Project Final Report Final Publishable Summary Report

Emmanuelle Grousset, Valérie Issarny, Amel Bennaceur, Antonia Bertolino, Daniela Mulas, Illaria Matteucci, Paul Grace, Gordon Blair, Youssouf Mhoma, Paola Inverardi, et al.

CONNECT

Emergent Connectors for

Eternal Software Intensive Networked Systems

ICT FET IP Project

Project Final Report

Final Publishable Summary Report

SEVENTH FRAMEWORK PROGRAMME

http://www.connect-forever.eu

Inria · NTT docomo DOCOMO Euro-Labs · Amblentic · Lancaster University · Thales · Universita' dell'Aquila · technische universität dortmund · University of Oxford · Uppsala Universitet

Project Number	:	231167
Project Title	:	CONNECT – Emergent Connectors for Eternal Software Intensive Networked Systems
Deliverable Type	:	Report

Deliverable Number	:	Final report
Title of Deliverable	:	Project final report – Final publishable summary report
Nature of Deliverable	:	R
Dissemination level	:	PU
Internal Document Number	:	V3
Contractual Delivery Date	:	31 January 2013
Actual Delivery Date	:	10 January 2013
Contributing WPs	:	all
Editor(s)	:	Emmanuelle Grousset, Valérie Issarny (Inria)
Author(s)	:	Emmanuelle Grousset, Valérie Issarny, Amel Bennaceur (Inria) Antonia Bertolino, Daniela Mulas, Illaria Matteucci (CNR) Paul Grace, Gordon Blair (LANCS) Youssouf Mhoma (TCF) Paola Inverardi, Romina Spalazzese, Massimo Tivoli (UNIVAQ) Maik Merten, Bernhard Steffen (TUDO) Hongyang Qu, Marta Kwiatkowska (UOXF) Bengt Jonson, Sofia Cassel (UU) Yun Ma (PKU) Pierre Guillaume Raverdy, Roberto Speicys-Cardoso, Emil Andriescu (AMBIENTIC)
Reviewer(s)	:	Nikolaos Georgantas (Inria)

Abstract

This document is the final publishable summary report, part of the CONNECT final report.

Document History

Version	Type of change	Author(s)
V0	Initial template	Emmanuelle Grousset
V1	Contribution from partners	All
V2	Integration	Valerie Issarny
V3/Final	Final editing	Valerie Issarny

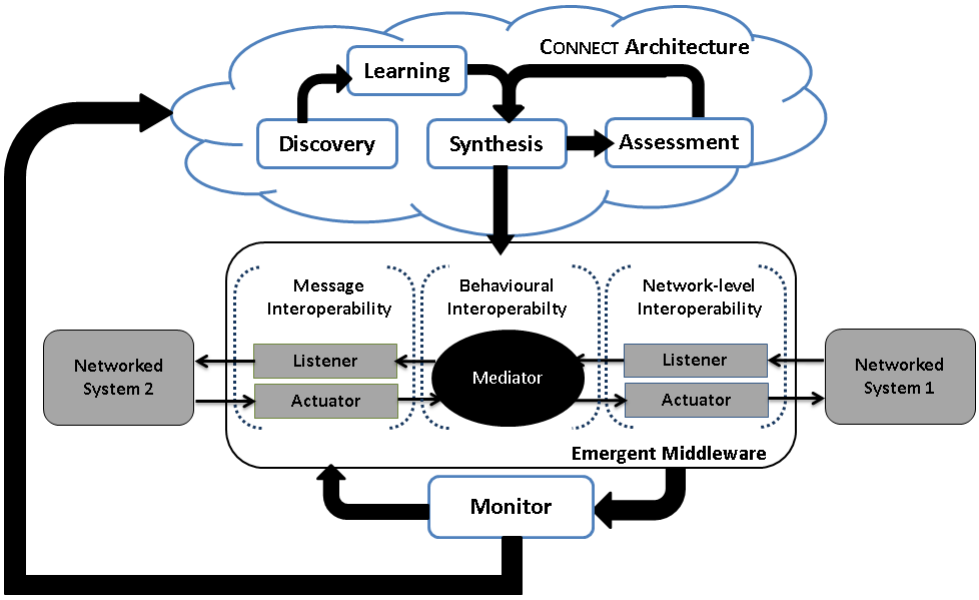
Table of Contents

Executive Summary

Emergent Middleware for Eternal Software Intensive Networked Systems

The pervasive computing vision is hampered by the often extreme level of heterogeneity in the underlying infrastructure, which in turns impacts on the ability to seamlessly interoperate. Further, the fast pace at which technology evolves at all abstraction layers increasingly challenges the lifetime of networked systems in the digital environment. Overcoming the interoperability challenge in pervasive computing systems is at the heart of CONNECT.

CONNECT is an EU Future and Emerging Technologies project involving partners whose expertise covers middleware, software engineering, formal methods, machine learning, and systems dependability. The aim of CONNECT is specifically to overcome interaction protocol heterogeneity at all layers, on-the-fly, by using a revolutionary approach that dynamically generates the necessary interoperability solution to connect two heterogeneous systems. We term this new style of middleware: **Emergent middleware**.



Emergent middleware *aka* CONNECTors are developed through a comprehensive dynamic process, which is supported by dedicated enablers that:

- **Extract knowledge** from, **Learn** about, and **Reason** on the interaction behaviour of networked systems, so as to:
- **Synthesize** new interaction behaviours out of the ones exhibited by the systems to be made interoperable, and further:
- **Generate and deploy** corresponding CONNECTor implementations to actually realize networking of the involved systems; and
- **Analyze dependability and security** exposed by the realized networking, both at pre-deployment time and at runtime.

The above raises a set of unique challenges in the area of software systems engineering, from theoretical foundations for specifying the interaction behaviour of networked systems to run-time methods and tools for turning specifications into running protocols, and vice-versa.

It is overcoming the above challenges that CONNECT concentrated on.

To know more, visit the project Web site at www.connect-forever.eu/

Context & Objectives

We are moving towards a world where everything is connected. Yet such a goal highlights the deficiencies of today's systems platforms in achieving a fundamental property of distributed systems, namely interoperability. Faced with the extreme heterogeneity of computational devices and networks, how can we ensure that every system can talk to every other system?

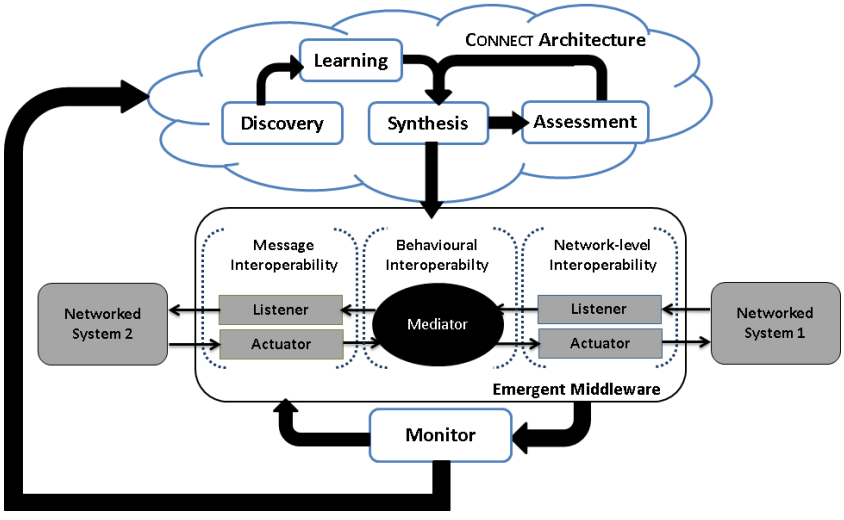
It is the above question that CONNECT addresses. CONNECT is an EU Future and Emerging Technologies –FET– project, which began in February 2009 and concluded end 2012. To meet its ambitious objective, CONNECT involved experts in middleware, software engineering, formal methods, machine learning, and systems dependability.

CONNECT Context:

Emergent middleware facing the interoperability challenge

Interoperability is the ability for two systems to exchange, understand and use each other's data, and is a long-standing problem in the field of distributed systems. However, the emergence of pervasive computing and the Internet of Things have brought about new challenges to achieving universal interoperability. Extreme heterogeneity and spontaneous interactions are characteristics of today's complex distributed systems. Computational devices ranging from embedded devices, sensors, and smartphones through to cluster machines and the Cloud use a wide range of communication networks and middleware protocols to communicate with one another. However, as soon as two systems adhere to heterogeneous protocols (from application down to network layers) to interact with each other, interoperability is impossible. Standards are a well-established approach to rectifying these types of problems. Where two systems agree upon a standard, interoperability can be guaranteed. However, systems may encounter one another spontaneously where no such agreement is possible, and hence where the communication protocols differ they cannot interoperate.

The aim of CONNECT is to overcome interaction protocol heterogeneity at all layers, on the fly, by using a revolutionary approach that dynamically generates the necessary interoperability solution to connect two heterogeneous systems. We term this new style of middleware: *Emergent middleware.* The figure below illustrates an emergent middleware solution, which ensures interoperation between two networked systems by combining *message interoperability*, i.e., the ability to interpret messages from/toward networked systems and *behavioural interoperability*, i.e., the ability to mediate the interaction protocols run by the communicating networked systems, under specified non-functional properties, e.g., reliability, performance and security. The CONNECT *architecture* then introduces the necessary *enablers* for on-the-fly production of emergent middleware.



Enabling emergent middleware

CONNECT Concept:
Network behaviourally for eternal communication

In our view, the key to eternal interoperability is to make the networking of systems agnostic to their specific technological middleware. Then, networked systems should seamlessly integrate and compose with networks of systems according to functional and non-functional properties each one of them provides to and requires from the digital environment rather than according to their underlying middleware technology.

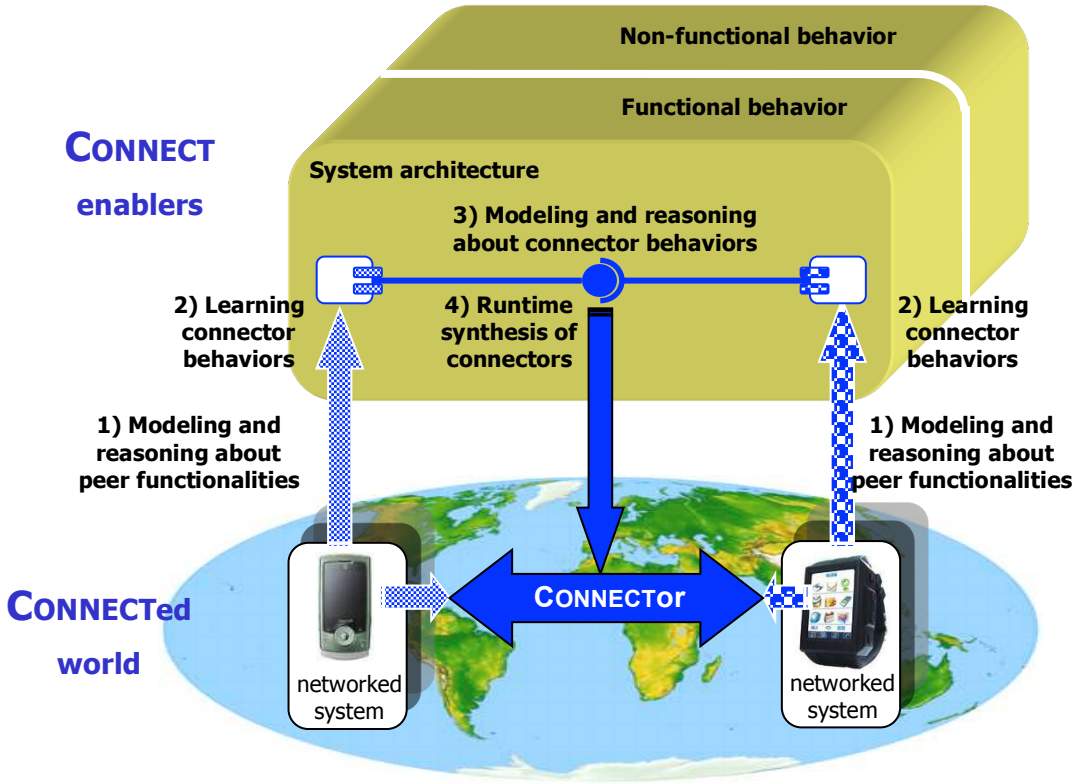
Specifically, in the CONNECTed world, networked systems run discovery protocols to advertise their presence and locate systems with which they need to interact at a specific time and place. The initial networking association of systems is then solely based on the systems' provided and required application-specific behaviour, which is characterized semantically, thanks to ontologies. Following, CONNECT enablers present in the networked environment set up needed emergent middleware, aka CONNECTors, among the interacting systems, at run-time. CONNECTors effectively bridge semantically the protocols of networked systems, from the application down to the middleware layer. As a result, networked technology-dependent systems interact behaviourally within the CONNECTed world, based on their respective interaction semantics. Key concepts of CONNECT are as follows:

- As pioneered by the pervasive computing domain, the *dynamic networking of digital systems* is at the heart of making networked applications evolvable and therefore eternal. In this way, a networked system is able to compose with others, based on respective provided and required functionalities within the network, without requiring a priori knowledge about the systems that are actually networked at a given time and place. Then, our only assumption is that a networked system runs some discovery protocol and further characterizes provided/required (application-layer) networked functionalities using ontologies.
- CONNECT sustains future-proof dynamic networking among any digital systems, from the legacy to the yet-to-come, by dynamically generating CONNECTors, thus bringing universal interoperability. Emergent connectors are synthesized on the fly according to the behavioural semantics of application- down to middleware-layer protocols run by the interacting parties, thereby realizing the necessary protocol mediation. Also, emergent connectors are dependable, unobtrusive, and evolvable to indeed meet the promise of eternality, while not compromising the quality of software applications. Last but not least, emergent connectors are concrete system entities, being implemented on the fly to enable actual interactions.
- CONNECTors are implemented through a comprehensive dynamic process based on: (i) extracting knowledge from, (ii) learning about and (iii) reasoning about, the interaction behaviour of networked systems, together with (iv) synthesizing new interaction behaviours out of the ones exhibited by the systems to be made interoperable, also considering their respective provided and required non-functional specifications, finally (v) generating and deploying corresponding CONNECTors implementations to actually realize networking of the involved systems, and further (vi) continuously monitoring CONNECTors runtime behaviour, to timely detect needs of adaptation.

CONNECT Objectives & Challenges:
From formal foundations to networked enablers for emergent connectors

The core objective of the FET CONNECT project is to effectively support the aforementioned dynamic process for the actual implementation and deployment of emergent middleware (or CONNECTors). This requires devising: (i) a semantic foundation of connectors enabling automated reasoning and synthesis of their behaviour, 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.



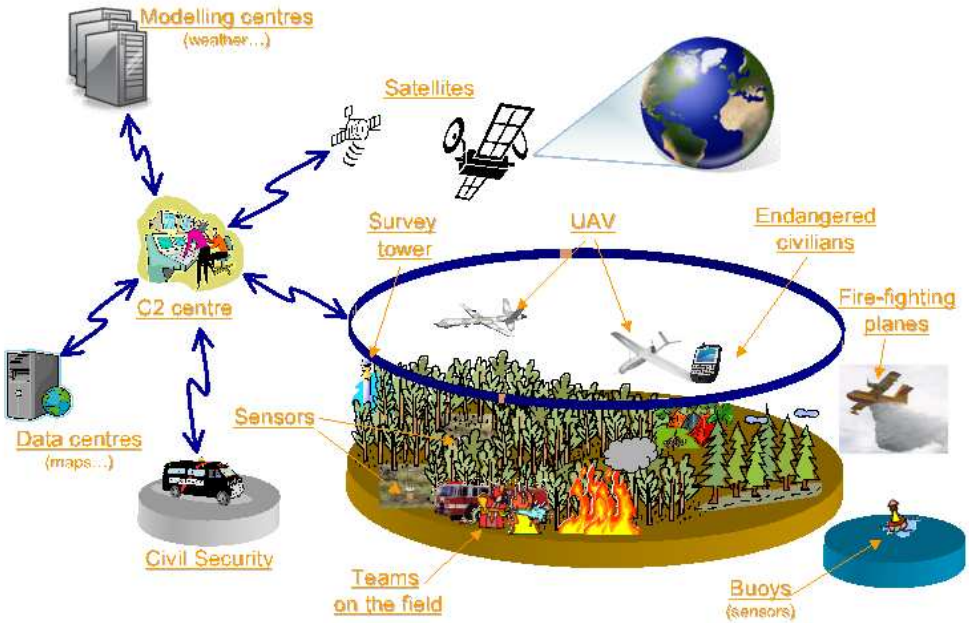
The CONNECT process & enablers

The above raises a set of unique challenges in the area of software systems engineering. Specifically, CONNECT focuses on the following issues and related challenges:

- **Modelling and reasoning about peer system functionalities:** Whilst the CONNECT project focuses on interaction protocol interoperability, the semantic description of networked systems functionalities is still crucial. Indeed, networked systems shall primarily meet according to their functional properties/capabilities. Ontologies pose as a natural base choice. However, using ontologies for establishing common understanding of networked system functionalities in the open CONNECTed world remains challenged by various concerns, starting with devising unambiguous yet flexible system descriptions.
- **Modelling and reasoning about connector behaviours:** A modelling framework where connector specifications can be built in an incremental way is required to be able to reason about interaction protocol matching and further generate new protocols out of the ones supported by interacting parties, in an automated way.
- **Runtime synthesis of connectors.** Given the specifications abstracting the interaction protocols run by networked systems and thanks to the composability of such specifications, the specifications of mediators bridging interacting parties need to be synthesized and actual implementations derived. The resulting emergent connector then implement (application down to middleware layer) interaction protocols that mediate the protocols run by the communicating systems, and further match required functional and non-functional (e.g., reliability, performance, security) properties.
- **Learning connector behaviours:** One cannot expect to have all the networked systems coming along with the formal specification of their interaction behaviour, and that this will match the specific modelling language underlying connector synthesis within CONNECT. It is then needed to put in place learning algorithms and techniques to dynamically infer

specifications from the networking/communicating behaviour of digital systems, which may range from complex application-level behaviour, implemented using some middleware technology, to primitive network messaging.

- **Dependability assurance:** Enabling the seamless networking of systems comes at risk from the standpoint of dependability. Two complementary issues 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.
- **Connecting eternal systems:** The above mentioned CONNECT enablers shall be effectively distributed within the network. The resulting system architecture may range from fully distributed with (partial) support for generating CONNECTors embedded on each node to logically centralized with support embedded on a few specific (server) nodes. Key requirement is to support the CONNECTion of legacy systems so that CONNECT does not introduce yet another technological island compromising eternal interoperable communication.



CONNECT enables GMES

- **Experiment.** The success of the proposed approach is certainly related to the resulting efficiency of emergent CONNECTions. However, CONNECTor generation takes place at the time networked systems compose, and so the execution of applications is only affected by the efficiency of synthesized CONNECTors. Still, it is crucial to assess the effectiveness of the overall approach, integrating the various pieces together.

  The assessment of CONNECT solutions is carried out based on various applications scenarios, including key applications for industry. In particular, as exemplified by the GMES (Global Monitoring for Environment and Security) scenario depicted above, CONNECT's capabilities are in general needed by the class of systems known as "*systems of systems*" (SoS) with real-life applicability in various fields such as transportation, military, security systems and so forth. Another key application area for CONNECT is that of mobile collaborative applications that must deal with interoperability across heterogeneous mobile platforms as well as heterogeneous Cloud services.

As outlined hereafter, key CONNECT scientific and technology contributions span a number of research areas to enable on-the-fly synthesis of emergent middleware, from architectural framework to formal foundations to protocol learning and synthesis. This effectively leads to the introduction of *Enablers* that build upon dedicated formal foundations for CONNECTors and are integrated within the CONNECT architecture.

Formal Foundations for CONNECTors

The research concentrates on the formal modelling of interaction behaviours to support automated learning, reasoning and synthesis, hence laying the theoretical underpinnings for generating CONNECTors. Our research on formal foundations for CONNECTors has led to the accomplishment of three streams of work, as summarized below.

Compositional specification theory for modelling and reasoning on component and connector behaviours

We have formulated a comprehensive specification theory for modelling components and connector behaviours. The theory introduces formal notions such as refinement for comparing components and determining when they can be substituted for one another, and useful composition operators. The operation of parallel composition allows users to examine the structure of composed component-based systems, while conjunction supports independent development, a necessary feature for supporting compositionality in CONNECT. The operation of quotient supports the principles of synthesis, and can be used for proving technical results about the properties of mediators synthesized using the CONNECT synthesis enabler, such as deadlock and error freedom. The theory allows the compositional verification of safety properties for probabilistic systems. In particular, we have developed an algebraically elegant framework for reasoning compositionally about the safety properties satisfied by specifications of components, primarily in the form of Assume-Guarantee (AG) specifications. Our work is more general than existing attempts, as we do not insist on equality of interfaces under satisfaction, do not require components and guarantees to be input-enabled, provide the first definition of conjunction and quotient on contracts for this model type, and present strong algebraic properties for the operations defined directly on contracts. All of the AG rules are shown to be both sound and complete.

Quantitative extension of the specification theory

We have proposed a quantitative extension of the above specification theory. This framework supports the modelling of real-time components with critical timing constraints. As for the non-quantitative specification theory, we define the full collection of compositional operators, including quotient for synthesis, and provide the weakest substitutive refinement preorder maintaining absence of safety and bounded liveness errors. Our theory is elegant in comparison to existing approaches, based on our equating of safety and liveness errors through a single inconsistent state. This means that we need only use a single transition system, rather than two, as in state-of-the-art approaches. Subsequent work has led us to look at the realizable sub class of timed components, which are not permitted to stop the global system clock. This involves redefining the compositional operators, in order to avoid encountering a time stop state from which time may not progress.

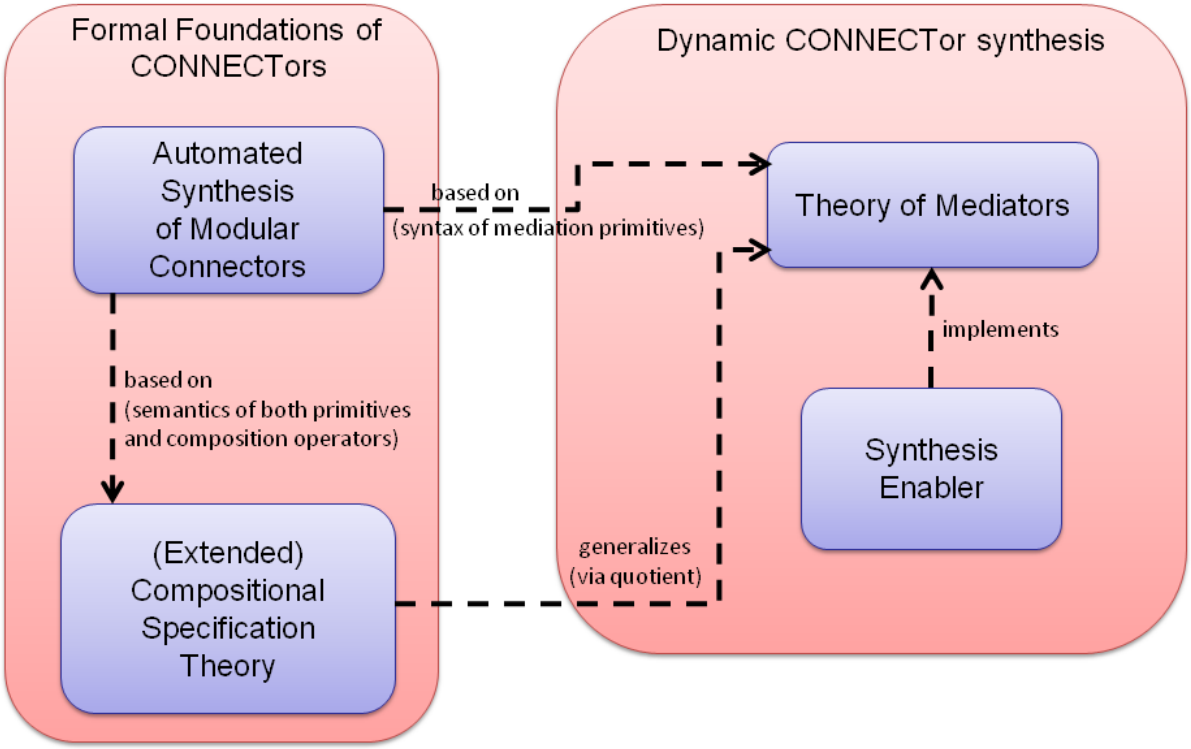
Automated synthesis of modular connectors

With the aim of supporting CONNECTor synthesis, we have formulated a method for the automated synthesis of modular connectors. A modular connector is synthesized as a composition of independent mediators, each of them corresponding to the solution of a recurring protocol mismatch. A synthesized modular connector is equivalent to its monolithic version as produced by the specification theory via quotient. We have proven that our

connector decomposition is correct and we have shown how it promotes connector evolution. An overall advantage of our approach with respect to the work in the state of the art is that our connectors have a modular software architecture organized as a composition of fundamental mediation primitives. This supports connector evolution and automated generation of the connector's implementation code. In particular, we have recently released a first implementation (http://www.connect-forever.eu/software-API-Mediator.html) of both the algebra primitives and the composition operators. This implementation is based on the use of Enterprise Integration Patterns (http://www.eaipatterns.com/) and is developed through the Apache Camel framework (http://camel.apache.org/). Because of the way a modular connector is structured, the automatic generation of its actual code written in terms of our algebra implementation is viable and can be achieved with little effort. We have started to show, through its application to real world case studies, that our method supports connector evolution.

From CONNECTor foundations to dynamic synthesis

Summing up, the above streams of work have produced a set of original research results about formal specification and analysis of components and connectors whose application in CONNECT provides the formal foundations for CONNECTors used to support automated learning, reasoning and synthesis. As far as CONNECTor synthesis is concerned, as pictorially sketched in the figure below, our method for the automated synthesis of modular connectors relies on both the (extended) compositional specification theory discussed above and the theory of mediators underpinning *dynamic CONNECTor synthesis* (see the next section), and establishes some relationships among them (see the thicker dashed arrows). In particular, each mediator synthesized as a basic constituent of a modular connector is an instance of a specific mediation pattern (e.g., reordering of messages) from the theory of mediators discussed in the next section.



Integration of formalisms and theories to support dynamic CONNECTor synthesis

To enable automated reasoning on our connector decomposition (e.g., to prove that a modular connector is equivalent to a monolithic CONNECTor in the theory of mediators, or is free of mismatches), we have formally defined the semantics of protocols, as well as of mediators and connector, by using the (extended) compositional specification theory.

We have proven that a modular connector for two protocols P and R enjoys the same correctness properties of its respective monolithic connector obtained by expressing the synthesis problem as a quotient problem in the specification theory. Since, connectors synthesized through quotient generalizes CONNECTors in the theory of mediators, this also means that our synthesis algorithm supports dynamic CONNECTor synthesis.

Dynamic CONNECTor Synthesis

Here, the research focuses on one of the Enablers of CONNECTors generation, that is, the automated synthesis of mediators according to the interaction behaviours of Networked Systems –NSs– seeking to communicate. The produced mediators abstractly specify the behaviour of CONNECTors and are run by a dedicated CONNECT engine to enact emergent middleware.

Even though there exist many interoperability solutions, they are inappropriate for one of the two following reasons: either (i) they deal with application heterogeneity and generate corresponding mediators but fail to deploy them on top of heterogeneous middleware, or (ii) deal with middleware heterogeneity while assuming the same application atop and rely on developers to provide all the translations that need to be made. Within CONNECT, we have been arguing that seamless interoperation is a crosscutting concern and interoperability solutions must consider conjointly application and middleware layers:

- The application layer provides the appropriate level of abstraction to reason about interoperability and automate the generation of mediators;
- The middleware layer offers the necessary services for realizing the mediation by selecting and instantiating the specific data structures and protocols.

The core contribution of our work is then to enable interoperability between highly heterogeneous systems by reconciling behavioural discrepancies, or mismatches, that occur between functionally compatible systems, from the application down to the middleware layer. The contribution is both theoretical and practical, decomposing into:

- A *theory of mediators* that formalizes the process of mediator synthesis, and in particular highlights the central role of ontologies in reasoning about functional and behavioural matching of networked systems.
- A *synthesis enabler* that is an implementation of the proposed theory, further targeting fully automated mediator synthesis, which can then be performed on the fly following the run-time discovery of NSs that implement functionally matching affordances.

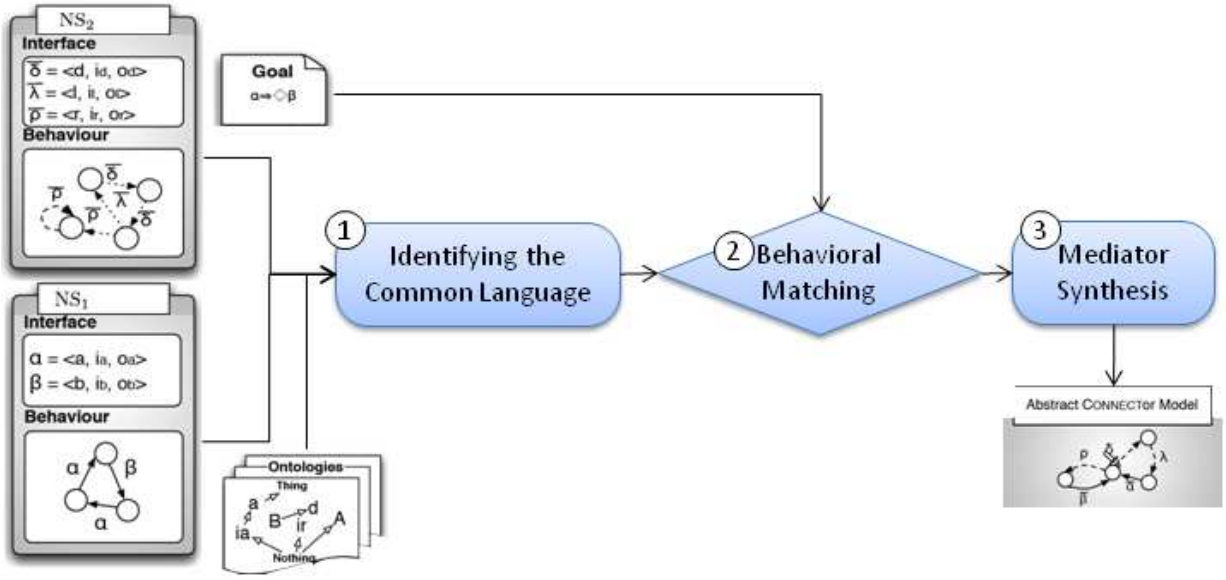
The CONNECT theory of mediators

The mediator synthesis process takes as input the NSs models and the ontologies describing the domain-specific knowledge about the NSs. In particular, the ontology describes actions and data concepts within the domain of interest and also relations among these concepts, e.g., the subsumption relation. The process is further assumed to be preceded by Functional Matching and Middleware Abstraction phases:

- The former consists in checking whether, at a high level of abstraction, the functionality (*aka* the *affordance*) required by one system can be provided by the other;
- The latter abstracts/translates (sequences of) middleware functions into the input/output qualification of actions.

The mediator synthesis process is made up of three phases or steps (see figure below):

1. *Identification of the Common Language* makes comparable the NS behaviours by determining their common language and, possibly, reduces their size and speeds up the reasoning about them.



The mediator synthesis process

2. *Behavioural matching* checks the NS compatibility, possibly identifying behavioural mismatches.

   In a nutshell, the behavioural matching step takes as input the specification of the NS protocols as Labelled Transition Systems –LTS– that are relabelled using the common language, so as to verify the behavioural compatibility of NSs. Such a check amounts to seeking compatible traces modulo: ordering mismatch, third party interactions and extra actions. A successful behavioural matching implies that a mediator exists and it will be automatically synthesized at the subsequent step.

   The output of the matching includes: (i) the behavioural compatibility relation (i.e., non-matching, intersection, inclusion, and total matching); (ii) the compatible traces –identifying sub-LTSs – labelled by actions and data concepts of the domain ontology including silent actions τs for third party actions; (iii) an LTS defined over the domain-specific ontology including τ that realizes a skeleton that then needs to be refined to become a mediator.

3. *Mediator Synthesis* produces a mediator that address the identified mismatches between the two NSs and allows them to communicate.

   In more details, given two behaviourally compatible protocols *P* and *Q*, we want to synthesize a mediator *M* such that the parallel composition *P || M || Q || E* where *E* is the environment, allows *P* and *Q* to evolve to their final states. Actions of *P* and *Q* can belong either to the common language or the third party language, i.e., the environment language. We build the mediator in such a way that it lets *P* and *Q* evolve independently for the portion of the behaviour to be exchanged with the environment (denoted by the τ action) until they reach a "synchronization state" from which they can synchronize over compatible actions. We recall that the synchronization cannot be direct since the mediator needs to perform suitable manipulations as, for instance, actions reordering or translation used to identify the common language.

The CONNECT synthesis enabler

The CONNECT synthesis enabler implements the mediator theory in a way that allows the fully automated synthesis of mediators. The synthesis enabler specifically addresses the "identification of the common language" phase as an interface mapping problem.

1. *Mapping interfaces*: A sequence of actions required by an NS can be achieved using a sequence of actions provided by the other NS only if some constraints are verified.

The first constraint is that the provided operations are a specialization of the required ones, according to the ontological subsumption relation.

In addition, when an NS requires an action, it sends the input data and receives the corresponding output data. Hence, we can allow the NS to progress if the input data are cached and it does not expect any output data. Let us set *I* as the position of the first required action that necessitates some output data. To provide this output data, the corresponding provided actions have to be executed. Therefore, the input data of the first action can be obtained using the input data already received. Once this action has been executed, then the cached data are augmented with its output data. Once all the provided actions have been performed, we can execute the remaining required actions if all their output are available, where the cache is augmented with the input data of the required actions.

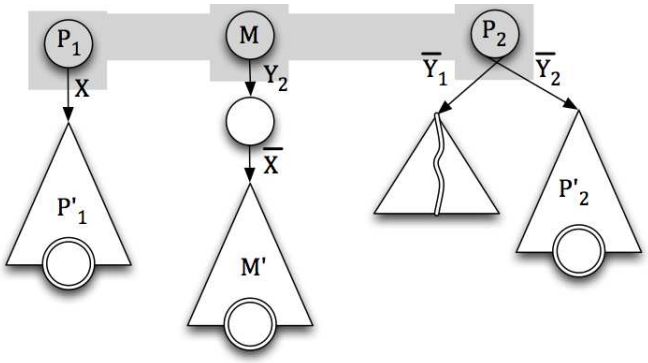
We use constraint programming to compute the mapping efficiently, where the above conditions on the operations and input/output data represent the constraints. The solver implements intelligent search algorithms such as backtracking and branch and bound to optimize the time for finding the mapping. However, no existing CP solver deals with ontology-based operators. To this end, we have defined a bit vector encoding of the ontology that is correct and complete regarding the subsumption and disjunction axioms.

2. *Synthesizing abstract mediators:* While the computation of interface mapping ensures that a sequence of actions from one NS can safely be achieved using another sequence of actions from the other NS, it does not specify when each mapping has to be performed. Furthermore, the interface mapping might be ambiguous. Hence, the mappings need to be combined such that the interaction of the two NSs does not lead to erroneous states, e.g., a deadlock. We analyse the behavioural specifications of the two NSs, which are specified using LTSs (say P_1 and P_2, respectively) and generate a third LTS, the mediator *M*, such that the mediated NSs reach their final states, which implies that the system made up of the parallel composition of P_1, P_2, and *M* is free from deadlocks, or we determine that no such mediator exists. If a mediator exists, then we say that P_1 and P_2 can interoperate using a mediator *M*.

   We inductively build a mediator *M* by forcing the two NSs to progress consistently so that if one requires the sequence of actions *X*, the interacting NS must provide a semantically-compatible sequence of actions *Y*. Given that an interface mapping guarantees the semantic compatibility between the actions of the two NSs, the mediator is able to compensate for the differences between their actions by performing the necessary transformations. The base case is that where both P_1 and P_2 are final states, in which case the mediator is made up of one final state. Then, at each state we choose an eligible mapping, i.e., both processes can engage in the sequences specified by this mapping and moves to states where a (sub-)mediator can be generated.

   The above implies that when multiple mappings can be applied, we select one of them and check if it allows the two processes (P_1 and P_2) to reach their final states. Otherwise, we backtrack and select another mapping.

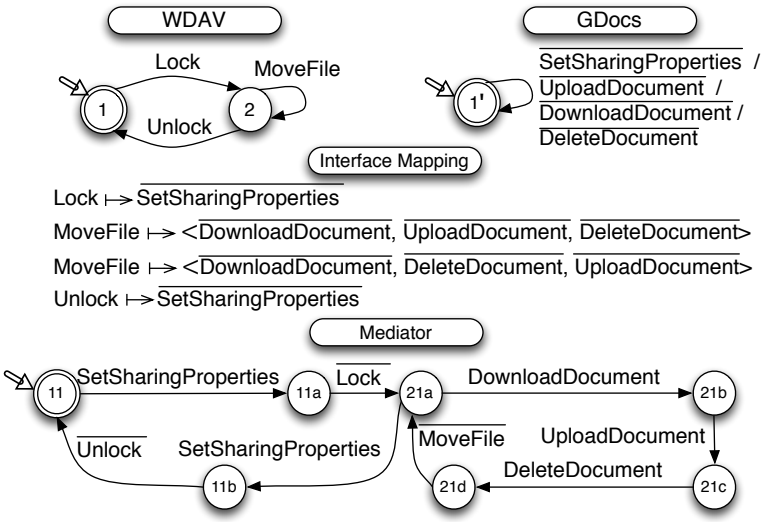
   Consider the example depicted below. P_1 is ready to request for *X*, and P_2 can provide (denoted by the overbar) either Y_1 or Y_2, and *X* can be semantically mapped to both Y_1 and Y_2. If we select the first mapping, i.e., *X* is realized by Y_1, the final state (denoted by the circle) of P_2 cannot be reached. Hence, we backtrack and select the second mapping. To compute the mediator *M*, we append the mapping of X to Y_2 to M' that is the mediator computed between P'_1 and P'_2. If none of the mappings is applicable, then we are not able to generate the mediator. Note however, that the mediator is not unique since many mappings may lead P_1 and P_2 to their final states. In this case, we only keep the first valid mapping.



Incremental synthesis of a mediator

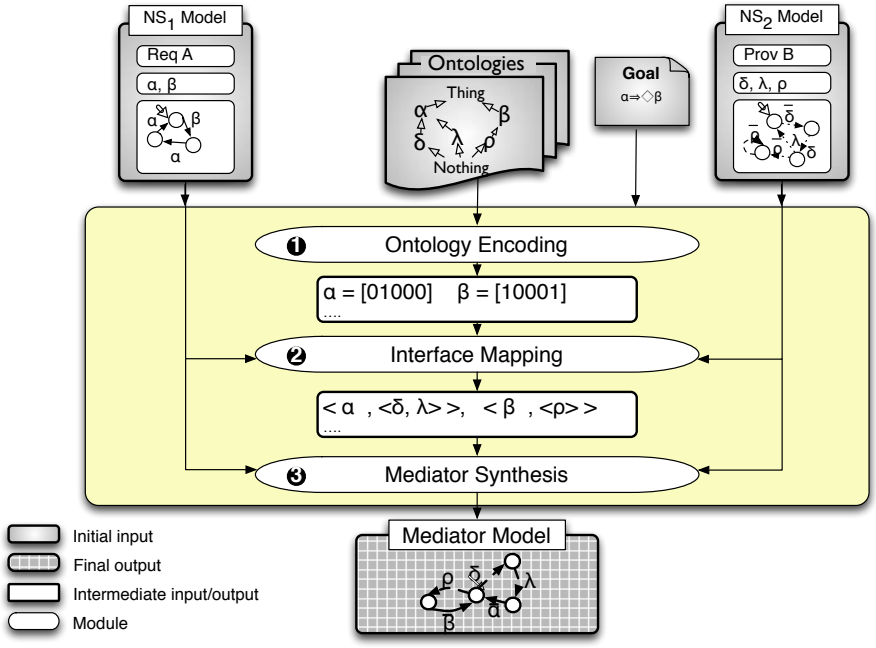
As an illustration, consider the CONNECTion between a WebDAV client and a GoogleDocs server. WebDAV (Web Distributed Authoring and Versioning) is an IETF specification that extends the Hypertext Transfer Protocol (HTTP) to allow users to create, read and change documents remotely. The Google Docs service lets users create, store, and search Google documents and collections. Although these two systems offer similar functionalities and use HTTP as the underlying transport protocol, they are unable to interoperate.

The figure below depicts an excerpt of the LTSs representing the behaviour of the WebDAV client (WDAV) and that of the GoogleDocs server (GDocs). To simplify the presentation, the actions are defined using the operation concept only. As a result of the interface mapping computation, the *Lock* and *Unlock* operations of WDAV map to *SetSharingProperties* of GDocs, and the *MoveFile* operation maps to *DownloadDocument*, *UploadDocument* , and *DeleteDocument* while the last two operations can be executed in any order. When both processes are at their initial states (1 and 1' respectively), the only applicable mapping is *Lock to SetSharingProperties* since WDAV is only able to perform this action. After applying this mapping, WDAV goes to state 2, GDocs remains in state 1', and a partial trace of the mediator is created from 11 –> 11a -> 21a. Then, WDAV can loop on the *MoveFile* required action, one of the possible mappings is chosen since both are applicable as GDocs loops on the three provided actions, *DownloadDocument*, *UploadDocument*, and *DeleteDocument*. WDAV stays in state 2, GDocs also remains in 1' while the mediator is augmented with the trace 21a->21b->21c->21d->21a. WDAV can also branch on the *Unlock* operation, which maps to *SetSharingProperties* and results in the trace 21a->11b->11 in the mediator. Finally, both processes reach their final states and the mediator is successfully created.



Synthesis of a WebDAV-GoogleDocs (partial) mediator

The CONNECT mapping-based synthesis enabler is available for download from the CONNECT Web site and has been integrated within the overall CONNECT architecture to support the realization of emergent middleware.



The CONNECT enabler for mediator synthesis

Interaction Behaviour Learning and Monitoring

Our research on machine learning and monitoring devises the other key Enabler of CONNECT, i.e., discovering the specifications of the interaction behaviours of networked systems. Related achievements have advanced the S&T state-of-the-art in several directions, as described in the following.

Extending the power of active learning: From the finite-state case to the infinite-state case

Prior to the CONNECT project, active automate learning algorithms had been developed for finite-state models of components that utilize a finite set of primitives for interaction. To support learning of realistic networked systems, one challenge of CONNECT has been to extend learning techniques to handle richer models. Proposed techniques include:

- *Symbolic abstraction techniques* that have been introduced into the setting of active automata learning. Abstractions transform the problem of learning a rich, possibly infinite-state model, to the problem of learning a finite-state model.

- *Canonical automaton models* based on Nerode-like congruences allow guaranteeing that the learning of a component will succeed with a bounded number of test inputs, where the bound reflects the complexity of the component. We first developed canonical models for register automata, where data values could only be tested for equality. In spite of the restriction to canonicity, our approach provided natural and succinct models on typical applications. This approach was further generalized to a richer class of automata, covering also tests on sequence numbers, membership of objects in lists, etc.

- A framework for defining *learning algorithms as systematic search* for a least refined abstraction among the possible canonical automata models. This framework allows developing learning algorithms systematically, from the finite-state case by exploiting the fact that the automaton model is based on a Nerode congruence, which induces a symbolic abstraction. Our active learning algorithm is unique in that it directly infers the effect of data values on control flow as part of the learning process.

We have applied the proposed techniques to a range of examples. Not only are the inferred models much more expressive than finite state machines, but the prototype implementation also drastically outperforms the classic L* algorithm, even when exploiting optimal data abstraction and symmetry reduction. Thus, this work represents a breakthrough in the area of automata learning, and outperforms previous approaches to learning models that combine control and data.

We further developed a number of additional enhancements to active automata learning:

- A concept of *"effect-centric" learning*, which aims at decreasing time spent on learning while still guaranteeing that the models contain relevant information.

- *Combining active automata learning with syntactic analysis of WSDL descriptions*, which provides a means to infer types while learning black-box systems. With this technology, we can generate test-harnesses and input alphabets to support learning from interface descriptions in an automated fashion.

- *Algorithmic improvements in the search for new test inputs* that allowed to win the Zulu competition (http://labh-curien.univ-st-etienne.fr/zulu/index.php).

LearnLib: A flexible and powerful learning framework

During the CONNECT project, our framework for active automata learning, LearnLib, has been thoroughly reengineered in order to make the learning functionality available as reusable components.

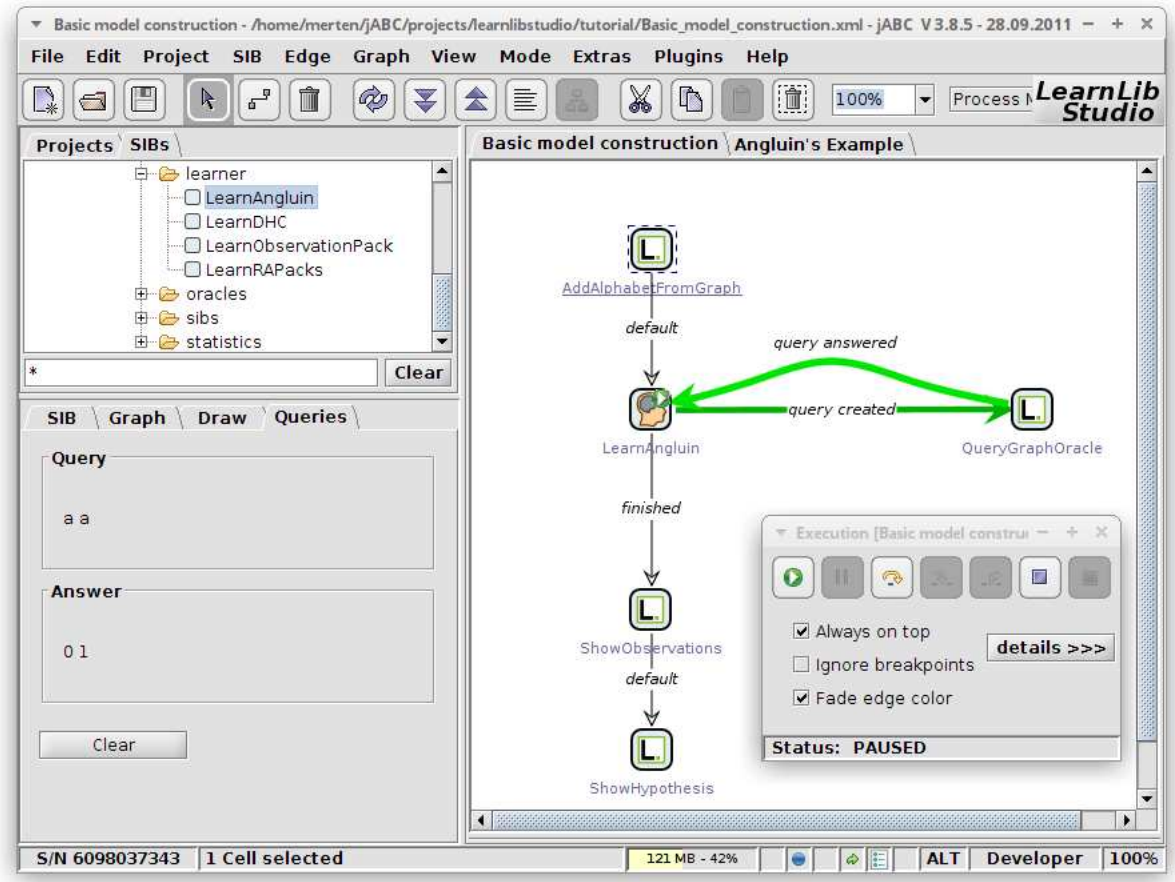
The resulting modularized framework is structured according to the basic principles of the MAT (Minimally Adequate Teacher) paradigm, accounting for structures such as learning algorithms, test drivers, equivalence checks and application-specific filters. This ensures that a multitude of learning algorithms following that paradigm can be implemented within a rich supporting infrastructure, enabling the composition of application-fit learning setups, utilizing interchangeable components provided by LearnLib.

The overall structure is also fit to support new automata models such as Register Automata and their respective learning algorithm, witnessed by implementations within the framework provided by LearnLib.

Being publicly available for free, LearnLib has seen adoption by groups that are not part of the CONNECT project.

Further, we have developed techniques specifically for the realization of the Learning Enabler, leveraging LearnLib as a component in the CONNECT architecture, thereby integrating with the various other CONNECT enablers. The Learning Enabler builds on the integration of a number of tasks, including to: (1) Receive interface descriptions, and generate corresponding test harnesses, (2) Applying semantic and syntactic analyses of interface descriptions, in order to tailor and adapt test harnesses and learning algorithms, (3) Equipping test harnesses with probes that gathers additional information, including non-functional properties, (4) Storing results, (5) Producing models in suitable CONNECT (LTS-based) formats, and (6) Annotating the generated models.

From the perspective of networked systems, these operations constitute a single learning operation to be invoked solely by the Discovery Enabler.



The LearnLib framework

GLIMPSE: A modular, flexible and lightweight monitoring infrastructure

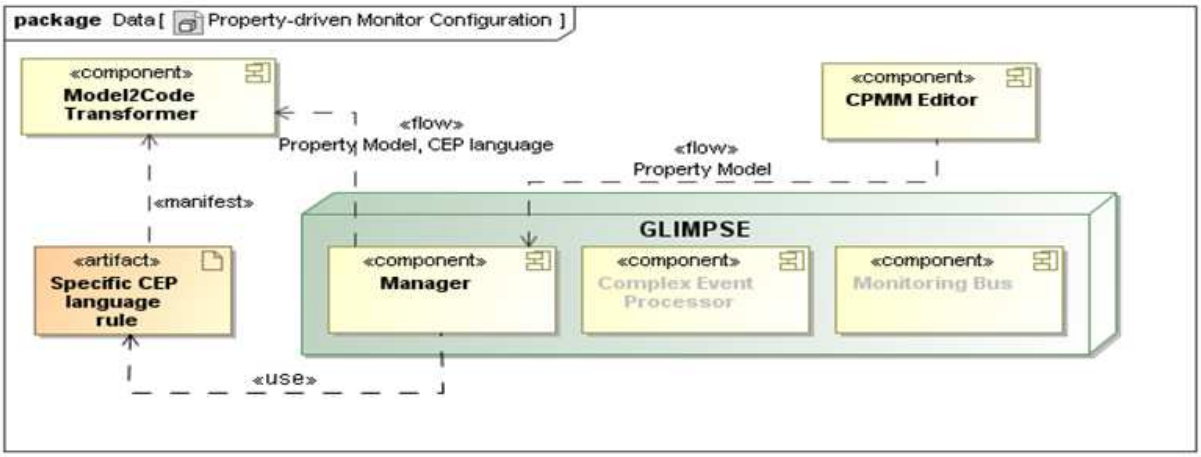
GLIMPSE conveniently complements the active learning enabler with monitoring capabilities. GLIMPSE is an acronym for Generic fLexIble Monitoring based on a Publish-Subscribe infrastructure. GLIMPSE is a comprehensive event-based publish-subscribe infrastructure for monitoring, which is the central instrument within the CONNECT architecture to provide self-awareness and to drive runtime adaptation.

The major technical contribution of GLIMPSE is its *model-driven orientation*, which is based on two key elements:

1. A generic monitoring infrastructure that offers the greatest flexibility and adaptability, and
2. A coherent set of domain-specific languages, expressed as meta-models, which enable us to exploit the support to automation offered by model-driven engineering techniques.

Concerning the former element, GLIMPSE is totally generic and can be easily applied to different contexts. Such infrastructure supports all the main functionalities required to a monitoring infrastructure, including data collection, local interpretation, data transmission, aggregation, and reporting. It has been fully integrated within the CONNECT architecture, and is now able to cooperate with all the CONNECT enablers involved in the off-line and on-line analysis of CONNECTors: namely, the Deployment Enabler, DePer, the Security Enabler, and the Learning Enabler.

Concerning the latter element, the monitor fully implements a Model-driven approach, so that the functional and non-functional properties to be observed can be specified by models conforming to the CPMM meta-model defined for specifying non-functional properties. In this respect, GLIMPSE is thus an advancement over the state-of-the-art in runtime monitoring infrastructures, as no similar approaches with the maturity and comprehensiveness of CPMM exists. A translator from CPMM (Connect Property MetaModel) models to GLIMPSE rules (Drools) has also been released.



The GLIMPSE monitoring enabler

Dependability Assurance

A considerable effort in CONNECT has been devoted to ensure that the non-functional properties required at each side of the CONNECTion going to be established are fulfilled, including dependability, performance, security and trust, or, in one overarching term, CONNECTability.

More precisely, CONNECTability refers to the ability of CONNECT enablers to provide a CONNECTed system with the intended properties of Dependability, Performance, Security and Trust in justifiable way.

Meta-modelling of non-functional properties, metrics and events

To model the non-functional properties, we have introduced the CONNECT Property Meta-Model –CPMM– that supports a model-driven approach to the specification of non-functional properties. This meta-model defines elements and types to specify prescriptive (required) and descriptive (owned) quantitative and qualitative properties that CONNECT actors may expose or must provide.

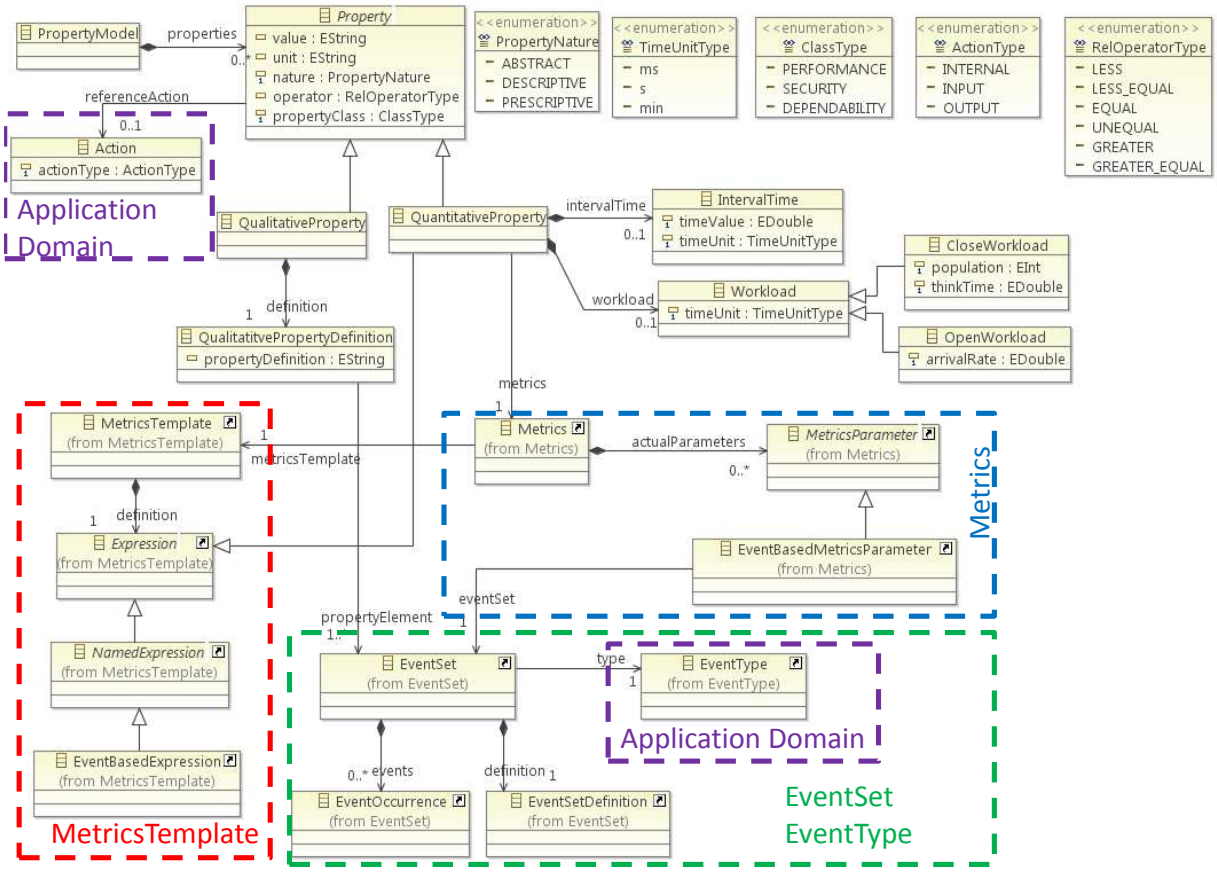
The key concepts of CPMM are: *Property*, *MetricsTemplate*, *Metrics*, *EventSet*, and *Event-Type*. We separated the property definition from the application domain and its specific ontology. The ontology is linked to the Property meta-model via the *EventType* entity that models a generic event type where the terms of the application-domain ontology will be used. Similarly, we distinguish the *Metrics* and *MetricsTemplate* concepts. The *MetricsTemplate* is defined upon a generic set of events/operations, which is not coupled with a particular application domain. This general part of the definition is specialized by the metric that instantiates those general concepts (*templateParameters*) with application-based events/operations (*metricsParameters*).

CPMM also includes a machine processable specification language that allows for defining complex events models involved into non-functional properties. The proposed language is

specified as part of CPMM, but it can be used in isolation to specify events that are not necessarily tied to a property model. This language combines features of two existing event specification languages that are GEM and Drools Fusion and in addition presents new features not included in them. In particular, we have defined operators that allow for modelling a temporal relationship (as those of Drools Fusion) and operators that allow for combining simple or complex events (as those of GEM). In addition, we have identified situations of interest not covered by the operators of GEM and Drools Fusion that have been formalized through new operators.

We make available the CPMM eCore model equipped with a dedicated editor (as an Eclipse Plugin), for using existing models and deriving new property and metrics from CPMM.

Finally, we also released a Model2Code transformer that translates models conforming to CPMM into Drools rules, which are used to configure the GLIMPSE monitoring infrastructure for run-time verification of CONNECT properties. The aim of such transformation is to allow for the dynamic configuration of GLIMPSE whenever a new property to be monitored is specified and introduced in CONNECT. This transformation has been implemented using the Acceleo technology.



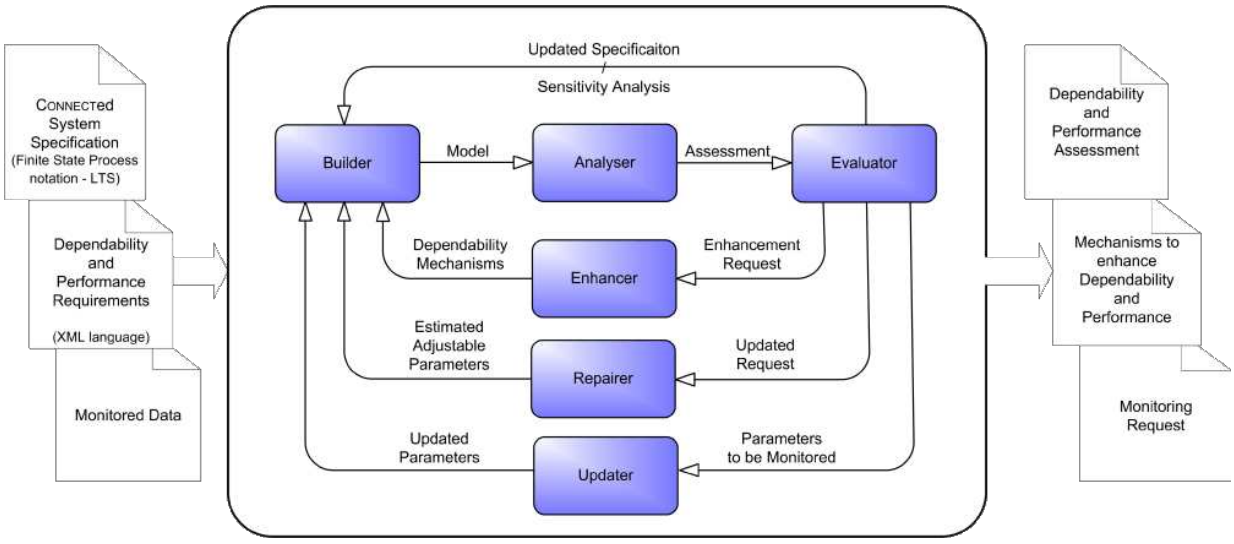
CPMM - The CONNECT Property Meta-Model

Dependability and performance analysis

To verify the specified Dependability and Performance properties, we have developed model-based analysis techniques and tools. In particular, both state-based stochastic methods and stochastic model checking have been exploited in CONNECT to enrich the variety of dependability analyses. We have conducted a detailed comparison between these two approaches, using PRISM, a stochastic model checker, and Mobius, a state-based stochastic analysis tool. The two approaches are complementary in assessing dependability and

performance properties. Indeed, the different formalisms and tools implied by the two methods allow: (i) on the one hand, to serve as a basis for a detailed analysis, by focusing on aspects such as the level of abstraction, scalability and accuracy, for which the two approaches may show different capabilities; and (ii) on the other hand, through the inner diversity, provide cross-validation to enhance confidence in the correctness of the analysis itself.

To assist the synthesis and deployment of a CONNECTor that can satisfy the non-functional requirements, we have developed the DePer Enabler. As depicted below, DePer has been designed following a modular approach, and comprises: Builder, Analyser, Evaluator, Enhancer (for the state-based stochastic analysis engine), Repairer (for the stochastic model checking analysis engine) and Updater.



DePer - The Dependability & Performance enabler

The dynamicity and evolution of the CONNECT environment lead to potential sources of uncertainty, which undermine the accuracy of the pre-deployment analysis. It is therefore crucial to investigate adaptive dependability assessment, with re-calibration and refinement of the dependability and performance prediction along time. DePer can combine classical pre-deployment analysis with on-line adaptation through recalibration of model parameters on the basis of data relative to real executions along time. This is achieved through the cooperation with GLIMPSE, the CONNECT monitoring enabler, which is properly instructed by the Updater module of DePer to observe relevant events at run-time and convey related data on such observed events.

Finally, we can embed selected dependability mechanisms in the CONNECTor model under assessment, to evaluate their efficacy to improve the synthesised CONNECTor, in case the analysis reveals that dependability or performance metrics are not satisfied. If the extended CONNECTor model is considered adequate to satisfy the stated non-functional requirement, the employed dependability mechanism is indicated to Synthesis as a suitable means to enhance the CONNECTor. This support provided by DePer aims at adapting the CONNECTor in response to problems arising from the execution environment, the uncertainties about the environment itself as known at pre-deployment time and evolution of the working context. The Enhancer module of DePer, implemented in the prototype with a selection of some basic dependability mechanisms, is in charge of this activity.

Quantitative incremental run-time verification

Due to the continuous evolution of the context with which a CONNECTor interacts, offline (static) analysis results may be invalidated after a CONNECTor is deployed. The accuracy of the analysis results can also be limited because unpredictable phenomena may affect the system during its operation. Therefore, the analysis typically needs to be refined or repeated with data

obtained from real system executions. We introduced the concept of (online) quantitative run time verification, so that changes to the evolving system can be taken into account and verified at run time. When running verification tasks at run time, it is important to optimise the execution time, particularly when the system is evolving, in case subsequent changes occur before the task terminates. We presented an approach for incremental verification, which improves the performance of verification at run time by reusing results from previous verification runs to obtain fast accurate results during the evolution of a CONNECTed system. Our approach decomposes a model into strongly connected components (SCCs) and identifies unaffected SCCs during system evolution, which do not need to be processed during incremental verification. In addition, our approach also improves the efficiency of SCC-based offline probabilistic model checking, and minimises the computation across multiple verifications using the improved technique.

Our incremental verification technique can handle CONNECTed systems that are subject not only to changes in probability values but also in the system structure. We first developed incremental methods for constructing models from high-level system descriptions and then designed an SCC-based incremental policy iteration to speed up numerical computation when the system structure is changed. The incremental model construction technique computes all states that have to be visited during incremental verification. The incremental policy iteration technique decomposes the system into SCCs, and is performed incrementally by reusing policies between verification runs.

Incremental verification can speed up verification at run time. However, an important question relevant to verification remains unaddressed. That is, what can we do if some non-functional properties are not satisfied? The (conceptual) Repairer module in the stochastic model checking engine proposes a means to proceed when the synthesised CONNECTor does not satisfy a given non-functional property. This module focuses on the situation where certain parameters in the CONNECTor can be adjusted. It attempts to compute new values for these parameters such that the CONNECTor synthesised using the new values ensures that the non-functional properties are satisfied. In order to effectively determine new parameter values for a new CONNECTor, we assume each adjustable parameter is defined in a bounded domain.

To enable usage at run-time, we aim to achieve fast performance of repair. We propose three efficient approaches to solve the parameter synthesis problem with respect to quantitative reachability properties in parametric probabilistic systems. We employ techniques from Monte Carlo sampling and evolutionary computation to obtain inexact, randomised, algorithms that perform well in practice. In particular, we show how to apply two sampling based approaches, i.e., Markov chain Monte Carlo and the cross entropy method, and a swarm-intelligence based method, i.e., the particle swarm optimisation, to the parameter synthesis problem.

Security and Trust modelling and assessment

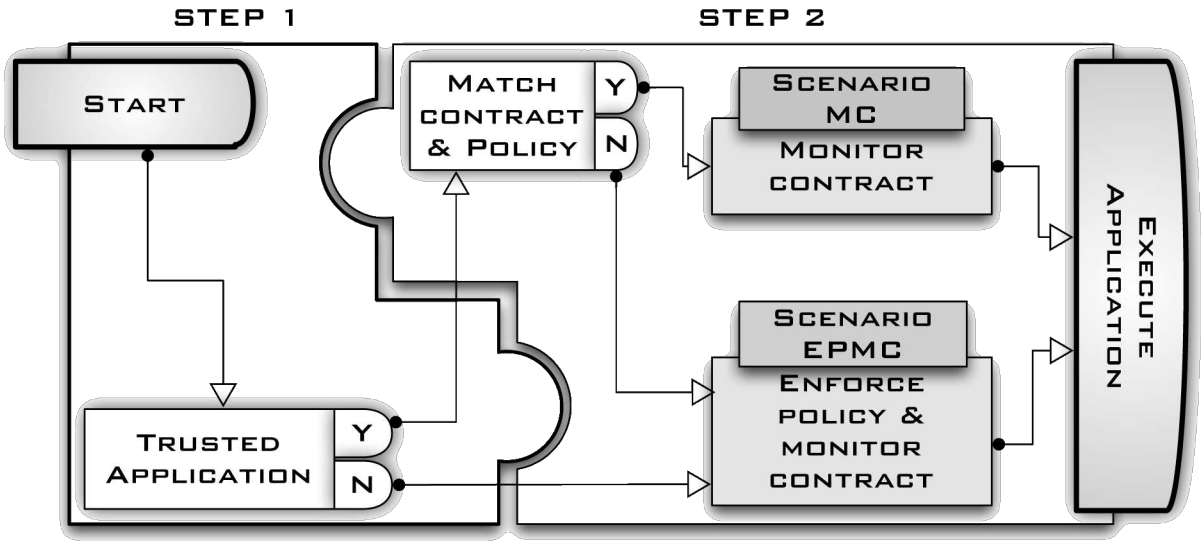
One major goal in CONNECTability assurance is to guarantee that whenever a CONNECTor is provided, the connection among networked systems works as expected from the security perspective. We have analysed threat models in the CONNECT framework by investigating the trust relations among networked systems, CONNECTor, and CONNECT enablers. This allows us to point out the security aspects that have to be taken into account.

We work at two different levels within the CONNECT architecture. On the one hand, we improve the synthesis of a CONNECTor by providing an automatic generation strategy for security features, in order to deal with cryptographic primitives. This aims to guarantee that the synthesized CONNECTor not only works as expected with respect to the required functional properties, but is also able to maintain or add cryptographic functionalities that preserve privacy and confidentiality aspects of the communication.

Once the CONNECTor is deployed, one should guarantee that it works as expected at run-time. To this aim, we extend the Security-by-Contract framework by taking into account trust

relations. This resulted in a novel framework that we called Security-by-Contract-with-Trust, SxCxT for short.

Security-by-Contract defines a methodology for guaranteeing security at run-time. This ought to be obtained by comparing the behaviour of the CONNECTor and the considered security policy in order to check whether they match, and consequently that the CONNECTor behaves as required by the policy. The main addition of SxCxT to this approach is the introduction of a trust module, taking care of managing the relations of the involved entities. SxCxT integrates security and trust for guaranteeing security policy at run-time. In particular, SxCxT provides a trust negotiation framework for establishing and managing trust relations among entities. Once the trust relations are studied from the trust module, SxCxT is able to decide if it is possible to execute the synthesized CONNECTor without any active control. Otherwise, correct execution of the CONNECTor has to be enforced. This guarantees at run time the satisfaction of local private policies of the Networked System (NS), that are shared neither with the CONNECT infrastructure, nor among local participants. Examples of local private policies are those regarding the GPS coordinates of the NS itself, or access rights to the private data stored on the device of the NS. Furthermore, using SxCxT we assure that no NS tries to attack the others by manipulating the part of the CONNECTor that it executes. The final product of the project is the implementation of the security enabler that works according to the Security-by-Contract-with-Trust framework in such a way that it is integrated with the rest of the enablers of the CONNECT architecture.



SxCxT - Security-by-Contract-with-Trust

Concerning the assurance of desired trust levels, we have introduced TMDL (Trust Model Description Language) as the basis to express and to compose a wide range of trust management systems and thereby support trust management across heterogeneous networked systems. Using TMDL, two heterogeneous trust models from two NSs willing to be CONNECTed can be modelled, composed and mediated. The composition is specified in terms of mapping rules between roles of the original models. Rules are then processed by a set of mediation algorithms to overcome the heterogeneity of the trust metrics, relations and operations associated with the composed trust models. We have provided a complete XML representation for TMDL. We also implemented several dedicated tools that (i) guide developers to check and create a valid TMDL description; (ii) automatically generate such description for the Java code of the corresponding trust management system; and (iii) enables the composition of any given trust management system according to given mapping rules. As part of such framework, we have also developed a TMDL editor that guides developers to create a valid and correct TMDL description that can serve to automatically generate the Java code of the corresponding trust management system.

CONNECTing Eternal Systems

This part of the research investigates the supporting architecture for on-the-fly CONNECTor generation and deployment, thereby providing a point of integration for the results from the other aspects of CONNECT.

Extending the state of the art in middleware and distributed systems

The work on the overall CONNECT architecture has led to a number of results that both add to and extend the current state of the art in middleware and distributed systems with respect to achieving interoperability in highly heterogeneous and dynamic pervasive computing environments:

- *Rich analysis of the state of the art in interoperability middleware and semantic web technologies.* The consortium first produced a comprehensive state of the art of middleware and data interoperability solutions; this identified five types of heterogeneity that are a barrier to achieving interoperability in networked systems:
    - *Discovery protocol* heterogeneity, where different protocols are used to advertise and search for services (e.g., SLP, Jini, UPnP).
    - *Interaction protocol* heterogeneity, where services use different protocols to exchange and use data, e.g., RPC protocols such as SOAP, and IIOP or different messaging protocols such as JMS or MSMQ.
    - *Data heterogeneity*, where applications may use data that is represented in different ways and/or have different meanings.
    - *Application heterogeneity*. The application interfaces may be different in terms of the descriptions of operations, e.g., the behaviour provided by one operation in one interface may be provided by multiple operations in the other interface.
    - *Heterogeneity of non-functional properties*. Systems may have particular non-functional properties, e.g., latency of message delivery, dependability measures and security requirements that must be resolved.

  The subsequent analysis of the state of the art showed that no solution from either the middleware or semantic interoperability fields addressed all of these heterogeneity dimensions.
- *The use of cross-cutting ontologies*. We identified the pivotal role that ontologies play in developing interoperability solutions. The semantic technologies support the automated reasoning about networked system meaning and then inform the matching, classifying and mapping of heterogeneous systems. We further demonstrated via the CONNECT architecture that ontologies can cross-cut system architectures and be used in the discovery and matching phase, the mapping and software synthesis phase, and in the final concrete application to middleware protocols.
- *Achieving Interoperability in the face of alternative middleware styles*. Previous interoperability solutions have focused on interoperability between networked systems that only employ a similar middleware style, e.g., RPC-based middleware protocols, messaging based. In the CONNECT architecture we investigated interoperability between alternative styles, and in particular built suitable implementation to achieve interoperability between networked systems involving the interaction of a publish-subscribe middleware-based application with a corresponding RPC-based application. In our evaluation we were able to show that interoperability was achievable. This was a significant result as it identified the increased breadth of the CONNECT solution beyond what the state of the art is capable of.

The CONNECT architecture & software

A key result of our work on the definition of the CONNECT architecture was the bringing together of the key disciplines and sub-disciplines within the consortium, to achieve an agreed and consistent overall approach:

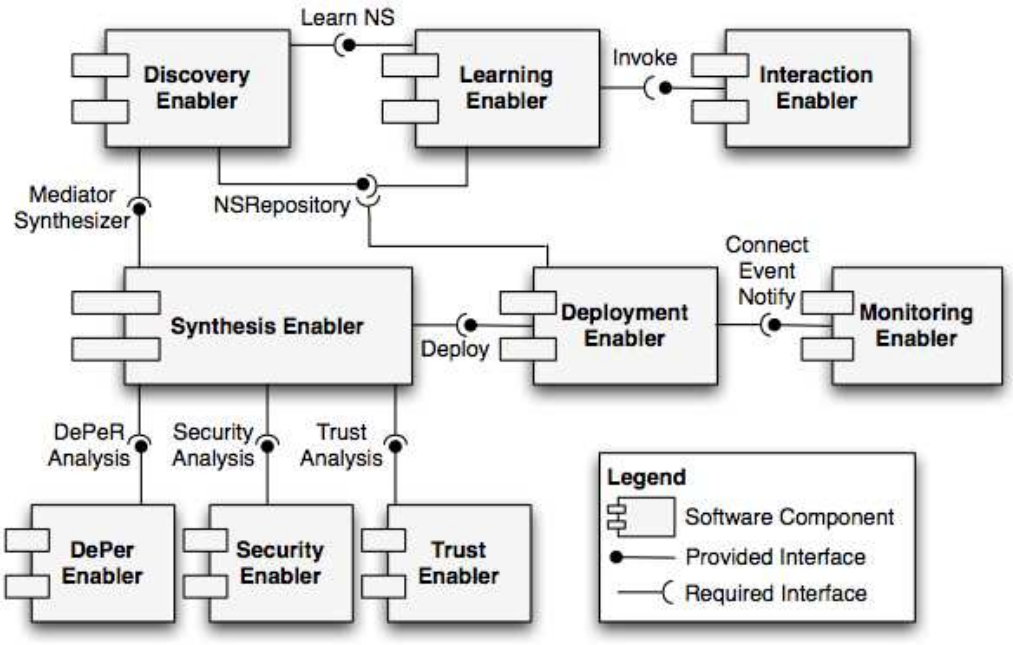
- *Modelling of networked system behaviour*. The CONNECT Networked System Model defines a runtime software artefact that accurately describes each networked system in order to inform the CONNECT process. The model instances contain a rich semantic description of each individual networked system in terms of their: role, interface syntax, behaviour and non-functional properties.
- *Integration of software enablers*. The Enabler Architecture describes how enablers (software components that perform the CONNECT process of creating CONNECTors) are composed and co-operate to complete the functionality of CONNECT. The Enabler Architecture is split into two phases:

  i. The *CONNECTion phase* performs the initial behaviour required to generate a new CONNECTor; and

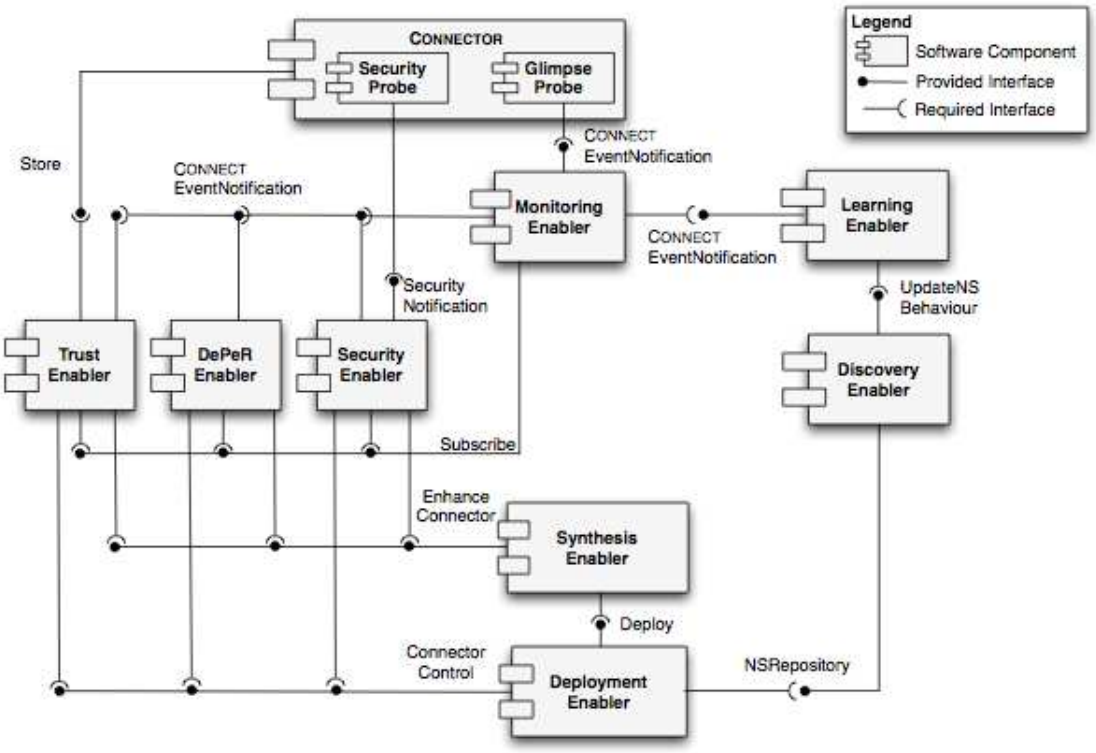
  ii. The *CONNECTability phase* monitors and adapts the CONNECTor in order to maintain dependability, performance, security and trust requirements.

  This final architecture was evaluated using interoperability case studies; these highlight how the CONNECT solution achieves correct interoperability.

  The results from the CONNECT architecture also demonstrate that the work of CONNECT exceeds existing state of the art middleware or semantic Web solutions because it is able to handle each of the five identified heterogeneity barriers to interoperability.

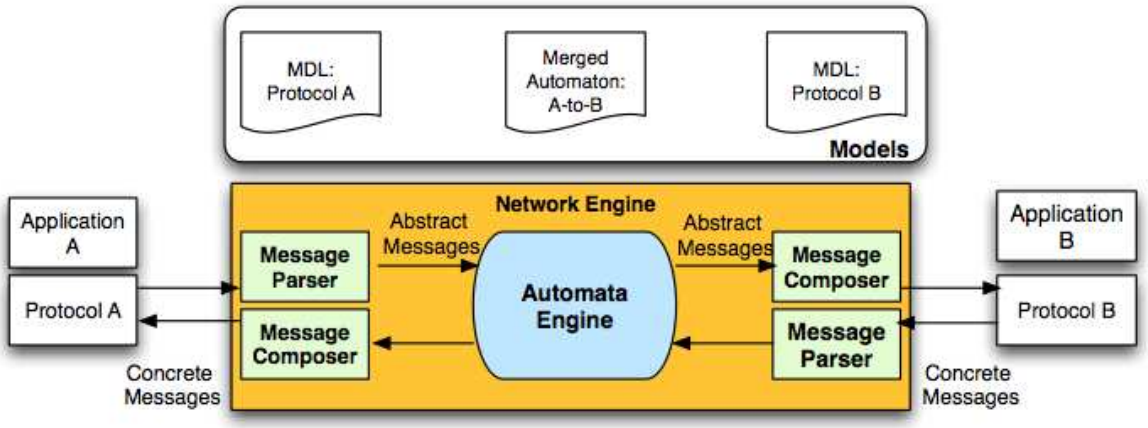


The configuration of the CONNECT architecture for the CONNECTion phase



The configuration of the CONNECT architecture for the CONNECTability phase

- *Starlink*. In order to create concrete CONNECTors, the project introduced the concept of k-Coloured automaton as the concrete model that specifies the mediation between two networked systems in terms of both application and middleware heterogeneity. The Synthesis enabler produces this model. Starlink then provides the software tools in order for these models to be interpreted; at interpretation time the application actions are bound to the annotated middleware protocol related to the action. This binding ensures that an application action transition in the k-Coloured automaton is transformed into the sending and receiving of middleware protocol messages. Starlink is now available to the wider distributed systems community as an open source project on SourceForge, and contains a number of existing middleware protocols: SOAP, XML-RPC, IIOP, SLP, UPnP, Bonjour, XMPP, AMQP, and Stomp.

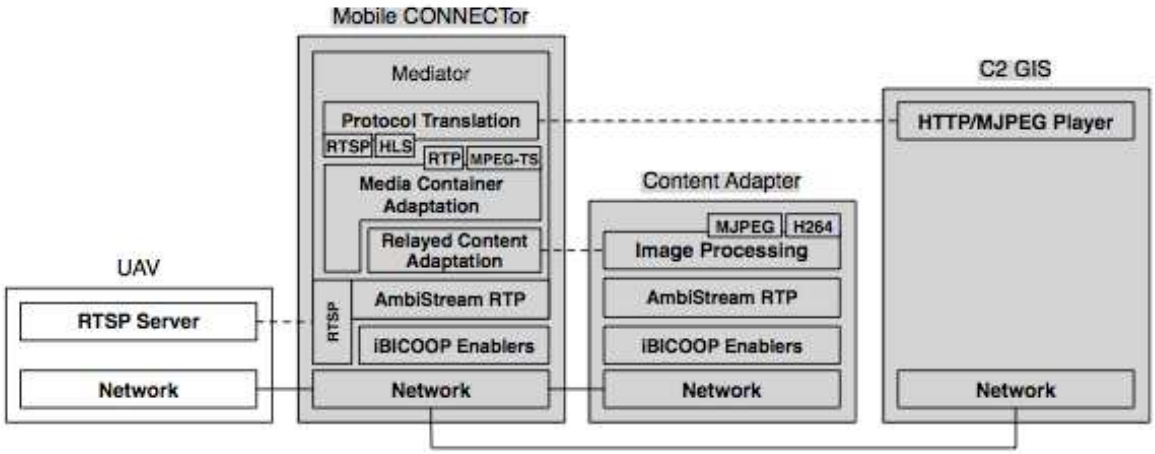


Architecture of the Starlink framework

Further Application of the CONNECT Results

Beyond the integration of the core CONNECT architecture and its subsequent evaluation using industry-strength use cases, we have also investigated the wider application of the architectural principles that form the core of the CONNECT process:

- *Mobile media streaming*. For the further exploitation of the application of the CONNECT results, we investigated an approach to interoperable media streaming applications in the mobile environment. Networked systems providing streaming services, e.g., video streaming using a streaming communication protocol (e.g., RTSP) offer different challenges for CONNECT or construction in order to address interoperability. The integration of CONNECT concepts with the iBICOOP-AMBISTREAM mobile collaborative middleware have illustrated that CONNECT can be applied to a wider range of protocol types (i.e., streaming) and within diverse domains such as mobile computing.



The AmbiStream CONNECTor architecture for interoperable media streaming

- *Models@runtime*. Explorative and collaborative investigation of the potential of Models@runtime to provide dynamic adaptation within the CONNECT architecture. This showed how runtime models could be used in the software engineering process to inform the regeneration/adaptation of produced CONNECTORS using the SM@RT tool from PKU; and similarly how to adapt the enabler architecture itself using SM@RT in order to optimize its use of resources, e.g., the load balancing of deployed CONNECTORS across distributed hosts.

Experiment and Assessment

CONNECT research was assessed based on industry-strength use cases in the two following areas:

- **Systems of Systems**, where we more specifically focus on using CONNECT for addressing the interoperability requirements raised by scenarios related to the European programme on *Global Monitoring for Environment and Security.*
- **Mobile collaborative applications***,* where we focus on exploiting CONNECT to address the interoperability requirements that arise for mobile applications deployed on heterogeneous mobile platforms and interacting with increasingly diverse Internet-based services, among which Cloud services.

Enabling systems of systems: the GMES use case

GMES (Global Monitoring for Environment and Security) is the European Program for the establishment of a European capacity for Earth Observation started in 1998. The services provided by GMES address six main thematic areas: land monitoring, marine environment monitoring, atmosphere monitoring, emergency management, security and climate change.

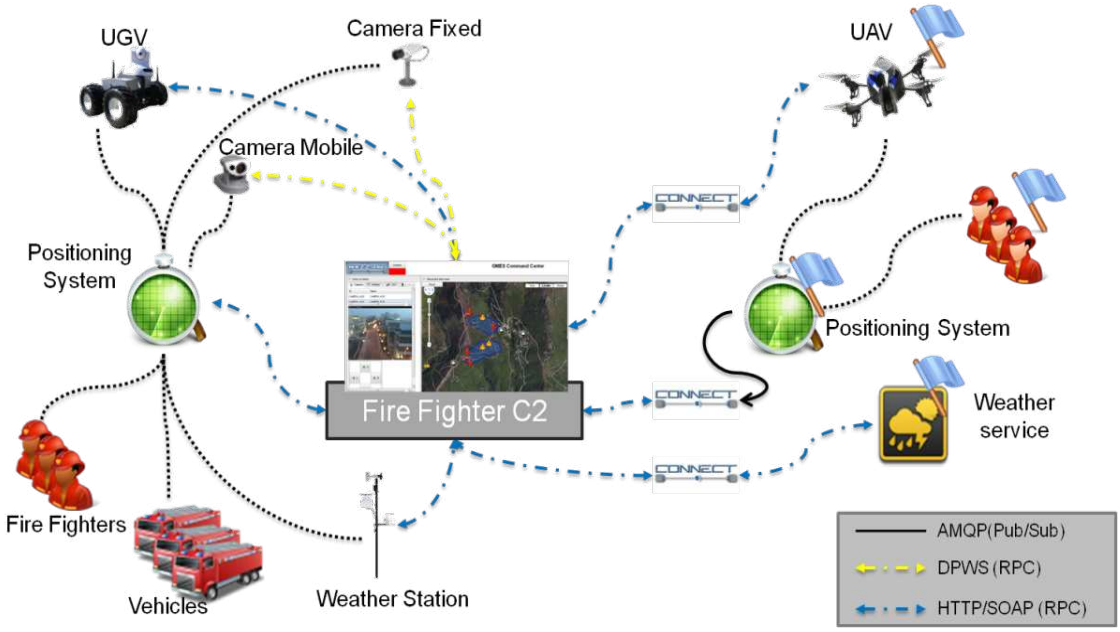
The emergency management service directs efforts towards a wide range of emergency situations; in particular it covers different catastrophic circumstances: floods, forest fires, landslides, earthquakes and volcanic eruptions and humanitarian crises.

Within CONNECT, we concentrated on the Forest fire situation. The scenario illustrates the management of forest-fire, close to a border village and a factory, between country A and country B. Forest monitoring and forest fire management in the country A are the responsibility of the Country A Command and Control fire operations centre (C2-A). For example in France, the CODIS, led by a professional fire fighter, is the regional authority in charge of coordination of operational forces during fires and disasters for one French department. Then, the C2-A centre must interoperate with a number of Networked Systems (NSs) of Country B so as to monitor the area. The table below lists the networked systems that are more specifically considered for the experiment.

Networked Systems	Middleware-layer protocols	
	Protocol	Exchange Pattern
NS 1 UAV	SOAP	RPC
	RTSP	Stream
NS 2 UGV	SOAP	RPC
	MJPEG	Stream
NS 3.1 Camera Fixed	DPWS	RPC
	RTSP	Stream
NS 3.2 Camera Mobile main	DPWS	RPC
	MJPEG	Stream
NS 4 C2 GIS	SOAP	RPC
	MJPEG	Stream
NS 5 Weather Station	SOAP	RPC
NS 6 Weather Service	SOAP	RPC
NS 7.1 Position System A	SOAP	RPC
NS 7.2 Position System B	AMQP	Pub/Sub

GMES NSs and protocols

The following figure depicts the CONNECTion we have been focusing on (see dashed lines labelled with the CONNECT logo).



CONNECTions in GMES

Based on the experiment using the GMES use case, we have been assessing the possible exploitation of CONNECT for the system of systems domain.

As defined in Wikipedia's page on Systems of Systems[1], several traits are inherent to System of Systems: (i) Operational Independence of Elements, (ii) Managerial Independence of Elements, (iii) Evolutionary Development, (iv) Emergent Behaviour, (v) Geographical Distribution of Elements, (vi) Inter-disciplinary Study, (vii) Heterogeneity of Systems, and (viii) Networks of Systems. While some of these traits are not relevant to CONNECT (such as Inter-disciplinary Study), the others have been gathered in four main objectives given the commonalities of their assessment methodology:

(1) Operational and Managerial Independence of Systems;
(2) Evolutionary Development and Emergent Behaviour;
(3) Heterogeneity of Systems; and
(4) Networks of Systems and Geographical Distribution of Elements.

It appears that the CONNECT solution to interoperability effectively complies with the four above requirements.

Enabling mobile collaborative applications

While in the desktop and Internet domains, services are loosely bound and easily interchangeable, the mobile domain exhibits very specific constraints and behaviours that go towards integrated and heavily controlled services. Indeed, mobile platform vendors, such as Google with its associated manufacturers with Android, and Apple with iOS, are intensively focusing on the vision of deeply integrating services within the mobile operating systems. This approach raises interoperability issues as older phones are not always updated, for technical or business reasons, and therefore cannot interact with newer phones due to protocol or content format mismatches.

At the same time, mobile devices store and manage much of the user's personal information such as contacts, timetable of personal appointments, location, or payment information that raise privacy-invasion concerns, which have led to a number of additional restrictions on mobile platform environments. Most notably, the distribution of mobile applications is heavily

[1] http://en.wikipedia.org/wiki/System_of_systems

controlled, and executable code cannot be generated and deployed at run time. For example, all iOS applications have to pass through a manual screening and approval process before being published on the Apple App Store, which is the only means of application distribution on Apple's platform.

Another major evolution in the design of collaborative applications in the mobile domain is the increasing dependence on Cloud services (or alternatively the availability and use of Cloud services on mobile devices thru native mobile applications). The multiplicity of personal or family connected devices (i.e., tablets, smart TV) is also to be accounted for, which results in the rising demand for exchanging, sharing, and synchronizing data at a global scale between different users and devices.

These major changes in the lifecycle of devices and applications, as well as constraints induced by mobility aspects, such as computational resources, connectivity, or battery life, introduce interoperability barriers between different platforms and also between applications.

We have thus extensively investigated the application of the CONNECT architecture in the mobile domain. In particular, the consortium was able to experiment and assess with a number of prototype implementations based on actual scenarios provided by the SME partner Ambientic. Ambientic develops solutions to enable collaboration among applications deployed on heterogeneous mobile phone platforms. The first product from Ambientic is a suite of mobile collaborative services, called U-Event, which is aimed at fostering communication among actors at any event (e.g., visitors, exhibitors and panellists at trade show or conference). In correlation with Ambientic business needs, we have implemented four use cases, which are representative of the innovations Ambientic aims to introduce in the U-Event platform. The goal of those use cases was to evaluate if the CONNECT architecture can sustain interoperability in real world mobile settings with particular constraints. Whenever the specific needs of the mobile environment required extensions to the CONNECT architecture, the further goal was to provide enhancements to the architecture so as to obtain a holistic approach for mobile interoperability.

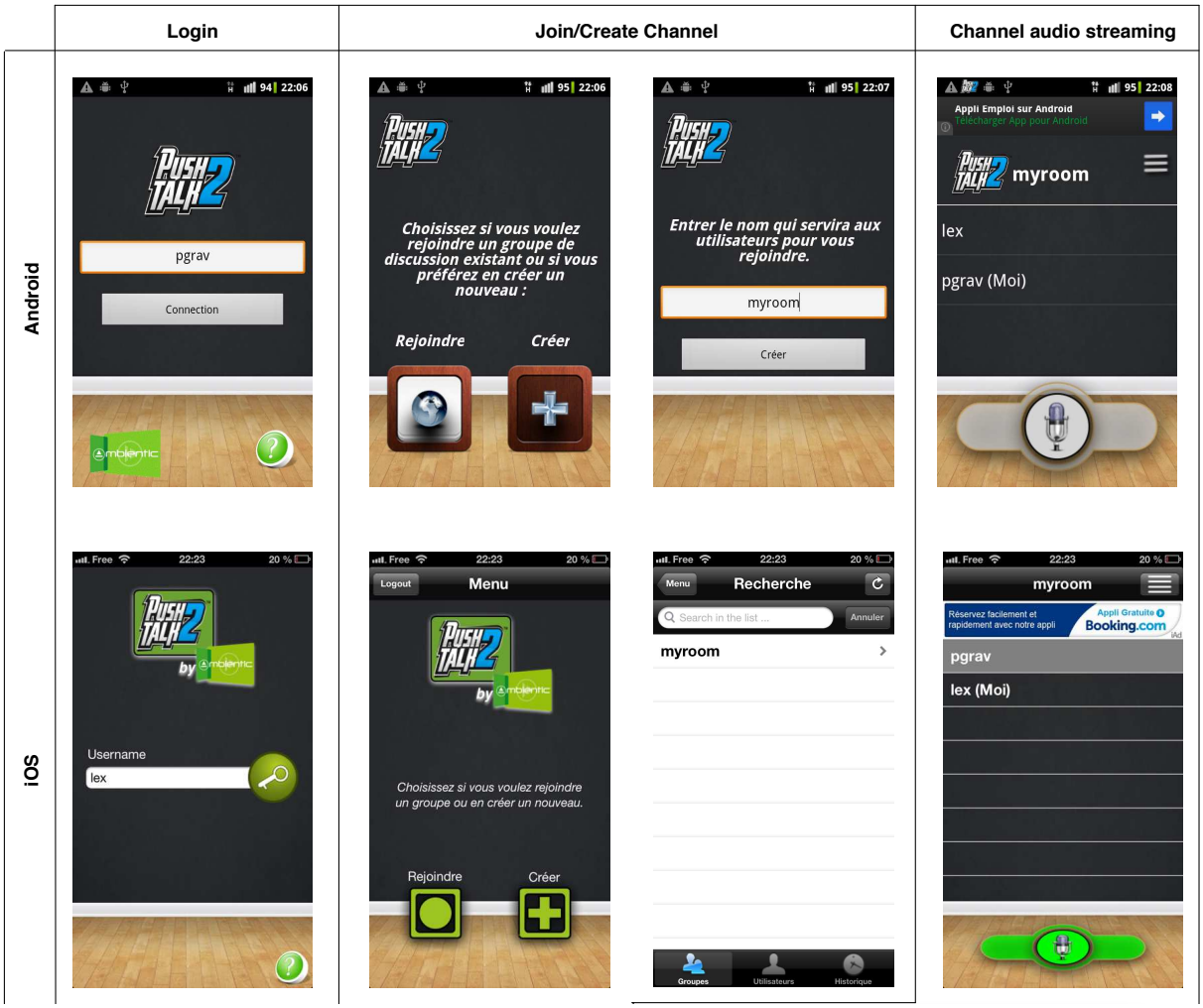
We specifically concentrated on the following use cases within CONNECT:

- In order to provide, through U-Event, conference's attendees and speakers the capability to use their smartphones to broadcast audio/video streams, we implemented two prototypes that highlight how CONNECT technologies help to achieve multimedia stream interoperability. The former is on mobile video streaming interoperability and highlights Control Protocol as well as Media Container interoperability. The latter, called Push2Talk, implements a "Walkie-talkie" application to be used among a group of people and highlights mobile group communication and interoperability with legacy audio clients.

  The release of the Push2Talk application on the AppStore and on the Google Play store further allowed us to experiment with a large real world deployment of a mobile application, which uses CONNECTors to solve on-the-fly audio streaming interoperability, and to assess group communication scalability.

- Another innovation consists of enabling U-Event to get access to heterogeneous Cloud providers to allow event participants to retrieve their documents and data from the Cloud and share them with other participants. To do so, we implemented a third use-case that demonstrates app-to-Cloud mediation.

- The U-Event application interacts with the Ambientic server, which is responsible to manage users' data and further bridging the U-Event application with organizer's event planning services. Currently, the bridging process is handmade and time consuming. This is why we envisioned to experiment CONNECT technologies through a real business case requiring interoperability with different event management systems, and involving complex interaction protocol stacks with cross-layer dependencies.



Push2Talk GUI on Android and IoS

The above use cases then allowed the consortium to assess the ability of the CONNECT solutions to overcome the interoperability issues that arise in the mobile domain:

1. **Interoperability between mobile deployed applications** where CONNECTors must mediate mobile collaborative applications that are either co-located or installed on separate mobile devices.
2. **Interoperability between mobile applications and Cloud services** where CONNECTors must enable mobile applications to get access to miscellaneous remote Cloud and networked services.
3. **Improvement of the mobile application development process** where the CONNECTor synthesis process should drastically ease the development of mobile collaborative applications, especially in terms of development effort due to automation.
4. **Handling of mobile context dynamicity** where collaborative mobile applications must face changing network conditions that are inherent to mobile environments.
5. **Mobile application scalability** where architecture and usage must scale with the number of users.

The table below matches the above issues with the four use cases that we considered for experiment.

Criteria/Use Case	Live video streaming	P2T	Cloud Storage	Event Management
1. Mobile app interoperability	X	X	X	
2. Mobile app-Cloud interoperability			X	X
3. Improved development process	X	X	X	X
4. Mobile context dynamics	X	X	X	
5. Scalability	X	X		

Mobile use cases requirements

Overall, CONNECT successfully bridged interoperability gaps existing in real world mobile applications with relatively small efforts on the part of application developers.

The number of different complex interoperability scenarios where we were able to effectively use CONNECT to simplify interoperability among software components makes us confident that the CONNECT architecture is a promising approach to achieve eternal CONNECTivity of networked components.

Impact, Dissemination, Exploitation & Collaboration

Impact

The impact of CONNECT relates to enabling Eternal systems as part of the FET ICT Forever yours initiative (see http://cordis.europa.eu/fp7/ict/fet-proactive/ictfy_en.html).

The table below first recalls the targeted contributions to sustaining eternal systems, as stated in the CONNECT description of work, and further assesses the extent to which the foreseen contributions are met.

Topics (from the call)	**CONNECT contributions**
Eternal systems	The CONNECT project specifically focuses on the eternality of systems with respect to their interaction with the networked environment. In our view, the key requirement for making networked systems eternal is to enable them to CONNECT with other networked systems according to their provided/required functionalities but also quality of service, while being agnostic to the specific (from application- down to middleware-layer) protocols run by the networked systems for interaction. The CONNECT project addresses this issue by introducing *emergent connectors* (or *emergent middleware*) that are dynamically generated according to the respective interaction behaviour of the dynamically linked systems, thanks to the dynamic learning and abstraction of the interaction behaviour of networked systems. As a result, the realized CONNECTion bridges possibly highly heterogeneous systems. More specifically, any system that uses some discovery protocol for advertising/requiring networked system functionalities and fulfils some minimal assumptions allowing the learning of its interaction behaviour will CONNECT. So, CONNECT is not yet another middleware: it is based on the assumption that the networked systems can be highly heterogeneous, retain their own interaction protocols, and are not required to come along with their complete specification following a global common schema. **Assessment:** CONNECT has delivered an overall S&T solution to emergent middleware / CONNECTors, from theoretical foundations to working software prototypes of the supporting enablers.
Theoretical and practical framework for extremely long-lived systems	CONNECT sets the basis for eternal, i.e., extremely long-lived, CONNECTion within systems' networks by introducing a theoretical framework for the formal specification of emergent connectors and related interaction behaviours. The CONNECT theoretical framework will support automated learning, reasoning and synthesis of interaction behaviours so that emergent connectors can be dynamically generated according to the interaction behaviours of networked systems. The CONNECT architecture then introduces the practical framework that builds upon the elicited connector theory to concretely generate

Topics (from the call)	CONNECT contributions
	emergent connectors within the network by offering: (i) an ontology-based language for the specification of networked systems semantics together with multi-protocol, interoperable system discovery, and (ii) networked enablers to learn the interaction behaviour of systems and thereupon synthesize emergent connectors to be deployed among them. **Assessment:** CONNECT has formulated a comprehensive specification theory for components and connector behaviours that introduces formal notions such as refinement for comparing components and determining when they can be substituted for one another, and useful composition operators. The operation of quotient in particular supports the principles of synthesis, and can be used for proving technical results about the properties of mediators synthesized using the CONNECT synthesis enabler, such as deadlock and error freedom. CONNECT has further delivered a number of enablers supporting the synthesis, deployment and execution of dependable emergent middleware for networked systems to interoperate on the fly.
Future-proof systems providing machine-independent functionalities	Through dynamic learning, reasoning, and synthesis, CONNECT provides machine-independent functionalities, i.e., covering any existing and potentially future system and middleware technology, which makes CONNECT itself and CONNECTed systems future-proof. **Assessment**: See above.
Self-sustaining and evolving systems	Emergent connectors are adaptable to cope with the ever-changing environment. As a result, networked systems in the CONNECTed world evolve with respect to their interaction with the networked environment. **Assessment**: Self-adaptive emergent middleware has been tackled upon based on models@runtime principles. However, further research is required for making emergent middleware self-adaptive.
Knowledge, diversity and time	CONNECT specifically deals with knowledge assets related to systems functional behaviour, interaction behaviour and behaviour ensuring non-functional properties. CONNECT introduces a comprehensive ontology-based specification for characterizing the semantics of networked systems so that they can meet according to provided/required functionalities and quality of service. CONNECT enablers deployed in the network then learn the interaction behaviour of networked systems, effectively synthesizing related specifications and exploiting this knowledge to generate emergent connectors enabling systems to communicate. The above process, from learning to connector generation, is fully dynamic, with specifications evolving as new behaviours are learned, thus making it "*robust against ageing*". **Assessment**: The CONNECT architecture integrating the various CONNECT enablers to realize emergent middleware, and associated software prototypes indeed support a fully dynamic creation of emergent middleware, on the fly.

Topics (from the call)	CONNECT contributions
New approaches for eternal and reliable access to knowledge assets	CONNECT enablers deployed within the network manage and store the knowledge about the systems' functional, interaction and non-functional behaviour, based on effective abstraction, discovery and a thorough learning process. As a result, the management of such knowledge is future proof, being dynamic abstraction-based and loosely coupled with networked systems, however, representative of existing and potentially future systems. **Assessment**: See above.
Secure and dependable	Since CONNECT aims at enabling open, highly heterogeneous networked systems to dynamically CONNECT, security and dependability are among its central concerns. Indeed, a significant part of the CONNECT solution lies in enablers for dependability assurance in open, highly heterogeneous and dynamic networked environments, including: • A thorough definition of dependability metrics accounting for the eternal dimension of networked systems, emphasizing risks rather than guarantees of properties in the networking of nodes; • On-line verification and validation of CONNECTed systems and emergent connectors; • A trust model and related framework for the CONNECTed systems and enablers; • Custom privacy-enhancing technology to be embedded within emergent connectors for dealing with privacy issues that arise in the seamless networking of systems. **Assessment**: CONNECT has delivered a CONNECTability framework toward ensuring the dependable CONNECTion of networked systems, including support for: dependability and performance analysis and enhancement of synthesized CONNECTors, and interoperable trust and security management. On the other hand, the elicitation of custom privacy-enhancing technology for emergent middleware remains an area for future work.
Methods and tools for high-level verifiably secure and dependable programming	CONNECT targets automated software engineering of the assembly/composition of networked systems, actually generating the connectors that bridge heterogeneous networked systems. By adopting a rigorous automated synthesis process and further devising networked enablers for dependability assurance, supporting among others online verification and validation, CONNECT directly contributes to offering methods and tools for dependable programming. **Assessment**: See above.
New metrics to aid assessability of the security & dependability of highly distributed & heterogeneous software or of ambient systems	CONNECT revisits the definition of dependability and security metrics to account for long-lived systems that network seamlessly. **Assessment:** To allow for model-driven specification of non-functional properties, CONNECT introduces the Property Meta-Model, that supports models of dependability, performance and security related properties as well as complex events. While the meta-model has been applied for synthesis and monitoring, its application to other phases of CONNECT process remains area of future work.

The next table assesses CONNECT results against the expected impact of the ICT Forever Yours initiative, still based on the a priori estimate sketched in the project's description of work.

In a nutshell, CONNECT meets all the expected contributions with the exception of: (i) enabling self-adaptive emergent middleware, and (ii) enforcing privacy, which mostly remain areas for future work.

Expected impacts (from the call)	**CONNECT contributions**
Versatile interaction with modules, systems and services in the environment	Emergent middleware make interaction with networked systems *versatile*, as systems are able to interact with any system that matches required/provided functional properties and quality of service, independent of the respective interaction protocols (from the application- down to the middleware-layer) that are run by the systems. Such a challenge is achieved by the CONNECT theoretical and practical framework enabling the automated learning and synthesis of networked systems' interaction behaviours. Thanks to CONNECT, networked systems are able to join and compose seamlessly in the network, without imposing requirements upon software technologies embedded within systems.
Adapting to change in the environment with minimal intervention	Again, thanks to the dynamic, seamless networking among systems allowed by emergent middleware, systems smoothly adapt to changes in the environment. Indeed, connectors emerge and further evolve according to the evolution of the interaction behaviours of the systems that seek to communicate. Still, as mentioned previously, enabling self-adaptive emergent middleware remains area for future work.
Harnessing disperse and dynamic content by exchanging knowledge at a semantic level	CONNECT deals with knowledge associated with functional, interaction and non-functional behaviours, which are addressed at a semantic level to provide the right abstraction for the learning and synthesis of interaction protocol behaviours. In detail, CONNECT addresses ontology-based partial description of system semantics, and introduces specialized formalisms for behavioural semantics underlying the specification of interaction protocols.
Preserving and even changing original functionality and properties over time	While the individual functionalities provided by networked systems are preserved by CONNECT, functionalities provided by the composition of networked systems using emergent middleware are emergent ones according to the specific systems that get CONNECTed. In addition, the realized CONNECTion of systems leads to the adaptation of the interaction protocols that are run, so that they can successfully coordinate.
Providing security to the environment	CONNECT introduces interoperable security mechanisms, dynamically synthesized and fully integrated in the CONNECT architecture.

Expected impacts (from the call)	CONNECT contributions
Offering assessable security of systems	CONNECT offers on-line dependability assessment mechanisms for both the dynamically networked systems and the emergent connectors. In particular, CONNECTors are enriched with security contracts, expressing the desired security features. Those may be *a priori* analysed and *a posteriori* monitored.

Dissemination

Dissemination has been one of the central activities within the CONNECT project and all partners have contributed to it. The objectives of the dissemination activity are to ensure the dissemination of the results of the project toward researchers and developers, at the international level, so as to in particular foster uptake of the project results.

The consortium actively disseminated CONNECT results through publications in major International venues and presentations toward international audience.

All the publications are further made available via the HAL open archive to ensure broader availability. In a nutshell, CONNECT publications include 110 conference papers and 33 book chapters and articles.

Simply visit https://www.connect-forever.eu/publication.html to browse the CONNECT publications.

In addition, CONNECT organized two major international events to foster research in the area of eternal software and next generation middleware, as investigated within CONNECT. The two events were:

(1) The CONNECT summer school on “Eternal networked software systems”, and

(2) The international workshop on the “Future of middleware”.

These events are detailed in the two following sections.

The SFM-11 summer school on Eternal Networked Software Systems

The CONNECT summer school, entitled SFM-11:CONNECT, the 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Connectors for Eternal Networked Software Systems, was held on 13-18 June 2011 at the University Residential Centre of Bertinoro (see http://connect-forever.eu/bertinoroA.html).

The school was organized within the framework of SFM (Int. School on Formal Methods for the Design of Computer, Communication, and Software Systems), which has hosted a series of successful summer schools on a wide spectrum of topics. We in particular thank the local organizer, Prof. Marco Bernardo at the University of Urbino, for excellent support.

The goal of the CONNECT summer school was to present and make students acquainted with important concepts and techniques for achieving interoperability in networking, and to present relevant material that is in the focus of the CONNECT process, i.e.: modelling, reasoning, synthesis, learning, and monitoring of networked systems. In addition, the school covered material on the architecture of networked systems, as well as a broader perspective on enabling eternal systems. Each day typically consisted of three lectures (typically 90 minutes each), one of which held by an invited speaker, followed by a lab session.

See https://www.connect-forever.eu/bertinoroA.html for more.

For the purpose of the summer school, a volume of lecture notes has been published as Lecture Notes in Computer Science, vol. 6659, by Springer, ISBN 978-3-642-21454-7.

See http://www.springer.com/?SGWID=0-102-24-00&searchType=EASY_CDA&queryText=lncs+6659 for more.

The school attracted 31 attendees (see picture below) from 12 countries, out of which 10 were female. More information on the origins of participants can be found at: http://www.sti.uniurb.it/events/sfm11connect/participants.html.

SFM-11 attendees and lecturers

The Future of Middleware Workshop at Middleware'2011

We are witnessing a period of rapid and intense change in distributed systems, at a rate that is unprecedented since the inception of the subject in the early 1980s. With the advent of Cloud computing, for example, we can see the deployment of very large-scale distributed systems offering a range of novel and exciting new services, including interesting new paradigms for large-scale computation. Distributed systems are also becoming significantly more heterogeneous, spanning very small devices embedded in the physical environment around us through data centres housing massive cluster computers. In addition, users of distributed systems are often on the move, resulting in significant context changes over time, which the system must adapt to. Networking technologies also continue to evolve with, for example, the emergence of a range of new ad hoc networking techniques and peer-to-peer approaches to implementing core network services.

Middleware retains a core role in distributed systems, offering key properties such as abstraction, interoperability, openness, and supporting a range of non-functional properties. However, middleware is coming under increasing pressure given the trends highlighted above:

- What are the right abstractions for the development of future distributed systems given the scale of complexity of the underlying infrastructure? How can we abstract over this complexity? What do we need in terms of middleware APIs, programming languages and associated software engineering methodologies?
- How do we achieve interoperability and openness in this new world we find ourselves in, especially given the extreme heterogeneity we encounter in the distributed systems of today? What principles and approaches do we need to deal with such extreme heterogeneity? Do existing approaches to interoperability and openness still work?

- How do we achieve the desired level of dependability and security? Again, what principles and techniques do we need to achieve such non-functional properties and how do we embed such techniques within the architectures of our middleware platforms?

These are very difficult questions to answer and it is clear that the middleware community faces some major challenges over the next few years in keeping up with the pace of change. These issues were specifically discussed at the Future of Middleware event (FOME'11) that was held in Lisbon on 13th December 2011 in conjunction with the ACM/IFIP/USENIX Middleware conference, the premier conference in the area of middleware principles, architecture and engineering (http://2011.middleware-conference.org/fome/fome2011).

The event brought together a number of invited leading researchers (see picture below) in the field, selected to offer comprehensive coverage of the key issues identified above. The aims of the event were to take stock of where we are in middleware, to address the key challenges facing the field, to establish an agenda for the next wave of middleware research and, most importantly, to stimulate researchers and build a community to address the significant challenges we face. In a nutshell, what should middleware look like in 2020?

FOME speakers and organizers

Next to the event a dedicated special issue of the Journal of Internet Services and Applications – JISA – was published on May 2012 (Volume 3, Issue 1).

See http://link.springer.com/journal/13174/3/1/page/1 for more.

Exploitation

Use cases developed within CONNECT focused on scenarios targeted by the industry partners, namely Thales and Ambientic, so as to foster take up of the project results in an industrial setting. Another exploitation path is that of open source software, as most of the CONNECT software prototypes are released open source.

We outline below the exploitation plan for CONNECT technologies based on potential industrial exploitation and open source software take-up. Obviously, these exploitation channels are complemented by the exploitation plan of the academic partners that will leverage the CONNECT research results within follow-up training, research and innovation activities.

Thales exploitation in the systems of systems domain

Thales is focusing its exploitation activities on improving its current operation in existing markets, and on the creation of and preparation for new markets.

In the markets of defence, security, space, aerospace and ground transportation, Thales provides the tools and technologies to gather, process and distribute information coming from an increasing number of heterogeneous sources (sensors, databases, business information) in

order to understand complex situations and decide and act in a timely fashion to obtain the best outcomes.

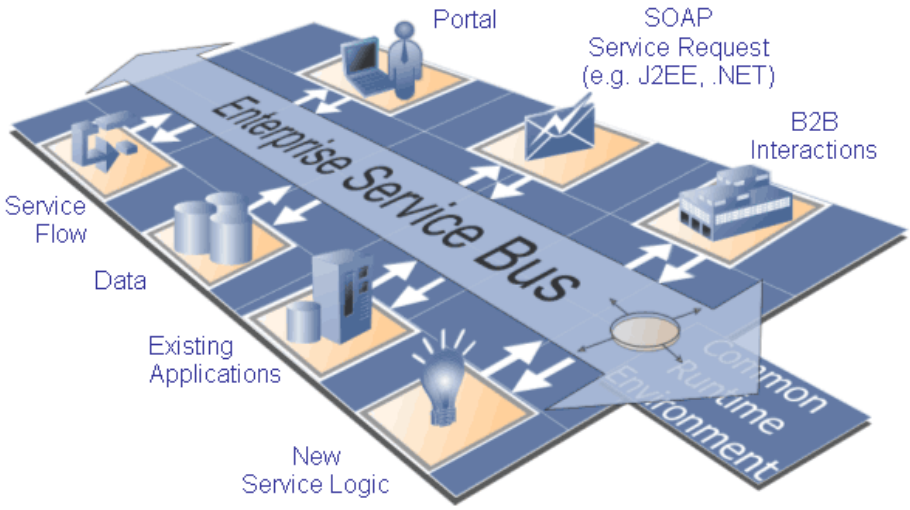
In a world with greater mobility and growing complexity, devising solutions that ease the connection with diverse sources of knowledge is a key objective of Thales, for which CONNECT S&T developments are much relevant.

Among other things, Thales is an integrator expert and one of its missions is to integrate existing system into a new system of systems providing new functionalities, while having no impact inside the already existing systems that continue to work independently.

The usual solution to achieve the above is to combine the functionalities of the old systems through an enterprise service bus (ESB) and create specific connectors from the old systems towards the ESB.

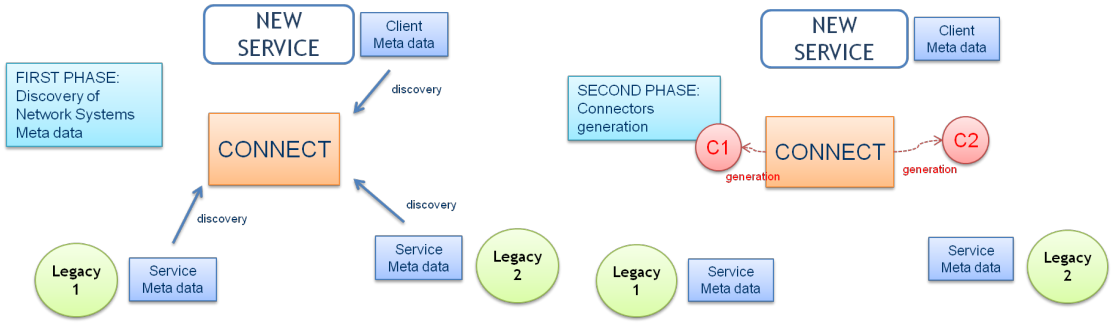
Still, much work needs to be done to make this solution functional:

- Deploy the ESB
- Develop specific connectors to existing systems in a specific protocol
- Develop the chain of transformation

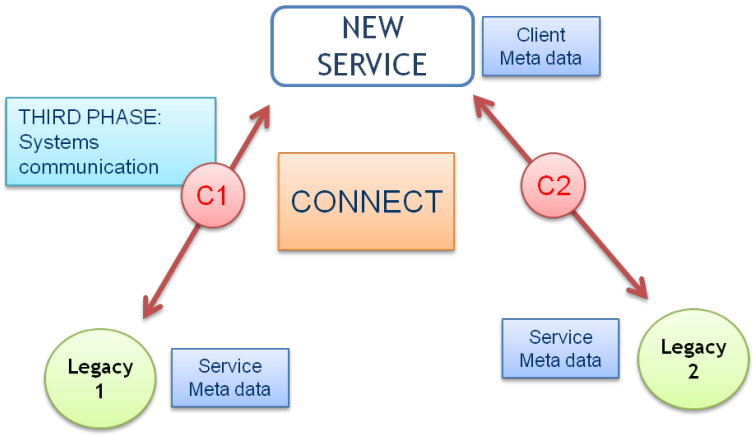


ESB style integration

CONNECT then appears as an appealing alternative. As CONNECT provides a way to establish seamlessly CONNECTors between networked systems that are providing and requiring matching functionalities, why don't we replace the ESB by CONNECT? In other words, why not just focusing the effort in specifying the meta-data of the existing systems and creating a system of systems based on client meta-data information that will require the use of these systems? The process is illustrated with the pictures below.



Systems discovery and CONNECTors generation



Seamless communication

Ambientic exploitation in the mobile collaborative applications domain

Ambientic (www.ambientic.com) develops solutions to enable collaboration among applications deployed on heterogeneous mobile phone platforms (i.e., iPhone, Android, Blackberry, WindowsMobile), including a middleware and other software tools to facilitate and accelerate the development of multi-platform applications. Ambientic thus offers innovative solutions to mobile ad hoc interaction. The first product from Ambientic is a suite of mobile collaborative services, called U-Event, which is aimed at fostering communication among actors at any event (e.g., visitors, exhibitors and panellists at trade show or conference). Given the large diversity and specifics of today's mobile platforms, and the high interoperability requirements of mobile applications, Ambientic is particularly interested in leveraging and further adapting CONNECT solutions to reduce the effort needed to create multi-platform collaborative applications.

As a first step, Ambientic investigates exploitation of CONNECT solutions for: (i) leveraging interoperable AV streaming to provide advanced streaming-based applications across mobile platforms, and (ii) the effective integration of mobile applications with cloud-based services.

Ambientic plans to leverage results in the area of interoperable AV streaming, thru 2013, to propose live multimedia services, in particular to event organizers and participants but also to other domains of activities. In particular, Ambientic plans to develop:

- Push-to talk and Video-calling multi-platform services that adapts automatically to local as well as 3G/4G network availability.
- Services for broadcasting audio and/or video during special events such as keynotes and panels. Integration with the Ambientic U-Event suite will enable the broadcasting of simultaneous translation/interpretation, multiple video feeds, or easily engaging participants during Q&A sessions without passing around microphones.

Delivering such services on participants' smartphones will significantly reduce organizers' costs related to the rental and management of dedicated devices.

Another exploitation path for the CONNECT technology relates to sustaining interoperability with Cloud services. Indeed, Cloud services complement the mobile experience by providing resources that are easily accessible and unavailable on current generation mobile phones and smartphones. It is widely recognized that mobile and Cloud integration will accelerate the consumerization of IT, and pave the way for the "Internet of things". However, the current trend of mobile software providers to create non-interoperable Cloud services to support their mobile products (e.g. iCloud, Google Docs, Office365, etc.) and the myriad of innovative services with similar purposes (e.g. DropBox, Wuala, Mozy, etc.) create yet another level of heterogeneity that hinders collaboration among mobile users. Building collaborative applications thus require mobile apps to interoperate with the various Cloud services currently used by mobile users.

In the particular scenario of event organization, dozens of Cloud-based services are now available to handle a number of organization activities such as registration, ticketing, mailing, contractor management, etc. Currently, each organizer selects the best Cloud-based tools according to their requirements. Integrating these tools with each other as well as with the tools used by other actors cooperating to organize an event (e.g. contractors, visitors, exhibitors, etc.) is a time and resource intensive process. For effective mobile and Cloud integration, it is necessary to deal with several issues that arise as a consequence of this heterogeneity, such as:

- The differences between mobile platforms and the protocols they support,
- The differences between the formats of data, communication paradigms and APIs associated with the services accessed from mobile terminals and in particular the Cloud services,

Within CONNECT, we worked on the above issues, focusing on design-time solutions for mobile-Cloud interoperability, as run-time solutions are not feasible due to mobile platform constraints as well as limitations imposed by application stores. We first focused on server-based mediation for mobile-Cloud interoperability, using actual conference management services. We investigated preliminary solutions to deal with middleware interoperability in terms of stacks of composable protocols, and to handle cross-layer dependencies between protocols and data interoperability. We then investigated mobile-based mediation for dynamically replacing Cloud services used to support collaborative mobile applications. This work identified a specific interaction pattern for mobile applications and services on smartphones that can be leveraged to deploy and use mediators directly on the mobile.

Ambientic plans to leverage these CONNECT results to enable the automatic integration of services into any event IT context, and therefore speed-up the integration process with event organizers. More generally, as individuals are increasingly relying on Cloud storage services to seamlessly access their content across their mobile and fixed devices, and new services are emerging constantly, we expect CONNECT results to allow us to quickly deliver compelling collaborative applications that empower users to share content regardless of their terminals or online service providers.

Beyond the event organization market, Ambientic expects CONNECT results to ease the development of collaborative applications involving mobile-Cloud interactions, and gain the capability to introduce novel collaborative applications that take the Cloud into account when enabling users to share content. Ambientic expects new market opportunities to open, as the Internet of Things and Digital Cities become realities.

Collaboration

The CONNECT consortium took part in the FET EternalS Coordination Action (https://www.eternals.eu/) that was launched in March 2010.

Latest research work within ICT has allowed to pinpoint the most important and urgently required features that future systems should possess to meet users' needs. Accordingly, methods making systems capable of adapting to changes in user requirements and application domains have been pointed out as key research areas. Adaptation and evolution depend on several dimensions, e.g., time, location, and security conditions, expressing the diversity of the context in which systems operate. A design based on an effective management of these dimensions constitutes a remarkable step towards the realization of Trustworthy Eternal

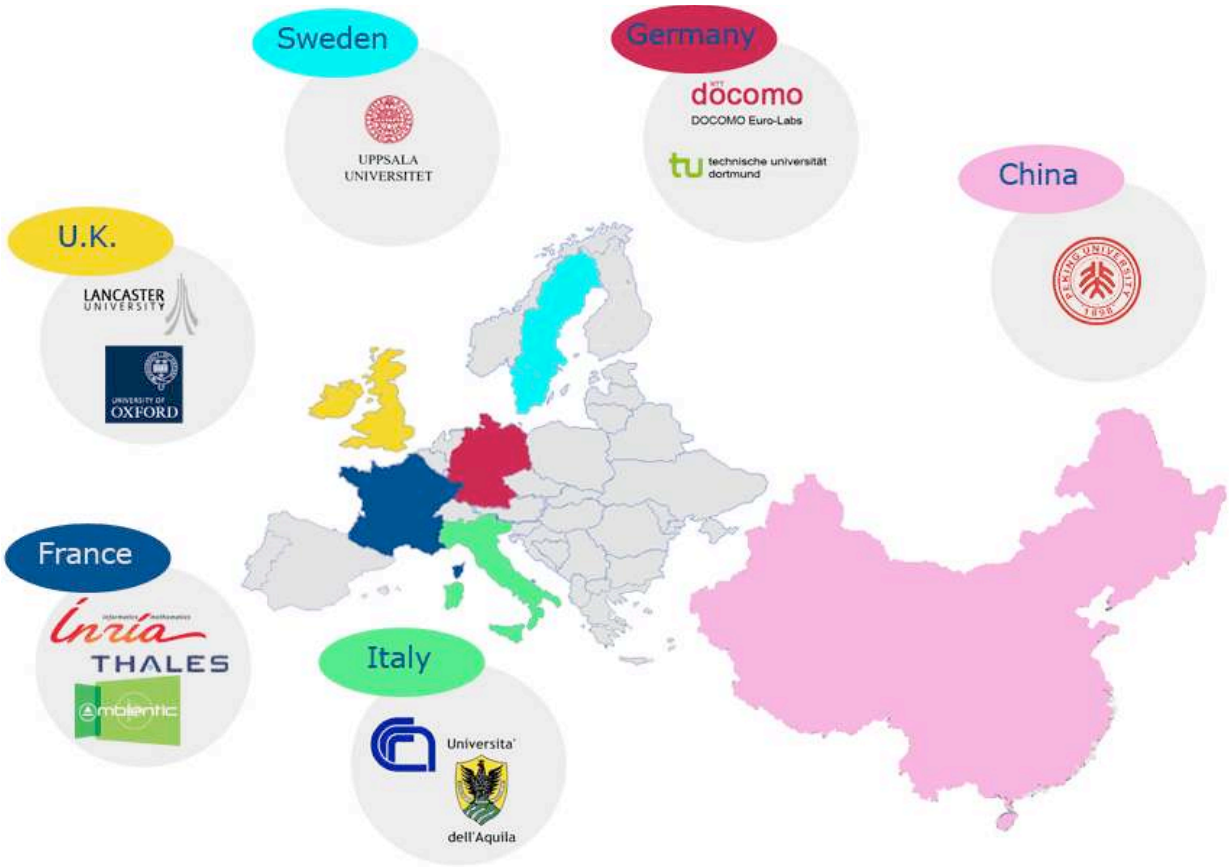
Systems. The EternalS Coordination Action specifically aims at coordinating research in that area based on a researcher Task Force together with community building activities, where the organization of large workshops and conferences is just one of the tools that will be used to conduct a successful CA.

EternalS created the conditions for mutual awareness and cross-fertilization among the 4 ICT-Forever Yours - FET projects (FP7-ICT-Call 3): LivingKnowledge, HATS, CONNECT and SecureChange. These projects have been conducting research in key ICT areas: (i) automatic learning of systems capable of analysing knowledge and diversity with respect to their complex semantic interactions and evolution over time (e.g., diversity of opinions due to sequences of events), (ii) exploitation of formal methods for the design and networking of adaptive and evolving software systems, where security policies and fully connected environment represent fundamental properties of effective present and future systems. Thanks to the indirect participation of the above-mentioned projects, EternalS actively involved many researchers from both academic and industrial world in its action. This allowed for (a) thoroughly studying eternal systems' dimensions such as diversity & time awareness and self adaptation & evolution by automatic learning, in fields of relevance, i.e., Knowledge, Software, and Networked and Secure Systems; and (b) writing the roadmap for interesting and successful future emerging technology.

Within EternalS, the CONNECT consortium actively contributed to the international workshops organized and/or supported by EternalS and to the EternalS task forces. Further, active collaboration in the area of machine learning have been undertaken with partners of EternalS, beyond the CONNECT consortium.

Contact Details

The CONNECT Consortium



The FP7 ICT FET CONNECT Project

- **FP7 Theme:** ICT-2007.8.6, FET Proactive 6: ICT Forever Yours
- **Grant agreement:** 231167
- **Budget:** Total cost: 6.5M€ - EC contribution: 4.8M€
- **Starting date:** February 1, 2009
- **Duration:** 46 months

CONNECT Contact Information

Project coordinator: Valérie Issarny, Valérie.Issarny@inria.fr.

Project Web site: www.connect-forever.eu/