBA)$	*Passenger*
$N^{-}_{\mathsf{C}}(TBA)$	*Passenger*

In the following, the example will only focus on the testing-based service rank described in Section 6.2; however similar results, and considerations can be achieved by using the reputation-based service rank presented in Section 6.3.

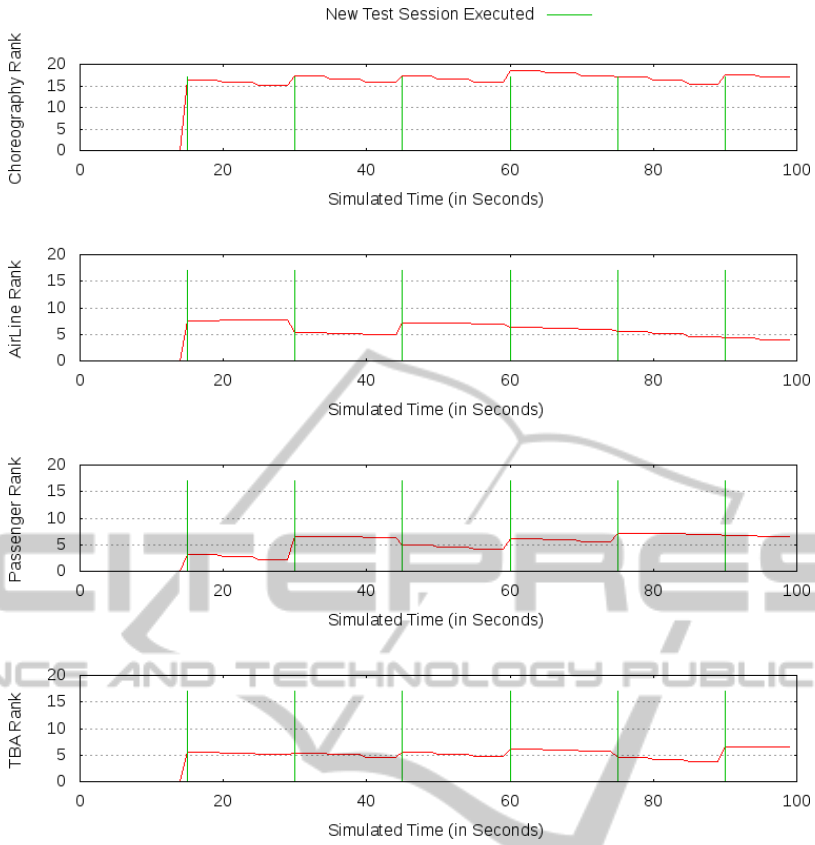

Figure 5: Results of the ranking simulation on the passenger-friendly-airport choreography.

Thus let us assume several services can implement each role in C. Specifically, this example considers : 2 services acting as *Passenger*, 4 services acting as *Airline*, and 5 services acting as *TBA*. In this scenario, the example foresees that each of these services is associated with a specific test suite, whose periodical execution provides the index r_ω presented in Section 6.2.

Clearly, for each service, the experiment can refer to a specific timeout simulating the execution of a new testing session, and thus the definition of new values for the indexes r_ω. Also, for each of service, the experiment can also use the parameter h_{Offset} introduced in Equation 9 in order to control the decaying process of the results from the latest testing session simulated.

Nevertheless, in order to keep the presentation of the results simple, the simulation we run over this example considered for all the services involved both the same h_{Offset}, and the same timeout for launching the new testing sessions.

Figure 5 depicts the results of the simulation we executed. In particular, the up-most diagram of the figure depicts the rank for the whole choreography computed according to Equation 7; while the other diagrams show the computed rank for the three roles (i.e. *Passenger*, *Airline*, *TBA*). In each diagram, the vertical green bars denote when the execution of a new testing session occurred. After each testing session, the ranks of each role can either grow or decrease depending on the results collected from the testing of the services playing it. Between two testing sessions, each role fall-off in ranking as prescribed by Equation 10.

9.2 A Simple Example about Test Cases Selection

This section describes how a V&V Manager can define V&V policies concerning the derivation test cases from the service choreography. Specifically, the V&V Manager could define test cases selection policies by using a counter-example approach similarly to (De Angelis et al., 2010).

The very general idea of test cases selection policies assumes that the V&V Manager specifies the characteristics that a test case should have through a test purpose. Often, a common objective in dealing with service choreography is trying to derive test suites for the choreography participants that are strongly focused on integration issues. For example,

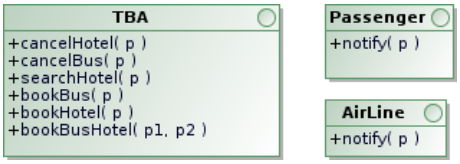

Figure 6: Interfaces associated with the participants the choreography at Figure 3.

the V&V Manager can define the test case selection process by means of the formulation of test purposes in form of reachability properties. According to the choreography specification introduced in Figure 3, a simple property defined as a test cases selection policy, could be:

> *"The test cases must include task ②, followed (in a future) by task ⑤, and then either by task ⑥, or ⑦"* (17)

As detailed in (De Angelis et al., 2010), such a policy can be expressed in a temporal logic formula (such as in LTL, CTL, Hennessy-Milner Logic) so that it could be automatically processed by a test cases generation framework.

Nevertheless, as more participants are involved in a choreography task, from the same property the policy can select different test suites; usually one for each service participating in the choreography. Thus, in addition to the property in (17) driving the generation of test cases over the service choreography, a test cases selection policy also has to specify the target of such generation process. This process is usually referred as projection of a trace on a service test case.

In this specific example, we assume that any participant to the choreography in Figure 3 implements one of the interfaces depicted in Figure 6. Listing 1, and Listing 2 show the projection of two integration test cases that could be derived for a service that is willing to play as *TBA* from the property in (17). If a service candidate to *TBA* fails one of such tests, it cannot be integrated within the choreography at Figure 3.

Concluding, the V&V Manager can plan the test cases generation process by means of the test case selection policies. The section shown how a test case can target in revealing potential integration issues of the service choreography. Indeed, the test cases validates if the usage of a service implementation as participant of a choreography is compliant with the service pre-conditions or with service protocol. The V&V Manager can refer to the test cases selected by such policies as an acceptance validation for the implementation of the choreographed services.

```
1  b = searchBus(p1);
2  if (! isValid(b) )
3    return TestFailed;
4  h = searchHotel(p2);
5  if (! isValid(h) )
6    return TestFailed;
7  r = bookBusHotel(b,h);
8  if (r.isAKindOf(BookingInformation) || r.isAKindOf(
       BookingFailure))
9    return TestPassed;
10 else
11   return TestFailed;
```

Listing 1: Example of integration test case derived for *TBA*.

```
1  s = searchBusHotel(p1,p2);
2  if ((!isValid(s.b)) || (!isValid(s.h)))
3    return TestFailed;
4  r = bookBusHotel(b,h);
5  if (r.isAKindOf(BookingInformation) || r.isAKindOf(
       BookingFailure))
6    return TestPassed;
7  else
8    return TestFailed;
```

Listing 2: Another integration test case derived for *TBA*.

10 CONCLUSIONS AND FUTURE WORK

Choreography and SOA governance are important instruments to enable the dynamic and flexible interoperability among independent services which is at the basis of the FI. Our work concerns the on-line V&V of services and choreographies, which relies on a disciplined policy-based governance approach. Implementation of such a framework is a huge undertaking, facing theoretical and technical challenges. We have identified in this paper a set of policies, including activation, rating, ranking, enactment and test case selection policies. We have discussed how such policies can support V&V and provided some preliminary examples. The policies illustrated here are undergoing implementation in the context of the European project CHOReOS[1].

Future work will include the implementation of a governance registry for the management of such policies, and their instantiation within the CHOReOS demonstration scenarios.

Furthermore, another important motivation for V&V policies is to enable the scaling up of online testing as the dimensions of the tackled choreographies increases. Indeed, within the FI any software

[1] http://www.choreos.eu/

will be by nature characterized as an Ultra-Large-Scale (ULS) software system, where a ULS system "*is ultra-large in size on any imaginable dimension*" (Pollak, 2006), like in the resulting number of lines of codes, in the number of people employing the system for different purposes, in the amount of data stored, accessed, manipulated, and refined, in the number of hardware elements (i.e. heterogeneity), and finally in the number of possibly cooperating organizations. Therefore, future work will also include the identification of further policies to help mitigate ULS effects in V&V of FI choreographies.

ACKNOWLEDGEMENTS

This paper is supported by the the European Project FP7 IP 257178: CHOReOS, and partially by the European Project FP7 NoE 256980: NESSoS.

REFERENCES

Arroyo, S. and Sicilia, M.-Á. (2008). Sophie: Use case and evaluation. *Information & Software Technology*, 50(12):1266–1280.

Baresi, L., Nitto, E. D., and Ghezzi, C. (2006). Toward open-world software: Issue and challenges. *Computer*, 39:36–43.

Barker, A., Walton, C. D., and Robertson, D. (2009). Choreographing web services. *IEEE T. Services Computing*, 2(2):152–166.

Bartolini, C., Bertolino, A., Elbaum, S. G., and Marchetti, E. (2011). Bringing white-box testing to service oriented architectures through a service oriented approach. *Journal of Systems and Software*, 84(4):655–668.

Bertolino, A., De Angelis, G., Kellomäki, S., and Polini, A. (2012). Enhancing service federation trustworthiness through online testing. *IEEE Computer*, 45(1):66–72.

Bertolino, A., Frantzen, L., Polini, A., and Tretmans, J. (2006). Audition of Web Services for Testing Conformance to Open Specified Protocols. In *Architecting Systems with Trustworthy Components*, number 3938 in LNCS, pages 1–25. Springer.

Bertolino, A. and Polini, A. (2009). Soa test governance: Enabling service integration testing across organization and technology borders. In *Proc. of ICSTW*, pages 277–286.

De Angelis, F., De Angelis, G., and Polini, A. (2010). A counter-example testing approach for orchestrated services. In *Proc. of ICST 2010*, pages 373–382, Paris, France. IEEE Computer Society.

Eler, M., Delamaro, M., Maldonado, J., and Masiero, P. (2010). Built-in structural testing of web services. In *Proc. of Brazilian Symp. on Soft. Engineering*, pages 70–79.

Eler, M. M., Bertolino, A., and Masiero, P. (2011). More testable service compositions by test metadata. In *Proc. of SOSE*, Washington, DC, USA. IEEE Computer Society.

Fiedler, M., Gavras, A., Papanikolaou, N., Schaffers, H., and Wainwright, N. (2011). Future Internet Assembly Research Roadmap – Towards Framework 8: Research Priorities for the Future Internet. Technical report, Future Internet Assembly Working Group.

Ghezzi, C. (2011). The fading boundary between development time and run time. In *Keynote at ECOWS*, page 11. IEEE.

Greiler, M., Gross, H., and van Deursen, A. (2010). Evaluation of online testing for services: a case study. In *Proc. of PESOS*, pages 36–42, New York, NY, USA. ACM.

Jøsang, A. and Ismail, R. (2002). The Beta Reputation System. In *Proc. of the Bled Electronic Commerce Conference*.

Jøsang, A., Ismail, R., and Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644.

Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H. (2003). Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University.

MacKenzie, C., Laskey, K., McCabe, F., Brown, P., and Metz, R. (2006). Reference model for service-oriented architecture, version 1.0. Technical report, OASIS.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report SIDL-WP-1999-0120, Stanford University.

Pan, J., Paul, S., and Jain, R. (2011). A survey of the research on future internet architectures. *Communications Magazine, IEEE*, 49(7):26 –36.

Pollak, B., editor (2006). *Ultra-Large-Scale Systems – The Software Challenge of the Future*. Software Engineering Institute – Carnegie Mellon.

Trossen, D., Briscoe, B., Mahonen, P., Sollins, K., Zhang, L., Mendes, P., Hailes, S., Jerman-Blaciz, B., and Papadimitrou, D. (2009). Starting the Discussions: A white paper from the EIFFEL Think-Tank. Technical report, Eiffel TT.

Tsai, W. T. et al. (2003). Scenario-based web service testing with distributed agents. *IEICE Transaction on Information and System*, E86-D(10):2130–2144.

Verhulst, P.-F. (1838). Notice sur la loi que la population poursuit dans son accroissement. *Correspondance mathématique et physique*, 10.

Woolf, B. (2007). Introduction to soa governance. Accessed on 15 Jun 2011.