Network–aware programming and interoperability

A Joint Project of Microsoft Research Europe
In collaboration with the Departments of Computer Science
of the Universities of Bologna, Firenze, Pisa, and Politecnico di Milano

General Introduction to the Project

The convergence between telecommunications and computing offers new opportunities and challenges for a large market of innovative network–wide services. However, most assumptions about traditional asynchronous distributed services over Local Area Networks are being challenged over the Internet by technological advances along two lines. On one hand, the growth and pervasiveness of the Internet is enabling novel asynchronous services to a much larger community of users; this calls for a sharper attention to issues related to fault tolerance, flexibility, and customizability. On the other hand, the recent advances in the area of wireless and ad hoc networks and the availability of powerful portable info appliances at a reasonable price is fostering the diffusion of mobile computing, which provides support to nomadic users; and this calls for software architectures including context–aware components.

Software design methods for the new architectures are different not only for scalability, connectivity, heterogeneity, and autonomy, but specifically for the ability of dealing with dynamic and unpredictable changes of their underlying network environment relatively to the sites where applications are executed (e.g. with respect to network connectivity, lack of resources, node failures, network reconfigurations and so on). The essential point is that these applications must be capable of reacting and adapting to dynamic changes of their network environment.

Some success has been achieved in the development of applications over the Web. However, the current applications are often collections of ad hoc modules glued together with ad hoc mechanisms without adequate integration or exploitation of the reference software architecture of the Web. From a programming language viewpoint a major issue of network–aware programming models is that the focus of the programming effort is less and less algorithmic, and more and more structural, or even architectural. In fact, a major problem is defining a basic computational and structural model to enable users, programmers, and designers to reason at the right level of abstraction by considering all and only the relevant aspects of the domain and the underlying infrastructure. Another main problem concerns the interoperability between different, independently developed applications and components, guaranteeing a high quality of service.

Our goal is to identify proper tradeoffs to relieve the programmer as much as possible from the burden of dealing explicitly with low–level events taking place in the distributed system. Our research will be cast in three areas:

– Models of Computation,

– Programming Languages,

– Distributed System Technology.

These three areas are indeed a comprehensive framework for network–aware programming that provides both methods and patterns to develop applications and certify their properties.

Models of Computation

A few foundational models that formally characterize some issues of network–aware programming have been recently developed; however, there is no consensus on a unified formal model of network–aware programming. A clear example is offered by languages for mobile agents: there is no basic model for logical mobility. This makes it difficult to understand all the novel issues involved in the development of network–aware applications including mobile components.

Coordination is a key concept for understanding the model of computation of network–aware applications. Coordinators are the basic mechanism to adapt components to changes of their environment. For instance, coordinators are in charge of supporting and monitoring the execution of dynamically loaded modules. The formal description of coordination languages offers a natural framework for stating, checking and proving behavioral properties of open ended systems.

The project will exploit the idea that coordination primitives provide programmable supports to adapt applications to the network environment. For instance, languages like Ambient, Jada, KLAIM, and Lime support programming of coordination policies. The primitives of the Ambient calculus provide the abstractions to coordinate the interactions among mobile agents and different authorities (or administration domains). Similarly, in KLAIM, Jada and Lime the control of access rights of mobile code is the result of a programmable coordination policy. In these approaches, the behavior (and the properties) of network–aware applications depend on the underlying coordination policy. An important issue addressed in this proposal is therefore to understand the relationships between authorities and coordinators and how their interaction may be programmed.

In this project, we plan to develop mechanisms to specify coordination policies and the related verification infrastructure to certify interesting properties of coordinators. We will propose a layered model, each level corresponding to different degree of homogeneity and interoperability among network components and to tighter or coarser coordination policies. Different properties will be made expressible at different levels, and we will experiment on verification methods and verification infrastructures for them.

The basic building block of the proposed model is given by the notion of naming and the corresponding algebra. The main idea is to identify all the principals (e.g. authorities, software components) involved within a coordination activity by naming schemes. A programmable coordination policy is modeled as a structured operation acting over names. For instance, the coordination policy which redirects bindings to resources of mobile agents roaming the net is naturally described by a structured action over names (bindings) of resources. Similarly, the coordination policy which detects the accessible resources (namely the current execution environment) of incoming mobile agents fits in this framework. We expect that both the programming language guidelines and the design principles guidelines will gain from this semantical understanding.

Interestingly, models of computation are likely to uncover similarities and differences among the physical and logical forms of mobility. These two forms, at a sufficiently high level of abstraction, can both be modeled as components that move across some space. However, at some point, the forms of mobile agents and mobile hosts are bound to diverge. Our research will investigate models of computation that on one hand are able to capture the key aspects related with mobility at large, by providing a unified framework encompassing both forms of mobility, and on the other hand provide support for the characteristics that are specific of each kind of mobility.

Programming Languages

The emerging network–aware programming languages provide mechanisms to support mobility of code and computations, and, at the same time, effective infrastructures to support coordination and control of dynamically loaded software modules.

Many programming languages and systems provide supports for mobility. These supports range from simple mechanisms with primitives for downloading code for execution, to more sophisticated mechanisms which allow the migration of entire computations (code and state) or agents (code, state and authority). In these languages, the issue of dealing with physical mobility is mostly pushed on the underlying network layers and protocols, although the scenarios enabled by wireless computing and ad hoc networking demand for novel abstractions to deal with dynamically and rapidly changing contexts, particularly as far as connectivity is concerned.

A major difference between network–aware and traditional programming languages is the importance given to security issues. For instance, programmable security policies are an essential component of the formers. In this respect, an issue of this project will be to identify ''secure'' programming abstractions which are most suitable for network–aware programming.

We intend to use our expertise for studying the requirements of a new language for the specification and programming of advanced distributed applications. Distributed, network aware systems are built by looking at problems from many different perspectives, including concurrency, memory management, network distribution, load balancing, error checking, profiling, security, which all represent aspects of the systems separated from their functionality. For this reason, we will pay particular attention to programming language techniques developed within the area of aspect–oriented programming.

We also plan to design and implement the corresponding secure abstract machine. In other words, the characterizing feature of our approach is that of programmable security where the security issues are taken into account in the early design stages and not considered later on top of existing unsecured abstract machines and infrastructures.

The semantics foundations play a central role in our developments. In particular, to prove correctness of the security infrastructures. Indeed, ''secure'' programming languages should come equipped with both formal calculi and compositional logics to reason about and to certify security properties. Compilers and development environments will, therefore, be equipped with semantic–based verification infrastructures that programmers may use to prove properties of network–aware applications.

Distributed System Technology

While a number of useful network–aware applications can be developed by sophisticated programmers using standard programming paradigms (e.g. client/server), Internet and World Wide Web have revolutionized the way applications are designed and implemented. Indeed, new programming paradigms (thin client and application servers, collaborative ``peer–to–peer'', code–on–demand, mobile agents) have emerged as more appropriate to develop network–aware applications. However, the current applications are often collections of ad–hoc modules glued together with mechanisms without adequate conceptual foundations.

We will investigate the requirements, designs, implementation, and on–field validation of a new generation of middleware that provides programmers with abstractions for Internet–wide and mobile computing, with particular emphasis on context awareness and communication. To pursue this goal on, we will build on our knowledge in the areas of event–based, tuple–based middleware, and mobile code technologies. The main issue addressed will therefore be the identification and development of software design principles and design patterns for network aware programming.

The design patterns will identify the components (both stationary and mobile), the authorities involved, together with their coordination constraints, i.e. security constraints and network environment constraints. Semantic foundations shall play a central role, not just for proving and certifying correctness of the design principles, but also to provide guidelines for sound development of the design patterns themselves.

The starting point will be the middleware technologies that have been proposed in the last decade to support the development of distributed applications. Middlewares like RPC, MOMs, and more recently DCOM, CORBA, RMI, Jini and Microsoft SOAP, are being successfully used to develop distributed applications. However, several experiences have made evident the difficulties one encounters when using these platforms in scenarios involving WANs and mobile entities, be them physical (eg. handheld computers) or logical ones (eg. software agents). This because the above technologies essentially rely upon the LAN assumption, made of immobile workstations and reliable connections. On the contrary, the new domains calls for mechanisms and primitives that allow to cope with scaling, faults, and context−aware computing. The question of which one is the right set of abstractions to program a wide area, highly reconfigurable network, and how to reconcile the conflicting requirements posed by contemporary and future networks will be one of the main issues of our research.

In fact, since networking in the 90s become ubiquitous, programmers have to deal with network protocols to engineer and deploy their applications. From a programming language viewpoint most network protocols are low level: there is no explicit programming model. This makes life difficult for programmers, who prefer to abstract away from network protocols. Thus, novel programming models and tools to deal with issues like interoperability and QoS are very welcome.

Another major novelty is the fast advent of XML technologies, which we forecast will storm and enhance several aspects of computer science, from data structures to programming languages, from database design to programming environments. Even if we restrict ourselves to programming languages, we see that the actual languages have specific mechanisms to declare and manipulate data structures, which are totally unaware of Web−centric data description languages like XML. We believe that it is increasingly important to study the impact of this new technology on programming languages and design environments. In fact, programming languages like Java are evolving towards encapsulating XML−like structures, and design languages like UML are being redefined using XML−based meta−models. We expect that we can do even better, by studying and developing new UML−oriented languages relaying on XML.

The research group

This proposal is being presented by researchers in four different departments in Italy. All people involved are well known in the national and international communities of language design and software engineering.

Contribution from the University of Bologna

Scenario and past experiences

The advent of the Internet has promoted the Web to be a potentially new, programmable computing model. This model melds two key features: mobility and interaction through semi−structured data. Mobility is due to the fact that there is a continuously changing network of active components. Namely, new components may dynamically come into the scene, old ones may exit and the scope of each component is circumscribed to a finite environment. Interaction indicates that there is a basic common mechanism supporting cooperations among components. In order to fully exploit the new potentialities, Internet programming languages should include concepts such as migration, agenthood, security, coordination. In Internet languages, components should be the basic units, and programs should be built out of them. This commitment towards compositionality can be also used as support for abstraction, ease of program verification, low cost of use.

The introduction by Sun of the Jini technology shows the road for yet another quantum leap in the field of Internet programmable platforms. Jini is an example of "connection bus for info appliances". More precisely, Jini consists of the integration of several technologies which form a software platform to support novel distributed applications. A key technology in Jini is called JavaSpaces. JavaSpaces relies on a well−known concept in the programming languages community: a "coordination model", called "flat multiple tuple spaces", integrated with events, transactions, and a security system.

A widely supported Internet standard is XML, a means for structuring data which has been defined by the WWW−Consortium. XML offers ways to organize, program and edit the information. More relevant, XML is becoming the main way to share information between Internet components.

Our research group has a strong background in the areas of agent coordination, mobile objects, and XML−based software architectures. In the recent past we have developed one of the first reference architectures for Web−based agents, and one of the first architectures for developing XML−based active documents. We also have a strong background on the study and comparison of coordination models.

Proposed research

Nowadays XML documents are passive entities that encode information. In our view this is a very poor use of such a powerful standard. The goal of the Department of Computer Science in Bologna is to make XML into action by design a virtual machine directly interpreting XML documents. This machine should support an Internet programming language as powerful as the current and emerging ones (such us Jini), in a perspective suitable with respect to Microsoft technologies.

To design our XML−based Internet Language we intend to analyze both the theoretical models, such as Ambient Calculus, Coordination Languages, Join Calculus, and the new services offered by the Internet computing models. Indeed the touchstones for our design choices will be of two types. On one end, the simplicity of the model, the presence of an intelligible static semantics, the support for verification tools. On the other end, the abstraction degree and the expressivity of the language primitives. Such primitives must be network−aware and, at the same time, must hide details such as the physical machine where the component runs, its operating system, etc. Following previous research experiences of our group on the coordination languages Linda and Manifold, we plan a deep understanding of the expressive power of every feature of XML−based language.

People involved

Cosimo Laneve, Paolo Ciancarini, Nadia Busi, Fabio Vitali, Gianluigi Zavattaro.

Relevant recent papers

1. F. Arbab, J.W. de Bakker, M.M. Bonsangue, J.J.M.M. Rutten, A. Scutellà, and G. Zavattaro, "A transition system semantics for the control−driven coordination language MANIFOLD", Theoretical Computer Science, 240(1):3−47, Elsevier, 2000.
2. N. Busi and G. Zavattaro "On the Expressiveness of Event Notification in Data−driven Coordination Languages" In Proc. of ESOP2000, LNCS 1782, Springer−Verlag, 2000.
3. P.Ciancarini and D.Rossi. Jada: Coordination and Communication for Java agents. In J.Vitek and C.Tschudin, editors, *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes in Computer Science*, pages 213−−228. Springer−Verlag, Berlin, 1997.
4. P.Ciancarini, R.Tolksdorf, F.Vitali, D.Rossi, and A.Knoche. Coordinating Multiagent Applications on the WWW: a Reference Architecture. *IEEE Transactions on Software Engineering*, 24(5):362−−375, 1998.
5. P.Ciancarini, F.Vitali, and C.Mascolo. Managing complex documents over the WWW: a case study for XML. *IEEE Transactions on Knowledge and Data Engineering*, 11(4), July/August 1999.
6. C. Fournet, C. Laneve, L. Maranget, D. Remy: "Inheritance in the join−calculus", March 2000. Available at http://www.cs.unibo.it/~laneve/papers/UBCLSinhjoin.ps
7. G. Bigliardi, C. Laneve: "A Type System for JVM Threads". Technical Report UBCLS 2000−06, June 2000. Available at http://www.cs.unibo.it/~laneve/papers/UBCLSjvmc.ps

Contribution from the University of Firenze

Scenario and past experiences

A major effort in in the last two years of the group in Firenze has been the design and the implementation of a language for the development and the proof of properties of concurrent mobile systems. This language as been named KLAIM, namely a Kernel Language for Agents Interaction and Mobility. The extension of KLAIM, the tuning of its semantics and of its security primitives, together with the actual implementation and development of case studies (examples from the fields of electronic commerce and load balancing of distributed CPU's will be used to assess our modelling and ourlinguistic choices) will be the main objective of the Firenze group.

Proposed research

As it is, KLAIM permits describing essentially "flat" computer networks a la Ethernet; the language will be generalized to permit descriptions of nets hierarchies (that more faithfully represent architectures a‘ la Internet). We plan, moreover to enrich the language for supporting Quality of Service (QoS) management in the WWW environment thus allowing to specify, at the linguistic level, management policies for the various aspects of the QoS, including security, response time, communication bandwidth, reliability.

As for the security properties, the actual type mechanism used in Klaim to guarantee that the access rights fixed by the coordinator are not violated is a solution that may turn out to be too restrictive because it imposes one to fix statically and independently for all processes the access rights of processes to systems resources. We plan to relax these constraints and guarantee greater dinamicity, possibly permitting communication of capabilities together with data. We shall also consider the possibility of structuring localities at different security levels, the correspondent structure of the type system should then enable use to control non−interference properties between security levels.

Starting from the type system used in KLAIM for specifying and enforcing relative access rights of processes over a distributed store we shall aim at defining type system for controlling access and interaction within pi−calculus/ambient−calculus, which do not possess yet satisfactory verification and analysis techniques as well as typing systems. This calculi will be generalized by endowing processes with capabilities (namely, operations to be performed by them at specific locations) that need to be controlled in order to ensure data integrity by granting access to resources and data only to agents that have the right to do so. We shall also develop software tools that allow automatic type inference and type checking.

We shall also study observational semantics based on equivalences or preorders. More abstract semantics are necessary to support modular programming and program verification. We shall define behavioural equivalences, temporal logics and equational systems that are a necessary support for the statement and the verification of systems properties.

People involved

Rocco DeNicola, R.Pugliese, B.Venneri.

Relevant recent papers

1. R. De Nicola, G. Ferrari, R. Pugliese. Klaim: a Kernel Language for Agents Interaction and Mobility. IEEE Transactions on Software Engineering, 24(5):315−330, 1998.

2. L. Bettini, R. De Nicola, G. Ferrari, R. Pugliese. Interactive Mobile Agents in X−Klaim. Proc. of the 7th IEEE Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'98), pp.110−115, IEEE Computer Society, 1998.

3. R. De Nicola, G. Ferrari, R. Pugliese, B. Venneri. Types for Access Control. Theoretical Computer Science special issue on Coordination (D. Le Metayer, Ed.), 2000. (to appear)

4. R. De Nicola, G.−L. Ferrari, R. Pugliese. Types as Specifications of Access Policies. In Secure Internet Programming: Security Issues for Distributed and Mobile Objects (J. Vitek, C. Jensen, Eds.), LNCS 1603, pp.117−146, Springer, 1999.

5. R. De Nicola, G. Ferrari, R. Pugliese. Programming Access Control: The Klaim Experience. In Proc. of the 11th International Conference on Concurrency Theory (CONCUR'00), Lectures Notes in Computer Science, Springer, 2000. (to appear)

6. R. De Nicola, R. Pugliese. Linda based Applicative and Imperative Process Algebras. Theoretical Computer Science, 238, Elsevier Science, 2000, pp. 389−437

7. R. De Nicola, R. Pugliese, A. Rowstron. Proving the correctness of optimising destructive and non−destructive reads over tuple spaces In Proc. of the 4th Int. Conf. on Coordination Models and Languages (COORDINATION'00), LNCS, Springer, 2000. (to appear).

Contribution from the University of Pisa

Scenario and past experiences

Different foundational models that formally characterize some issues of Web programming have been recently developed; however no unified model that formalizes the model of computation of Web programming exists. This makes it difficult to understand all the novel issues involved in the development of Web applications as described in the previous section.

Web applications distinguish themselves from traditional distributed applications also for their relationship with the data they access and manage. Traditional applications are designed together with the data they manipulate, and evolve together; hence, they can rely on strong assumptions about the structure of data. New network applications have to be able to operate on data which has been created by different applications, and whose structure evolves in a way which may be difficult to predict. These applications need the ability to query both the data and its structure, and to tolerate some deviations between the expected and the actual structure of data. More generally, they have to be able to discover both the availability and the structure of the data which is accessible from their current environment. Some languages (eg. XMLQL) have been designed to query and analyze semi−structured data, defined as self−describing data whose structure is not fully known by the applications that operate on them. These languages are characterized by the ability of computing over the structural information extracted from the data, and by the ability of navigating through the data with just a limited knowledge of its structure. However, many of them lack of a precise semantics definition, and none is associated with a type system, that would allow to write safer and more efficient applications.

We emphasize the fact that coordination is a key concept to design, implement and certify Web applications. The idea is that coordination primitives provide programmable supports to adapt Web applications to the network environment. The KLAIM language (developed in cooperation with the group in Firenze) and the Ambient calculus both support programming of coordination policies. For instance, in KLAIM checking access rights of mobile code is the result of a programmable coordination policy. An important issue is to understand the relationship between authorities and coordinators and how their interaction may be programmed (layers of coordinators, how to control authorities).

Our research group has a strong background in the areas of concurrency, mobility, and coordination. Concerning models of computations, a general and flexible model of computation, called tile logic was developed and applied to a variety of application fields. Tiles are inference rules (reminiscent of SOS rules) defining the behavior of open configurations, i.e. system components, which may interact and synchronize through their interfaces. Tiles have been studied as a foundation for open, distributed and interactive systems. Especially interesting are applications to software architectures, where the flexibility and compositionality aspects of the tile model can be valuable.

Proposed research

In this project we propose to develop a framework which will provide the semantic foundations and the abstractions to describe and define programmable coordination policies. We plan to develop mechanisms to formal specify programmable coordination policies and semantic−based verification tools to certify properties of coordinators. We expect that the design of both languages for Web programming and Web software architecture patterns will gain from this semantic understanding.

We will propose several levels for our framework, which will correspond to different levels of homogeneity and interoperability among network components and to tighter or coarser coordination policies. Existing calculi and languages for Web programming will be mapped into our framework in order to compare, merge and generalize them. Different properties relevant for Web programming will be made expressible at different levels, and we will experiment on verification methods and tools for them.

Another major issue concerns the problem of data access. This can be faced by assuming that information shared by different applications is represented in an application independent and self−describing format, such as XML. In this context, the language needs abstractions to query this kind of data, with the possibility to represent and exploit all its knowledge about the data structure, and of acquiring more knowledge by inspecting the data itself. These abstractions have to be amenable to optimization and efficient implementation in presence of large amounts of data. We plan to reach these goals by exploiting three strategies. We will design a language which uses Cardelli's ambient logic to analyze data and its structure. Ambients can be used to represented semi−structured information, or XML data; the logic, enriched with ambient variables, can be used to describe the data one wants to match and to extract the relevant information. The flexibility of the ambient logic allows one to describe either a vague or a precise knowledge of the structure of data. We will define a type system to describe assumptions about the structure of data, and will use this type information both to increase the efficiency of data access and the reliability of applications. We will define an algebra to compile the language into a form which is amenable to cost−based optimization.

People involved

Gianluigi Ferrari, **Ugo Montanari**, Giorgio Ghelli, F. Gadducci.

Relevant recent papers

1. Bruni, R., Meseguer, J. and Montanari, U. Executable Tile Specifications for Process Calculi. In: Jean−Pierre Finance, Ed., FASE'99, Springer LNCS 1577, pp. 60−76.
2. L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for Mobile Ambients. In Proc. of the 26th International Colloquium on Automata, Languages, and Programming (ICALP), Prague, Czech Republic, July 1999.
3. L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In Proc. of IFIP International Conference on Theoretical Computer Science (IFIP TCS), Sendai, Japan, August 2000.
4. G.Ferrari, S.Gnesi, U.Montanari, M.Pistore, G.Ristori, Verifying Mobile Processes in the HAL Environment. Computer Aided Verification (CAV'98), LNCS, 1998.
5. G.Ferrari, U.Montanari, Tile Formats for Located and Mobile Systems, Information and Computation, to appear, 2000.
6. F. Gadducci, U. Montanari, The Tile Model. In: G. Plotkin, C. Stirling and M. Tofte, Eds., Proofs, Languages and Interaction: Essays in Honour of Robin Milner, MIT Press, to appear.
7. Hirsch, D., Inverardi, P. and Montanari, U. Modeling Software Architectures and Styles with Graph Grammars and Constraint Solving. Proc. Working IFIP Conference on Software Architecture, February 1999.

Contribution from the Politecnico di Milano

Scenario and past experience

People at Politecnico di Milano (PoliMi) have a strong background in the areas of mobile code technology, middleware, and programming languages.

Mobile code technologies: the research carried on this subject started with a classification of mobile code concepts with respect to technology, architectures, and applications. Then, it delved in research issues spanning from theoretical models for code mobility to the implementation of mobile code systems.

Middleware. Our most recent results in the area of distributed architectures and middleware technology have been the development of an event–based middleware called JEDI and of a tuple–based middleware called Lime. The first one provides support for scalable event delivery to parties which are possibly mobile agents, while the second is especially conceived for ad hoc networking. Both involve the use of, and provide support to, mobile code.

Programming languages. At PoliMi, research activity in the area of programming languages has a long tradition. Today, PoliMi people are investigating a new research area called aspect–oriented programming languages. Current efforts are targeted to assessing the expressivity and usability of aspect–oriented constructs in network–aware and concurrent applications.

JEDI is a Java Event–based Distributed Infrastructure that can be used to build distributed applications whose components communicate through the event–based paradigm (also known as the publish/subscribe paradigm). The components of an event–based application cooperate by sending and receiving events, a particular form of messages. The sender delivers (i.e., publish) an event to an event dispatcher. The event dispatcher is in charge of distributing the event to all the components that have declared their interest in receiving it (i.e., to the components that subscribed to the event). Thus, the event dispatcher supports a high degree of decoupling between the sources and the recipients of an event.

µCode is a lightweight and flexible mobile code toolkit, fully implemented in Java. The main goal of µCode is provide a minimal set of primitives that cover a broad set of relocation strategies for code and state. Mobile agents are provided as an example of the power of the primitives provided by µCode. Building a mobile agent on top of µCode takes about a dozen lines, and is just a matter of storing the agent object and desired classes into a group. µCode provides also methods for determining (and trimming) the full class closure of a given object, i.e., the whole set of classes and interfaces (e.g., belonging to the inheritance hierarchy or used by the object,s methods) that are needed by an object. µCode has been used successfully in many projects and teaching experiences, some of which successfully exploited it in combination with JEDIwithout any need for modifications in either system.

Lime is a middleware conceived to enable coordination and communication among mobile components, that can be indifferently logical (e.g., mobile agents) or physical (e.g., mobile hosts with wireless connectivity). Lime borrows and adapts the coordination model made popular by Linda. In Linda, processes communicate through a shared tuple space that acts as a repository of elementary data structures–––the tuples. The model underlying Lime introduces a shift from the fixed context of conventional distributed computing, as assumed by Linda, to the dynamically changing context fostered by mobility. This accomplished by breaking up the Linda tuple space in many tuple spaces, each permanently associated to a mobile unit, and by introducing rules for transient sharing of the individual tuple spaces based on connectivity.

Proposed research

The main focus of the activities Politecnico di Milano will be the realization of a middleware for mobility. The emphasis will be on programming abstractions that allow a suitable representation of and access to the dynamically changing context determined by mobility. Under this perspective, we will explore adaptations of models based on event dispatching and on Linda. The research strategy we will pursue will encompass a tight coupling between the modeling the formal semantics of our primitives, implementing them, and validating their effectiveness on the field.

Ideally, our middleware will integrate both forms of mobility, i.e., logical and physical. However, we expect that most of the technical challenges will come from physical mobility, in particular from coping with unannounced disconnection, and from implementing our solutions on resource constrained devices.

Besides the slant towards system research, we expect this study to provide also significant conceptual insights. As we model formally our middleware we will likely uncover similarities and differences between the two forms of mobility. Furthermore, another perspective we plan to explore is the applicability of the abstractions conceived in the mobile environment back to the fixed one.

People involved

Carlo Ghezzi, GianPietro Picco, G.Cugola, A. Fuggetta, M.Monga.

Relevant recent papers

1. G. Cugola, C. Ghezzi, and M. Monga, "Language Support for Evolvable Software: An Initial Assessment of Aspect−Oriented Programming," in Proceedings of International Workshop on the Principles of Software Evolution, (Fukuoka, Japan), July 1999.
2. Fuggetta, G.P. Picco, and G. Vigna, "Understanding Code Mobility," IEEE Transactions on Software Engineering, vol. 24, pp. 342{361, May 1998.
3. Mascolo, G. P. Picco, and G.−C. Roman, "A Fine−Grained Model for Code Mobility," in Proceedings ESEC/FSE ’99, LNCS, Springer, September 1999.
4. G.P.Picco, "_Code: A Lightweight and Flexible Mobile Code Toolkit," in Proc. 2nd Int. Workshop on Mobile Agents (K. Rothermel and F. Hohl, eds.), LNCS 1477,1998.
5. G.Cugola, E.Di Nitto, and A. Fuggetta, "Exploiting an event−based infrastructure to develop complex distributed systems," in Proceedings of the 20th International Conference on Software Engineering (ICSE98), (Kyoto, Japan), April 1998.
6. Cugola, E. Di Nitto, and A. Fuggetta, "The JEDI event−based infrastructure and its application to the development of the OPSS WFMS," IEEE Trans. on Software Engineering. To appear.
7. G.P. Picco, A. L. Murphy, and G.−C. Roman, "Lime: Linda Meets Mobility" in Proceedings of the 21st International Conference on Software Engineering (ICSE99), May 1999.

Papers are available at http://www.elet.polimi.it/~picco and http://www.elet.polimi.it/~cugola.