INFORMATION SOCIETIES TECHNOLOGY (IST) PROGRAMME

Contract for:

Shared-cost RTD

Annex 1 - "Description of Work"

Project acronym: PEPITO
Project full title: PEer-to-Peer Implementation and TheOry
Contract no.: IST-2001-33234
Related to other Contract no.:

Date of preparation of Annex 1: September 13, 2001

Operative commencement date of contract:

Contents

1 Project summary

Project Acronym	PEPITO	Proposal No.	IST-2001-33234

A2.	**Project Summary**

Objectives (maximum 1000 characters)

Peer-to-peer computing (P2P) is a paradigm in which applications are connected to a shared network as peers, i.e., with the same capabilities and responsibilities. Current P2P applications are limited to information exchange. The objectives are to remove this limitation by: developing formal models to understand P2P computing, developing the distributed algorithms required for implementation, implementing a language-independent set of basic services, implementing languages and devising programming techniques and convincing demonstrator applications. Further objectives are: better using resources at the network's edge, scaling better than server-centric computing, allowing device mobility (independence of IP address), allowing individuals to publish information and services, and to allow individuals to collaborate while remaining anonymous.

Description of the work (maximum 2000 characters)

PEPITO will assume a completely decentralized architecture in which a peer can have four simultaneous roles: it may use services, provide services, forward requests, and provide caching of information. We also assume that peer nodes connect through a virtual network that is dynamic and intermittent, and that nodes do not possess a fixed IP address. To successfully deal with the complexity of P2P systems (in which failure, reconfiguration and security are central) it is important to pursue use-model analysis, theoretical work and prototyping in a closely-linked style. The complementary expertise of the PEPITO partners will make this possible: the objectives will be addressed, but the enabling of interaction between them is crucial. Use-model analysis of this type of system will investigate how they are perceived by users, and what new applications are possible. Theoretical work will study the foundational concepts of P2P systems. This includes mathematical models (calculi, proof techniques, and resource models) and new distributed algorithms (decentralized algorithms for diffusing information, and for coping with multi-consistent computing – with simultaneous inconsistent views of entities). System Design and Prototyping will develop prototypes of programming languages and programming platforms (middleware) suitable for peer-to-peer computing (such platforms are lacking today; those existing are server-centric). One aspect will be a scalable and robust name/directory service based on our algorithms. Together, all these will enable development of applications that: handle dynamic connectivity and device mobility; allow individuals to become publishers of information and services; permit full use of existing network resources at the edge of the network; and permit applications to scale better than server-centric designs.

Milestones and expected results (maximum 500 characters)

- Develop a formal model for peer-to-peer computing.
- Develop novel distributed algorithms for peer-to-peer computing.
- Develop a scalable and robust name/directory service based on these algorithms.
- Provide a language independent peer-to-peer distribution subsystem.
- Provide languages and platforms for peer-to-peer computing.
- Provide convincing demonstrators for peer-to-peer applications.

2 Project objectives

Peer-to-peer computing is a new paradigm in which single-user applications are connected to a shared network as *peers*, i.e., with potentially the same capabilities and responsibilities. When considered together, a group of collaborating peers forms one, much larger application, which we call a *peer-to-peer* application. There is a small but rapidly growing class of applications that can be classified as peer-to-peer. Up to now, these applications have been primarily intended for brokering and exchanging resources between peers. Some, like ICQ, Napster, and SETI@home, still use centralized coordination to mediate services. Others, like FreeNet and Gnutella, no longer require central control. Because of the continued exponential growth of the Internet, we postulate that future peer-to-peer applications will become much more important and will cover a much wider spectrum of user needs.

The PEPITO project will investigate completely decentralized models of user-centric peer-to-peer computing. By this we mean that users and application nodes connect and disconnect dynamically and are present in potentially unlimited numbers (thousands to millions) that can change on very short time scales (seconds to minutes). A major advantage of peer-to-peer computing is that it can tap the abundant resources of user devices (including PCs) at the edge of the network. We assume that a peer can have four simultaneous roles: it may use services (be a client), provide services (be a server), forward requests (be a router), and provide caching of information (be a cache). We also assume that peer nodes connect through a virtual network that is dynamic (may change topology) and intermittent (may have temporary failures), and that nodes do not possess a fixed IP address. Given this basic context, PEPITO will be organized according to the following cyclic structure:

- Study the *use models* of this type of system, i.e., how it is perceived by users and what new applications are possible. Clearly there are other possible applications than mere resource sharing, in particular applications where collaboration is important.
- Study the foundational concepts of this type of system. This includes formal foundations (calculi, proof techniques), new distributed algorithms, and coping with multi-consistent views of information. In particular, we see the following two new basic issues that appear:
  - Multi-consistent computing: allowing simultaneous inconsistent views of entities.
  - Diffusion-based computing: distributed computing that is completely decentralized, i.e., it has no centralized control point whatsoever.

  As we explain later, these two styles of computing are the most natural for large peer-to-peer applications: they have highest performance, greatest robustness, and are the most scalable.
- Design prototypes of programming platforms (middleware) suitable for peer-to-peer computing. Such platforms are lacking today, e.g., CORBA and .Net services are all server-centric and depend on fixed IP addresses [43]. Even SOAP, a recent protocol whose goal is to allow RMI with XML, is server-centric.
- Demonstrate new classes of applications in this framework and repeat the cycle.

Figure 1 illustrates this structure and how it relates to the project workpackages. From the definition of peer-to-peer computing given above, we see that it is a promising approach for at least five technical reasons:

- **Network resources**. It will permit full use of existing network resources. The bulk of computational and other resources exist at the edge of the network, i.e., with individual users.

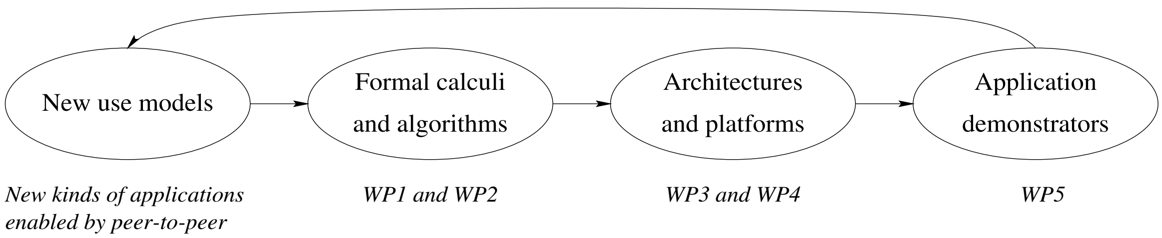


Figure 1: Structure of the PEPITO project

- **Scaling**. It will permit applications that scale better than server-centric techniques such as "server farms". Currently, the responsibility of handling growing numbers of users is placed on the provider of a service, who has to install ever more costly server farms. Letting the users be peers will allow them to share the load.
- **Device mobility**. Because peer-to-peer by its nature handles dynamic connectivity, it will permit device mobility, e.g., PCs that change their connection points and IP addresses.
- **Individual publishers**. It will allow individuals to become publishers of information and services. Currently, an individual cannot directly publish either. He or she must go through an intermediary, e.g., a service provider.
- **Anonymous networks**. It will allow creating any number of anonymous networks, i.e., virtual private networks without centralized control. This also allows individuals to remain anonymous, while still publishing information and collaborating with others.

We propose the following four measurable and verifiable objectives to judge the success of PEPITO:

- The project will develop a formal model for peer-to-peer computing that is accepted by the research community and applied to peer-to-peer applications.
- The project will develop novel distributed algorithms for peer-to-peer computing. As one concrete criterion to judge their quality and relevance, we propose to realize a scalable and robust name/directory service based on these algorithms.
- The project will provide a language-independent distribution subsystem tailored to peer-to-peer computing (e.g., it provides IP-independent addressing and mobility of services).
- The project will provide languages and platforms for peer-to-peer computing and show that they are useful by implementing convincing demonstrator applications.

It is our view that peer-to-peer computing will have a large commercial impact in the near future. It is therefore important to develop competence in Europe. This competence cannot be developed by individuals, nor even by one research group. The subject matter is simply too vast. It requires expertise in many complementary areas (including concurrency theory, distributed algorithms, programming languages, type theory, fault tolerance, security, and implementation). We believe that the partners of PEPITO provide this complementary expertise and that the project is the right size to make a significant impact. The project will educate many people to peer-to-peer computing, including researchers, students, as well as commercial developers. We intend to release the project software under an Open Source license to permit third-party users to take advantage of it.

3 Participant list

List of participants

Partic. role	Partic. no.	Participant name	Participant short name	Country	Date enter project	Date exit project
C-F	1	Swedish Institute of Computer Science	SICS	Sweden	Start of project	End of project
P	2	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland	Start of project	End of project
P	3	Institut National de Recherche en Informatique et Automatique	INRIA	France	Start of project	End of project
C-S	4	Kungliga Tekniska Högskolan	KTH	Sweden	Start of project	End of project
P	5	The Chancellor, Masters and Scholars of the University of Cambridge	UCAM	United Kingdom	Start of project	End of project
P	6	Université catholique de Louvain	UCL	Belgium	Start of project	End of project

4 Contribution to programme/key action objectives

We examine peer-to-peer computing and the PEPITO project from the viewpoint of four major characteristics of the Global Computing Initiative, giving the relevant workpackages:

- *The systems are composed of autonomous computational entities where activity is not centrally controlled.* In a peer-to-peer application, the peers are autonomous computational entities. Their activity is normally decentralized, as peers represent different owners. Occasionally in a group of peers one participant may act as coordinator of some activity, but this is dynamic and temporary. Others must be prepared to take over on coordinator failure. WP1 and WP2 attack these issues.

- *The computational entities are mobile.* We attack mobility in two respects. First, nodes in peer-to-peer applications will not assume any fixed IP address. Therefore, nodes can move from one part of the network to another without changing the logic of the applications. Second, computations hosted by individual nodes will have the ability to move to other hosts. WP2 and WP3 attack these issues.

- *The configuration varies over time.* Peer nodes are connected by a virtual network, i.e., a network that is built as a software layer on top of the real network. The virtual network is not based on IP addresses but on logical names and uses diffusion-based routing and resource discovery. WP2 attacks these issues.

- *The systems operate with incomplete information about the environment.* Information cached in peer nodes might be out-of-date. Therefore an important question is how applications can operate under this assumption. The study of probabilistic algorithms and multi-consistency in WP2 addresses this question.

We examine the focus of the research from the point of view of the four stipulated issues that the global computing initiative emphasizes:

- *Design of such systems.* WP1 and WP2 will design the formal models and algorithms underlying such systems. WP3 and WP4 will design and implement peer-to-peer programming platforms using these models and algorithms.

- *Analyzing and reasoning about their behavior.* WP1 develops formal models to analyze and reason about behavior of peer-to-peer applications.

- *Avoiding and/or detecting undesirable behavior through control of the system and/or its environment.* Failure in a peer-to-peer context is considered in many of the workpackages. One of the key features is the greater likelihood of transient failures. The algorithms of WP2 have fault-tolerant requirements. This is leveraged into the platforms in WP3 and into the distribution subsystem in WP4. There are also specific tasks dealing with transactions in WP1. The work on resource control in WP3 is concerned with avoiding runaway resource consumption by mobile components. This is important for the class of peer-to-peer applications where resource brokerage is central.

- *Understanding their limits and potential.* The models in WP1 and the distributed algorithms in WP2 will be studied both theoretically and experimentally, showing under what conditions they apply. In addition, the demonstrator applications of WP5 will provide direct experience of the potentials and limitations of peer-to-peer computing as viewed by PEPITO.

5 Innovation

We list the innovative aspects of PEPITO and compare each with the state-of-the-art. We give the workpackages that focus on each aspect.

5.1 Multi-consistent computing (WP2)

There is a spectrum of computation styles, from completely consistent (the traditional approach) to "multi-consistent", i.e., allowing simultaneous inconsistent views of entities. Multi-consistent computation is the "natural" style for a large peer-to-peer application: it has highest performance, robustness, and scalability. We briefly review the state of art in consistency models and point out the areas where PEPITO is expected to innovate.

Consistency models In distributed and parallel computing systems, a wide variety of consistency models have been defined: sequential consistency, release consistency, causal consistency, and other forms of weak consistency [17, 3, 4]. For brevity, the rest of this section refers to this work as "DSM (Distributed Shared Memory) computing". The goal of DSM computing is to allow to do exact computation as much as possible, while relaxing the synchronization constraints between processors to improve performance, scalability, and fault tolerance. For example, release consistency maintains coherence only between synchronization objects, not between all data structures like sequential consistency does. If the application is carefully written, then this is enough to guarantee that it does an exact computation, while improving performance with respect to a system respecting sequential consistency.

Some work is being done to use these consistency models for peer-to-peer computing. For example, Reiher *et al* shows how to do file sharing in a peer-to-peer fashion [47]. This work assumes that eventually a consistent view is achieved (e.g., at a synchronization point). In the general form of multi-consistent computation, this assumption no longer holds. For example, in a distributed virtual environment, the inconsistency does not eventually go away. The degree of inconsistency depends on the logical distance (spatial or temporal) and is always present.

New issues raised by peer-to-peer computing PEPITO will investigate how to use different consistency models together in a single application, where the inconsistency may be visible and stay visible at the application level. We will evaluate the feasibility of known consistency models (e.g., causal consistency) under the new constraints posed by peer-to-peer computing. We see the main differences between peer-to-peer and DSM computing as follows:

- The main issue in peer-to-peer computing is collaboration, in DSM computing it is performance.
- The possible failure modes are much richer in peer-to-peer computing; in particular partial failure is much more common.
- The network behavior (bandwidth and latency) is more variable in peer-to-peer computing.
- Application nodes come and go more frequently in peer-to-peer computing.
- Permanent failure detection is possible for DSM computing, but mostly impossible for peer-to-peer computing (except for special cases such as peers on the same LAN).
- The number of nodes can be much larger for peer-to-peer computing.

Because of these differences, and because the basic assumption of eventual consistency no longer holds, we cannot assume that the scientific results of DSM computing will continue to hold in the area of peer-to-peer computing. Therefore it is important to investigate multi-consistent computing as a topic by itself. We will review the area of DSM computing and apply its results as much as possible, but we will develop new science whenever the assumptions of DSM computing do not apply.

5.2 Diffusion-based computing (WP2)

A distributed system can be characterized by its degree of centralization. It can range from completely centralized (the traditional, server-centric approach) to completely decentralized (all participants are equal). Completely decentralized is the "natural" style for a large peer-to-peer application, since global coordination is expensive. This style uses algorithms based on diffusion of information and peer caching. "Standard" distributed algorithms that presuppose a fixed number of processes are no longer expected to hold in the context of an unbounded and dynamic number of processes. All failures are partial and potentially temporary. We briefly review the state of art in diffusion-based computing and point out the areas where PEPITO is expected to innovate.

Diffusion algorithms PEPITO will focus on a class of distributed algorithms called *diffusion algorithms* that are based on a fully decentralized structure in which the only communication allowed is from a node to its logical neighbors. Each node has a copy of the algorithm. The algorithm coordinates the nodes to perform a single task. These algorithms have been used for many tasks:

- Routing packets on an IP network. This is done by a diffusion algorithm called the distance vector algorithm [26]. Periodically, each routing node sends the contents of its routing table to its neighbors. It receives its neighbors' tables and uses them to update its own table.
- Gathering a coherent snapshot of global state based only on local information [33].
- Searching for a file in a global file system with no servers. Information is cached with a diffusion algorithm.

These algorithms are scalable, fault-tolerant, and dynamic. Our research in this area will revisit standard algorithms (membership, information search & retrieval, caching, replication, ...) and programming techniques (e.g., transactions), extending them for the requirements of peer-to-peer computing. For example, one challenging task in peer-to-peer computing is to develop diffusion algorithms for global directory services with high availability in the context of dynamic connectivity [52].

"Small World" routing To achieve efficiency in peer-to-peer computing, peer nodes have to configure themselves into a network graph with desirable properties. For example, some relevant topologies are given by the so-called "Small World" models [61]. These topologies guarantee properties regarding network diameter, degree of clustering, etc. This is needed to specify the performance and failure behavior of peer-to-peer routing. PEPITO will develop a generic framework for diffusion algorithms that dynamically maintain these properties in the peer-to-peer context. We will revisit existing algorithms to identify their limitations in the context of a dynamic environment.

5.3 Foundational model: P2P-calculus (WP1)

PEPITO will identify key issues of peer-to-peer computation and collaboration by distilling them in a formal process calculus. Key issues to be expressed in this calculus may include: multicast communication, replicated state, consistency models, distributed transactions, transient failures, mobility and dynamic network topology, and probabilistic reasoning. At present, not enough is known about the issues to enable us to distinguish fundamental from derived concepts and to define a concrete calculus at the outset. Instead, we will design and refine the calculus as the project progresses, based on the experiences and results we gain in it. For this we will draw on previous experience with calculi such as Join [21, 22], Functional Nets [41], Nomadic Pict [62], and Ambients [16]. The P2P-calculus will allow reasoning about peer-to-peer applications at a high level. It will likely be on a higher level than these calculi of mobile processes [35], but it will be connected to the lower level calculi via translations.

5.4 Innovations in languages and systems (WP3,WP4)

The implementation goals for the project are twofold: first, to build a distribution subsystem to provide the basic functionality needed for peer-to-peer computing and second, to design and implement different programming languages on top of it. The distribution subsystem will support both multi-consistent computing and diffusion-based computing. We summarize these two areas and explain where PEPITO is expected to innovate.

5.4.1 Distribution subsystem (WP4)

The distribution subsystem provides the language-independent functionality needed to implement peer-to-peer applications. This includes the following:

- **Basic services and protocols**. Reliable message sending, failure detection and network partitioning detection, and dynamic connectivity protocols.
- **Consistency protocols**. These must permit the simultaneous existence of inconsistent state.
- **Distributed garbage collection**. This should work on failure-prone WANs without needing global synchronization and without noticeably impacting network performance.
- **Capabilities and security**. The distribution subsystem must be hardened against malicious attack. This requires unforgeable references, encrypted communication at the protocol level, and security awareness (avoiding security overhead within a trust boundary).

No existing product or research prototype exists that provides these services. We summarize the state-of-the-art in support for peer-to-peer computing. There are three approaches:

- **Message-passing approach**. Messages are sent directly between peers, using protocols such as TCP/IP or UDP/IP. Higher-level protocols include HTTP and XML. An example of this is the Java Messaging Service. This approach means that each application needs to reimplement much of the functionality of the distribution subsystem.
- **Middleware approach**. A software layer is placed between the application and the operating system. Middleware platforms include Java with RMI [54], .Net, ObjectSpace Voyager, Mozart [36], and many others. Characteristic for these platforms is that they merge to some degree the language constructs with distributed algorithms. Java RMI is a simple example: it supports distributed synchronous stationary objects. Voyager and Mozart provide for asynchronous method invocations. Mozart provides for cached objects. None of these platforms support multi-consistent computing or diffusion-based computing.

- **Special-purpose platform approach**. This includes platforms designed for a particular purpose such as distributed interactive simulation, which provides a very specialised object model, heavily influenced by requirements on scalability and real-time interactions (such as DIVE [23]) or requires the programmer to explicitly handle aspects of the object distribution, for instance the "broadcast" or handling of update notifications after a modification of a replicated object (such as NetZ [46], HLA [19], TerraPlay [55]). There is typically some support for multi-consistent computing, tailored to the application.

WP4 will develop a distribution subsystem using the middleware approach.

5.4.2 Languages (WP3)

It is important that the underlying peer-to-peer support be made available to the language level in the proper way. PEPITO will explore two approaches that attack this problem in complementary ways. In this way, we expect to discover innovative language constructs and programming techniques that facilitate building peer-to-peer applications. The two approaches are as follows:

- *Integration approach.* We will extend the dynamically-typed concurrent functional/object language Oz with peer-to-peer abilities in a fundamental way. Oz already addresses distribution in a fundamental way: each primitive language operation is extended with a distributed semantics. The result is an efficient distributed implementation that has been used in industrial projects and deployed applications. In PEPITO we will apply this approach to peer-to-peer computing.

- *Library approach.* We will build extensions and/or libraries for Java, using the DACE library as a starting point. Java has basic support for distribution and dependable computing. In PEPITO we will see how this support can be extended for peer-to-peer.

Both approaches will use the language-independent distribution subsystem of WP4. For each approach, we will add language support for peer-to-peer computing and we will explore the programming techniques that it implies. We will then critically compare the two approaches, showing the advantages and disadvantages of each.

5.5 Applications (WP5)

In this workpackage we will develop two small applications that show the power of our approach. The applications will be chosen based on two criteria: for their intrinsic interest and for their interest in exploiting the techniques of diffusion-based computing and multi-consistent computing.

5.6 High availability (WP1,WP2)

Traditionally, recovery in highly available systems has been coordinated by all surviving processors working together. For example, in clusters all surviving processors use an agreement algorithm to decide that a particular processor has failed. In peer-to-peer computing this approach is no longer feasible. PEPITO will develop approaches to high availability based on diffusion algorithms and caching.

6 Community added value and contribution to EU policies

EU has formulated eight important principles shaping the EU policy for the information society. Out of these eight, this proposal directly addresses four. These are the following:

1. Market forces must drive progress to the information society. This means opening up to competition information services and infrastructure. This focus of this proposal is to develop foundational techniques for non-centralized cooperation. The intent is to provide critical components, that in turn spur a rapid development of more and more advanced and creative components. The focus on heterogeneity and the support for gradual development of interoperability makes is relatively easy to become a service provider, giving equal opportunities for large companies and single entrepeneurs to supply proactive and creative novel services. As part of this objective, we will aim to provide all results from the project as Open Source Software.

2. Universal service must be ensured together with network interconnection and the interoperability of services and applications throughout the Union. Similar measures are needed in other regions of the world which also guarantee equal access. The emphasis on market drive could counteract universal interoperability by encouraging centralized solutions. Concrete examples of this have been seen, when large companies deliberately avoid current or emerging standards, aiming to make their own model a de facto standard. In this proposal we avoid this risk by making the de facto standard the norm. Instead of integrating large-scale standardisation efforts, we allow for competing standards to live in parallel in the same environment. An advantage of the P2P approach to the situation of today is that in P2P nets there is no immediate advantage for one large company supplying an integrated set of services. Rather, users may prefer to use integrated services from small suppliers, since such services could more readily adapt to changes, detect opportunities for new services and supply solutions, etc.

3. The job of financing the information society lies chiefly with the private sector. The support for services developed within this project will serve to make both market push and market demand much more readily inspectable. This will in turn make potentials for profitable development very clear to investors. The effect will be that the rapid development we hope for, will almost entirely be financed by the private sector. The process will be similar of that of the WWW of today: a relatively small initial investment is followed by massive investment from a large number of private investors.

4. Cultural and linguistic diversity should be protected and promoted. Heterogeneity and diversity is a fundamental assumption of our proposed solution. This includes cultural and linguistic diversity. The requirement on cultural diversity could potentially be counteracted by the demands on market drive and financing from the private sector. One of the purposes of this proposal is however to leverage between the roles of a service producer and a consumer. Essentially, it should be possible for anyone that has something to offer, to set up a service. Similarly, a community of users can collaborate in forming interactions between services that share a common language or a common cultural perspective. This way, cultural and language diversity can still be protected and promoted.

7 Contribution to Community social objectives

This project has some indirect impact on Community social objectives, in particular those concerning employment. The major goals are stated as:

- combat long term unemployment and exclusion from the labour market
- develop the professional skills and qualifications of potential job seekers
- help young unemployed persons to enter the labour market
- promote equal opportunities between men and women in the labour market
- foster the creation of new jobs
- preempt unemployment by adapting workers to industrial change
- improve education and training systems.

The move from the industrial society into the post-industrial information society requires large changes in the work patterns and nature of employment models for people. One of the aims of this project is to provide a set of usage scenarios centred on the P2P model for a global information ecology. In these scenarios we will envision a number of new roles and work patterns for people. But it is also important that by supporting the development of services as we do, we also contribute to the shaping of this culture. Our focus on highly distributed and heterogeneous solutions makes it possible for virtually anyone to become a service provider. This approach will contribute to social goals such as promoting equal opportunities between men and women, help young unemployed persons to enter the labour market, and foster the creation of new jobs.

8 Economic development and S&T prospects

Peer-to-peer computing is a new paradigm in which single-user applications are connected to a shared network as peers. When considered together, a group of collaborating peers forms one, much larger application, which we call a peer-to-peer application. There is a small but rapidly growing class of applications that can be classified as peer-to-peer, e.g., ICQ, Napster, SETI@home, FreeNet, and Gnutella. Up to now, these applications have been mainly limited to brokering and exchanging resources between peers. Because of the continued exponential growth of the Internet, we postulate that future peer-to-peer applications will become much more important and will cover a much wider spectrum of user needs.

Peer-to-peer computing is a promising approach for at least five technical reasons:

- *Network resources.* It will permit full use of existing network resources. The bulk of computational and other resources exist at the edge of the network, i.e., with individual users.
- *Scaling.* It will permit applications that scale better than server-centric techniques such as "server farms". Currently, the responsibility of handling growing numbers of users is placed on the provider of a service, who has to install ever more costly server farms. Letting the users be peers will allow them to share the load.
- *Device mobility.* Because peer-to-peer by its nature handles dynamic connectivity, it will permit device mobility, e.g., PCs that change their connection points and IP addresses.
- *Individual publishers.* It will allow individuals to become publishers of information and services. Currently, an individual cannot directly publish either. He or she must go through an intermediary, e.g., a service provider.
- *Anonymous networks.* It will allow creating any number of anonymous networks, i.e., virtual private networks without centralized control. This allows individuals to remain anonymous, while still publishing information and collaborating with others.

These advantages will potentially result in new and improved applications, thus increasing the competitiveness of the European software industry. PEPITO will help realize this in the following ways:

- The project will disseminate Open Source software (platforms, tools, demonstrator applications).
- The project will educate many people to peer-to-peer computing, including the partners' laboratories, students, as well as companies with which we have contacts.
- The project will disseminate its results in both the scientific and industrial communities, at conferences, workshops, and other events.
- The project will organize open workshops both to disseminate results to interested outsiders (both in university and industry) and to get valuable feedback.
- The project partners will assess the potentials for exploitation. If this assessment finds potentials for commercialisation, partners will form a spin-off company to exploit project results. This activity has already happened in previous work on Mozart, e.g., the FriarTuck company [29].

9 Workplan

9.1 General description

We now make concrete the basic project structure of Section 2 and give the workpackages, their tasks, and their interrelationships. PEPITO is organized into six workpackages:

WP1 Formal Models. This workpackage will devise formal models for peer-to-peer group collaboration (P2P-calculus), transactions, versioning, and modularity.

WP2 Distributed Algorithms and Services. This workpackage will study consistency and approximate replication models, network topology and diffusion algorithms, a distributed directory service, probabilistic algorithms for agreement, and failure detectors.

WP3 Programming Languages and Systems. This workpackage will implement language front ends for Oz and Java (using DACE library). The workpackage will also implement mobile computation and resource control.

WP4 Distribution Subsystem. This workpackage will provide the generic distribution services needed for peer-to-peer computing.

WP5 Demonstrator Applications. This workpackage will implement two demonstrator peer-to-peer applications that have enough functionality to be a convincing proof-of-concept. The applications are chosen to exercise the diffusion and multi-consistent aspects of peer-to-peer computing.

WP6 Project Management with Assessment and Evaluation. See Section 9.7.

We now discuss this organization. WP1 gives the foundational underpinning that feeds into the algorithmic work of WP2. This work will feed into WP4 (for the language-independent part) and WP3 (for the language-dependent part). The platforms of WP4 will be used to write the applications of WP5. Results from these applications will feed back into the other workpackages.

Workpackages WP1, WP2, and WP4 are language independent. Workpackages WP3 and WP5 are language dependent. We justify this as follows. WP1 and WP2 treat general issues of peer-to-peer computing. They will result in algorithms and formal models of aspects that are relevant in the context of peer-to-peer computing. WP4 provides the basic services needed for any peer-to-peer application. WP3 integrates peer-to-peer aspects into Oz and Java. Both will use the services of WP4. WP5 concretizes the potential of the languages in WP3 by implementing two demo applications.

We now discuss each workpackage separately.

9.1.1 WP1: Formal Models

Models of peer-to-peer group collaboration We will try to identify key issues of peer-to-peer computation and collaboration by distilling them in a formal calculus. Key issues to be expressed in this model may include: multicast communication, replicated state, consistency models, distributed transactions, transient failures, mobility and dynamic network topology, and probabilistic reasoning. At present, not enough is known about the issues to enable us to distinguish fundamental from derived concepts and to define a concrete calculus at the outset. Instead, we will design and refine the calculus as the project progresses, based on the experiences and results we gain in it. For this we can draw on previous experience with calculi such as Join [21, 22], Functional Nets [41], Nomadic Pict [62], and Ambients [16]. The P2P-calculus will likely be on a higher level than these calculi of mobile processes [35], but it will be connected to the lower level calculi via translations.

The work in this package will evolve from a collaboration of several project partners. It will clarify and guide work to be done on transactions, consistency, and probabilistic algorithms (all in WP2), and resource control (WP3) It will also influence the programming models of peer-to-peer computing (in WP3).

As a case study for the P2P-calculus we will take the model of collaboration exemplified by Groove.net [1], which can be summed up as follows: Sites participate in circles of collaborator processes; such circles come into existence and grow dynamically. In each circle, common state is managed in a decentralized way. Communication between sites means sending state deltas with the aim of establishing a shared view of what the current state is. Because of transient failures there is however no guarantee that global consistency will be reached.

Transactions and the semantics of failure Transactions are one of the most fundamental concepts in modern reliable distributed computing. Roughly speaking, a transaction provides a computational unit for concurrency and failure recovery. Transactions are typically defined through their ACID properties: Atomicity, Consistency, Isolation and Durability. While the transaction concept is powerful, its implementation in a peer-to-peer context poses difficulties: the level of partial failure implies that the ACID properties will sometimes be impossible to guarantee, as the algorithms required (e.g. 2-phase commit) will not be implementable.

We propose to axiomatize each of the ACID properties of the transaction concept, and to explore whether each of those properties can be represented by a first class abstraction. This deconstruction will help us study the relationship between transactional semantics and messaging, and the orthogonality of the ACID properties.

We will develop rigorous semantic models of the low-level interactions and failures that may occur in a peer-to-peer system – not in the rather abstract styles of traditional distributed algorithm theory [33] or process calculi [9, 22, 48], but for the actual networking APIs used. This will provide a basis for formal reasoning about implementations of transaction idioms, and guide the failure semantics of the P2P-calculus of WP1. We also plan to study the generalization of an existing formalization of a 2 phase commit algorithm in a mobile process calculus [13] in the context of the P2P-calculus.

Together, this work will precisely capture the notion of failure and the idioms required for programming with it.

Versioning and modularity As connectivity changes, there will inevitably be multiple coexisting versions of software components, both of peer-to-peer application programs and of the underlying distributed services. To construct reliable systems therefore requires precise notions of when two components are sufficiently consistent to interact or when it is possible for them to negotiate a common interface.

In previous work, partner UCAM has demonstrated that static type systems for modular programming can provide fine-grain versioning control of abstract types, supporting interoperation of old and new code. In this context, we define a *module* as a unit of potential version-change. This must now be extended to address version subtyping, by the introduction of polarised interaction types [44], and dynamic linking [31, 20]. The modular construction of distributed services is also an issue – we will use the work of WP2 and WP4 as a testbed for considering what type-theoretic support for structuring is required.

9.1.2 WP2: Distributed Algorithms and Services

We will extend the state of the art in diffusion-based and probabilistic algorithms and propose a generic framework in the context of dynamic peer-to-peer computing. We will also study consistency models and models for nonexact (approximate) replication of data. This workpackage is language-independent and will be applied to several quite different languages in the context of WP3. Many algorithms will be used either in WP4 or for libraries in WP5.

Consistency and approximate replication models For many peer-to-peer applications it is impossible to achieve adequate performance using traditional exact consistency models. This task will review the current state-of-art in consistency models to see how it is applicable in the context of peer-to-peer applications. In addition it will investigate models for approximate replication of data based on spatial and time metrics. We will also investigate special-purpose platforms for non-exact computation, e.g. gaming platforms, 3D visualization in order to understand the application needs. The partners have extensive experience in this class of applications. An important point is to clarify the division of labor between applications, languages, and the distribution subsystem. The input will be used in the WP3 and WP4 and WP5.

Network topology and efficient diffusion algorithms To achieve efficiency in peer-to-peer computing, system nodes have to configure themselves automatically so that the resulting network topology has desirable properties. Some relevant topologies are those given by the so-called "Small World" models [61]. These topologies guarantee properties regarding network diameter, degree of clustering, etc.

This task will develop a generic framework for diffusion algorithms that dynamically maintain these properties in the peer-to-peer context. We will revisit existing algorithms to identify their limitations in the context of a dynamic environment. We will investigate how to optimize diffusion algorithms given certain topologies that occur in a peer-to-peer context, e.g., Small World networks. The work here will be used in WP3 and/or WP4 and will also feed into the distributed directory service (see below).

Distributed directory service This task will provide a completely decentralized generic service directory. This is the basis for more specific services, like content-based routing, naming, or resource discovery. This task will use the results obtained from Task 9.1.2. There are several challenges:

- Achieving an acceptable access performance for the directory.
- Maintaining the consistency of local information in an environment where nodes fail and recover frequently.
- Maintaining a high degree of availability.

Given that the notion of initial global consistent state does not exist, the algorithms have to be self-stabilizing [18, 37, 11, 32, 53, 10].

Probabilistic algorithms In peer-to-peer systems one generally cannot assume any underlying mutual exclusion abstraction. Therefore, it is necessary to develop different kinds of agreement that make sense in a dynamic environment.

We shall explore here a probabilistic approach to agreement and use gossip-based algorithms that seem to particularly fit a dynamic and disconnected environment. We already foresee some solutions to specify and implement the semantics of a probabilistic form of reliable broadcast, but it is not clear how and whether that would make sense for a total order broadcast for example. Accompanying to this line of work, we plan to continue our study on the formalization of mutual exclusion algorithms in the context of mobile process calculi [40, 39, 38], where recent advances [30] have profited from probabilistic techniques. The aim of this work is to help bridging the gap between the above-mentioned algorithmic development and the P2P-calculus of WP1.

A probabilistic form of reliable broadcast could be stated as follows. There is a probability P that: (1) if a correct process broadcasts a message, it eventually delivers a message; (2) if a process delivers a message, a fraction F of correct processes eventually delivers the message; and (3) a message is delivered at-most-once and only if it was broadcast. Such a specification of a probabilistic primitive makes sense but finding out the right tools to measure the reliability of practical broadcast algorithms with respect to this specification is non-trivial. Furthermore, extending the specification to include traditional properties like causal or total order properties does not seem straightforward.

Failure detectors We will study the problem of failure detection and network partitioning detection for peer-to-peer environments. The usual failure detection methods may not work well in such environments: peer-to-peer networks are extremely large, they change rapidly over time, and their end-to-end communication delays are very large and highly variable. We will seek failure detection algorithms that provide Quality of Service guarantees in such networks (in terms of the time and accuracy with which failures are detected). We will also seek to design and implement a failure detection service that can be shared by peer groups, and that provides to each group the QoS guarantees that it needs.

9.1.3 WP3: Programming Languages and Systems

This workpackage integrates peer-to-peer abilities into two languages, Oz and Java. The Java integration will be through a library and external interface. The Oz integration will be at the level of the Mozart virtual machine. Each language is designed to use the programming models of WP1 and WP2 in its own way and thus explore a different part of the programming space. Each language is implemented on top of its own virtual machine, which is responsible for all non-distributed execution. Each virtual machine will delegate all distributed execution to the distribution subsystem of WP4.

It is important to note that resource management is a language issue and not a distribution subsystem issue, and as such belongs in this workpackage. This can be shown as follows. Consider the *stationary object view* of a resource as an object that can be remotely accessed. The language or the application defines what messages can be passed to the object. The only role of the underlying distribution subsystem in this scenario (see WP4) is to allow secure message passing, e.g., with capabilities and encryption.

Mobile computation Peer-to-peer systems are presently largely concerned with distributed *data*. For more flexibility, one requires both mobile *code* and mobile *computation*. Having both is clearly useful in a non-peer-to-peer environment. For example, it can substantially ease system management tasks and is crucial for mobile devices. It is an open question how much peer-to-peer applications would benefit from the added power of mobility. One obvious peer-to-peer use for mobile computation is to provide scalable computing resources by allowing load balancing between peers in group collaboration. We will investigate this and other as-yet-unknown uses of mobility by implementing it in the Mozart system.

Resource control Resources dependent on physical locations are dynamically linked to mobile code and computations. This presents challenges for scope and type analysis for the programming language. It is important to have assurance that mobile computations call resources in the correct way. Current typing disciplines only guarantee that mobile agents do not call local resources incorrectly. To solve the problem, we will use a formal model of resource allocation and use [49]. The solution will involve resolving a critical language-design issue for both high-level and low-level languages: process calculi with mobile features are classified either as lexically scoped (the π-calculus and its derivatives [34, 51]) or as dynamically scoped (Ambients [16]).

Oz This task will extend the Mozart system for peer-to-peer computing. Mozart is an open-source platform with a large user community. Mozart implements Oz, a dynamically-typed concurrent functional/object programming language based on the concurrent constraint model. Mozart provides a network-transparent distributed implementation of Oz that extends each language operation with a distributed semantics [36]. This is done carefully to provide predictable control over network communication patterns. The result is a practical system that has been used in industrial projects and deployed applications. Oz was chosen by Microsoft as one of the languages of the .Net effort. The Mozart system provides reflective failure detection (fail-stop site failure and network inactivity detection, both synchronous and asynchronous), with which we have programmed useful fault tolerance abstractions. The Mozart virtual machine has to be adapted to the distribution subsystem in WP4. Additionally, the Oz language must be extended to reflect new programming abstractions that use the protocols of WP2.

Java/DACE/Funnel This task will provide a frontend to the distribution subsystem as a set of Java library classes and will adapt the Funnel language and the DACE framework to use these classes. DACE (Distributed Asynchronous Computing Environment, `www.d-a-c-e.com`) provides support for publish/subscribe styles of distributed interaction in Java. As such, it is intended to serve as a benchmark against which more advanced and/or special-purpose languages can be compared.

Funnel [42, 8] is a small statically-typed composition language which supports functions, objects and concurrency. Funnel has an operational semantics based on the join calculus. Funnel's syntax and type system are designed to allow a large degree of flexibility for writing extensible software systems. For instance, it is possible to support transaction-related concepts in a library, without having to extend the language proper. The new implementation of Funnel compiles Funnel programs to Java bytecodes and enables a smooth cooperation between Funnel and Java components.

9.1.4 WP4: Distribution Subsystem

This workpackage is concerned with the design and implementation of a distribution subsystem (DSS) for peer-to-peer computing. The DSS offers a wide range of services including basic services (message sending, secure network pointers, distributed garbage collection) and a full suite of consistency protocols for stateful data. The DSS will be built on top of standard low-level communication protocols, i.e., TCP/UDP. Although IP multicast could be used, the DSS cannot rely on its always being present.

The DSS is language independent. It can be either made available directly to applications (as libraries) or to a (centralized) virtual machine, e.g., JVM, .Net, or OzVM. An important design goal is that the DSS be easy to use by any centralized VM, by extending it with a small set of new features and important hooks. The two components, extended VM and distribution subsystem, together realize a general-purpose peer-to-peer programming platform.

We now explain the tasks of this workpackage.

Basic services This task will provide the basic services, based on previous work. The task here is to extract a generic interface out of the existing Mozart implementation [15]. This includes the following services:

- Reliable, scalable, message sending (including network pointer management and generic marshaling/unmarshaling support).
- Basic failure detection including network partitioning detection. In general, in a peer-to-peer context, permanent failure cannot be discovered and so failure detection must rely on timeouts (e.g., heartbeat).
- Extend protocols for network awareness, based on previous work. This includes the following protocols:
  - Mobility protocol for state (e.g., cached state).
  - Other mobility protocols (e.g., thread mobility).
  - Invalidation protocol for state.
  - Caching protocols for stateless information.
- Fault containment. This consists of isolating the presumably faulty node. The involved nodes must first agree on failure and then isolate the faulty node to initiate error recovery.
- Dynamic connectivity. This consists of process connection mechanisms that do not depend on IP addresses. An important issue is correct failure detection of nodes with dynamically changing connectivity. It is important to understand exactly what guarantees can be provided.

Diffusion-based routing and caching protocols Two basic operations of a peer will be routing and caching. These operations will in many cases be independent of the application. Therefore the distribution subsystem needs to support diffusion-based routing and caching algorithms.

Multi-consistency protocols There may be several useful consistency protocols that permit the simultaneous existence of inconsistent states. We see at least the need for a causal replication protocol for state. For performance reasons these protocols will probably use an unreliable message-sending component (on top of which arbitrary levels of reliability can be built).

Distributed garbage collection This task will provide distributed garbage collection. Because of the interaction with memory management in the centralized VM, there will be an interface between the two. Distributed garbage collection should work on failure-prone wide-area networks without needing global synchronization and without impacting network performance noticeably. These are difficult requirements to satisfy together. Possibly, some of the distributed memory management will be made visible to the centralized VM or the application, for garbage collection of partially failed data structures and distributed statefull cycles. There may be a time-lease mechanism to help collect these cycles in the context of failures.

Capabilities and security The distribution subsystem (including basic services, protocols, and garbage collection) must be hardened against malicious attack. To achieve this, the task will add at least the following security mechanisms:

- Unforgeable references, i.e., capabilities.
- Encrypted communication for protocols, e.g., authenticated protocol messages.
- Security awareness, i.e., the ability to control whether or not the security support is used.

While PEPITO will not fully address the issue of security, it is important to provide at least the above implementation support to have a credible platform for P2P computing.

9.1.5 WP5: Demonstrator Applications

This workpackage will build two demonstrator applications. These are small applications that have enough functionality to be a convincing proof-of-concept. The following descriptions are tentative; it is almost certain that experience gained during the project will change the application scenarios.

Distributed auction system We will build a prototype implementing a completely decentralized auction system. This prototype is intended to explore diffusion-related concepts (WP4), the work on transactions (WP1), and dynamic forms of agreement (WP2). It will be implemented using an adaptation of the existing DACE framework. The demonstrator requires an interface between DACE and the DSS via the Java VM (WP3).

Collaborative game in a virtual environment This application explores multi-consistent computation (WP2). The application allows for game-playing in collaborative environment (virtual world) context. This will exploit previous work on distributed interactive environments that heavily exploited inexact consistency as well as previous work on collaborative worlds that exploited exact consistency. This application will be implemented in Oz.

9.2 Workpackage list

Workpackage list

Work-package No.	Workpackage title	Lead contractor No.	Person-months	Start month	End month	Phase	Deliverable No.
WP1	Formal Models	UCAM	92 *(20)*	0	36		D1.1 D1.2 D1.3 D1.4 D1.5 D1.6
WP2	Distributed Algorithms and Services	KTH	82 *(14)*	0	36		D2.1 D2.2 D2.3 D2.4 D2.5 D2.6
WP3	Programming Languages and Systems	EPFL	78 *(10)*	0	36		D3.1 D3.2 D3.3 D3.4 D3.5 D3.6 D3.7 D3.8 D3.9
WP4	Distribution Subsystem	SICS	62 *(4)*	0	36		D4.1 D4.2 D4.3 D4.4 D4.5 D4.6 D4.7 D4.8
WP5	Demonstrator Applications	UCL	36 *(0)*	12	36		D5.1 D5.2
WP6	Project Management	SICS	12 *(0)*	0	36		
	TOTAL		362 *(48)*				

Note: the italicized numbers in parentheses refer to the nonfunded contributions of the AC partners. That is, EPFL (12 spread evenly over WP1, WP2, WP3), KTH (12 spread evenly over WP2 and WP3), UCAM (12 added to WP1), and UCL (12 spread evenly over WP1, WP2, WP4).

9.3 Workpackage descriptions

Workpackage description: Formal Models

Workpackage number:	WP1	**Start date or starting event:**	

Participant number:	1	2	3	4	5	6	
Person-months per participant:	0	16	6	0	52	18	

Objectives The objective of this workpackage is to provide formal models for peer-to-peer applications. These models should cover the issues of replicated state, consistency models, transactions, failure, mobility, dynamic network topology and distributed transactions.

Description of work

The workpackage will study models for group collaboration and transactions in a peer-to-peer context. The work is divided into the following tasks.

Models of peer-to-peer group collaboration (EPFL:6, INRIA:6, UCAM:10, UCL:6) This task will identify key issues of peer-to-peer computation and develop a formal calculus over these issues.

Transactions and the semantics of failure (EPFL:10, UCAM:18, UCL:12) This task will apply the multifaceted concept of "transaction" to the peer-to-peer context and see how the different facets can be used there.

Versioning and modularity (UCAM:24) This task will study what type-theoretic support is needed to support programs using multiple versions of a module, which is expected to be a common occurrence in a peer-to-peer application.

Deliverables

D1.1 Requirements for a peer-to-peer calculus.

D1.2 A formal calculus for peer-to-peer computation and collaboration.

D1.3 Axiomatisation of transactions.

D1.4 Study of possible semantics of failures.

D1.5 Requirements for peer-to-peer versioning support.

D1.6 Type system for version change.

D1.7 First progress report on formal models.

D1.8 Second progress report on formal models.

D1.9 Final report on formal models.

Milestones and expected result

Major milestones correspond to above deliverables.

Workpackage description: Distributed Algorithms and Services

Workpackage number:	WP2	Start date or starting event:				
Participant number:	1	2	3	4	5	6
Person-months per participant:	6	18	10	36	0	12

Objectives The objective is threefold. First, targeting peer-to-peer application requirements we aim to advance the state of the art in diffusion-based and probabilistic algorithms. Second, to develop a generic framework in the context of peer-to-peer computing for multi-consistent modeling. Third, to solve the problem of failure detection for peer-to-peer requirements.

Description of work

Consistency and approximate replication models (SICS:6, UCL:12) This task will do the following:

- Study the current state-of-art in consistency models.
- Study the consistency models (however adhoc) used in state-of-art applications.
- Develop an integrated model for multi-consistency.
- Determine the appropriate level of application support for multi-consistency (i.e. language, distribution subsystem, library).

Network topology and efficient diffusion algorithms (KTH:18) This task will do the following:

- Study existing diffusion algorithms in a peer-to-peer context.
- Develop a generic framework for these diffusion algorithms.
- Advance the state of the art in diffusion algorithms.
- Study and develop techniques for optimizing diffusion algorithms under typical peer-to-peer topologies.

Distributed directory service (KTH:18) This task will develop a completely decentralized generic service directory.

Probabilistic algorithms (EPFL:18) This task will study probabilistic forms of agreement in the peer-to-peer context, where intermittent failures make exact agreement difficult or impossible.

Failure detectors (INRIA:10) This task will study the problem of failure detection and network partitioning detection for peer-to-peer environments and develop failure-detection algorithms that provide QoS guarantees in peer-to-peer networks.

Deliverables

D2.1 Multi-consistency in peer-to-peer computing.

D2.2 Diffusion algorithms for peer-to-peer computing.

D2.3 Distributed directory service.

D2.4 Specification of probabilistic reliable broadcast.

D2.5 Algorithms and applications of probabilistic reliable broadcast.

D2.6 Failure detectors in peer-to-peer computing.

D2.7 First progress report on distributed algorithms.

D2.8 Second progress report on distributed algorithms.

D2.9 Final report on distributed algorithms.

Milestones and expected result

Major milestones correspond to above deliverables.

Workpackage description: Programming Languages and Systems

Workpackage number:	WP3	Start date or starting event:				
Participant number:	1	2	3	4	5	6
Person-months per participant:	6	24	24	12	0	12

Objectives The objective of this workpackage is to extend several different programming languages for peer-to-peer computing. Two languages are chosen (Oz and Java) for research and interoperability purposes.

Description of work

Mobile computation (KTH:12) This task is to extend the Oz programming language with facilities for mobile computation (thread groups).

Resource control (INRIA:24) This task will study how a typing discipline can provide support for correct use of dynamically-linked resources, in particular for mobile code and mobile computations.

Oz (SICS:6, UCL:12) This task will integrate the distribution subsystem developed in WP4 into the Oz programming language and its Mozart implementation.

Java/DACE/Funnel (EPFL:24) This task will provide a frontend to the distribution developed in WP4 as a set of Java library classes. We will adapt the Funnel language and the DACE framework to use these classes.

Deliverables

D3.1 Mobile computation design.

D3.2 Mobile computation implementation.

D3.3 Resource control design.

D3.4 Resource control implementation.

D3.5 Peer-to-peer Oz platform (1).

D3.6 Peer-to-peer Oz platform (2).

D3.7 Extending the Oz language for peer-to-peer computing.

D3.8 An interface between the Java VM and the Distribution Subsystem.

D3.9 A library for distributed peer-to-peer applications in Funnel.

D3.10 Progress report on languages and systems.

D3.11 Final report on languages and systems.

Milestones and expected result

Major milestones correspond to above deliverables.

Workpackage description: Distribution Subsystem

Workpackage number:	WP4	**Start date or starting event:**				
Participant number:	1	2	3	4	5	6
Person-months per participant:	42	0	6	8	0	6

Objectives The objective of this workpackage is to provide a language-independent distribution subsystem. This distribution subsystem will offer a wide range of distributed services/protocols/ algorithms. The distribution subsystem is specifically designed to allow close coupling with centralized virtual machines. This will allow the integration of distribution services at the language level.

Description of work

Basic services (SICS:18, UCL:6) This task will provide the basic distribution services already existing in the Mozart implementation by doing three things: extracting a generic interface out of the current system, extending existing protocols for network awareness, and providing fault containment and dynamic connectivity support.

Diffusion-based routing and caching protocols (SICS:12) This task will implement basic support for diffusion-based routing and caching into the distribution subsystem.

Multi-consistency protocols (SICS:12) This task is by guided by the work in WP2 to integrate multiple consistency protocols into the distribution subsystem.

Distributed garbage collection (INRIA:6) The goal of this task is to provide distributed garbage collection that works on failure-prone networks without requiring global synchronization and without impacting network performance noticeably.

Capabilities and security (KTH:8) This task will add security support to the DSS. This includes at least the following three topics: unforgeable references, encrypted communication, and security awareness, i.e. the ability to control whether or not the security support is used.

Deliverables

D4.1 Basic distribution subsystem.

D4.2 Report on basic distribution subsystem.

D4.3 Security services in distribution subsystem.

D4.4 Report on security services in distribution subsystem.

D4.5 Extended distribution subsystem.

D4.6 Report on extended distribution subsystem.

D4.7 Distributed garbage collection design.

D4.8 Distributed garbage collection implementation in DSS.

D4.9 Progress report DSS.

D4.10 Final report on DSS.

Milestones and expected result

Major milestones correspond to above deliverables.

Workpackage description : Demonstrator Applications

Workpackage number:	WP5	**Start date or starting event:**				
Participant number:	1	2	3	4	5	6
Person-months per participant:	12	6	6	0	0	12

Objectives The objective of this workpackage is to develop two peer-to-peer demonstrator applications to test, evaluate, and validate the work in WP1 through WP4. The applications will focus on diffusion-based computing and multi-consistent computing.

Description of work

We give here two tentative application descriptions. We expect that the experience gained in the project will modify them, or even replace them by suggesting other applications that will better illustrate peer-to-peer computing or more thoroughly challenge the platforms developed in WP3 and WP4.

Distributed auction system (EPFL:6, UCL:6) This application explores diffusion-based computing. It is a completely decentralized auction system implemented using an adaptation of the existing DACE framework.

Collaborative game in a virtual environment (INRIA:6, SICS:12, UCL:6) This application explores multi-consistency. We will develop a collaborative environment and incorporate a game involving 3D visualization. This application will need both exact and inexact consistency models for shared state.

Deliverables

D5.1 A demonstrator for a distributed auction service.

D5.2 A demonstrator for a collaborative game.

D5.3 Report on demos.

Milestones and expected result

Major milestones correspond to above deliverables.

Workpackage description: Project Management

Workpackage number:	WP6	Start date or starting event:	0			
Participant number:	1	2	3	4	5	6
Person-months per participant:	11	0	0	0	0	0

Objectives

To manage the project, disseminate results in a timely manner, and assess the project success.

Description of work

Project management This task will do the following:

- Carry out the overall project management, including administrative management.
- Define the project standards and guidelines in relation to deliverables, presentations, information and communication flow.
- Organize official project meetings and reviews.
- Coordinate, compile, and distribute project reports.

Workpackage management This task will do the following:

- Carry out the technical project management.
- Coordinate the developments in the different workpackages.
- Ensure system integration and resolve technical conflicts.

Dissemination and exploitation This task will do the following:

- Create and maintain a Web site with presentations, project reports, publications, and related matters, informing scientific and industrial communities of the project progress.
- Explore potential applications of the project results in Internet applications.

Assessment and evaluation This task will do the following:

- Workpackage coordinators report to project coordinators, to provide in-time input on the evolution of the project according to assessment criteria provided for each workpackage
- Provide evaluation of the project results during the project at months 12, 24 according to the assessment criteria for each workpackage.
- Provide final evaluation at month 36 according to the four measurable objectives given in Section 2.

Deliverables

D6.1 First project workshop.

D6.2 Second project workshop.

D6.3 Third project workshop.

D6.4 Dissemination and use plan

D6.5 Technological implementation plan

D6.6 Assessment criteria for workpackages

D6.7 First progress and assessment report

D6.8 Second progress and assessment report

D6.9 Final report

Milestones and expected result

Major milestones correspond to above deliverables.

9.4 Deliverables list

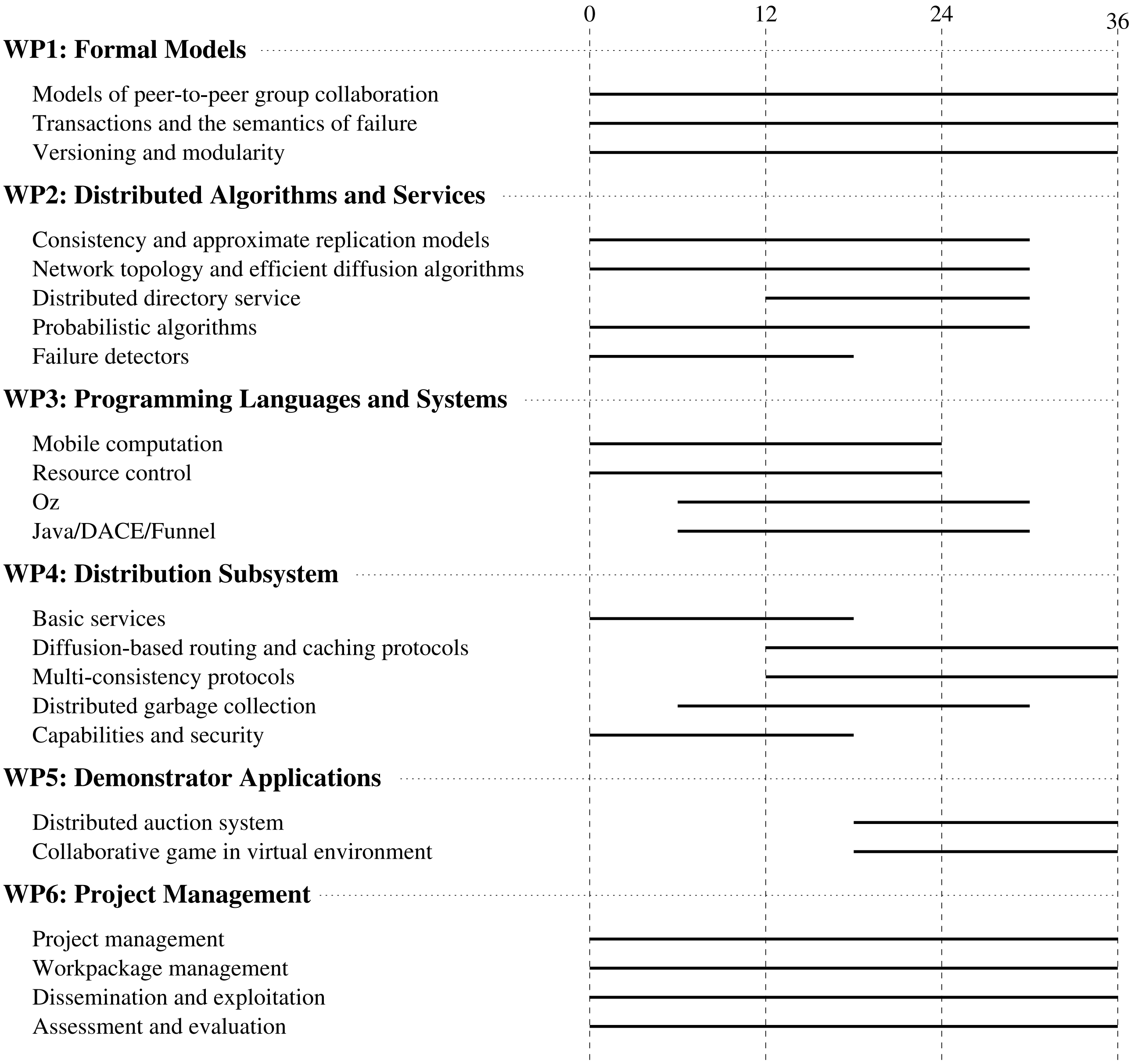
Deliverables list

Del. no.	Deliverable title	WP no.	Lead part.	Est. person months	Del. type	Security	Delivery (proj. month)
D1.1	Requirements for a peer-to-peer calculus	1	UCAM	13	report	Pub.	18
D1.2	A formal calculus for peer-to-peer computation and collaboration	1	UCAM	13	report	Pub.	36
D1.3	Axiomatisation of transactions	1	EPFL	20	report	Pub.	18
D1.4	Study of possible semantics of failures	1	UCAM	19	report	Pub.	36
D1.5	Requirements for peer-to-peer versioning support	1	UCAM	12	report	Pub.	36
D1.6	Type system for version change	1	UCAM	12	report	Pub.	36
D1.7	First progess report on formal models	1	UCAM	1	report	Pub.	12
D1.8	Second progess report on formal models	1	UCAM	1	report	Pub.	24
D1.9	Final report on formal models	1	UCAM	1	report	Pub.	36
D2.1	Multi-consistency in peer-to-peer computing	2	UCL	18	report	Pub.	36
D2.2	Diffusion algorithms for peer-to-peer computing	2	KTH	16	report	Pub.	36
D2.3	Distributed directory service	2	KTH	17	report	Pub.	30
D2.4	Specification of probabilistic reliable broadcast	2	EPFL	9	report	Pub.	18
D2.5	Algorithms and applications of probabilistic reliable broadcast	2	EPFL	9	report	Pub.	36
D2.6	Failure detectors in peer-to-peer computing	2	INRIA	10	report	Pub.	18
D2.7	First progress report of distributed algorithms	2	KTH	1	report	Pub.	12
D2.8	Second progress report of distributed algorithms	2	KTH	1	report	Pub.	24
D2.8	Final report of distributed algorithms	2	KTH	1	report	Pub.	36
D3.1	Mobile computation design	3	KTH	6	report	Pub.	24
D3.2	Mobile computation implementation	3	KTH	6	proto	Pub.	24
D3.3	Resource control design	3	INRIA	12	report	Pub.	12
D3.4	Resource control implementation	3	INRIA	12	proto	Pub.	24
D3.5	Peer-to-peer Oz platform (1)	3	UCL	4	proto & report	Pub.	18
D3.6	Peer-to-peer Oz platform (2)	3	UCL	4	proto & report	Pub.	30
D3.7	Extending the Oz language for peer-to-peer computing	3	UCL	4	report	Pub.	36

Del. no.	Deliverable title	WP no.	Lead part.	Est. person months	Del. type	Security	Delivery (proj. month)
D3.8	An interface between the Java VM and the Distribution Subsystem	3	EPFL	11	proto & report	Pub.	18
D3.9	A library for distributed peer-to-peer applications in Funnel	3	EPFL	11	proto & report	Pub.	30
D3.10	Progress report on languages and systems	3	EPFL	1	report	Pub.	24
D3.11	Final report on languages and systems	3	EPFL	1	report	Pub.	36
D4.1	Basic distribution subsystem	4	SICS	16	proto	Pub.	12
D4.2	Report on basic distribution subsystem	4	SICS	6	report	Pub.	12
D4.3	Security services in distribution subsystem	4	KTH	4	proto	Pub.	24
D4.4	Report on security services in distribution subsystem	4	KTH	4	report	Pub.	24
D4.5	Extended distribution subsystem	4	SICS	18	proto	Pub.	30
D4.6	Report on extended distribution subsystem	4	SICS	6	report	Pub.	30
D4.7	Distributed garbage collection design	4	INRIA	3	report	Pub.	18
D4.8	Distributed garbage collection implementation in DSS	4	INRIA	3	proto	Pub.	30
D4.9	Progress report on DSS	4	SICS	1	report	Pub.	24
D4.10	Final report on DSS	4	SICS	1	report	Pub.	36
D5.1	A demonstrator for a distributed auction service	5	EPFL	12	proto	Pub.	36
D5.2	A demonstrator for a collaborative game	5	SICS	22	proto	Pub.	36
D5.3	Report on demos	5	SICS	2	report	Pub.	36
D6.1	First project workshop	6	SICS	1	workshop	Pub.	12
D6.2	Second project workshop	6	SICS	1	workshop	Pub.	24
D6.3	Third project workshop	6	SICS	1	workshop	Pub.	36
D6.4	Dissemination and use plan	6	SICS	1	report	Pub.	6
D6.5	Technological implementation plan	6	SICS	1	report	Pub.	24
D6.6	Assessment Criteria	6	SICS	1	report	Pub.	6
D6.7	First progress and assessment report	6	SICS	1	report	Pub.	12
D6.8	Second progress and assessment report	6	SICS	1	report	Pub.	24
D6.9	Final report	6	SICS	1	report	Pub.	36

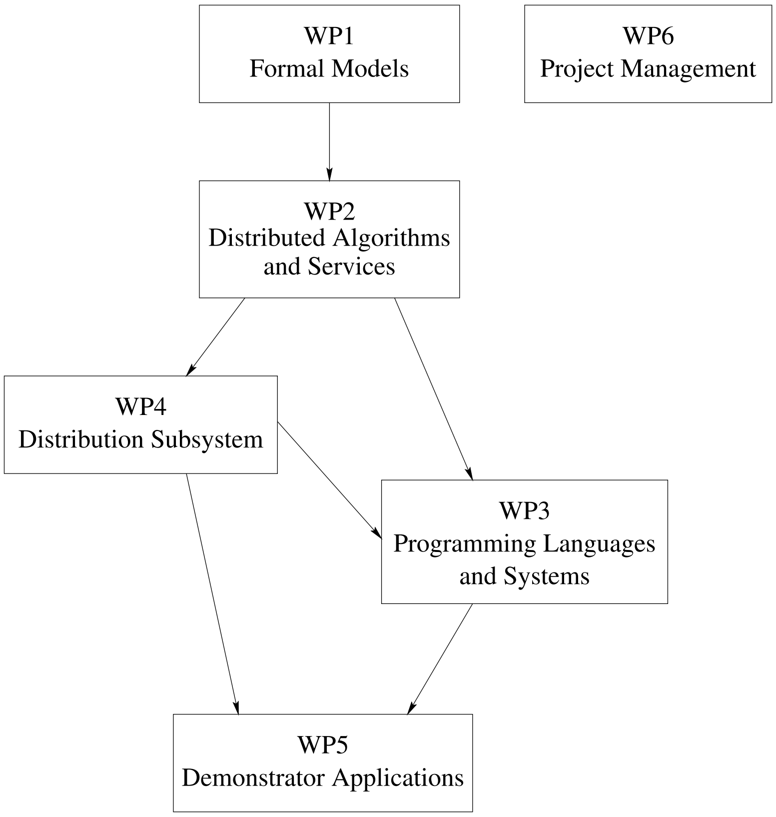
9.5 Project planning and timetable

The following Gantt chart shows the timing of the workpackages and their components.



9.6 Graphical presentation of project components

The following Pert diagram shows the interdependencies among the workpackages. An arrow from a to b means that there is work in a which is a prerequisite for b.



9.7 Project management

9.7.1 Top-level management structure

The project will be managed globally by two coordinators:

- The scientific coordinator is Prof. Seif Haridi from KTH. He is responsible for the scientific progress of the project, the coordination between workpackages, and all conflict resolution in the project.
- The administrative coordinator is Thomas Sjöland from SICS. He is responsible for respecting all contractual obligations with respect to the Commission. This includes timely submission of contract information, cost statements and timetables, deliverables, progress and management reports, financial arrangements between the partners and the Commission, and information dissemination (Web sites, documents, etc.).

9.7.2 Internal management structure

Each workpackage has a lead contractor, which is a partner who is responsible for the correct operation of the workpackage. The lead contractor reports to the administrative and scientific coordinators. The lead contractors are as follows:

- WP1: UCAM.
- WP2: KTH.
- WP3: EPFL.
- WP4: SICS.
- WP5: UCL.
- WP6: SICS.

Each partner has one person, the partner manager, who is responsible for the correct operation of the partner with respect to all its tasks including that of lead contractor. The partner manager reports to the administrative and scientific coordinators. The partner managers are as follows:

- SICS: Per Brand.
- EPFL: Uwe Nestmann.
- INRIA: Jean-Jacques Lévy.
- KTH: Seif Haridi.
- UCAM: Peter Sewell.
- UCL: Peter Van Roy.

The project board consists of the scientific coordinator, the administrative coordinator, and all partner managers.

9.7.3 Mechanisms for assessment and evaluation

The project board meets regularly to assess the progress of the project and to discuss corrective measures if the progress is insufficient. Objective evaluation of the project will be done according to the four criteria listed in Section 2, namely:

- Development of a formal model of peer-to-peer accepted by the research community. Acceptability can be judged, e.g., by publications in respected international conferences and journals.
- Advancement of the state-of-the-art in diffusion-based algorithms and multi-consistency algorithms. Acceptability can be judged as for the previous item.
- Development of a language-independent distribution subsystem.
- Implementation of convincing demonstrator applications.

The project will organize three workshops, one per year, open to all interested members of the research and industrial communities. In addition to their role for dissemination, these workshops will provide valuable feedback for the continuation of the project.

10 Clustering

This project could benefit from other projects, in the global computing programme, that are complementary. In particular projects that deal with issues related to security, language design, and distributed algorithms are of interest. Possible projects include SECURE, MIKADO, MRG, FLAGS, and DBGLOBE.

11 Other contractual conditions

The partners intend to travel outside the European Union and Associated States for attending conferences (workshops) and visiting research groups motivated by the project and for presenting PEPITO results. This will be requested in a case by case basis.

A Consortium description

A.1 Consortium

The proposed consortium consists of the following institutions with the following project roles:

Partner 1 Swedish Institute of Computer Science (SICS), Sweden. SICS is project administrative coordinator. At SICS, the DSL and ICE laboratories participate in the project proposal.

Partner 2 Ecole Polytechnique Fédérale de Lausanne (*Federal Polytechnic Institute of Lausanne*) (EPFL), Switzerland.

Partner 3 Institut National de Recherche en Informatique et Automatique (*National Institute for Research in Computer Science and Control*) (INRIA) at Rocquencourt, France.

Partner 4 Kungliga Tekniska Högskola (*Royal Institute of Technology*) (KTH), Sweden. KTH is project scientific coordinator. At KTH, the participant is the LECS of the Department of Microelectronics and Information Technology (IMIT), Sweden.

Partner 5 University of Cambridge (UCAM), United Kingdom.

Partner 6 Université catholique de Louvain (*Catholic University of Louvain at Louvain-la-Neuve*) (UCL), Belgium.

The PEPITO project is divided into the following workpackages, with in parenthesis the partner leading each workpackage:

- (WP1) Formal models (UCAM)
- (WP2) Distributed algorithms and services (KTH)
- (WP3) Programming languages (EPFL)
- (WP4) Distribution subsystem (SICS)
- (WP5) Demonstrator applications (UCL)
- (WP6) Project management (SICS)

EPFL are the main responsible for the models part of WP1 with collaboration from most partners. In WP1 models for versioning and modularity are the responsibility of UCAM, distributed transactions will be a co-operation between EPFL and UCL, where EPFL will deal with formal modelling and UCL with algorithmic parts. In WP2 KTH will be responsible for the diffusion algorithms and distributed directory services, EPFL for probabilistic algorithms, INRIA for failure aspects, and UCL for consistency models. In WP3 INRIA, EPFL, and SICS will work on adaptations for different languages of peer to peer applications assisted by KTH and UCL. WP4 on the distribution subsystem will be the main responsibility of SICS. Security aspects in WP4 are handled by KTH. The partners will collaborate in adapting the systems in WP3 to use the prototype of WP4. The WP5 demonstrator applications will use all workpackages to illustrate the results of the project.

A.2 Participating organizations

A.2.1 Swedish Institute of Computer Science

General information SICS, Swedish Institute of Computer Science AB, is a non-profit research organization. SICS mission is to contribute to the competitive strength of Swedish industry by conducting advanced and focused research in strategic areas of computer science, and actively promoting industrial use of new research ideas and results in industry and society at large. SICS works in a close collaboration with industry and the national and international research community.

Research SICS research focus is on distributed and networked interactive real-time multimedia systems and applications – spanning from infra-structural issues to software methodologies to human-computer interaction. SICS collaborates actively with industrial and academic partners. The core of SICS research is supported by major companies associated with Föreningen för Datateknisk Forskning (FDF), the government agency for innovation, VINNOVA, and the state-owned Ireco Holding AB supporting industrial research institutes in Sweden. Research contracts assigned by the FDF members and other organizations, Swedish and foreign, speed up dissemination and uptake of research results in industry and public sector. SICS also participates actively in collaborative R&D programs, both national and international, such as Esprit and ACTS funded by the European Commission, and Real World Computing funded by the Japanese government. SICS has a well developed collaboration pattern with high-tech SMEs in Sweden, carrying our joint projects, and acting as an external R&D resource for selected SMEs. SICS has also a proven record of disseminating and promoting industrial deployment of ts research findings, including establishing of spin-off companies, as well as licensing of its software and patents. SICS is a member of the following organisations:

- IRIS (Industrial research institutes in Sweden)
- ERCIM (European Research Consortium in Informatics and Mathematics)
- RWCP (Real World Computing Partnership, Japan)
- W3C (WorldWideWeb Consortium)

Facts Turn-over 1999: 74 MSEK. Staff: In January 1999 SICS had a research staff of 90, thereof 30 PhDs.

Output

- Research Results: approximately 70 refereed papers and articles in international journals and conferences per year, 2-4 research degrees (doctor and/or licentiate) by SICS employees per year, 1-3 people move to academia for tenured positions (professor, lecturer) per year.
- Industrial results: on average 10 people per year move to industry, 1-2 SICS projects and patents are transferred to industry, and 1-2 spin-off companies are formed. Software distribution is responsible for approximately 5% of the turn-over.
- Research Organization (labs):
  - Computer and Network Architectures
  - Distributed Systems Laboratory

– Formal Design Methods
– Intelligent Systems Laboratory
– Adaptive Robust Computing
– Human Computer Interaction and Language Engineering
– Interactive Collaborative Environments

The DSL laboratory at SICS The focus of the research activities of the Distributed Systems lab is on software tools and methods for supporting the development of distributed systems and applications, especially for networked and mobile computing. By networked systems we mean systems and applications that are distributed across the Internet, and by mobile we mean systems where some of the nodes are mobile, e. g. handheld wireless-devices. As part of our mission, we try out our tools, methods and software in realistic applications either as research prototypes (demos), or in cooperation with external partners (mainly industry). This area of software technology is very large; therefore we concentrate our efforts on the following areas where we believe we can make innovative contributions:

1. Design and implementation of network-enabled programming languages and systems.
2. New algorithms, middleware, libraries and techniques to support distribution, and in particular software-based high availability.
3. Intelligent agent technology.

There are a number of other research areas, in which the lab has and maintains expertise in order to build novel distributed applications that are also reliable and intelligent. Two examples are computer security and intelligent systems (knowledge-based systems, and machine learning). In these areas our research ismainly focused on incorporation of security techniques into middleware and programming systems, as well as incorporation of intelligent system techniques in agent platforms. The lab has designed and implemented the Mozart system, a platform for developing intelligent and distributed applications. Mozart is a novel programming system that integrates concurrency, functional and object-oriented programming, as well as logical and constraint inference in a coherent whole. Mozart is designed to simplify distributed application building, cleanly separating the functional and logical aspects of an application from aspects connected with distribution. Mozart is released as open source and there is an appreciable user-community that is continuously providing us with feedback on the system. Mozart is the lab's research vehicle providing a platform where new ideas, architecture and abstractions can be prototyped, tested and evaluated. This can be and is done on almost all levels of abstraction, from libraries and abstraction layers built on top of Mozart to low-level runtime support for distribution on the level of the abstract machine.

The ICE laboratory at SICS The Interactive Collaborative Environments Laboratory (ICE) at SICS focuses on the development of future computational environments that will support and promote cooperative interaction. Inhabitants of these environments will require support independently of physical location as well as having interaction and communication with fellow inhabitants to occur both synchronously and asynchronously. Designing the next generation of control and interaction surfaces will bring together work in tangible and physical interfaces coupled with distributed virtual reality technology to explore how a number of heterogeneous devices and interfaces can be used in unison. The ICE laboratory is currently a partner in the EU financed PING (IST), KidStory (I3) and ACCORD (IST) projects, and has previously participated in the COVEN (ACTS), eSCAPE (I3) and COMIC (ESPRIT) projects.

A.2.2 Ecole Polytechnique Fédérale de Lausanne

EPFL is primarily concerned with training engineers and architects capable of maintaining a high degree of excellence in industry, trade and culture. The 4,500 students and doctorate candidates are trained by highly-qualified teaching staff and researchers in a dynamic and internationally-oriented environment, and have access to very sophisticated equipment and installations. The EPFL also caters for experienced professionals, who wish to keep themselves abreast of the latest developments in technology. A comprehensive post graduate programme offers possibilities for decision-makers to increase and update their knowledge, so that Switzerland can remain competitive in the world economy.

There are 12 fields of teaching and research at the EPFL: civil engineering, rural engineering, mechanical engineering, electricity, physics, chemistry, mathematics, computer science, materials science, architecture, micro-engineering and communication systems. The EPFL is the only Swiss Institute of higher education to award a university level diploma in the last two subjects. It is the Institute's policy not to dehumanise, but to emphasize the human factor in technology by putting it to work for the present and future well-being of the society. With this goal in mind, the EPFL helps to solve current problems such as energy supply environmental protection and the effect of technology on the environment.

The high standard of basic and applied research at the EPFL contrives to attract the most eminent specialists each year, the increasing cooperation with industry makes it a leader among the Swiss higher education establishments in terms of industrial projects. The important international programmes in which it participates—almost 100 with the European Union alone, and several dozen in Africa, Asia and America—indicate a factor which has been present throughout the Institute's 140 years of history: an outward-looking attitude. All these elements help to make the EPFL a land mark centre of excellence at the heart of Europe.

Within the 4th framework programme, EPFL participated in 150 projects (33 MEUR); within the 5th framework, also more than 60 projects (around 20 MEUR) perform research together with EPFL-partner projects.

Training The EPFL offers a thorough scientific and technical training to its future engineers and architects. An EPFL diploma obtained after four and a half years of study is equivalent to those awarded by the best schools of higher education in Europe. The Institute also trains researchers and encourages students to continue to doctorate level. Young women are encouraged to prepare a doctorate by offering the conditions necessary to make post-graduate study compatible with family life.

Research A high standard of research is achieved by concentrating strengths. The EPFL focuses on materials science, production methods, information technologies, basic sciences, techniques and environment. The EPFL is the only site in Switzerland for thermonuclear fusion and it organises large inter-disciplinary projects in specialised sectors such as nanoscience, supercomputer applications, bioengineering and solar energy.

Scope Its teaching and research activities link the EPFL to networks ranging from regional to worldwide. Through its exchange of technology and cooperation agreements in Switzerland, in the industrialised North and in the developing South, the EPFL is in contact with widely differing circles: academic, economic, industrial, political. It also welcomes visits from the general public and schoolchildren, and is open to secondary school pupils for training periods.

The team consisting of the professors Rachid Guerraoui, Martin Odersky and Uwe Nestmann, has developed significant expertise in (1) understanding the fundamentals of distributed computing, (2) the foundations, design, and implementation of programming languages, and (3) the theory and modeling of concurrent computing. They have joined together in order to study problems underlying distributed and transactional algorithms, and to provide high-level abstractions for these. The team and their supporting research groups has a strong interest to join its complementary forces from both practical and theoretical points of view, which is exhibited by the various contributions through international publications of a high rank as well as through software systems with industrial impact.

A.2.3 Institut National de Recherche en Informatique et Automatique

INRIA is a French public-sector scientific and technological institute operating under the dual authority of the Ministry in charge of Research and the Ministry in charge of Industry. General Headquarters are situated at Rocquencourt near Versailles in France. INRIA has five research units at Grenoble, Nancy, Rennes, Rocquencourt, and Sophia-Antipolis (near Nice). INRIA's missions are "to undertake basic and applied research, to design experimental systems, to ensure technology and knowledge transfer, to organize international scientific exchanges, to carry out scientific assessments, and to contribute to standardization".

The research carried out at INRIA brings together experts from the fields of computer science and applied mathematics covering the following areas: networks and systems; software engineering and symbolic computing; man-machine interaction; image processing, data management, knowledge systems; simulation and optimization of complex systems. INRIA gathers in its premises around 2100 persons including 1300 scientists, many of which belong to partner organizations (CNRS, industrial labs, universities) and are assigned to work in common research teams called "projects". On INRIA's budget, around 500 full-time equivalent R&D positions can be accounted for. A large number of INRIA senior researchers are involved in teaching and their PhD students (about 550) prepare their thesis within the different INRIA research projects (currently 83). Its budget is roughly 90 MEuro, 25% of which comes from research and development contracts, royalties and sales. Technology transfer is addressed at INRIA via industrial contracts (400 running), European Projects (already 300 executed), Framework agreements with specific industrial companies, and finally the creation of Technology companies (40 created since 1984). In 1998, INRIA has launched a subsidiary to promote high-tech start-up companies: INRIA-TRANSFERT deals with early accompaniment of the future companies and is the instigator of the I-Source Gestion company, in charge of setting up start up funds in the field of Information Technology.

Together with MIT for America and the University of Keio for Asia, INRIA leads the European branch of the World Wide Web Consortium (W3C). The W3C is in charge of developing common protocols for the evolution of the World Wide Web. The European W3C research team is hosted at the INRIA premises in Sophia-Antipolis.

Expertise INRIA projects have much expertise in the areas related to Architectures and Systems, Networks and Telecommunications, Distributed and Real-Time Programming, Semantics and Programming Languages. This expertise notably leads to the participation of EC funded projects addressing issues relevant to OZONE concerns. For specific ongoing IST projects in which INRIA participates, let us mention: ADVISOR (Annotated Digital Video for Intelligent Surveillance and Optimised Retrieval), BRAIN (Broadband Radio Access for IP Based Networks), CARSENSE (Sensing of Car Environment at Low Speed Driving), DSoS (Dependable Systems of Systems), EVENTS (Computer Vision Enables non-constrained Transmission of Scenarios), INTERLIVING (Designing Interactive, Intergenerational Interfaces for Living Together), MOUMIR (Models for Unified Multimedia Information Retrieval), OPENISE (Open for Enhanced Interactive Services), OZONE (New Technologies and Services for emerging nomadic societies), PROXITV (Proxy Infrastructure for Internet Interactive TV), SALT (Standard-Based Access Services to Multilingual NPL-Lexicon), SONG (Portals of Next Generation), VESPER (Virtual Home Environment for Service Personalization and Roaming Users), VISOR BASE (Video Sensor Object Request Broker open Architecture for Distributed Services). At INRIA, research is organized into Projects. A project is a team of researchers - from INRIA and other establishments, associate researchers, foreign visitors and Ph. D. students, who are conducting research activities on a common subject. The projects are grouped into four major topics, all represented at INRIA Rocquencourt. See `http://www-rocq.inria.fr/en/recherche/index.htm`.

Project team INRIA participation to PEPITO will be through involvement of INRIA projects working in the fields of networks and systems, software engineering, man-machine interaction, image processing, data management, and knowledge systems. An overview of the research activities that are being carried out in the above fields may be found at

`http://www.inria.fr/Recherche/activites-eng.html`.

The project team focuses on the development, the compilation and the semantics of concurrent and functional languages for distributed environments, with possible migrations. Currently, the team is investigating a new model of distributed programming, derived from the pi-calculus, called the Join-calculus. We have developed two implementations of the JoCaml language, which realizes this model, and produced some theoretical results. The team also has an interest in the validation and the debugging of real-world concurrent programs, such as the Caml concurrent garbage collector or the Ariane 5 on-board software.

A.2.4 Kungliga Tekniska Högskola

KTH will participate through the Laboratory of Electronics and Computer Systems (LECS) in the IMIT department. The activities of LECS concern Computer Systems at large, but in particular parallel computer systems, computer architecture, and distributed computer systems including programming systems. The research is mainly motivated by technical challenges in distributed and parallel systems. In all of our projects, we focus as well on experimental design and evaluation as on sound design principles.

Prof. Haridi's group in LECS co-develops the programming system Mozart for transparent programming of distributed and mobile applications. The system greatly simplifies the development of net based services. The research is now focused on aspects such as scalability, security and support for fault tolerance (in cooperation with SICS and Ericsson). There is also work on adaptation of Mozart to small mobile units (PDAs) and algorithms for distributed systems and fault-tolerant systems, in particular work on self-stabilising algorithms and also program analysis of parallel and concurrent systems.

In 1994, LECS researchers developed a an extensible object-oriented platform NUTS for distributed computing [60]. The language of NUTS is a concurrent object-oriented programming language with coarse-grained parallelism and distributed shared memory model implemented on a distributed memory architecture, e. g. network of workstations. NUTS processes can be arranged into structured collections: grids which enable to program data-parallel computations on a high level. The distributed layer of NUTS supports passing objects, classes and scripts among distributed NUTS processes, and is based on serialization technique similar to the Java object serialization developed later at SUN Microsystems.

In the late 90s, CSLAB, IMIT/KTH, carried out research in HW and SW support for fine-grained communication and synchronization of parallel and distributed processes. The goal of this research was to provide and to implement a flexible and efficient model of inter-task communication and synchronization via synchronizing shared memory with aggressive support for fine-grained communication and synchronization. Within this project, CSLAB developed and implemented on top of PVM the mEDA shared memory model [5]. The model provides a unified approach for communication and synchronization of parallel processes using memory operations of different synchronization types.

More information about the LECS research group, project and publications can be found at `http://www.it.kth.se/labs/cs`.

A.2.5 University of Cambridge

The University of Cambridge is one of the leading Universities in the United Kingdom. The Computer Laboratory was founded in 1937, as the Mathematical Laboratory. It continues to be in the forefront of Computer Science Research, with approximately 60 academic and research staff, and 80 postgraduates engaged in research for the PhD degree. It received a 5* rating (the top rating) in the most recent UK research assessment exercise.

Two groups are directly involved in PEPITO: the Theory and Semantics group (studying mathematical models for reasoning about systems, and programming language design), and the Opera distributed systems group. The main participants are Peter Sewell and Gavin Bierman, with valuable context from the other Opera group work led by Jean Bacon and Ken Moody. Further context will come from contacts with the strong research groups in Automated Reasoning (studying the development and application of theorem proving methodologies), Security (studying security, cryptology, and their applications), and Systems, studying networks, operating systems, multimedia, distributed systems, and computer architecture.

A.2.6 Université catholique de Louvain

The Université catholique de Louvain (UCL), is one of Belgium's largest research and educational institutions. Founded in 1425, the mother university bifurcated in 1971, giving birth to two independent entities, UCL and its sister university KUL (Katholieke Universiteit Leuven). While the KUL remained in Louvain, UCL established itself in the new town of Louvain-la-Neuve. UCL currently has more than 20000 students (including 4000 foreign students), 3000 teaching and research personnel, and 1800 technical and administrative personnel. UCL hosts more than 1000 external research contracts with an annual turnover of 70 million Euro and is the nucleus of an industrial park containing almost 100 companies. UCL is an active participant in the ERASMUS (now SOCRATES) and TEMPUS programs, and is a founding member of MED-CAMPUS, the Coimbra Group, CLUSTER, FIUC, NATURA, CEMS, etc.

The Department of Computing Science and Engineering at UCL (INGI) is part of the Faculty of Applied Science. INGI has nearly 50 teaching and research personnel, with a twofold mission: education (offering engineering and doctoral degrees) and research. The department is recognized world-wide for its contributions to dependable distributed computing, constraint and logic programming, and software engineering. Major releases by the department include the KAOS methodology and its tools for requirements engineering, and the Mozart Programming System for distributed application development and constraint programming.

INGI collaborates closely with research groups in many areas of the world including the USA, France, Germany, Sweden, and the third world. INGI personnel are major contributors and catalysers of computer technology and applications in Belgium. For example, INGI is a key player in the creation and continuing development of the CEDITI s.a., a software consulting and services company specializing in multimedia and Web development and software engineering, in the WIN, the Wallonia High-performance Intranet, and in the CETIC a.s.b.l., a new applied research laboratory.

Current INGI projects include TELESUN (computer-based distance learning; European Union), PIRATES (distributed application development; Walloon Region of Belgium), FRISCO (formal reasoning in software construction; French Community of Belgium), BIZNET (catalysing Internet commerce; Walloon Region), and IGLOO (technology transfer in software engineering; Walloon Region).

A.3 CVs of key persons

A.3.1 SICS

Per Brand (partner manager) Senior researcher at SICS and director of the Distributed Systems Laboratory since October 2000. He has done research on OR-parallel Prolog Systems and was part of the Aurora project team. He is co-designer of the Mozart Programming System and was the chief developer of the distribution subsystem of Mozart [36]. His current research interests are programming languages and distributed algorithms and the merging of the two into general-purpose distributed programming platforms [15].

Lennart E. Fahlén Director of the Interactive Collaborative Environments Laboratory, ICE. He has several scientific publications in the domain of tools and techniques for distributed virtual environments as well as being a well-known presenter at international conferences. The ICE lab is the main developer and maintainer of the DIVE VR system, an effort originating back to 1991. Fahlén's current interests lie in the areas of future interactive television, new metaphors for computing and man-machine interfaces and he has a special interest in the use of VR and associated technology for artistic purposes and in combination with disciplines such as dance and music. Fahlén holds a patent in the area of computer architecture.

Thomas Sjöland (project administrative coordinator) Acting lecturer at KTH and senior researcher at SICS. Sjöland is involved in research on logic programming and analysis of concurrent constraint programming languages, for instance in the ESPRIT-project ParForce where he designed and implemented an analyser for type/alias-analysis of AKL-programs. He managed the Swedish project on industrialisation of SICStus Prolog and coordinated the ESPRIT-project ACCLAIM. He is a member of the executive committee of the COMPULOG Network of Excellence. He has publications on program analysis, applications of constraints in planning and scheduling, object oriented logic programming, and computer music and has designed several data analysis systems for medical research and implemented communication protocols as well as real-time control software.

A.3.2 INRIA

Jean-Jacques Lévy (partner manager) Born 1947. Ph. D. in Theoretical Computer Science (1974, University of Paris). Doctorate in Sciences: Mathematics and Computer Science (1978). He has held visiting or consultant positions at Ecole Polytechnique in Paris, Xerox PARC, DEC/SRC and was permanent member of the research staff at DEC/PRL for 1 year. He headed the Para project during 12 years at INRIA and served as member and vice-chair of the national Evaluation Committee of INRIA for 6 years. He is "Directeur de Recherche" at INRIA, and Professor at the Ecole Polytechnique. He has been coordinator of Esprit projects Confer-1 and Confer-2. Jean-Jacques Lévy has research interests in general computer science. For his Ph. D. , he has worked in the lambda-calculus and syntax of abstract programming languages. He has also worked in operating systems, CAD for VLSI, interactive graphics, term rewriting systems, functional languages and their parallel implementation. He has been supervisor of 10 Ph.D.'s, and reviewer of 30 Ph. D. theses, habilitations, and D. in Sc.

Georges Gonthier Born 1962. He completed his Ph.D. thesis on the Esterel synchronous reactive programming languages in 1988 at the Ecole des Mines in Sophia-Antipolis (France). He went on to work at the AT&T Bell Labs in Murray Hill, on transport protocols. He returned to France at INRIA Rocquencourt in 1991, where he has worked on optimal reductions in the lambda calculus, design and verification for memory management algorithms, and the design and theory of mobile computation; he also took part in the revalidation of the Ariane 5 flight software following the maiden flight failure. His current research focuses on modularity and security in distributed computation. Georges Gonthier is currently head of the Moscova project at Inria, on the programming of concurrent and mobile systems. He is also vice-chairman of the scientific steering comittee of INRIA Rocquencourt and Professor at the Ecole Polytechnique.

A.3.3 EPFL

The EPFL team consists of the programming methods group with Profs. Odersky and Nestmann as well as the distributed computing group with Prof. Guerraoui. Short vitae of the principal researchers follow.

Uwe Nestmann (partner manager) Born 1967. Ph.D. in Theoretical Computer Science (1996, University of Erlangen, DE), Assistant Professor at EPFL since October 2000. Nestmann has been working mainly in the area of concurrency theory, in particular on research subjects addressing calculi for mobile processes. His emphasis has been on the development and exploitation of the calculus' theory for the study of correctness concerning implementations of primitives for mutual exclusion and mobile objects. Nestmann enjoyed post-doc positions at INRIA Rocquencourt and at BRICS, Aalborg. In his current position at EPFL, he is building up a research group at the EPFL with focus on the investigation of cryptographic protocols and distributed algorithms by means of process calculi.

Rachid Guerraoui Born 1967. Ph.D. in Distributed Computing (1992, University of Orsay, France). Habilitation: Programming with Transactions (1996, University of Grenoble, Fr) Assistant Professor at EPFL since 01/06/1998. Guerraoui is working on distributed algorithms and distributed programming. One the one hand, his emphasis has been on the development and exploitation of object-oriented languages for distributed and transactional programming. On the other hand, Guerraoui has worked on non-blocking atomic commitment, indulgent algorithms and the X-ability theory of replication. Guerraoui has previously been affiliated with the Centre de Recherche de l'Ecole des Mines in Paris, the Commissariat a l'Energie Atomique in Saclay, and HP Labs Palo Alto. He leads the distributed programming group at EPFL (`www.d-a-c-e.com`), focusing on probabilistic algorithms, hard agreement problems and linguistic support for publish/subscribe interaction.

Martin Odersky Born 1958. Ph.D. in Computer Science (1989, ETH Zuerich). Full professor at EPFL since May 1999. Martin Odersky's research concentrates on programming languages and methods. In particular, he has worked on functional programming, type systems and type theory, concurrency, constraint programming, and issues related to the Java programming language. He has co-designed and implemented the Pizza and GJ extensions to Java. He is the author of the javac reference compiler for Java, which is distributed by Sun Microsystems.

A.3.4 KTH

Seif Haridi (project scientific co-ordinator and partner manager) Born 1953. Ph. D. in 1981, currently professor of computer systems at KTH and chief scientific advisor of SICS. He has done research on implementation of logic and constraint-based languages and on scalable cache-coherent parallel computers. He led the development of SICStus Prolog, a high-quality Prolog system, the most widely-used world-wide for education and research on UNIX workstations. He was member of the Aurora project team, which implemented the first parallel Prolog on shared memory machines. He led the Andorra Kernel Language (AKL) team and was co-designer of the concepts of AKL [27], the first existing complete concurrent constraint language. He is a co-inventor of COMA architectures, a scalable cache-coherent multiprocessor with only caches, for which he holds a patent. This concept has been taken up by SUN Microsystems [25]. His current research focuses on programming systems for multi-paradigm programming, most notably the Mozart platform, of which he is co-designer [36].

Luc Onana Alima Born 1964. Ph. D. in Computer Science (1999, Université catholique de Louvain, Belgium). Currently, he holds an ERCIM fellowship for a postdoc. He is principal investigator in a project on high availability computer systems. He has worked as teaching assistant at the Department of Computing Science and Engineering of the Université catholique de Louvain from 1992 to 1999. His research interests include: distributed computing, fault-tolerance, self stabilization, and mathematical methods in program design. He has developed two frameworks for constructing self-stabilizing systems in a systematic manner [6, 7]. His research and his results have led to the development of a useful tool for simulating concurrent systems [45].

Vlad Vlassov Born 1957. Ph. D. in Computer Science (1984, Saint Petersburg's Electrotechnical University (LETI), Russia). He was an assistant and associate professor (1985-93) at LETI, visiting researcher at Royal Institute of Technology, KTH, Stockholm, Sweden (1993-96). In 1998, he was a visiting scientist in the Laboratory for Computer Science, Massachusetts Institute of Technology (MIT). He is currently an associate professor in Computer Science at KTH. His research interests include parallel and distributed computing, programming environments, (parallel) computer architecture, performance modeling and evaluation. In 1990-93, he was a project leader of three research projects on transputer technologies performed at LETI. He developed a run-time environment for a multiprocessor system distributed shared memory parallel programming environments at KTH, a distributed layer API for the distributed object-oriented platform NUTS at KTH.

A.3.5 UCAM

Peter Sewell (partner manager) He has held a Royal Society University Research Fellowship since 1999, titled "Foundations of Distributed Programming: Wide-Area Systems and Semantics". He was awarded his Ph.D. from the University of Edinburgh in 1995, after which he worked as a research associate in Cambridge. He is the Principal Investigator of the EPSRC research project "Wide-area programming: Language, Semantics and Infrastructure Design" and has organised the Cambridge activity of the ESPRIT CONFER-2 Working Group. His recent research has studied infrastructure and reasoning techniques for distributed systems, including: mobile computation languages and infrastructure, in the Nomadic Pict project [50]; modularity and versioning; and secure encapsulation of untrusted code.

Gavin Bierman Born 1969. He received a Ph.D. in Computer Science in 1994 from the University of Cambridge. He was an EPSRC Research Fellow at the University of Cambridge Computer Laboratory (1994-1995), a Research Fellow of Gonville and Caius College, Cambridge (1995-1999), and a lecturer at the University of Warwick (1999-2000). He is currently a lecturer at the University of Cambridge and a fellow of St. John's College, Cambridge. His research interests lie broadly in the area of software systems and programming language issues, in particular establishing formal techniques that allow programmers to reason about their code. His Ph.D. established properties of intuitionistic linear logic [14], which can, amongst other things, provide a type system for an expressive semantic metalanguage [12]. As well as work on parallel [2] and object databases [56], he is currently working on Iota, a concurrent, XML-based scripting language.

Jean Bacon and Ken Moody They lead the Opera group, which will provide valuable context. Jean Bacon is a Reader in Distributed Systems in the Computer Laboratory; Ken Moody is a Lecturer. They study distributed systems, focussing on system support for emerging network-based services. Jean Bacon was Editor in Chief of IEEE Concurrency during 1999 and 2000 and is now EIC of IEEE Distributed Systems Online. They have studied asynchronous middleware (the Cambridge Event Architecture, now integrated with CORBA) and an open access control architecture for secure interworking (OASIS [63]); the two are interdependent. Feasibility studies for this technology are underway, including studies with the UK National Health Service. They are involved in the SECURE project.

A.3.6 UCL

Peter Van Roy (partner manager) Born 1960. Ph. D. in Computer Science (1990, University of California, Berkeley). French Habilitation à Diriger des Recherches (1996, University of Paris VII). Van Roy was member of research staff of DEC/PRL (1991-1994), researcher at the University of Saarbrücken (1994-1996), and visiting professor at the University of Nice/Sophia-Antipolis (1995). He is currently professor at the Université catholique de Louvain and part-time researcher at SICS. He has general research interests in programming techniques and programming languages, including advanced techniques in distributed computing [59, 28], constraint programming, and human-computer interaction [24]. For his Ph.D., he developed Aquarius Prolog, the first Prolog compiler that generates code competitive in performance with C compilers [57]. He holds one patent in graphic design. He is codesigner of the distribution model of the Mozart system [36]. Together with Seif Haridi, he is currently working on a textbook on programming concepts [58].

B Contract Preparation forms

References

[1] Groove Networks, Inc., March 2001. http://www.groove.net/.

[2] M. Afshar, J. Bates, G. M. Bierman, and K. Moody. A new general purpose parallel database system (abridged version). In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 2–8. IEEE Press, 1997.

[3] Mustaque Ahamad, Ranjit John, Prince Kohli, and Gil Neiger. Causal memory meets the consistency and performance needs of distributed applications!, 1994. ACM SIGOPS European Workshop.

[4] Mustaque Ahamad and Rammohan Kordale. Scalable consistency protocols for distributed services. *IEEE Transactions of Parallel and Distributed Systems*, 10(9), 1999.

[5] H. Ahmed, L.-E. Thorelli, and V. Vlassov. mEDA: A parallel programming environment. pages 253–260, September 1995.

[6] Luc O. Alima, J. Beauquier, A. K. Datta, and S. Tixeuil. Self-stabilization with global rooted synchronizers. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, May 1998.

[7] Luc Onana Alima. A framework for constructing self-stabilizing systems. In *International Dagstuhl Seminar on Self Stabilization*, October 2000.

[8] Philippe Altherr, Martin Odersky, Michel Schinz, Matthias Zenger, and Christoph Zenger. Funnel distribution. Available from `http://lampwww.epfl.ch/funnel`, July 2000.

[9] R. M. Amadio and S. Prasad. Localities and failures. In P. S. Thiagarajan, editor, *Proceedings of* 14^{th} *FSTTCS. LNCS 880*, pages 205–216. Springer-Verlag, 1994.

[10] A. Arora and M. G. Gouda. Closure and convergence: A foundation of fault-tolerant computing. *IEEE Transactions on Software Engineering*, 19(10):1015–1027, 1993.

[11] A. Arora and D. Poduska. A timing-based schema for stabilizing information exchange in networks. In *Proceedings of the Third International Conference on Computer Networks*, 1995.

[12] P. N. Benton, G. M. Bierman, V. C. V. de Paiva, and J. M. E. Hyland. A term calculus for intuitionistic linear logic. In *Proceedings of First International Conference on Typed lambda-calculi and applications*, volume 664 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.

[13] Martin Berger and Kohei Honda. The two-phase commitment protocol in an extended pi-calculus. In Luca Aceto and Björn Victor, editors, *Preliminary Proceedings of EXPRESS '00*, volume NS-00-2 of *BRICS Notes Series*, pages 105–130, 2000. The Final Proceedings will be published as volume 39 of ENTCS, Elsevier Science Publishers.

[14] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of *lily*, a polymorphic linear lambda calculus with recursion. In *Proceedings of International Workshop on Higher Order Techniques in Semantics, Electronic Notes in Theoretical Computer Science*, volume 41, 2000.

[15] Per Brand, Seif Haridi, Erik Klintskog, and Konstantin Popov. An architecture for distributed programming platforms. Technical report, Swedish Institute of Computer Science, 2000.

[16] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS '98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS '98, Lisbon, Portugal, March 28 – April 4, 1998, Proceedings*, volume 1378 of *LNCS*. Springer, 1998.

[17] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems Concepts and Design 2nd ed.* Addison-Wesley, Reading, Mass., 1994.

[18] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communication of the ACM*, 17:643–644, 1974.

[19] DoD Defense Modeling and Simulation Office. *High Level Architecture Interface Specification, Version 1.3*, April 1998.

[20] Sophia Drossopoulou. Towards an abstract model of java dynamic linking and verification. In *ACM SIGPLAN Workshop on Types in Compilation (TIC00)*, September 2000.

[21] Cédric Fournet and Georges Gonthier. The reflexive chemical abstract machine and the join-calculus. In *Proceedings of the 23rd ACM Symposium on Principles of Programming Languages*, pages 372–385, St. Petersburg Beach, Florida, January 21-24 1996. ACM.

[22] Cédric Fournet, Georges Gonthier, Jean-Jacques Lévy, Luc Maranget, and Didier Rémy. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR'96)*, pages 406–421, Pisa, Italy, August 26-29 1996. Springer-Verlag. LNCS 1119.

[23] Emmanuel Frcon and Mårten Stenius. Dive: A scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal*, 5(3):91–100, 1998.

[24] Donatien Grolaux, Peter Van Roy, and Jean Vanderdonckt. `QTk` – a mixed declarative/procedural approach for designing executable user interfaces. In *8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'01)*, Lecture Notes in Computer Science, Toronto, Canada, May 2001. Springer-Verlag. Short paper.

[25] Erik Hagersten, Anders Landin, and Seif Haridi. DDM–a cache-only memory architecture. *IEEE Computer*, September 1992.

[26] Fred Halsall. *Data Communications, Computer Networks, and Open Systems.* Addison-Wesley, 1996.

[27] Seif Haridi and Sverker Janson. Kernel Andorra Prolog and its computation model. In *7th International Conference on Logic Programming*, pages 31–48. MIT Press, June 1990.

[28] Seif Haridi, Peter Van Roy, Per Brand, Michael Mehl, Ralf Scheidhauer, and Gert Smolka. Efficient logic variables for distributed computing. *ACM Transactions on Programming Languages and Systems*, May 1999.

[29] Martin Henz, Sevugan Alagappan, and Tobias Müller. FriarTuck Corporation, 2001. See `http://www.friartuck.net`.

[30] Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In Jerzy Tiuryn, editor, *Proceedings of FoSSaCS 2000*, volume 1784 of *LNCS*, pages 146–160. Springer-Verlag, 2000.

[31] Michael Hicks, Stephanie Weirich, and Karl Crary. Safe and flexible dynamic linking of native code. In *Preliminary Proceedings of the Third Workshop on Types in Compilation (TIC 2000). CMU Technical Report CMU-CS-00-161*, September 2000.

[32] Mehmet Hatan Karata and Pranay Chaudhuri. A self-stabilizing algorithm for bridge finding. *Distributed Computing*, 12:47–53, 1997.

[33] Nancy Lynch. *Distributed Algorithms.* Morgan Kaufmann, San Francisco, Calif., 1996.

[34] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I + II. *Information and Computation*, 100(1):1–77, 1992.

[35] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Information and Computation*, 100:1–77, September 1992.

[36] Mozart Consortium. The Mozart Programming System version 1.2.0, May 2001. Available at `http://www.mozart-oz.org/`.

[37] H. Yu N. Chen and S. Huang. A self-stabilizing algorithm for constructing spanning trees. *Information Processing Letters*, 39:147–151, 1991.

[38] Uwe Nestmann. On the expressive power of joint input. In Catuscia Palamidessi and Ilaria Castellani, editors, *Proceedings of EXPRESS '98*, volume 16.2 of *ENTCS*. Elsevier Science Publishers, 1998.

[39] Uwe Nestmann. What is a 'good' encoding of guarded choice? *Information and Computation*, 156:287–319, 2000. An extended abstract appeared in the *Proceedings of EXPRESS '97*, volume 7 of *ENTCS*.

[40] Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Information and Computation*, 163:1–59, 2000. Also available as report BRICS-RS-99-42, Universities of Aalborg and Århus, Denmark, 1999. An extended abstract appeared in the Proceedings of *CONCGUR '96*, LNCS 1119, pages 179–194.

[41] Martin Odersky. Functional nets. In *Proc. European Symposium on Programming*, number 1782 in LNCS, pages 1–25. Springer Verlag, March 2000.

[42] Martin Odersky. Programming with functional nets. In *The School of Niklaus Wirth – The Art of Simplicity*, pages 173–200. DPunkt, 2000.

[43] Andrew Oram, Nelson Minar, and Clay Shirky. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies.* O'Reilly & Associates, 2001.

[44] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

[45] Christophe Ponsard and Luc Onana Alima. FACS: A framework for animating concurrent systems. Technical Report RT-112, Department of Computing Science and Engineering, Université catholique de Louvain, September 2000.

[46] Proksim Software Inc. *NetZ Technical Overview*, May 2000.

[47] Peter Reiher, Gerald Popek, Michial Gunter, John Salomone, and David Ratner. Peer-to-peer reconciliation based replication for mobile computers, June 1996. ECOOP 2nd Workshop on Mobility and Replication.

[48] James Riely and Matthew Hennessy. Distributed processes and location failures. In *Proceedings of ICALP '97. LNCS 1256*, pages 471–481. Springer-Verlag, July 1997.

[49] Alan Schmitt. Safe dynamic binding in the join calculus. in preparation, April 2001.

[50] P. Sewell, P. Wojciechowski, and B. Pierce. Location-independent communication for mobile agents: a two-level architecture. In *Internet Programming Languages*, Lecture Notes in Computer Science, vol. 1686. Springer Verlag, October 1999.

[51] Peter Sewell, Paweł T. Wojciechowski, and Benjamin C. Pierce. Location-independent communication for mobile agents: a two-level architecture. In *Internet Programming Languages, LNCS 1686*, pages 1–31. Springer-Verlag, October 1999.

[52] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnanan. Chord: A scalable peer-to-peer lookup service for Internet applications, 2000.

[53] Arobinda Gupta Sukumar Ghosh and V. Pemmaraju. A self-stabilizing algorithm for the maximum flow problem. *Distributed Computing*, 10:167–180, 1997.

[54] Sun Microsystems. *The Java Series*. Sun Microsystems, Mountain View, Calif., 1996. Available at `http://www.javasoft.com`.

[55] Terraplay Systems. *Terraplay CNI Manual, Version 2.0*, March 2001.

[56] A. Trigoni and G. M. Bierman. Inferring the principal type and schema requirements of an OQL query. In *British National Conference on Databases*, Lecture Notes in Computer Science. Springer Verlag, 2001.

[57] Peter Van Roy and Alvin Despain. High-performance logic programming with the Aquarius Prolog compiler. *IEEE Computer*, pages 54–68, January 1992.

[58] Peter Van Roy and Seif Haridi. *Concepts, Techniques, and Models of Computer Programming – with Practical Applications in Distributed Computing and Intelligent Agents*. 2001. Book in preparation. Latest draft always available at `http://www.info.ucl.ac.be/people/PVR/book.html`.

[59] Peter Van Roy, Seif Haridi, Per Brand, Gert Smolka, Michael Mehl, and Ralf Scheidhauer. Mobile objects in Distributed Oz. *ACM Transactions on Programming Languages and Systems*, 19(5):804–851, September 1997.

[60] V. Vlassov, M. Addibpour, and E. Tyugu. NUTS: A distributed object-oriented platform with high level communication functions. *Computers and Artificial Intelligence*, 17(4):305–335, 1998.

[61] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, 1999.

[62] Paweł T. Wojciechowski and Peter Sewell. Nomadic Pict: Language and infrastructure design for mobile agents. *IEEE Concurrency*, 8(2):42–52, April–June 2000. Invited submission for ASA/MA 99.

[63] Walt Yao, Ken Moody, and Jean Bacon. A model of OASIS role-based access control and its support for active security. In *Proceedings Sixth ACM Symposium on Access Control Models and Technologies (SACMAT)*, May 2001.