DELIVERABLE SUMMARY SHEET

Project Number:	IST-2001-33100
Project Acronym:	PROFUNDIS
Title :	Proofs of Functionality for Mobile Distributed Systems

Deliverable N. :	7
Due Date :	
Delivery Date :	

Partners Owning:	INRIA
Partners contributed :	FFCT, INRIA, Pisa
Made available to:	

Summary

The title of this deliverable, *novel type systems* (taken from the TA) reflects the theme of Task 3.1 and the expectation that at the end of Year 1 all written papers should come from this task. It is indeed true that most of the papers for the deliverable come from Task 3.1. However, some of the work in the other tasks has already produced readable papers and are therefore included in the deliverable.

Most of the papers tackle specific problems of distrubuted mobile systems, in the π-calculus or distributed versions of it, and propose solutions using type systems. These problems are: interference [LS02] (classification of the possible interferences among processes, use of types to rule out certain classes of interference); control of the migration of processes and of the access to local resources that a migrated process is granted [GF02a]; resource allocation [TZH02] (spefically, the control of the number of processes that can be phisically present on a given site at the same time); message-deliverability [RMAL02] (the fact that every emitted message has a chance of being received); the specification of guarantees on the services that a process can offer, in cases where such services can evolve and change during the execution of the processes [RRV02]; termination [San02, San02b] (the fact that a process never reaches a divergence, that is, a point in which an infinite sequence of internal steps can be produced).

All the above mentioned papers introduce novel type systems. The only exceptions are [San02, San02b] where, to get a simpler solution, the desired property is ensured by means of a combination of known types and of syntactic conditions.

A new type system is also proposed in [GF02b], where the target of the study is MetaKlaim, the distributed version of a coordination language equipped with staging mechanisms. Here, the security guarantees on a given process are maintained even when the process interact with untrusted components. To ensure this, the type checks are partly performed statically, and are partly enforced dynamically.

In contrast, [VVR02, GVR02] are about a different strand of work, namely the scalabilility of techniques based on types. The types considered here are the "session types", developped by a number of people in the π-calculus (Honda, Vasconcelos, Gay). Their scalability has been examined in the setting of object component models (for which CORBA is the best known) and of functional languages (precisely, an enriched form of lambda-calculus).

Finally, [GSV02] shows that certain controls of interferences guaranteed by some of the types mentioned above are also useful at the level of the implementations of languages.

Comments The papers [LS02, GSV02, RMAL02] are the continuation of work initiated before Profundis, summary of which had appeared at conferences. The papers contains refinements and new technical developments (in some cases quite substantial) over the earlier work, which we believe fully justify the inclusion in the Profundis output and deliverable.

Concerning the problem of termination, two papers are included in the deliverable: [San02] and [San02b]. The former, which has been written as accompanying paper to an invited talk, contains a summary of the results in the latter.

1 Contributions

We present the contributions according to the tasks they belong to.

Task 3.1: interference and access control

Classification of interference Inria and Pisa [LS02] (see Appendix 3.1.1) have studied the problem of process interference in Cardelli and Gordon's *Mobile Ambients* (MA). Two forms of interferences are individuated: *plain interferences*, which are similar to the interferences one finds in CCS and π-calculus; and *grave interferences*, which are more dangerous and may be regarded as programming errors. To control interferences, the MA movement primitives are modified; the resulting calculus is called *Mobile Safe Ambients* (SA). The modification has also computational significance. In the MA interaction rules, an ambient may enter, exit, or open another ambient. The second ambient undergoes the action; it has no control on *when* the action takes place. In SA this is rectified: any movement takes place only if both participants agree. Existing type systems for MA can be easily adapted to SA. The type systems for controlling mobility, however, appear to be more powerful in SA, in that (i) type systems for MA may give more precise information when transplanted onto SA, and (ii) new type systems may be defined. Two type systems are presented that remove all grave interferences. Other advantages of SA are: a useful algebraic theory; programs sometimes more robust (they require milder conditions for correctness) and/or simpler. In the paper, all these points are illustrated on several examples.

Access control Pisa [GF02a] (see Appendix 3.1.2), has introduced formal model capable of supporting specification of multiple dynamic security policies. The basic idea of the model is the separation between the computational mechanisms (communication and mobility primitives) and the managers, called *guardians*, that enforce the security policies. Each guardian basically defines a local security context that controls only a portion of the current state. However, a certain degree of coordination among guardians is needed to support inter-networking security policies. The coordination of guardians enables to support a rich set of security policies. In other words, the security policies are programmable distributed entities. The main features of the approach are summarized as follows

- The distinction between the computational mechanisms (processes act) and the security supports (guardians allow and deny) permits describing a variety of local security policies. Keeping the specification of actions and that of permissions separate is important also from a software engineering prospective.
- A guardian provides a unit to describe the local security policy: the set of permissions that may be granted to the agents entering, exiting and communicating in the monitored computational environment.

- Security decisions are the outcome of coordination activities among guardians which are location-aware. The coordinated security decisions control the propagation of access rights by ensuring that the permissions are in accordance with the local security policies.

Behavioural types Lisbon [RRV02] (see Appendix 3.1.3) has developed a process algebra, the Algebra of Behavioural Types. A type is a higher-order labelled transition system that characterises all possible life cycles of a concurrent object. States represent interfaces of objects; state transitions model the dynamic change of object interfaces. Moreover, a type provides an internal view of the objects that inhabit it: a synchronous one, since transitions correspond to message reception. To capture this internal view of objects a notion of bisimulation is defined, strong on labels and weak on silent ac tions. Several algebraic laws are studied and completeness results for finite types are obtained.

Resource allocation Inria [TZH02] (see Appendix 3.1.4) has studied the problem of controlling the number of processes that can migrate at the same time on a given site, on a version of the Safe Ambient calculus. For instance, a type can specify a number n for a given ambient, and then typing ensures that at any time this ambient will never host more than n subambients.

The message deliverability property requires that every emitted message has a chance of being received. In the context of the asynchronous pi-calculus, Inria [RMAL02] (see Appendix 3.1.5) has introduced a discipline of non-uniform receptivity that entails this property. Adopting this discipline requires a style of programming where resources are persistent. A general method is given to transform (in a fully abstract way) a process so that it complies with the discipline.

Secure Composition of processes The contribution [GF02b] (see Appendix 3.1.6), from Pisa, describes the design and the semantics of MetaKlaim, an higher order distributed process calculus equipped with staging mechanisms. MetaKlaim integrates MetaML (an extension of SML for multi-stage programming) and Klaim (a Kernel Language for Agents Interaction and Mobility), to permit interleaving of meta-programming activities (like assembly and linking of code fragments, and dynamic checking of security policies at administrative boundaries) and “traditional” computational activities on a wide area network (like remote communication and agent mobility). MetaKlaim exploits a powerful type system (polymorphic types á la system F) to deal with highly parameterized mobile components and to dynamically enforce security policies: types are metadata which are extracted from code at run-time and are used to express trustness guarantees. The dynamic type checking ensures that the trustness guarantees of wide are network applications are maintained whenever computations interoperate with potential untrusted components.

Task 3.2: Integration of types with operational and logic techniques

Inria [San02, San02b] (see Appendixes 3.2.1 and 3.2.2) has studied the problem of termination of mobile processes (the fact that a process never reaches a divergence, that is, a point in which an infinite sequence of internal steps can be produced). Termination is a useful property in concurrency. For instance, a terminating applet, when loaded on a machine, will not run for ever, possibly absorbing all computing resources (a 'denial of service' attack). Similarly, termination guarantees that queries to a given service originate only finite computations. Termination of a non-trivial subset of the pi-calculus is ensured by a combination of conditions on types and on the syntax. The proof of termination is in two parts. The first uses the technique of logical relations – a well-know technique of lambda-calculi – on a small set of non-deterministic 'functional' processes. The second part of the proof uses techniques of process calculi, in particular techniques of behavioural preorders.

Task 3.3: Practicality and scalability

Lisbon [VVR02, GVR02] (see Appendixes 3.3.1 and 3.3.2) has worked on the scalability of the idea of "session types", developped by a number of people in the π-calculus (Honda, Vasconcelos, Gay) to more advanced programming constructs. Two lines of work have been pursued. The first has led to a proposal for typing the behavior of objects in component models. The main motivation for this is that most component models, CORBA in particular, do not offer any support for expressing behavior al properties of objects beyond the "static" information provided by IDLs. It is shown how session types can be effectively used for describing protocols, extending the information currently provided by object interfaces. It is shown how session types not only allow high level specifications of complex object interactions, but also allow the definition of powerful interoperability tests at the protocol level, namely compatibility and substitutability of objects.

The other line has focused on a language based on the lambda-calculus with side-effecting input/output operations and recursive functions. The typing judgements statically describe dynamic changes in the types of channels, the channel types statically track aliasing, and the function types not only specify argument and result types but also describe changes in channel types. After formalising the syntax, semantics and typing rules of our language, and proving a subject reduction theorem, some possibilities for adding references, objects and concurrency are we outlined.

Task 3.4: Case studies and tool development

INRIA [GSV02] (see Appendix 3.4.1) has investigated the correctness and the implementation of an abstract machine based on some of the type systems for interference control in task 3.1. One of the main objectives here was to show that the control of interferences guaranteed by types is also useful in implementations. For this, an abstract machine for a distributed implementation of an ambient calculus is presented. The abstract machine is different from, and simpler than, previous implementations of ambient-like calculi, mainly because: the underlying calculus is typed Safe Ambients rather than the untyped Ambient calculus and therefore does not present certain forms of interferences among processes (the *grave interferences*); the logical structure of an ambient system and its physical distribution are separated. The proof of correctness and a description of a distributed implementation of the abstract machine in Java are given.

References

[GF02a] R. Pugliese G. Ferrari, E. Moggi. Guardians for ambient based monitoring. In *Proc. Foundations of Wide Area Network Programming*, ENTCS. Elseviers, 2002.

[GF02b] R. Pugliese G. Ferrari, E. Moggi. Metaklaim: A type safe multi-stage language for global computing. Technical Report Under revision for Mathematical Structures in Computer Science, 2002.

[GSV02] P. Giannini, D. Sangiorgi, and A. Valente. A distributed abstract machine for Safe Ambients. Extended and refined version of a paper appeared in *Icalp.01*, 2002.

[GVR02] S. Gay, V. T. Vasconcelos, and A. Ravara. Session types for inter-process communication. Preprint, Department of Computer Science, University of Lisbon, Campo Grande, Edifcio C5, 1749-016 Lisboa, Portugal, 2002. Submitted for publication.

[LS02] F. Levi and D. Sangiorgi. Mobile safe ambients. To appear in the TOPLAS journal. Extended and refined version of a paper appeared in *Proc. 27th POPL*, ACM Press, 2002.

[RMAL02] Grard Boudol Roberto M. Amadio and Cdric Lhoussaine. On message deliverability and non-uniform receptivity. Research report 05-2002, LIF, Marseille, France, May 2002. Accepted for pubblication in Fundamenta Informaticae.

[RRV02] A. Ravara, P. Resende, and V. Vasconcelos. An algebra of behavioural types. Preprint, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2002. Submitted for publication.

[San02] D. Sangiorgi. Types, or: Where's the difference between CCS and π? In *Proc. CONCUR '02*, volume 2421, 2002.

[San02b] D. Sangiorgi. Termination of processes Submitted, 2002.

[TZH02] D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources. In *Proceedings of the 13th Int. Conf. in Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 288–303. Springer Verlag, 2002.

[VVR02] A. Vallecillo, V. T. Vasconcelos, and A. Ravara. Typing the behavior of objects and components using session types. In Antonio Brogi and Jean-Marie Jacquet, editors, *Electronic Notes in Theoretical Computer Science*, volume 68. Elsevier Science Publishers, 2002. presented at FOCLASA'02 - 1st International Workshop on Foundations of Coordination Languages and Software Architectures.

WP3 Scientific Contributions

APPENDIX

Appendix 3.1.1
F. Levi and D. Sangiorgi. Mobile safe ambients. To appear in the TOPLAS journal. Extended and refined version of a paper appeared in *Proc. 27th POPL*, ACM Press, 2002.

Appendix 3.1.2
R. Pugliese G. Ferrari, E. Moggi. Guardians for ambient based monitoring. In *Proc. Foundations of Wide Area Network Programming*, ENTCS. Elsevier, 2002.

Appendix 3.1.3
A. Ravara, P. Resende, and V. Vasconcelos. An algebra of behavioural types. Preprint, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2002. Submitted for publication.

Appendix 3.1.4
D. Teller, P. Zimmer, and D. Hirschkoff. Using Ambients to Control Resources. In *Proceedings of the 13th Int. Conf. in Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 288–303. Springer Verlag, 2002.

Appendix 3.1.5
Gérard Boudol, Roberto M. Amadio and Cdric Lhoussaine. On message deliverability and non-uniform receptivity. Research report 05-2002, LIF, Marseille, France, May 2002. Accepted for pubblication in Fundamenta Informaticae.

Appendix 3.1.6
R. Pugliese G. Ferrari, E. Moggi. Metaklaim: A type safe multi-stage language for global computing. Technical Report Under revision for Mathematical Structures in Computer Science, 2002.

Appendix 3.2.1
D. Sangiorgi. Types, or: Where's the difference between CCS and π? In *Proc. CONCUR '02*, volume 2421, 2002.

Appendix 3.2.2
D. Sangiorgi. Termination of processes Submitted, 2002.

Appendix 3.3.1
A. Vallecillo, V. T. Vasconcelos, and A. Ravara. Typing the behavior of objects and components using session types. In Antonio Brogi and Jean-Marie Jacquet, editors, *Electronic Notes in Theoretical Computer Science*, volume 68. Elsevier Science Publishers, 2002. presented at FOCLASA'02 - 1st International Workshop on Foundations of Coordination Languages and Software Architectures.

Appendix 3.3.2
S. Gay, V. T. Vasconcelos, and A. Ravara. Session types for inter-process com-

munication. Preprint, Department of Computer Science, University of Lisbon, Campo Grande, Edifcio C5, 1749-016 Lisboa, Portugal, 2002. Submitted for publication.

Appendix 3.4.1
P. Giannini, D. Sangiorgi, and A. Valente. A distributed abstract machine for Safe Ambients. Extended and refined version of a paper appeared in *Icalp.01*, 2002.