CONNECT Challenges: Towards Emergent Connectors for Eternal Networked Systems

Valérie Issarny, Bernhard Steffen, Bengt Jonsson, Gordon Blair, Paul Grace, Marta Kwiatkowska, Radu Calinescu, Paola Inverardi, Massimo Tivoli, Antonia Bertolino, et al.

CONNECT Challenges: Towards Emergent Connectors for Eternal Networked Systems

Valérie Issarny
INRIA, France
Valerie.Issany@inria.fr

Bernhard Steffen
Technische Universitat Dortmund, Germany
steffen@cs.uni-dortmund.de

Bengt Jonsson
Uppsala Universitet, Sweden
bengt@it.uu.se

Gordon Blair,
Paul Grace,
Lancaster University, UK
p.grace@lancaster.ac.uk,
gordon@comp.lancs.ac.uk

Marta Kwiatkowska,
Radu Calinescu
University of Oxford, UK
Marta.Kwiatkowska@comlab.ox.ac.uk,
Radu.Calinescu@comlab.ox.ac.uk

Paola Inverardi,
Massimo Tivoli
Universita dell'Aquila, Italy
Paola.Inverardi@gmail.com,
Massimo.tivoli@gmail.com

Antonia Bertolino,
Antonino Sabetta
CNR, Pisa, Italy
antonia.bertolino@isti.cnr.it,
antonino.sabetta@isti.cnr.it

***Abstract*—The CONNECT European project that started in February 2009 aims at dropping the interoperability barrier faced by today's distributed systems. It does so by adopting a revolutionary approach to the seamless networking of digital systems, that is, synthesizing on the fly the connectors via which networked systems communicate. CONNECT then investigates formal foundations for connectors together with associated automated support for learning, reasoning about and adapting the interaction behavior of networked systems.**

I. INTRODUCTION

Our everyday activities are increasingly dependent upon the assistance of digital systems that pervade our living environment. However, the current ubiquity of digital systems is technology-dependent. The efficacy of integrating and composing networked systems is proportional to the level of interoperability of the systems' respective underlying technologies. This leads to a landscape of technological islands of networked systems, although interoperability bridges may possibly be deployed among them. Further, the fast pace at which technology evolves at all abstraction layers increasingly challenges the lifetime of networked systems in the digital environment.

The CONNECT project that started in February 2009 aims at dropping the heterogeneity barriers that prevent networked systems from being eternal, thus enabling the continuous composition of networked systems to respond to the evolution of functionalities provided to and/or required from the networked environment, independently of the embedded software technologies. CONNECT specifically targets the dynamic synthesis of connectors via which networked systems communicate. The resulting *emergent connectors* (or CONNECTors) then compose and further adapt the interaction protocols run by the connected systems, which realize application- down to middleware-layer protocols.

The CONNECT synthesis process relies on a formal foundation for connectors, which allows learning, reasoning about and adapting the interaction behavior of networked systems. CONNECT operates a drastic shift by learning, reasoning about and synthesizing connector behavior *at run-time*. Indeed, the use of connector specifications pioneered by the software architecture research field has mainly been considered as a *design-time* concern, for which automated reasoning is now getting practical even if limitations remain. On the other hand, recent effort in the semantic Web domain brings ontology-based semantic knowledge and reasoning at run-time but networked system solutions based thereupon are currently mainly focused on the functional behavior of networked systems, with few attempts to capture their interaction behavior as well as non-functional properties. In the approach taken by CONNECT, the interaction protocols (both application- and middleware layer) behavior will be learnt by observing the interactions of the networked systems, where ontology-based specification and other semantic knowledge will be exploited for generating CONNECTors on the fly.

CONNECT raises a set of unique challenges in the area of software systems engineering, from theoretical foundations to specify the interaction behavior of networked systems to run-time methods and tools to turn specifications into running protocols, and vice versa. This paper surveys key challenges addressed by CONNECT, discussing the research background and required advance over state of the art. In more detail, the paper first stresses the challenge of interoperability that increasingly needs to be overcome by networked systems (Section II). This calls for a paradigm shift that goes beyond today's middleware solutions and effectively lies in the dynamic synthesis of emergent connectors. Related research challenges are then sketched, spanning theory for emergent connectors (Section III), and dynamic connector behavior learning (Section IV) and synthesis (Section V). CONNECT is further concerned with ensuring dependability of the overall synthesis process as well as experimenting with actual case studies (Section VI).

II. THE CHALLENGE OF INTEROPERABILITY

Interoperability is defined by Tanenbaum and van Steen [1] as "the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other's services as specified by a common standard". This is actually the

principal goal of a middleware solution and, indeed, if common standards were agreed and adopted universally then this problem would be largely solved. However, in practice, different standards inevitably co-exist for either technical, commercial, or legacy reasons. This implies that solutions for interoperability must be found across different domains where different middleware solutions have been implemented. More specifically, interoperability must be achieved in several key areas such as *discovery* where multiple service discovery protocols co-exist (e.g., UPnP, SLP, Jini, etc); *interaction* between different interaction styles such as RMI and publish-subscribe; and to achieve end to end *quality of service* in highly heterogeneous environments.

A. State of the Art

In terms of tackling heterogeneous discovery, a number of experimental middleware have emerged, among which are proposals by the CONNECT partners. ReMMoC [2] from Lancaster University is a client-side middleware that uses environmental information to dynamically adapt its behavior to ensure that lookup requests are sent using all appropriate protocols (in parallel). Reference [3] identifies that the range of discovery protocols in ad hoc networks is equally diverse and presents an extension to ReMMoC that copes with this cross-domain heterogeneity. INDISS [4] from INRIA deploys composers and parsers that translate discovery requests and responses between applications tied to individual discovery protocols, hence, legacy UPnP and SLP applications are visible to one another. MUSDAC [5] from INRIA leverages bridges that translate requests from one network domain to another in order to tackle heterogeneous discovery across broader deployments. Similarly, the Open Service Discovery Architecture [6] is a cross-domain discovery middleware for wide-area networks. A peer-to-peer indexing overlay is utilized to create an inter-domain and inter-network space where services are advertised and queries are solved. All of these solutions utilize syntactic matching for discovery, which is problematic because even simple differences in the data format may lead to a discovery failure. An alternative is to abstract syntactic differences by expressing the semantics of the data in a logical formalism such as OWL [7]. Attempts to use semantics in the definition of services take as starting points the semantic Web services efforts and the discovery algorithms, such as [8] that have been proposed in that context. S-Ariadne [9] from INRIA is a semantically enriched service discovery protocol, which allows Web Services to be discovered using both semantic behavior and Quality of Service (QoS) information, rather than just the syntactic information described in the WSDL description. This solution illustrates how semantics can be utilized with individual middleware technologies, rather than integrated within the middleware to better support the performance of the system.

Interoperability between middleware using heterogeneous communication protocols is a long-standing problem. Initial solutions were restricted to a single middleware framework, e.g., CORBA's interoperability framework based on GIOP/IIOP [10]. Solutions based on software bridges then achieved bi-directional interoperability between distributed object technologies [11]. Model Driven Architecture (MDA) [12] also provides (partial) support for interoperability and integration throughout the systems lifecycle. MDA defines how to model systems in terms of platform-independent functionalities (PIM), separated from platform specific (PSM) ones, with the goal to generate solutions for a given middleware. Through this approach, services can be deployed on particular platforms, and interoperability bridges are statically generated before deployment. In contrast, Uniframe [13] and uMiddle [14] present dynamic bridging based upon translations to a common intermediate format. Vinoski [15] provides an interesting perspective on such mapping problems between middleware platforms, and PKUAS proposed an architecture-driven approach to guiding the interoperability protocol substitution [16]. Service-oriented middleware offers alternative approaches to interoperability, effectively used to span between different domains in a complex system of systems. This approach achieves interoperability by mandating a particular protocol at the bridging level, which inevitably ties the service to a single messaging platform (i.e., SOAP), preventing interoperability with alternate platforms. To address this problem Enterprise Service Buses (ESB) specify a service-oriented middleware with a message-oriented abstraction layer atop different messaging protocols (e.g., SOAP, JMS, SMTP); databases, Web Services, or JMS queues are hosted with a messaging endpoint, and the bus translates data messages to the appropriate message type of the service being interoperated with. Example ESBs are Artix[1], BEA Aqualogic[2], and IBM Websphere Message Broker[3]. Services are discovered using UDDI; clients search UDDI servers for services that match their functional service requirement. In addition, a number of solutions utilize the mobile code capabilities of Java to support interoperability. For instance, Jini [4] supports the downloading of components on to the client platform to provide appropriate access to networked resources. This can include the use of particular underlying protocol solutions in a manner that is transparent to the user, hence enabling interaction with multiple styles of devices/systems. Reflective middleware also presents a framework and underlying set of mechanisms to partially tackle interoperability. In such platforms, the middleware communication protocol can be dynamically selected depending upon the communicating endpoint and context. Hence, if the service is implemented atop CORBA, the client adapts itself to use the IIOP communication protocol; ReMMoC [2] follows this philosophy.

However, there are still profound problems that are virtually un-researched. Whereas the vast majority of work on middleware support for quality of service is targeted at achieving a given property such as security, timeliness or dependability in a given domain, there has been very little

[1] http://www.iona.com/products/artix/
[2] http://dev2dev.bea.com/aqualogic/
[3] http://www-01.ibm.com/software/integration/wbimessagebroker/
[4] http://www.sun.com/software/jini/

research addressing Interoperability and QoS in terms of achievement of QoS properties spanning multiple properties and domains. A second open problem is Semantic interoperability. Protocol level interoperability guarantees that two or more systems can exchange messages, but it does not guarantee that the receiving systems can process the messages that have been sent. To address this problem, semantic mediators have been proposed, e.g. [67,68], but a theory and knowledge on the automatic generation of such mediators is still missing.

B. Baseline to CONNECT Interoperability

With respect to achieving the goals of interoperable middleware in CONNECT, the work will build upon the key cutting-edge technologies from the state of the art that go beyond a common interoperation format, or static bridging. Notably, the CONNECT partners have strongly influenced this field:

- Lancaster University [2, 3] use reflection and component architectures to tackle discovery and communication interoperability within and across diverse application domains.
- INRIA [4, 5, 9] investigate software architecture techniques, protocol translation, semantic discovery and dynamic bridging to connect discovery and interaction protocols.
- Peking University [16] use reflection and architectural models to dynamically ensure communication interoperability.

These dynamic solutions will be analyzed for improvements, and their principles and technologies will aid in the creation of a richer interoperation architecture, which captures broader protocol styles, semantic interoperability and end-to-end non-functional properties maintained across domains.

C. Research Challenges

As can be seen from the review above, there has been a considerable volume of research on interoperability in distributed systems. Equally, it can be seen that while progress has been made, the state of the art remains rather patchy, particularly when addressing the complexity of contemporary, highly heterogeneous distributed systems. CONNECT will make a significant impact on the state of the art by (1) *identifying a common framework* to deal with interoperability in highly heterogeneous systems covering discovery, interaction and quality of service; (2) *tackling the problem in all its dimensions* by considering the various levels of abstraction in a system and across multiple domains; (3) as such, *broadening the scope of interoperability solutions* to encompass pervasive computing (cf. the current central focus on enterprise computing); (4) developing techniques to *ensure semantically correct discovery and interaction*; (5) *seeking transparent solutions* based on automatic synthesis of underlying middleware. These are big challenges and areas where solutions can really impact on and revolutionize the state of the art in distributed systems in general and middleware more specifically. CONNECT will address those challenges by abstracting the behavior of networked middleware based on a formal foundation for connectors, and further synthesizing interaction behaviors that solve mismatch arising among the protocols run by the middleware.

III. TOWARDS A THEORY FOR EMERGENT CONNECTORS

The CONNECT foundations lie in a theory for connectors, which will enable learning, reasoning, analyzing, and synthesizing interaction behaviors that solve interaction protocol mismatches arising among communicating networked systems.

A. State of the Art

Within the Software Architectures (SA) community, a number of techniques have been proposed for the analysis, monitoring, detection/recovery of system anomalies, and software system synthesis [17,18]. The first frameworks introducing the notion of a software connector were proposed in early 1990s in the seminal works of [19] and [20]. Connectors facilitate the conceptual separation between computation (i.e., system components) and coordination (i.e., interaction protocol), defining a set of *roles* that specific named entities of the components must *play* plus a *glue* specification. Each role determines the obligations that the interacting parts have to fulfill to become instances of the role (an interface). The glue describes how the activities of the role instances are coordinated (i.e., it describes the interaction protocol that the role instances have to follow to properly interact). Typically, interaction protocols deal with a relatively small set of basic interaction primitives, such as synchrony/asynchrony, mutual exclusion, atomicity, ordering, etc. A related notion is that of interface automata [21]. Following the original presentation in [20], where connectors were formalized using the CSP process algebra, the majority of existing formalizations are based on process algebras. In different formalisms (e.g., Reo, BIP), connectors may vary depending on the interaction capabilities (direct/indirect message passing, exogenous) and whether they encapsulate computation. In [22] (and references therein) concepts such as component composition, behavior, and interaction protocol or glue code are formally defined. In [23], both computation and interaction are reduced to a common low-level formalism, whereas other approaches study interaction separately from computation, with a separate (higher-order) algebraic framework to study interactions [24,25]. Virtually all formalizations of connectors consider their functional behavior only, where connectors are represented as relations between ports [27], each with synchronization types, and system executions are induced from the classical labeled transition system (LTS) obtained from the operational semantics rules.

There are a number of functional properties that can be established for architectural descriptions. For *basic* interactions, these typically focus on compatibility (i.e., safety of communication via a particular interaction primitive) and avoidance of mismatches, e.g., deadlocks [26]. Another issue is that SAs specified using connectors are often complex and hierarchical, and therefore not amenable to an exhaustive state-space exploration of the full LTS but

instead a divide-and-conquer approach called *assume-guarantee reasoning* should be used. In an assume-guarantee approach [27], a component is considered in conjunction with the context (for example, another component or the environment) and it is verified under the assumption that the context behaves correctly, without the need to build the full LTS of the context. At the University of Oxford, a variant of assume-guarantee reasoning called protocol conformance verification was developed for CSP/FDR model checking [28]. At the University of L'Aquila an assume-guarantee approach has been proposed to compositionally verify middleware based applications [29].

Automated verification via model checking is firmly established as a tool for hardware and protocol verification, and more recently software, but has had limited application in the SA domain. For non-functional properties, efficient algorithms have been devised as part of the PRISM[5] model checker and extensively tested on case studies from the networking/QoS domain [30], and many others. However, all the protocols analyzed were static and the adopted modeling language does not have the expressive power required for connectors (e.g., the broadcast primitive, higher-order connectors).

B. Baseline to the CONNECT Foundations

The above approaches can be considered as a starting point for CONNECT, to enable reasoning about functional interoperability. However, within the CONNECT scenario, we must also take into account the many dimensions (not only functional behavior) of the connected components' interaction spanning from middleware- up to application-layer.

Key to successful outcome is a formalization that supports *compositional*, *adaptive* architectures and reasoning. We will base our investigation on the static connector structure [20] and its extension to the adaptive systems domain in [31]. We will use these together with [25], which admits a notion of equivalence for connectors. As already proposed in [20], compatibility can be formulated in terms of CSP process refinement (or alternatively simulation) and can be automatically checked by systematic state-space exploration with the help of model checkers such as FDR for CSP or similar. We are aware of only three model checking approaches for connectors [32,33,69] modeled as constraint automata [22]. For more complex interactions, one has to consider the unfolding of the underlying LTS into executions and perform temporal logic model checking [34] to establish richer properties, such that every request will eventually be granted.

Connectors should not only guarantee functional interoperability, but also *non-functional* properties. The latter require the enhancement of the formalism/model with quantitative information, such as the time an activity takes (hard deadline) or the probability that it is received by the deadline (soft deadline, QoS). We will use quantitative extensions of constraint automata with time [35] and QoS [36] that have been formulated, and specifically model checking algorithms for compatibility devised for the timed case [22]. However, in order to provide automated analysis of a full range of non-functional properties we must involve a *quantitative* verification approach [37,38,39] (pioneered at the University of Oxford). We will extend the quantitative algorithms of [30,40] which establish, for example, the expected probability or cost to reach a goal. We will also formulate an assume-guarantee reasoning paradigm for quantitative verification, currently in early phases of development [41]. An alternative to achieve scalability for assume-guarantee is by means of abstraction and approximation. In this direction, we will use modular analysis techniques for timing and resource properties being implemented at the University of Uppsala. This work originates from the work on the Real Time Calculus developed at ETHZ [42].

C. Research Challenges

CONNECT will find appropriate and sufficiently expressive formalisms for connectors, both to describe a broad range of interaction primitives, as well as to quantify the desired QoS levels for end-to-end properties among the networked systems. Furthermore, such non-functional specifications should be integrated/compatible with the formal functional definition, to allow not only for designing/modeling but also learning/reasoning/synthesis, and later comprehensive and efficient validation. From the SA perspective, the formalism must facilitate the compositional reuse of sub-parts of already synthesized and composed CONNECTors, hence providing mechanisms that keep traceability all along the steps of the composition process. CONNECT will devise such a formalism and its quantitative extension, together with algorithms and proof-of-concept implementation of automated compatibility checking for connectors, both functional and non-functional properties. This will serve as a foundation for dynamic CONNECTor synthesis and interaction behavior learning.

Besides providing a formal foundation for connectors enabling connector behavior learning/reasoning/synthesis, CONNECT will introduce associated enablers, which will concretely support learning the interaction behavior of networked systems (Section IV) and further generating CONNECTors at run-time (Section V).

IV. LEARNING CONNECTOR BEHAVIOR

The difficulties and overhead involved in formulating specifications of program components have spurred an interest in developing techniques for automated support in generating or "discovering" such specifications. Some techniques generate specifications as abstract models of source code through static analysis of source code. However in the CONNECT setting, networked systems are not assumed to provide access to source code or detailed formal behavioral specification, and must in the general case be analyzed by observing their external behavior. Then, dynamic inference techniques are the only means to gain information about the behavior of systems.

[5] www.prismmodelchecker.org

A. State of the Art

Dynamic inference, or learning, techniques have been developed and adapted in order to generate specifications of reactive component behavior. These techniques have been demonstrated to provide useful support for various software engineering tasks, including bug finding, test suite generation and evaluation, formal verification, specification generation, and software maintenance. The task of learning component behavior as accurately as possible can be performed using techniques for *regular inference*. Such techniques have been used, e.g., to create models of environment constraints with respect to which a component should be verified [43], for regression testing to create a specification and a test suite [44], for program analysis [45], and for formal specification and verification [46]. In regular inference, one infers a regular language (in the form of a deterministic finite automaton, DFA) from the answers to a finite set of membership queries, each of which asks whether a certain word is in the language or not. There are several techniques, which use essentially the same basic principles. Given "enough" membership queries, the constructed automaton will accept the "correct" language. A check whether the regular inference procedure is completed can be abstractly represented by a so-called equivalence query [47], and concretely carried out by, e.g., run-time monitoring or conformance testing [70]. To respect the difference between input and output events, inference of Mealy machines has been developed by the University of Dortmund [48]: the underlying principles of inference algorithms remain the same. More recent work concerns relational model construction [49], synthesis of design models from scenarios, with human interaction [50,71], and further optimizations of regular inference techniques for assume-guarantee reasoning [51].

Another application for dynamic inference techniques is in extracting interesting properties of a component, often selected from a predefined set: these properties are useful in bug finding, software updates, validation, etc. The properties of interest are typically simple, often pattern-based, and the main problem is to detect the properties from a lot of complex "noise" in execution traces, rather than condensing all available information in an automaton model. A conceptually simple approach considers invariants that describe, e.g., ranges of program variables or simple relationships between variables, exemplified by the Daikon system [52]. Extensions of these works consider simple temporal patterns, such as the pairing of calls and returns, and acquisitions and releases of locks [53]. More complex temporal properties have also been considered, such as simple forms of scenario descriptions, or automata.

B. Baseline to Learning CONNECTor Behavior

Perhaps the most versatile state of the art implementation of dynamic inference techniques is the LearnLib toolbox, developed mainly at the University of Dortmund. Because of the increasing number of variants addressing profile specific needs, LearnLib is based on a framework for the construction of learning algorithms [50,72,75]. It encompasses a number of algorithmic modules, which can be effectively combined into powerful profile-specific learning algorithms. Besides components for different types of models, like DFAs and Mealy machines, and different practical variants for approximating the equivalence queries, the LearnLib is known for its library of optimizations [73,74], which may have a huge practical impact, hinting towards the applicability of learning technology to realistic system sizes.

CONNECT will need techniques for synthesizing connectors that interact using primitives with data parameters, and which take QoS properties, such as latency and throughput, into account. There are some preliminary approaches (e.g., [54]) on extending the regular inference paradigm with data, and work is in progress to develop them into a practical approach. A theoretical basis for handling latency and other timing properties has been provided by the University of Uppsala in their extension of regular inference techniques to timed automata [55].

C. Research Challenges

The capability to learn from observed behavior and to extrapolate with a good degree of confidence from the observed to a generic behavior is a central prerequisite for the synthesis of CONNECTors. In particular, protocol adapters rely on hypothesis models of the external behavior of the involved parties.

Existing approaches to learning the behavior of modules, whose specification is not given, assume that the module's interface, i.e., the set of events or ports that the module can use for interaction, is known to the learning algorithm. In the CONNECT setting, this interface cannot be assumed to be given a priori. A challenge in CONNECT is to develop protocols for behavior discovery, which employ a bootstrapping process; information about component interfaces need to be first obtained e.g., by a reflection mechanism, and then successively used and refined during the learning of the module behavior. An alternative to interface specifications in terms of signatures and types are `semantic specifications´ based on ontologies, which allow an easy symbolic treatment of compatibility from an abstract service perspective rather than a concrete programming point of view. Practical solutions will have to combine this semantic treatment with techniques classically used at the programming level.

CONNECT will develop techniques for synthesizing connectors that interact using primitives with data parameters, and which take QoS properties into account. Based on previous work, a practical approach will be developed, which can cope with data parameters in interaction primitives, as well as challenges in handling timing properties, such as limitations in the precision of measurement and finding a scalable timing representation.

Main theme for the learning technology to be developed for CONNECT is the ability to deal with incomplete, imprecise and/or changing information, be it concerning the interface, the alphabet, and/or the data. Thus the learning process must be able to reshape already inferred (extrapolated) models according to new insights/situations. In fact, learning needs to be considered a continuous process of observation

/monitoring, and corresponding model updates. In practice, this cannot be fully automated since new insights and observations may have an impact on the models' level of abstraction, and at least in the beginning, the arising abstraction/refinement process will have to be moderated by some expert, whose guidance may also be important for guaranteeing a good learning performance.

Practical impact of the intended incremental learning technology is the continuous availability of hypothesis models that concisely and consistently reflect the current knowledge about the system to be learned.

V. DYNAMIC CONNECTOR SYNTHESIS

With the aim of achieving eternal universal interoperability among heterogeneous networked systems, a key concept of CONNECT is synthesizing new interaction behaviors out of the ones implemented by the systems to be made interoperable, and further generating corresponding run-time CONNECTors to bridge protocols. In the following we discuss those approaches to the automatic connector[6] synthesis that have been developed during the last twenty years.

A. State of the Art

The first approaches to connector synthesis appeared in the 90s in the control theory domain [56]. The aim of these approaches is to automatically synthesize a controller whose aim is to restrict the behavior of the system so that it satisfies a given specification. Thereafter, these approaches have been revised to fit the domain of software (embedded) systems (e.g., [57]). As a common limitation, these approaches focus on failure prevention rather than on resolution issues. The idea of embedding the interaction protocol into the components by means of adapters has been introduced by [59] to solve incompatibilities between component interfaces. However this approach is not automatic. In, e.g., [59], LTSs are used to model the I/O behavior of the components and automatically synthesize a set of constraints on the components' environment that allow deadlock avoidance. A limitation of this approach is that the adapter actual code has to be written by hand by exploiting the synthesized constraints. Earlier approaches from the University of L'Aquila (e.g., [60, 61]), show how to automatically derive the (centralized or distributed) actual implementation of the adapter from a specification of the components' interaction and of the requirements that the composed system must fulfill. The adapter allows the prevention/resolution of incompatible interactions. However these approaches do not take into account high-level non-functional properties (e.g., security).

The University of L'Aquila proposed, in cooperation with others, an approach to the automatic adapter synthesis for real-time components [62]. Although this approach deals with both functional and non-functional information, its limit is that it synthesizes only a model of the adapter. Recently there has been a lot of interest in the formalization of wrappers for enforcing security properties [65]. An algebraic framework has been proposed in [63] to automatically synthesize a (centralized or decentralized) controller program for systems with several malicious components (whose behavior cannot be predicted a priori, e.g., at design-time). However, all these approaches suffer the *state explosion* phenomenon. Model Driven Engineering includes the concept of model transformation that enables the evolution and the automated synthesis of the system implementation from models. Still, one limit of the model-driven synthesis approaches is that the generated code is typically limited to code skeletons that may still require a very intensive coding activity.

B. Baseline to the CONNECTor Synthesis

With respect to the CONNECT goal, efficiency and dynamicity are crucial aspects of the CONNECTor synthesis process and represent the main innovations with respect to the state of the art and the main factors for a better exploitation of the synthesis approach. They might be achieved by following a compositional approach, e.g., following the assume-guarantee paradigm. To support compositional synthesis, we plan to define architectural CONNECTor patterns by considering as baseline the work described in [64], so that the CONNECTor may be possibly distributed in a set of wrappers. We intend to extend this approach to take into account also high-level non-functional properties of the networked systems' interaction by taking inspiration from [65]. In this respect, we consider as baseline also the work described in [61] and we plan to extend it by introducing model transformation techniques to automatically derive, from a networked system model, the assumptions that the networked system makes on its expected environment. Thus, taking inspiration from [66], the CONNECTor synthesis step may result in enforcing the behavior of the environment to satisfy those assumptions.

Furthermore, to achieve eternal interoperability, the CONNECTor code should be synthesized in a way that it can possibly evolve with respect to possible changes in the environment. Suitable reconfiguration mechanisms should be then investigated and combined with the monitors produced by the learning process. Compositionality will be then a critical property since it might allow an efficient synthesis step capable to (re-)build, at run-time, only the connector part affected by the change (i.e., re-synthesize or adjust the behavior of only some wrappers).

C. Research Challenges

The approaches considered as baseline for the CONNECTor synthesis exhibit four common limits: (i) they are static, i.e., possible run-time changes in the connector environment are not taken into account; (ii) they assume that the component comes together with the specification of its interaction protocol; (iii) they do not take into account the non-functional properties of the component interaction except for very low-level and domain-specific properties; (iv) they suffer the state explosion phenomenon. CONNECT overcomes these limitations promoting the development of automatic connector synthesis approaches that can be

[6] Depending on the application domain they are also called coordinators, adapters, converters, controllers, or wrappers.

efficiently performed at run-time. The resulting emergent connectors should enable the components to correctly interact with respect to their functional and non-functional characteristics that are inferred by relying on the CONNECT learning mechanisms.

VI. CONCLUSION

The core objective of the CONNECT project is to effectively support the dynamic synthesis of emergent connectors to overcome the increasingly high heterogeneity of the networking environment. This requires devising: (i) a semantic foundation of connectors enabling automated reasoning and synthesis of their behavior, and (ii) associated networked enablers, which are the actual actors of the connector generation dynamic process. Emergent connectors specifically result from a learning, reasoning and synthesis process that is able to elicit the "modus operandi" for carrying out the interoperable communication. This paper has discussed the CONNECT challenges associated with the elicitation of supporting formal foundation and associated learning and synthesis of connector behavior, in light of the relevant state of the art. Additional challenges will be investigated, which relate to dependability assurance for the overall synthesis process. Indeed, enabling the seamless networking of systems, as promoted by CONNECT, comes at risk from the standpoint of dependability. Two complementary issues then need to be addressed: (i) verification and validation techniques to ensure that networked systems as well as the generated bridging CONNECTors behave as specified with respect to functional and non-functional properties, and (ii) security, trust and privacy assurance for interacting parties in the open CONNECTed world. Finally, in order to assess the efficacy of the elaborated solutions, we will carry out experiment experimentation with future real-life scenarios, among which scenarios related to the European effort in the area of "Global Monitoring for Environment and Security" (GMES).

ACKNOWLEDGMENT

The CONNECT project (www.connect-forever.eu) acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the ICT theme of the Seventh Framework Programme for Research of the European Commission.

REFERENCES

[1] A. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, Prentice-Hall, ISBN 0-13-239227-5, 2007.

[2] P. Grace, G.S. Blair, and S. Samuel, A Reflective framework for discovery and interaction in heterogeneous mobile environments, ACM SIGMOBILE Mobile Computing and Communications Review, 9(1), 2005.

[3] C.Cortes, G.S. Blair, and P. Grace, An adaptive middleware to overcome service discovery heterogeneity in mobile ad hoc environments, IEEE Distributed Systems Online, 8(7), 2007.

[4] Y. Bromberg and V Issarny, "INDISS: Interoperable discovery system for networked services", IFIP/ACM/Usenix International Middleware Conference, Grenoble, France, November 2005.

[5] P. Raverdy, O. Riva, A. Chapelle, R. Chibout, and V. Issarny, Efficient context-aware service discovery in multi-protocol pervasive environments, 7th International Conference on Mobile Data Management (MDM'06), Nara, Japan, May 2006.

[6] N. .Limam, J. Ziembicki, R. Ahmed, Y. Iraqi, D. Li, R. Boutaba, and F. Cuervo, OSDA: Open service discovery architecture for efficient cross-domain service provisioning, Computer Communications, 30(3), 2007.

[7] D. L. McGuinness, and F. van Harmelen, "OWL Web Ontology Language, Overview"; W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210.

[8] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, Semantic matching of Web services capabilities, 1st International Semantic Web Conference, 2002.

[9] S. Ben Mokhtar, A. Kaul, N. Georgantas, and V. Issarny, Efficient semantic service discovery in pervasive computing environments, ACM/IFIP/USENIX 7th International Middleware Conference, Melbourne, Australia, 2006.

[10] Object Management Group, "Common Object Request Broker Architecture 3.0", OMG Document number Formal/02-12-02, 2002.

[11] Object Management Group, "COM/CORBA Interworking Specification Part A & B", OMG Document number ORBOS/97-09-07, 1997.

[12] J. Miller and J. Mukerji (eds.), "Model Driven Architecture", OMG Document number ormsc/2001-07-01, 2001.

[13] P. Shah, B. Bryant, C. Burt, R. Raje, A. Olson and A. Mikhail, Interoperability between mobile distributed components using the UniFrame approach, 41st Annual ACM Southeast Conference, Savannah, Georgia, 2003.

[14] J. Nakazawa, H. Tokuda, W. Edwards, and U. Ramachandran, A bridging framework for universal interoperability in pervasive systems, 26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), Lisbon, Portugal, 2006.

[15] S. Vinoski, S., It's just a mapping problem, IEEE Internet Computing 7(3), May 2003.

[16] G. Huang, H. Mei, Q. Wang, and F. Yang, A systematic approach to composing heterogeneous components", Chinese Journal of Electronics, Vol. 12, No. 4, 2003.

[17] D. Garlan, Formal Modeling and Analysis of Software Architecture: Components, Connectors, and Events. Springer, LNCS 2804, 2003.

[18] M. Bernardo and P. Inverardi (Eds), Formal Methods for Software Architectures, LNCS 2804. 2003.

[19] D. Perry and A. Wolf, Foundations for the study of software architectures, ACM SICSOFT Software Engineering Notes 17(4), 1992.40-52.

[20] R. Allen and D. Garlan, Formalizing architectural connection, ICSE'94. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[21] L. de Alfaro and T.A. Henzinger, Interface automata, ESEC/FSE 01, ACM Press, 2001.

[22] F. Arbab, C. Baier, M. Sirjani, and J. Rutten. Modeling component connectors in Reo by constraint automata. Science of Computer Programming, Elsevier, 61(2), 2006.

[23] F. Arbab, Abstract behavior types: a foundation model for components and their composition. Sci. Comput. Program. 55(1-3), 2005.

[24] R. Bruni, I. Lanese and U. Montanari. A basic algebra of stateless connectors, Theoretical Computer Science, 366, 2006.

[25] S. Bliudze and J. Sifakis, The algebra of connectors: structuring interaction in BIP, EMSOFT '07, ACM, New York, NY, 2007.

[26] D. Compare, P. Inverardi, and A. L. Wolf, Uncovering architectural mismatch in component behavior, Sci. Comput. Program, 33(2), 1999.

[27] R. Alur and T. A. Henzinge, Reactive modules, Formal Methods in System Design 15(1), 1999.

[28] X. Wang and M. Z. Kwiatkowska, On process-algebraic verification of asynchronous circuits, ACSD 2006.

[29] M. Caporuscio, P. Inverardi, and P. Pelliccione, Compositional verification of middleware-based software architecture descriptions, ICSE 2004. IEEE Computer Science Press.

[30] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, Symbolic model checking for probabilistic timed automata, Inf. Comput. 205(7), 2007.

[31] S. Cheng, D. Garlan, and B. Schmerl, Architecture-based self-adaptation in the presence of multiple objectives, SEAMS'06, 2006.

[32] S. Klüppelholz and C. Baier, Symbolic model checking for channel-based component connectors, Electr. Notes Theor. Comput. Sci. 175(2), 2007.

[33] M. Izadi, A. Movaghar, and F. Arbab, Model checking of component connectors, COMPSAC (1), 2007.

[34] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, Symbolic model checking: 10^20 states and beyond, Inf. Comput. 98(2), 1992.

[35] S. Meng and F. Arbab, On resource-sensitive timed component connectors, FMOODS 2007.

[36] F. Arbab, T. Chothia, S. Meng, and Y. Moon, Component connectors with QoS guarantees, COORDINATION 2007.

[37] C. Baier, E. M. Clarke, V. Hartonas-Garmhausen, M. Z. Kwiatkowska, and M. Ryan, Symbolic model checking for probabilistic processes, ICALP 1997.

[38] M. Z. Kwiatkowska, G. Norman, and D. Parker, Probabilistic symbolic model checking with PRISM: a hybrid approach, STTT 6(2), 2004.

[39] M. Z. Kwiatkowska, Quantitative verification: models techniques and tools, ESEC/SIGSOFT FSE 2007.

[40] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, Automatic verification of real-time systems with discrete probability distributions, Theor. Comput. Sci. 282(1), 2002.

[41] K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis, Multi-objective model checking of Markov decision processes, TACAS 2007.

[42] E. Wandeler, A. Maxiaguine, and L. Thiele, Quantitative characterization of event streams in analysis of hard real-time applications, In IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.

[43] J. Cobleigh, D. Giannakopoulou, and C. Pasareanu, Learning assumptions for compositional verification. TACAS 2003.

[44] H. Hungar, O. Niese, and B. Steffen, Domain-specific optimization in automata learning, CAV 2003.

[45] G. Ammons, R. Bodik, and J. Larus, Mining specifications, 29th ACM Symp. on Principles of Programming Languages, 2002.

[46] R. Alur, P. Madhusudan, and W. Nam, Symbolic compositional verification by learning assumptions. CAV'05.

[47] D. Angluin, Learning regular sets from queries and counterexamples. Information and Computation 75, 1987.

[48] B. Steffen, T. Margaria, H. Raffelt, and O. Niese, Effcient test-based model generation of legacy systems, HLDVT'04, IEEE Computer Society Press, 2004.

[49] E. Torlak and D. Jackson. Kodkod: A Relational Model Finder, TACAS 2007.

[50] T. Margaria, H. Raffelt, B. Steffen, and M. Leucker: The LearnLib in FMICS-jETI, ICECCS 2007, Auckland (NZ), IEEE CS Press, 2007.

[51] S. Chaki and O.Strichman, Optimized L* for assume-guarantee reasoning, TACAS 2007.

[52] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, The Daikon system for dynamic detection of likely invariants, Sci. Comput. Program, 69(1-3), 2007.

[53] J. Yang, D. Evans, D. Bhardwaj, T. Bhat, and M. Das, Perracotta: mining temporal API rules from imperfect traces, ICSE 2006.

[54] T. Berg, B. Jonsson, and H. Raffelt, Regular inference for state machines using domains with equality tests, FASE 2008, LNCS Vol. 4961.

[55] O. Grinchtein, B. Jonsson, and P. Pettersson, Inference of event-recording automata using timed decision trees, CONCUR 2006.

[56] P. J. Ramadge and W. M Wonham, The control of discrete event systems. Proceedings of the IEEE, 77(1), 1989.

[57] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime, UPPAAL-Tiga: Time for playing games!". CAV 2007.

[58] D. M. Yellin and R. E. Strom, Protocol Specifications and component adaptors, ACM Trans. Program. Lang. Syst. 19(2), 1997.

[59] L. de Alfaro and T.A. Henzinger, Interface automata, ESEC/FSE 01, 2001.

[60] P. Inverardi and M. Tivoli. Deadlock-free software architectures for COM/DCOM Applications. Elsevier Journal of Systems and Software, 2003.

[61] P. Inverardi and M. Tivoli, Failure-free coordinators synthesis for component-based architectures, Science of Computer Programming, 2008.

[62] M.Tivoli, P.Fradet, A. Girault, and G. Goessler, Adaptor synthesis for real-time components, TACAS 2007, LNCS 4424.

[63] F. Martinelli and I. Matteucci, Synthesis of local controller programs for enforcing global security properties, IEEE International Workshop on Advances in Policy Enforcement (APE'08), IEEE Press, 2008.

[64] M.Autili, P.Inverardi, A.Navarra, and M.Tivoli, SYNTHESIS: A tool for automatically assembling correct and distributed component-based systems, ICSE 2007.

[65] J. Ligatti, L. Bauer, and D. Walker, Edit automata: Enforcement mechanisms for run-time security policies, International Journal of Information Security, 4(1-2), 2005.

[66] P. Inverardi, D. Yankelevich, and A. Wolf, Static checking of system behaviors using derived component assumptions, ACM TOSEM 9, 2000.

[67] R. Vaculin and K. Sycar, Towards automatic mediation of OWL-S process models, IEEE International Conference on Web Services, 2007.

[68] T. Margaria, M. Bakera, C. Kubczak, S. Naujokat, B. Steffen: Automatic Generation of the SWS-Challenge Mediator with jABC/ABC, in *Semantic Web Services Challenge - Results from the First Year*, Springer Verlag, 2009, ISBN: 978-0-387-72495-9.

[69] C. Kubczak, T. Margaria, B. Steffen, R. Nagel: Service-oriented Mediation with jABC/jETI, in */Semantic Web Services Challenge - Results from the First Year,* Springer Verlag, 2009, ISBN: 978-0-387-72495-9

[70] T.Berg, O.Grinchtein, B.Jonsson, M.Leucker, H.Raffelt, B.Steffen: On the Correspondence Between Conformance Testing and Regular Inference. FASE 2005: 175-189

[71] T. Margaria, M. G. Hinchey, H. Raffelt, J.L. Rash, C A. Rouff, B. Steffen: Completing and Adapting Models of Biological Processes, BiCC 2006, IFIP Conference on Biologically Inspired Cooperative Computing, IFIP Series N. 216/2006, Springer Verlag

[72] H.Raffelt, B.Steffen: LearnLib: A Library for Automata Learning and Experimentation. FASE 2006: 377-380

[73] H. Hungar, O. Niese, B. Steffen: Domain-Specific Optimization in Automata Learning. CAV 2003: 315-327

[74] H.Hungar, B.Steffen: Behavior-based model construction. STTT 6(1): 4-14 (2004)

[75] H. Raffelt, B. Steffen, T. Berg, T. Margaria: LearnLib: A Framework for Extrapolating Behavioural Models, STTT, Int. Journal on Software Tools for Technology Transfer (in print).