Dependability in dynamic, evolving and heterogeneous systems: the CONNECT approach

Antonia Bertolino, Felicita Di Giandomenico, Paolo Masci, Antonino Sabetta, Fabio Martinelli, Ilaria Matteucci, Antinisca Di Marco, Valérie Issarny, Rachid Saadi

HAL Id: inria-00465229

https://hal.inria.fr/inria-00465229

Submitted on 19 Mar 2010

Dependability in dynamic, evolving and heterogeneous systems: the CONNECT approach

[Project Paper]

Antonia Bertolino
Felicita Di Giandomenico
Paolo Masci
Antonino Sabetta
CNR-ISTI, Pisa, Italy
surname@isti.cnr.it

Antinisca Di Marco
Università dell'Aquila, Italy
adimarco@univaq.it

Fabio Martinelli
Ilaria Matteucci
CNR-IIT, Pisa, Italy
name.surname@iit.cnr.it

Valérie Issarny
Rachid Saadi
INRIA, France
name.surname@inria.fr

ABSTRACT

The EU Future and Emerging Technologies (FET) Project CONNECT aims at dropping the heterogeneity barriers that prevent the eternality of networking systems through a revolutionary approach: to synthesise on-the-fly the CONNECTors via which networked systems communicate. The CONNECT approach, however, comes at risk from the standpoint of dependability, stressing the need for methods and tools that ensure resilience to faults, errors and malicious attacks of the dynamically CONNECTed system. We are investigating a comprehensive approach, which combines dependability analysis, security enforcement and trust assessment, and is centred around a lightweight adaptive monitoring framework. In this project paper, we overview the research that we are undertaking towards this objective and propose a unifying workflow process that encompasses all the CONNECT dependability/security/trust concepts and models.

1. INTRODUCTION

Our everyday activities are increasingly dependent upon the assistance of pervasive inter-connected digital systems. However, the efficacy of integrating and composing such systems is proportional to the level of interoperability achieved between the systems' respective underlying technologies. This leads to a landscape of technological islands of networked systems, among which ad hoc connection bridges are possibly deployed, which are strongly technology-dependent. Yet, the fast pace at which technology evolves, at all abstraction layers of networked systems, challenges the lifetime of interoperability in the digital environment. The European Project CONNECT[1] aims at dropping the heterogeneity barriers that prevent networked systems from being eternally CONNECTed and at enabling their seamless composition in spite of technology heterogeneity and evolution. CONNECT targets the dynamic synthesis of CONNECTors via which networked systems communicate. The resulting emergent CONNECTors then compose and further adapt the interaction protocols run by the CONNECTed systems.

The above prospected CONNECT approach comes at risk from the standpoint of dependability. Indeed, as described in [10], there exist many potential threats to the dependability of modern dynamic, evolving and heterogeneous systems such as those tackled in CONNECT. Recently, other European projects and Networks of Excellence specifically focused on achieving dependability against accidental and intentional failures, both in traditional settings [1], and in evolving systems [2], where an additional concern is how to face changes. In the latter perspective, which is very close to the CONNECT vision, dependability is more precisely referred to as *resilience* [14, 16]. In CONNECT, we are investigating a comprehensive approach, which combines dependability analysis, security enforcement and trust assessment, and is centred around a lightweight adaptive monitoring framework. In this project paper, we overview the research we are undertaking towards this goal and propose a unifying workflow process view that encompasses all the CONNECT dependability/security/trust concepts and models.

The paper is structured as follows: Section 2 presents the challenges of the CONNECT project; Section 3 surveys CONNECT dependability concerns and illustrates our monitoring-centric view; Section 4 gives a unifying workflow process view of the dependability related activities in CONNECT, and Section 5 concludes the paper.

[1]CONNECT: Emergent Connectors for Eternal Software Intensive Networked Systems (http://www.connect-forever.eu) is an FP7 FET Proactive IP Project, G.No.231167, addressing the Theme: ICT Forever Yours

2. CONNECT CHALLENGES

The core objective of CONNECT is to establish *eternal interoperability* among networked systems through on-the-fly synthesis, implementation and deployment of emergent CONNECTors.

We depict schematically in Figure 1 the architectural vision of CONNECT. Four types of entities populate the CONNECT world: (i) Networked Systems, which use CONNECT services; (ii) CONNECT Enablers, which encapsulate the CONNECT logic that allows to synthesise a communication bridge between heterogeneous Networked Systems; (iii) CONNECTors, i.e., the emergent communication bridges synthesised by Enablers; (iv) the CONNECTed System, which is obtained by CONNECTing different Networked Systems.

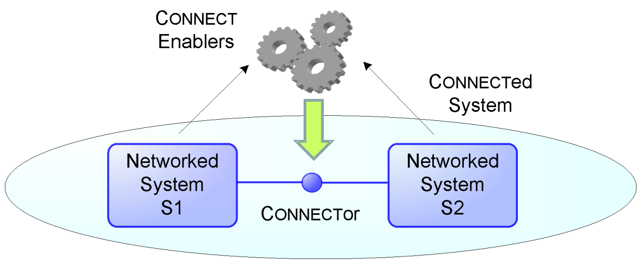

Figure 1: The CONNECT vision

CONNECT Enablers represent the core of the CONNECT approach: they can accept requests from Networked Systems, discover new Networked Systems, gather / learn information on their functional and non-functional behaviour, and synthesise a suitable CONNECTor that allows inter-operation among Networked Systems willing to interact. There are different types of Enablers, according to their provided functionality: Discovery Enablers, Learning Enablers, Synthesis Enablers, Monitoring Enablers, and so on.

To achieve its goals, the CONNECT project undertakes interdisciplinary research in the areas of behaviour learning, formal methods, semantic services, software engineering, dependability, and middle-ware. Among the above areas, this paper specifically focuses on dependability concerns.

In CONNECT, dependability addresses two complementary issues: (i) verification and validation techniques to ensure that Networked Systems and synthesised CONNECTors behave as specified with respect to their functional and non-functional properties, and (ii) security, trust, and privacy assurance for interacting parties.

3. DEPENDABILITY IN CONNECT

In CONNECT, we direct the investigation on dependability towards the peculiar aspects of the project, i.e., the threats deriving from the on-the-fly synthesis of CONNECTors. We explore appropriate means for assessing/guaranteeing that the CONNECTed system yields acceptable levels for different non-functional properties, such as dependability (e.g., the CONNECTor will provide continued communication without interruption), security and privacy (e.g., the transactions do not disclose confidential data), and trust (e.g., components are put in communication only with parties they trust). In order to have a label that includes all the above concerns, hereafter we will use the term "dependability" with two different senses: a stricter one, consistent with the classical definition given in [3], and a broader one, in which the term dependability is meant as a label inclusive of all the different concerns listed above. The context will make clear if we mean dependability in strict or in broad sense.

With reference to the four types of entities illustrated in Figure 1, dependability issues are of concern to all of them. Hence, the characterisation of the fault model and of the metrics of interest is specific to the individual entity considered. In general terms, and inspired by consolidated literature on fault types [3], we consider both development and operational faults, affecting both hardware and software components, caused by an accidental event or maliciously introduced by humans, and whose persistence may be either transient or permanent.

Classical dependability metrics defined in the literature are very useful to give a conceptual classification of different concerns in assigning reliance on a system. Starting from such concepts, we are working on a CONNECT metrics conceptual framework [9] that can be used to refine classical metrics with respect to some peculiar aspects of the CONNECT vision. Indeed, in CONNECT, Enablers may need to synthesise CONNECTors on-the-fly, even when the knowledge on the behaviour and capabilities of some Networked Systems is still incomplete. In these cases, CONNECT Enablers may initially synthesise a basic CONNECTor that permits only some elementary form of interaction, and an enhanced CONNECTor may be synthesised only in a second phase, when CONNECT Enablers have learnt the behaviour of the new Networked Systems. Two basic refinement dimensions are envisaged for dependability metrics: (i) a *CONNECT-dependent* dimension, which considers the four actors of the CONNECT architecture (Networked Systems, Enablers, CONNECTors and the CONNECTed System); (ii) a *context-dependent* dimension, which takes into account the application scenario and the heterogeneous and evolvable aspects of the different actors.

In the following, we introduce the four activities related to dependability in CONNECT: (i) Model-based analysis, (ii) Security enforcement and Privacy, (iii) Trust management, and (iv) Monitoring.

3.1 Model-based Analysis

Model-based analysis techniques are sought in CONNECT to ensure that Networked Systems and CONNECTors satisfy specified levels of accomplishment for dependability requirements, according to pertinent dependability metrics. Both off-line and on-line approaches to verification and validation and to fault forecasting are pursued, to cover a wider range of needs from the point of view of dependability assurance. As commonly intended in the literature, off-line analysis refers to activities devoted to analyse the system at hand before its deployment, or after deployment but in isolation with respect to the system in operation. On the contrary, on-line analysis refers to activities performed while the system is in operation, so accounting for the detailed system and environment aspects during that specific system execution. We adopt the off-line and on-line terminology with this meaning.

Methods considered in CONNECT belong to those for probabilistic, model-based quantitative evaluation, which aim at evaluating, in terms of probabilities, the extent to which the attributes of interest are satisfied. Research in dependability analysis has developed a variety of models, each of which focuses on particular levels of abstraction and/or system characteristics. Model-based approaches [18, 5], being based on the construction of a model of the system from the elementary stochastic processes that model the behaviour of the system components and their interactions, are very suited to early detect design errors and deficiencies, which could otherwise be very costly or even catastrophic when discovered at later stages. Therefore, with reference to the different CONNECT entities, such assessment methods could: (i) help in guiding the process towards the on-line synthesis of CONNECTors with the desired dependability accomplishment level (ii) allow to assess whether the emergent CONNECTor satisfies a desired dependability requirement; the provided assessment could be used as a further criterion for the optimal selection of the CONNECTor to deploy to satisfy specific interaction needs (iii) quantify metrics for end-to-end dependability, in order to verify whether desired levels are satisfied.

Model-based methods for dependability evaluation are currently applied as traditional off-line methods. However, investigations are undertaken for extending the approach to deal with the dynamic aspects involved in the generation of CONNECTors in CONNECT, so as to allow to some extent an on-line assessment of quantitative dependability properties. How to deal with model generation and, especially, model solution so as to provide feedback from the analysis in proper time to be profitably used are the big challenges in this context. Methods based on progressive model definition and refinement, so as to allow for partially pre-determined analysis to be refined/completed at run-time, seem to be promising directions to explore. Monitoring activities (see Section 3.4), by providing accurate information on those model parameters that cannot be estimated in advance, constitute a paramount support to on-line assessment. Also, monitoring can guide model refinement by revealing mismatches between the *actual* dependability level and the *expected* level estimated through the off-line analysis.

In addition, forms of partially-dynamic/partially-static methods will be analysed as well, such as Case Based Reasoning methodologies, where, e.g., a Knowledge Base (KB) repository (or simply a Look-up table) could be set up off-line, storing information on the most appropriate fault-tolerance solution for the CONNECTor to be synthesised for specific dependability requirements, on the basis of the results of a (off-line) model-based evaluation activity. At run-time, the KB is used to search for the best pre-determined solution mapping the requested non-functional properties (or the closest one, in case a proper match is not found). This way, fast decision-making is achieved. The KB is dynamically extended with new "cases" to be added, to account for interoperability requests of evolving Networked Systems.

It is worth noting that, in CONNECT, the activity on dependability assessment is complemented by a verification framework that includes on-line verification and quantitative compositional reasoning, which is part of the foundations and verification methods for composable CONNECTors [8].

3.2 Security and Privacy

CONNECT aims at guaranteeing that the communication between components is always secure. For that reason, we propose and elaborate the Security-by-Contract (S×C) paradigm [11, 12] for providing security in CONNECTed systems. The basic idea of the S×C framework is the concept of the *contract* of an application. The contract is a description of the behaviour of the application and it is provided with the application itself.

Consider two Networked Systems that want to communicate. Consider also that each Networked System has a security policy set on it, P_1 and P_2 respectively. In order to communicate, both Networked Systems send their communication request and their security policies to an Enabler that has to provide a CONNECTor to allow Networked Systems to communicate. Such a CONNECTor will be a mobile code, that the Enabler may chose among a set of already existing CONNECTors or it synthesises a new one on-the-fly. In both cases, the Enabler provides, to each Networked System, a CONNECTor and a contract C that describes the CONNECTor behaviour. Moreover C satisfies both P_1 and P_2.

We are considering the case in which both Networked Systems have also a private policy, P_{1priv} and P_{2priv} respectively. Before executing the CONNECTor, each Networked System locally checks if the contract C is compliant with the local private policy. If this is the case then the CONNECTor is executed, otherwise the local private policies are enforced. In both cases the communication is established.

The basic idea of the proposed enforcement architecture is the following: when the Enabler provides the CONNECTor to each Networked System, both of them verify if the code and the contract actually match by an *evidence checking* procedure (Check Evidence). This step is intended to provide a formal proof that the contract effectively denotes every possible behaviour of the running program. This step can be implemented, for instance, using the *model-carrying code* [22, 23] method. Briefly, the Enabler attaches a formal proof that the CONNECTor satisfies its contract. Then, the Networked Systems simply check whether the code satisfies the proof.

If the check fails, the user can decide to refuse the CONNECTor or to enforce the private security policy on it (Enforce Policies) by exploiting the *run-time enforcement* infrastructure (e.g., [7]). Otherwise, the Networked System can proceed to verify whether the contract (correctly representing the application) satisfies the private security policy (*Contract-Policy matching [13]* at deployment-time). Once again, if this step fails, the solution consists in enforcing the private security policy on the execution. Finally, if the previous checks were positively passed, the communication is established without any run-time monitoring (Execute Application).

3.3 Trust

Thanks to CONNECT Enablers, Networked systems get CONNECTed via CONNECTor(s) that can be composed and reused.

Thus, the CONNECT trust model is defined to allow: (i) Enablers to safely cooperate in order to build and deploy CONNECTors, (ii) Enablers to assess CONNECTor trustworthiness and hence provides CONNECTed systems with the most trusted CONNECTor and (iii) handle monitoring feedbacks to fairly update the trustworthiness of both Enablers and CONNECTors.

Enablers Assessment. In order to assess the trustworthiness of each Enabler, the CONNECT trust model applies a reputation mechanism. The trust reputation of each Enabler is computed from trust relations among Enablers and also from the Enabler's behaviour. The CONNECT trust model assesses for each Enabler a trust reputation value through a decentralised reputation mechanism. Thus, the reputation of each Enabler is managed by other Enablers, which have been selected with a distributed hash table, such as in CAN [20] or Chord [24]. We use several hash functions to replicate the reputation of each Enabler. This prevents against malicious Enablers and also keep the system more resistant to inherent dynamic network behaviour, namely, Enablers that unexpectedly disconnect.

Connector Assessment. In the CONNECT trust model, Synthesis Enablers assess the CONNECTors they produce by computing a trust recommendation value, which results from (i) the trust on the CONNECTor based on previous deployment and also from (ii) the trustworthiness of all the Enablers that are involved in the specific synthesis. The deployment history of each CONNECTor is maintained by its Synthesis Enabler. We take inspiration from trust assessment in Web Service composition [17, 19, 15], in which all trust relations that are involved in this composition are aggregated and composed in order to return the trust recommendation of the whole composition.

Feedbacks management. The CONNECT trust model has to update fairly (increase or decrease) the trustworthiness of each CONNECT stakeholder (i.e., the reputation of the involved Enablers and the trustworthiness of the Enabler on its synthesised CONNECTor) after each CONNECTor deployment. Thus, the CONNECT trust model deals with two parameters: (i) the degree of involvement (i.e., responsibility) of each Enabler in the process of synthesising and running CONNECTors and (ii) the recommendation value that is given by each Enabler for its contribution to the CONNECTor. Indeed, we consider that each Enabler must be rewarded or penalised proportionately to both its involvement and the value of its given recommendation. Therefore, in course of time, the CONNECT trust model will be able to identify trustworthy Enablers and hence will provide more efficient and relevant CONNECTors. However, after a while, the CONNECT system will mostly solicit Enablers with a high reputation (i.e., good history). This will preclude newcomers (without history) by making their participation to the running system very difficult or even impossible. Thus, in order to allow CONNECT networks to evolve with new capable Enablers, we endow the CONNECT trust model with an incentive Risk-based property, in which, Enablers that have a high reputation are pushed to reduce their recommendation values in order to maintain their reputation (i.e., minimise the penalisation). To implement this incentive property we use *The Behaviour function* that is defined in [21]. Thus, by adopting this behaviour, everyone wins. On the one hand, the entities with high reputation will save their position, and on the other hand, this incentive behaviour will boost the bootstrapping phase by giving the opportunity to new legitimate Enablers to be considered by CONNECT.

3.4 Monitoring

The very vision of CONNECT, i.e., achieving automated and eternal interoperability puts monitoring in a central position for the overall project. In CONNECT, monitoring is conceived as a common core service used by the other Enablers to implement feedback loops whereby approaches that are normally used off-line (e.g. techniques for dependability analysis, CONNECTor synthesis, behaviour learning) can be adapted to an on-line setting and can be enhanced to cope with change and dynamism. Monitoring is performed alongside the functionalities of the CONNECTed system and is used to detect conditions that are deemed relevant by its clients (i.e., the other CONNECT Enablers). Upon detecting the occurrence of such conditions, the monitoring system alerts the interested client which, in turn, triggers an update of the analysis, synthesis, and learning. In this way, powerful but expensive techniques are executed only when necessary.

As we intend to realise a monitoring system that can address different purposes, covering both functional and non-functional aspects, it must be designed with special emphasis on flexibility. Furthermore, although monitoring can provide valuable support to dependability assurance, it can easily incur in feasibility problems caused by excessive overhead. To cope with this issue, the performance penalty due to monitoring should be minimised, while achieving the intended observation goals. Approaches for reducing the impact of monitoring include using statistical sampling or self-tuning algorithms for directing the focus of monitoring to certain parts of the overall monitored system that are deemed especially critical or interesting [4]. As a matter of fact, most existing monitoring systems follow a best-effort policy, whereby overhead is kept as low as possible but is in fact unbounded. In CONNECT, we pursue efficiency by adopting predictable strategies to estimate the computational, storage and transmission resources that are demanded by a given set of monitoring goals. This means that the load caused by monitoring will not only be limited, but also predictable and controllable, along the lines of the approach advocated in [6]. Providing a reasoning framework to handle the trade-offs between monitoring precision and efficiency is one of the goals of our research in the remainder of the project.

4. WORKFLOW PROCESS VIEW

In CONNECT, the four activities related to dependability span over all stages of the CONNECT process, i.e., *discovery time*, *synthesis time*, and *execution time*. Combining and integrating the activities within the overall CONNECT process is a complex iterative endeavour. During this first year of the project, we envisaged a dependability-centric scheme of the CONNECT process. The high-level description of the scheme is depicted in Figure 2. In the figure, the CONNECT process is modelled as an activity diagram with separate swim-lanes. There is one swim-lane for each activity related to dependability, plus an additional swim-lane "other CONNECT activities", which represents the rest of the CONNECT

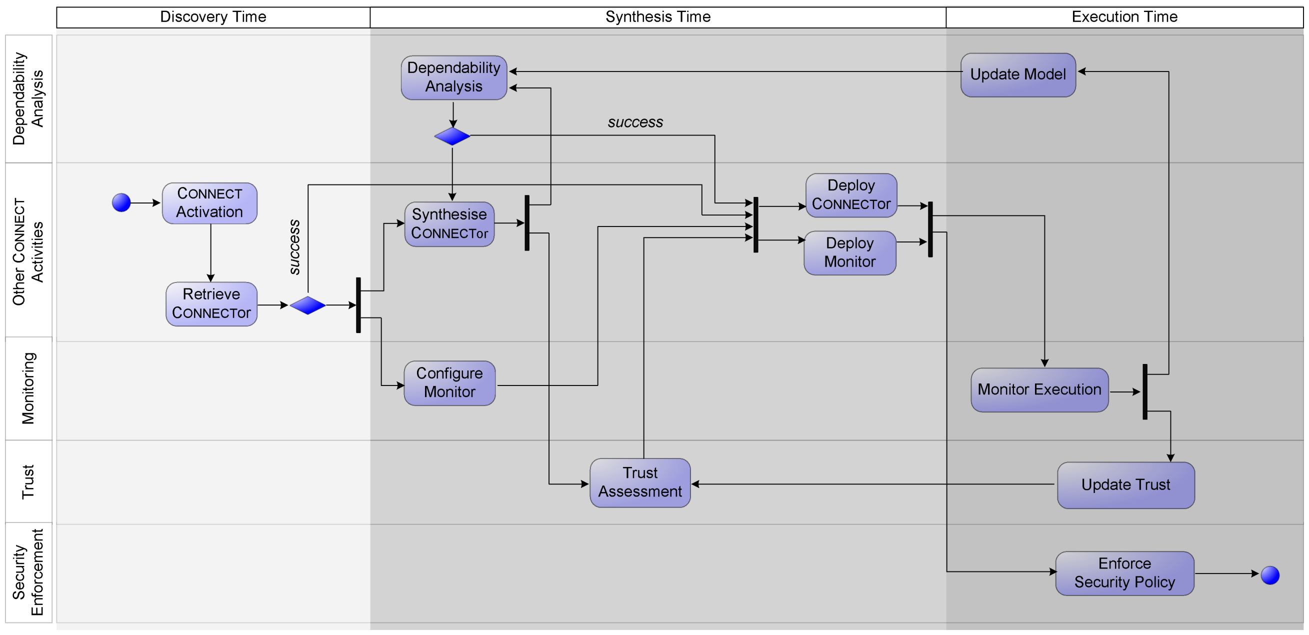

Figure 2: Life-cycle workflow of activities within the CONNECT process

activities other than dependability-related ones. Along this generic swim-lane, the CONNECT approach is triggered by a request of communication. This request, originating from a Networked System *S1*, is intercepted by CONNECT Discovery Enablers.

To serve *S1* request, we have here a first decision point: does there already exist a suitable CONNECTor that can be reused? So, when a Discovery Enabler *E* accepts the request, it will first search for a suitable CONNECTor already synthesised. *Suitable* here means that, among all the functional and non-functional characteristics of the CONNECTor, this yields adequate dependability properties and a trust level at least equal to the trust level associated to *E*. If the answer is positive, then the Enabler retrieves an implementation of the CONNECTor from a repository and deploys it, CONNECTing *S1* with a receiver Networked System *S2*. Moreover, the deployed CONNECTor is provided with a monitor that at execution time will warn the Enabler when and if the established communication is no longer satisfying *S1* needs (this may happen for many reasons: because either the environment or *S2* changed, or because the trust reputation decays, or due to negative reports from on-line analysis).

If no suitable CONNECTor is available, then the synthesis process is started by the Synthesis Enabler. The latter may interact with the Learning Enabler to infer the desired functional behaviour of the CONNECTor, and may also obtain some dependability requirements from *S1* interface description. Hence, during synthesis, the Enabler will interact with the Dependability Analysis Enablers to predict whether the built CONNECTor is satisfacing[2].

[2]The word *satisfice*, coined by Herbert Simon, blends “satisfy” and “suffice”, to highlight the aim to meet criteria for adequacy, rather than to identify an optimal solution.

At execution time, the monitoring mechanism is activated to keep track of the CONNECTor behaviour (monitoring at the Networked Systems interfaces) and of the CONNECTed System (end-to-end). Additionally, when security specifications are provided (security-by-contract), security enforcement mechanisms are activated at execution time. This discover and synthesis flow cycles whenever the communication is no longer satisficing. Throughout, the trust management model is pervasive; among the available Enablers, those yielding the highest trust reputation can be chosen, and trust reputation is updated at execution time according to monitoring feedback.

5. CONCLUSIONS

We have briefly introduced the European Project CONNECT and have focused on the unifying dependability framework that is currently under development [9]. The concept of dependability in CONNECT is quite broad, and includes four main activities: model-based V&V, security enforcement and privacy, trust management, and monitoring. A first proposal of a unified dependability-centric life-cycle has been outlined. The life-cycle spans over three phases of the CONNECT process (discovery time, synthesis time and execution time) and points out the role of the different activities related to dependability. At the time of writing, the CONNECT project has just concluded its first year, during which we focused on devising appropriate models and background material for the various dependability concerns. In the remaining years of the project, we will focus on developing the workflow process explained in this paper, which integrates different approaches to address the dependability challenges of dynamic, evolving, heterogeneous systems.

6. ACKNOWLEDGEMENTS

This work is partially supported by the European Commission Framework Program 7 Project CONNECT (FP7–231167)

7. REFERENCES

[1] Dependable Systems of Systems (DSoS) EU FP5 Project, 2000–2003.

[2] European Network of Excellence ReSIST . http://www.resist-noe.org/, 2006–2009.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable and Secure Computing*, 1(1):11–33, 2004.

[4] A. Bertolino, G. D. Angelis, A. Sabetta, and S. G. Elbaum. Scaling up sla monitoring in pervasive environments. In A. L. Wolf, editor, *ESSPE*, pages 65–68. ACM, 2007.

[5] A. Bondavalli, S. Chiaradonna, and F. Di Giandomenico. Model-based evaluation as a support to the design of dependable systems. In H. B. Diab and A. Y. Zomaya, editors, *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, pages 57–86. Wiley, 2005.

[6] S. Callanan, D. Dean, M. Gorbovitski, R. Grosu, J. Seyster, S. Smolka, S. Stoller, and E. Zadok. Software monitoring with bounded overhead. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.

[7] A. Castrucci, F. Martinelli, P. Mori, and F. Roperti. Enhancing java me security support with resource usage monitoring. In *ICICS*, pages 256–266, 2008.

[8] CONNECT Consortium. Deliverable 2.1 – Capturing functional and non-functional connector behaviours *(available soon)*, 2010.

[9] CONNECT Consortium. Deliverable 5.1 – Conceptual models for assessment and assurance of dependability, security and privacy in the eternal CONNECTed world *(available soon)*, 2010.

[10] G. Di Marzo Serugendo. Robustness and dependability of self-organizing systems - a safety engineering perspective. In R. Guerraoui and F. Petit, editors, *SSS*, volume 5873 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2009.

[11] N. Dragoni, F. Martinelli, F. Massacci, P. Mori, C. Schaefer, T. Walter, and E. Vetillard. Security-by-contract (SxC) for software and services of mobile systems. In *At your service: Service Engineering in the Information Society Technologies Program.* MIT Press, 2008.

[12] N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan. Security-by-contract: Toward a semantics for digital signatures on mobile code. In *EuroPKI*, pages 297–312, 2007.

[13] P. Greci, F. Martinelli, and I. Matteucci. A framework for contract-policy matching based on symbolic simulations for securing mobile device application. In *ISoLA*, pages 221–236, 2008.

[14] E. Hollnagel, D. D. Woods, and N. Leveson. *Resilience engineering: concepts and precepts.* Ashgate Publishing, Surrey, 2006.

[15] Y. Kim and K. Doh. Trust Type based Semantic Web Services Assessment and Selection. *Proceedings of ICACT, IEEE Computer*, pages 2048–2053, 2008.

[16] J. Laprie. From dependability to resilience. In *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, 2008.

[17] S. Nepal, Z. Malik, and A. Bouguettaya. Reputation propagation in composite services. In *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*, pages 295–302, Washington, DC, USA, 2009. IEEE Computer Society.

[18] D. M. Nicol, W. H. Sanders, and K. S. Trivedi. Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1:48–65, January-March 2004.

[19] S. Paradesi, P. Doshi, and S. Swaika. Integrating behavioral trust in web service compositions. In *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*, pages 453–460, Washington, DC, USA, 2009. IEEE Computer Society.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[21] R. Saadi, J. M. Pierson, and L. Brunie. T2D: A Peer to Peer trust management system based on Disposition to Trust. In *25th ACM Symposium On Applied Computing (SAC).* ACM Press, 2010 (to be appear).

[22] R. Sekar, C. R. Ramakrishnan, I. V. Ramakrishnan, and S. A. Smolka. Model-Carrying Code (MCC): a New Paradigm for Mobile-Code Security. In *NSPW '01: Proceedings of the 2001 Workshop on New security paradigms*, pages 23–30, New York, NY, USA, 2001. ACM Press.

[23] R. Sekar, V. Venkatakrishnan, S. Basu, S. Bhatkar, and D. C. DuVarney. Model-carrying code: a practical approach for safe execution of untrusted applications. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 15–28, 2003.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM.