Service-Oriented Design: The Roots

Tiziana Margaria[1], Bernhard Steffen[2], and Manfred Reitenspieß[3]

[1] Service Engineering for Distributed Systems, Universität Göttingen, Germany
margaria@cs.uni-goettingen.de
[2] Chair of Programming Systems, Universität Dortmund, Germany
steffen@cs.uni-dortmund.de
[3] Director Business Development, RTP 4 Continuous Services,
Fujitsu Siemens Computers, Munich, Germany
Manfred.Reitenspiess@fujitsu-siemens.com

Abstract. Service-Oriented Design has driven the development of telecommunication infrastructure and applications, in particular the so-called Intelligent Network (IN) Services, since the early 90s. A service-oriented, feature-based architecture, a corresponding standardization of basic services and applications in real standards, and adequate programming environments enabled flexibilization of services, and dramatically reduced the time to market. Today the current trend toward triple-play services, which blend voice, video, and data on broadband wireline and wireless services builds on this successful experience when reaching for new technological and operational challenges. In this paper, we review our 10 years of experience in service engineering for telecommunication systems from the point of view of Service-Oriented Design then and now.

1 Motivation

Service-Oriented Design has driven the development of telecommunication infrastructure and applications, in particular the so-called Intelligent Network (IN) Services, since the early 90s: IN services are customized telephone services, like e.g., 'Free-Phone', where the receiver of the call can be billed if some conditions are met, 'Virtual Private Network', enabling groups of customers to define their own private net within the public net, or credit card calling', where a number of services can be billed directly on a credit card account. The realization of new IN services was quite complex, error prone, and extremely costly until a service-oriented, feature-based architecture, a corresponding standardization of basic services and applications in real standards, and adequate programming environments came up: they set the market, enabled flexibilization of services, and dramatically reduced the time to market. Today the current trend moves toward triple-play services, which blend voice, video, and data on broadband wireline and wireless services. It builds on this successful experience when reaching for new technological and operational challenges.

In this paper, we review our 10 years of experience in service engineering for telecommunication systems from the point of view of Service-Oriented Design

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 450–464, 2005.

then and now. In particular, we aim at establishing a link to the notions used by the service-oriented programming (SO) community.

The central observation is that both communities pursue the same goals, a coarse granular approach to programming, where whole programs serve as elementary building blocks. However, they have quite a different view on what a service is and how it is organized. In the terminology of the SO-community, a service is a "nugget" of functionality (essentially a building block) that is directly executable and can be published for use in complex applications. In the telecommunication world, such elementary components are called Service-Independent Building blocks (SIBs), and the notion of service is typically used for the resulting (overall) application. In addition, in the telecommunication world the notion of feature is used to denote substructures of services (applications) that impose additional functionality (like e.g. call forwarding, or blacklisting) on the generic basic telecommunication functionality. In the IN-architecture, the basic functionality was POTS (plain old telephony service), and feature were typically only executable in the context of POTS.

It was always our point of view that some of these distinctions would disappear as soon as one lives in a fully hierarchical context, where services may themselves be regarded as elementary building blocks at a higher level of abstraction. In this scenario, which is supported by METAFrame, our service definition environment, the notion of service capture the corresponding notions of both the SO- and the telecommunication communities. Moreover, the notion of SIB simply characterizes services which cannot be refined, and the notion of feature characterizes subservices that cannot be executed on their own. The remainder of this paper is written from this unifying perspective and it focusses on the impact on formal methods to improve the service development process. This concerns in particular the idea of incremental formalization, which allows users to already exploit very partial knowledge about the service and its environment for verification. In turn, this enables a division of labour, which in particular enables the application expert to directly cooperate in the service definition process.

In the following, Sect. 2 introduces the traditional concept of services in an Intelligent Networks Architecture, Sect. 3 describes the current telecommunication perspective, and Sect. 4 presents a unifying feature-oriented description of services that goes beyond the IN understanding. Finally, Sect. 5 summarizes our conclusions.

2 Services in an Intelligent Networks Architecture

By integrating telecommunication and computer technology, the Intelligent Network concept (see [10] for an overview) helps (network) providers to make new and flexible telecommunication services available for their customers. Particularly complex examples of such services are Universal Personal Telecommunication (UPT), that combines personal mobility with the access to and from telecommunication over a unique number and account, and Virtual Card Calling (VCC), that allows subscribers to make calls from every private or public

telephone charging their own VCC account. Widely used IN services are Free-Phone (FPH, the family of 0180- or 800-services), Televoting (VOT, e.g. for selection of Saturday night movies via telephone or for the winner of the European song context), Universal Access Number (UAN, where service subscribers can be reached from anywhere under a unique universal, network-independent directory number), Premium Rate Service (PRM, which enables the service subscriber to supply any information under a unique number and against a usage fee), and Virtual Private Network Service (VPN, which allows subscribers to define a private number plan based on a public telephone exchange).

The underlying *intelligent networks* are composed of several subsystems that together implement the intended functionality. They form complex distributed systems, which require the cooperation of central computers, of databases, of the telephone network, and of a huge number of peripherals under real-time and performance constraints. In particular, the design of new services must take into account requirements imposed by the underlying intelligent network: e.g., system-dependent frame conditions must be obeyed in order to guarantee reliable execution of the new services. Figure 1 shows an abstract functional decomposition of an intelligent network, which comprehends management, control, switch and service creation units. A more detailed description of IN components and their functions can be found in [3,9].

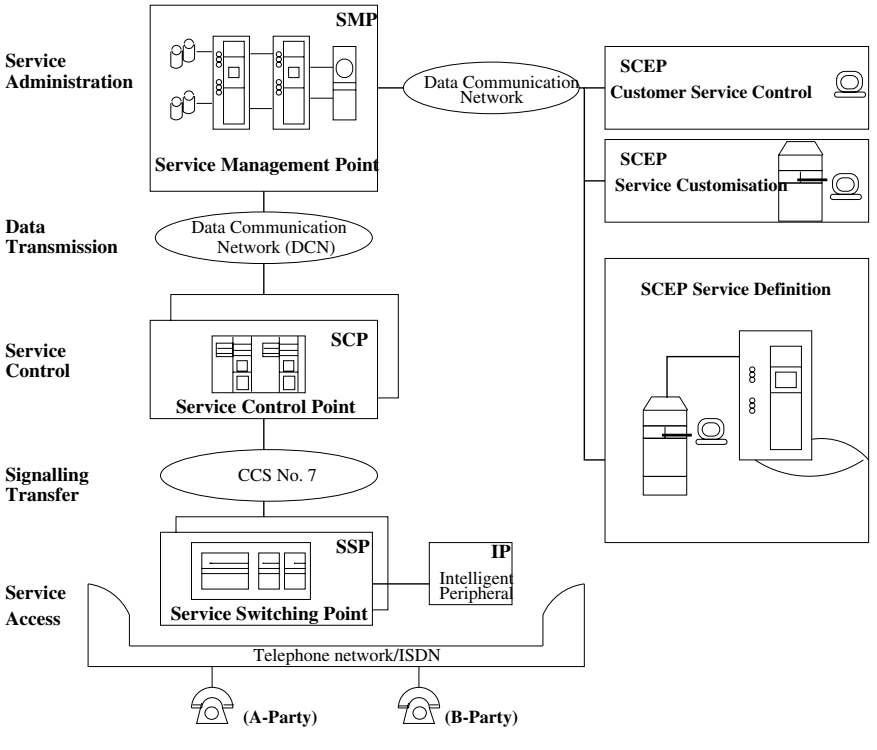


Fig. 1. Global Architecture of an Intelligent Network

- The *Service Management Point* (SMP) serves as the central component for the creation, customization and management of services and service subscribers/users. Based on a database system and on an advanced authorization system, the SMP allows the installation and administration of services and service customers by service subscribers and providers using the associated interfaces (Service Creation Environment, SCE). The SMP also provides interfaces for statistic raw data and for mass data entry.
- The *Service Control Point* (SCP) controls the Service Switching Point according to the control parameters provided by the SMP. The SCP also compiles statistical information of the calling activities and other call-related data and makes them available via the SMP for further processing. Information between SCP and SSP is exchanged via the INAP communication protocol.
- The CCS7 network is used to exchange the signalling information between the SCP and the *Service Switching Point* (SSP). The SSP sets up the call between the calling party and the called party in conjunction with the underlying telephone network (mobile or public exchange). An Intelligent Peripheral (IP) can be attached to the SSP for playing announcements or for other automatic voice services.
- The *Service Creation Environment Point* (SCEP) provides Customer Service Control (CSC), Service Customization (SC) and Service Definition (SD).
  - The *Customer Service Control* component supports the handling of the subscriber-specific service data such as parametrization of a subscribed service, modification and adaptation of service logic and statistics inquiries.
  - The *Service Customization* process serves to define which service functions and features the service subscribers are allowed to use according to their needs.
  - The aim of *Service Definition* is to establish the logic of a service and the parameters which control the processing of the service. The definition of a service begins with the creative process of detailed service specification, in which various aspects such as market requirements, technical performance (load criteria) and serviceability must be taken into account.

The complexity of the new services and the complexity of the distributed environment in which they must correctly function under strict real-time requirements of availability and performance currently make service definition intricate and error prone. In a pure direct programming based approach, the introduction of new complex services like the ones mentioned above used to take several expert years for development and testing.

A model-driven Service Definition approach has supplanted the programming style already in the '90s, supporting a reliable service design and development tailored to the specifica of the intelligent network. This has led to a much shorter time to market (days instead of months), with shortened development and testing phases. It has enabled low-cost development of high-quality services, boosting the differentiation of services to the richness we experience today.

2.1 The METAFrame Environment for Service Definition

The implementation of the Service Design Environment was based on the METAFrame® environment [29]. At that time, the service-oriented terminology was not yet defined, and the IN community defined and standardized own names for the entities they were working with. In the following we stick to that original terminology. The parallel to the modern SO-world is astonishing.

Behaviour-Oriented Development: Application development consists in the behaviour-oriented combination of Service Independent Building Blocks (SIBs) on a *coarse* granular level. SIBs are software components with a particularly simple interface. This kind of interface enables one to view SIBs semantically just as input/output transformations. Additional interaction structures can also be modelled, but are not subject to the formal synthesis and verification methods offered by the METAFrameenvironment. SIBs are here identified on a functional basis, understandable to application experts, and usually encompass a number of 'classical' programming units (be they procedures, classes, modules, or functions). They are organized in application-specific collections. In contrast to (other) component-based approaches, e.g., for object-oriented program development, METAFrame focusses on the dynamic behaviour: (complex) functionalities are graphically stuck together to yield flow graph-like structures called Service Logic Graphs (SLGs) embodying the application behaviour in terms of control. This graph structure is independent of the paradigm of the underlying programming language, which may, e.g., well be an object-oriented language: here the coarse granular SIBs are themselves implemented using all the object oriented features, and only their combination is organized operationally. In particular, we view this flow-graph structure as a control-oriented coordination layer on top of data-oriented communication mechanisms enforced e.g. via RMI, CORBA or (D)COM. Accordingly, the purely graphical combination of SIBs' behaviours happens at a more abstract level, and can be implemented in any of these technologies.

Incremental Formalization: The successive enrichment of the application-specific development environment is two-dimensional. Besides the library of application specific SIBs, which dynamically grows whenever new functionalities are made available, METAFrame supports the dynamic growth of a hierarchically organized library of *constraints*, controlling and governing the adequate use of these SIBs within application programs. This library is intended to grow with the experience gained while using the environment, e.g., detected errors, strengthened policies, and new SIBs may directly impose the addition of constraints. It is the possible *looseness* of these constraints which makes the constraints highly reusable and intuitively understandable. Here we consciously privilege understandability and practicality of the specification mechanisms over their completeness.

Library-Based Consistency Checking: Throughout the behaviour-oriented development process, METAFrame offers access to mechanisms for the verification of

libraries of constraints by means of model checking. The model checker individually checks hundreds of typically very small and application- and purpose-specific constraints over the flow graph structure. This allows concise and comprehensible diagnostic information in the case of a constraint violation, in particular as the information is given at the application rather than at the programming level.

These characteristics are the key towards a well-functioning distribution of labour, according to the various levels of expertise. The groups that crystallized were

- **Programming Experts**, responsible for the software infrastructure, the runtime-environment for the compiled services, as well as the programming of SIBs.
- **Constraint Modelling Experts**, who are experts of the protocols and frame conditions of the underlying infrastructure formulate the correctness conditions for services to properly run and interact.
- **Application Experts**, who develop concrete applications, by graphically combining SIBs into coarse-granular flow graphs. These graphs can be immediately executed by means of an interpreter, in order to validate the intended behaviour (rapid prototyping). Model checking guarantees the consistency of the constructed graph with respect to the constraint library.
- **End Users** may customize a given (global) service according to their needs by parametrization and specialization [6].

The resulting overall lifecycle for application development using METAFrame is two-dimensional: both the application and the environment can be enriched during the development process.

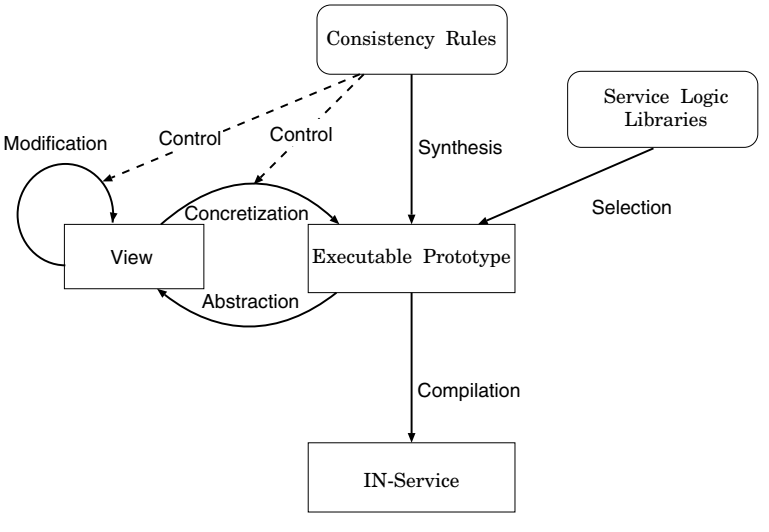


Fig. 2. The Service Creation Process

2.2 Service Definition in Practice

The Service Definition Environment is constructed for the flexible and reliable, aspect-driven creation of telephone services in a 'divide and conquer' fashion [34,4]. As shown in Fig. 2, initial service prototypes are successively modified until each feature satisfies the current requirements. The entire service creation process is supported by thematic views that focus on particular aspects of the service under consideration. Moreover, the service creation is constantly accompanied by on-line *verification*: the validity of the required features and of the executability conditions for intermediate prototypes are checked directly at design time. Design decisions that conflict with the *constraints* and consistency conditions of the intended service are immediately detected via model checking.

The novelty of this SD environment is due the impact of formal verification and abstract views on service creation [33,32]. In fact,

- Formal verification allows designers to check for *global consistency* of each design step with implementation-related or service-dependent frame conditions. Being based on model checking techniques [28], it is *fully automatic* and does not require any particular technical knowledge of the user. This simplifies the service design since sources for typical failures are detected immediately.
- Abstract or thematic *views* concentrate on the required global context and hide unnecessary details. They allow the designer to choose a particular aspect of interest, and to develop and investigate the services under that point of view. This supports a much more focussed service development, which concentrates on the design of the aspect currently under investigation. Of particular interest are error views that concentrate on the essence of a detected error.
- SIBs and reusable services are classified typically according to technical criteria (like their version or specific hardware or software requirements), their origin (where they were developed) and, here most importantly, according to their intent for a given application area. The resulting classification scheme (called *taxonomy*, [32,30,20]) is the basis for the constraint definition in terms of modal formulas.
- Both formal verification and abstract views are fully compatible with the *macro* facility of the environment. This allows developers to define whole sub-services (usually called *features*) as primitive entities, which can be used just as SIBs. Macros may be defined on-line and expanded whenever the internal structure of a macro becomes relevant: this way the SDE supports a truly hierarchical service construction [35].

The design of the taxonomies goes hand in hand with the definition of aspect-specific views, since both are mutually supportive means to an application specific structuring of the design process.

IN services soon reached sizes and complexities which demand for automated support for error detection, diagnosis, and correction. The IN-METAFrame environment encourages the use of the new methods, as they can be introduced

incrementally: if no formal constraints are defined, the system behaves like standard systems for service creation. However, the more constraints are added, the more reliable are the created services [31].

To allow verification in real time, we use finite-state model checkers [28,23] optimized for dealing with large numbers of constraints. The algorithms verify whether a given model satisfies properties expressed in a modal logic called the modal mu-calculus [18]. In the SD-IN setting:

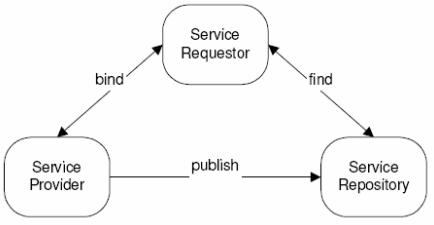
- the *properties* express correctness or consistency constraints the target IN service is required to respect. They are expressed in a natural language-like macro language, internally based on the temporal logic SLTL (Semantic Linear Time Logic, cf. [30]). This is a linear-time variant of Kozen's mu-calculus [18], which comes together with efficient verification tools;
- the *models* are directly the Service Logic Graphs, where SIB names correspond to atomic propositions, and branching conditions correspond to action names in the SLTL formulas.

Model checking a service, as shown in [35] on a concrete case, may lead to the discovery of paths in the graph that violate some constraints. When the model checker detects such an inconsistency, a plain text explanation of the violated constraint appears in a window. To ease the location and correction of the error, an abstract *error view* is automatically generated, which evidences only the nodes which are relevant to the error detection [5]. Errors can be corrected directly on the error view, and the subsequent view application transmits the modifications to the concrete model. Examples have been already discussed in previous papers [33,32].

3 The Current Telecommunication Perspective

During the last 10 years, smooth but steady transition has taken place from the switch-based IN-Architecture described in the previous section to Computer-Telephony Integrated solutions [12], and more recently to the integrated, open IT-based architectures that are being developed and deployed today as high-availability service solutions [26,24], or as distributed service integration and collaboration platforms [20]. The internet has supplanted the ISDN backbone as basis network, and the picture has reversed: cutting edge telecommunication services are being provided on an IP basic infrastructure.

The transition from the pure telecommunication scenario (which was still dominating the IN architecture) to a holistic service-oriented attitude, that includes also middleware and applications, has been actively pursued in the past years and it starts paying off. An example of this trend are the Open Specifications for Service Availability^{TM}, a collection of specifications spanning from hardware interfaces to the application level, which are available from the SAForum webpage [27]: they are increasingly influencing the way 3G telecommunication services are built. As such, they are a new success story for the feasibility and readiness of adoption of guidelines (a de facto standard) in shaping service



(a) Service Oriented Architecture (SOA)

Service Publication	Universal Description, Discovery and Integration (UDDI)
Service Description	Web Service Description Language (WSDL)
XML Messaging	Simple Object Access Protocol (SOAP)
Transport	Hypertext Transfer Protocol (HTTP)

(b) Web Service Standards Stack

Fig. 3. The Service-Oriented Architecture and the Web Services Standards Stack

interfaces with the aim of granting interoperability also in the software application domain!

The Service Availability Forum itself is a consortium of industry-leading communications and computing companies working together to develop and publish high availability and management software interface specifications. Member companies include e.g. Fujitsu Siemens Computers, IBM, Intel, Motorola, Oracle, Veritas Software, and a large number of smaller companies that offer solution that need to interwork with the platforms of the global players. The goal of the SAForum is to create complete and robust specifications to manage complex, highly-available platforms, with a specific focus on supporting high availability services. The SA Forum then promotes and facilitates specification adoption by the industry.

It is widely perceived by the participating companies that this goal by far exceeds the current aims of the SOA community: the SOA interaction structure establishes a simple, but very generic way of communication shown in Fig.3(a), and there are a number of layers for specific SOA architectures which are object of standardization efforts. SOAP [11] is a standard for XML messaging, the Web Services Description Language [8](WSDL) for service descriptions and UDDI for service repositories are de-facto standards for today's most popular implementation of service orientation: Web services [1]. The Web Services Standards Stack is summarized in Fig.3(b). A Web services implementation aims at allowing loose coupling between business partners. Since all the needed interactions can be automated, it allows also just-in-time queries to find available services.

Still, the standardization in Web services concerns interface programming languages (WSDL), data description languages (XML and derivatives like OWL-S), behaviour composition languages (BPEL4WS) and mechanisms (WSMO [37]), but not services. In other words, it tackles formats - not content, syntax - not semantics. A catalogue of service behaviours that

- service providers in one domain must provide,
- which must satisfy a given standard, but whose implementation might differ, e.g. resorting to different technologies or platforms, and

– which must be capable of interoperation, in the sense that they are guaranteed to be interchangeable,

is not yet in sight. Only such (domain-specific) standardizations will realize the full potential of service orientation in the sense of a major shift of development paradigm.

This is far away from what is customary in the telecommunication world, as described in the previous sections, and far weaker. The concept of concrete sets of *Features* and of *Services* as objects of standardization, which is natural and well accepted in the telecommunication world, is in fact still extraneous to the SOA community. For specific industrial sectors, names for categories and services are slowly becoming established and agreement is building up. Ontology-based approaches are one of the emerging technologies that are being intensively investigated within the semantic web paradigm to handle this. But they are still insufficient since they are island-solutions, not accepted standards. In the IN world, and consequently in the METAFrame environment, this standardized way of thinking was already realized in the '90s.

4 Feature-Oriented Service Description Beyond IN

There are many definitions of features, depending heavily on their context and their use. Although we too learned to know and appreciate the concept and the use of features in the context of Intelligent Networks [14,15,35], our notion of features is meanwhile more general in order to also capture a more general class of services like online, financial, monitoring, reporting, and intelligence services:

Definition 1 (Feature).

1. *A* feature *is a piece of (optional) functionality built on top of a base system.*
2. *It is* monotonic*, in the sense that each feature* extends *the base system by an* increment *of functionality.*
3. *The description of each feature* may *consider or require other features, additionally to the base system.*
4. *It is defined from an* external *point of view, i.e., by the viewpoint of* users *and/or* providers *of* services.
5. *Its granularity is determined by* marketing *or* provisioning *purposes.*

In the IN setting, the base system was a switch that offered POTS (plain old telephone service) functionality, and the features were comparatively small extensions of that behaviour. Instead, today we tend to have a *lean* basis service that deals with session, user, and role-rights management, and a very rich collection of features. Complex internet services with a strongly CSCW-oriented character and online decision support systems like the Online Conference Service described in [17], have been entirely developed this way. This brings a different perspective on the role and purpose of features.

Features were traditionally understood as local modifiers of the basic service: they were individually executed, i.e. a single feature was triggered by some event,

executed, and it retuned upon termination to the basic service. This is no longer sufficient: in order to account for complex evolutions of services, we allow in today's SD a *multilevel organization* of features, whereby more specialistic features build upon the availability of other, more basic, functionalities.

In order to keep this structure manageable and the behaviours easily understandable, we restrict us to *monotonic* features, which are guaranteed to add behaviour. Restricting behaviour, which is also done via features in other contexts (e.g. [13]), is done in an orthogonal way in our setting, via constraints at the requirements level.

Additionally, we distinguish between features as implementations and properties of feature behaviours. Both together give the feature-oriented description of services enforced in the ABC.

Definition 2 (Feature-Oriented Description).

1. *A* feature-oriented service description *of a complex service specifies the behaviours of a* base system *and a set of* optional features.
2. *The behaviour of each feature and of the basic system are given by means of Service Logic Graphs (SLGs) [15].*
3. *The realization of each SLG bases on a library of* reusable components *called Service Independent Building-Blocks (SIBs).*
4. *The feature-oriented service description includes also a set of* abstract requirements *that ensure that the intended purposes are* met.
5. Interactions *between features are regulated* explicitely *and are usually expressed via* constraints.
6. *Any* feature composition *is allowed that does not violate any constraint.*

The library of SIBs for IN services was itself standardized [16], thus leading to a well-defined set of capabilities that ensured interoperation between functionalities offered by the different vendors.

In contrast to the proposal by [7], which is still close to the IN point of view, we distinguish the description of the feature's behaviour from that of the legal use of a feature. Restrictions to behaviours are in fact expressed at a different level, i.e. at the requirements level, and they are part of an aspect-oriented description of properties that we want to be able to check automatically, using formal verification methods.

As we successively discovered, the INXpress SDE was already largely organized that way, since it was strongly influenced by more complex IN-services, which themselves were built from a combination of pre-existing individual services. Examples of such leading-edge IN services are the already mentioned UPT and VCC.

– The UPT service examined in [35] combines personal mobility with the access to and from telecommunication over a unique number and account. Using a personal identifier, a service subscriber can access telecommunication services at any terminal and use those services provided by the network which are defined in their own service profile. Personal mobility involves the

capability to identify the location of the terminal currently associated with the subscriber. Incoming UPT calls must be routed to the current destination address, and the associated charge may be split between the calling line and the UPT subscriber. Subscribers can use any terminal in the network for outgoing UPT calls, which are charged to their accounts. This requires user identification and authentication on a per-call basis. The use of the optional follow-on feature allows one authentication procedure to continue to be valid for subsequent calls or procedures. The service package can be tailored to the subscriber's requirements selecting from a comprehensive service feature portfolio.

– The VCC service allows subscribers to make calls from every private or public telephone charging their own VCC account. VCC calls are free of charge for the originating telephone line, so that cash or cards are no more needed at public telephones. After dialling the defined access code, VCC subscribers have to identify themselves by entering their virtual card number, used by the VCC service provider to determine the subscriber's account for billing purposes, and a Personal Identification Number (PIN) for personal authorization. If the virtual card number and the PIN are valid and match, the VCC user can dial the desired destination number and will be connected.

5 Conclusions

The INXpress Service Development Environment (SDE), the Siemens solution to Advanced Intelligent Networks that came out of our cooperation in 1995-1996, is a commercial product that shaped the state-of-the-art of IN-service definition in the late '90s. Presented at various international fairs (e.g. CeBIT'97), it was installed at a number of early-adopter customers (e.g. Deutsche Telekom, South Africa's Vodacom, Finnland's RadioLinja), while a number of further key contracts followed, where our SDE was a key factor for the decision of changing to Siemens technology. The success of the IN services since then has clearly demonstrated the validity and adequacy of the service-oriented way of thinking.

The same approach to service definition, composition, and verification has been meanwhile successfully applied in other application domains. With the ABC (Application Building Center) and the jABC (Java ABC)[1] we have meanwhile built internet based distributed decision support systems [19], an integrated test environment for regression test of complex CTI systems [25], a management infrastructure for remote intelligent configuration of pervasive systems [2], as well as many other industrial applications in e-business, supply chain management, and production control systems. In the area of internet-based service orchestration and coordination we have developed since 1997 the Electronic Tool Integration Platform, ETI [30], and its Web services based successor, jETI [21]. jETI is unique in providing (1) lightweight remote component (tool) integration by rregistration, (2) distributed component (tool) libraries, (3) a graphical coordi-

[1] The ABC and the jABC are the successors of the METAFrame environment.

nation environment, and (4) a distributed execution environment. Currently its application focus is on tools for program analysis, verification and validation.

The current challenge is to enhance this approach to today's Telecommunication scenarios, e.g. to 3G VoIP solutions, that blend voice, video, and data on broadband wireline and wireless services, and even beyond, reaching to the full range of unified communication and data management scenarios. This should encompasses correctness, interoperability, security, and other aspects which are not yet sufficiently supported by the standards and by the service design and validation environments. In particular, the security issue is new in this dimension to the telecommunication culture, since these concerns have been brought in by the transition to IP-based networks.

Service-level standardization efforts are still going to be the approach of choice in the telecommunication domain. Back then, the IN application consortia of competing vendors like FINNET Group, CSELT/STET, Deutsche Telekom AG, France Tlcom, Swiss Telecom PTT, Telecom Eireann, Telecom Finland Ltd., Telecom Portugal S.A. had joined forces into EURESCOM, and set the (still valid) standard of services and features in that specific domain.

This way of standardizing services and features according to their content is still infant in the area of Web services. Here the accent is still on the application indepedent infrastructure, as in the METEOR-S project [22], with initial catalogues of concrete services being developed (see e.g. the UN/SPSC service taxonomy).

We are convinced that combined approaches, that blend the flexibility of the current SO-scenario with the rigour and semantic standardization culture of the telecommunication community can be the key to the new generation of personalized, secure, and available "triple play" services. Incremental formalization and automatic verification techniques may be again the key to achieving confidence and reliability for services that interact and interoperate on a large distributed scale.

References

1. Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). Web Services - Concepts, Architectures and Applications. Springer Verlag.
2. M. Bajohr, T. Margaria: *MaTRICS: A Management Tool for Remote Intelligent Configuration of (Pervasive) Systems*, Proc. ICPS 2005, IEEE Int. Confer-ence on Pervasive Services, 11-14.July 2005, Santorini, Greece, pp. 457-460, IEEE Computer Society Press, 2005.
3. J. Biala: "*Mobilfunk und Intelligente Netze*," ISBN 3-528-15302-4, Vieweg, Braunschweig (D), 1995.
4. F.-K. Bruhns, V. Kriete, T. Margaria: "*Service Creation Environments: Today and Tomorrow*" tutorial, 4th Int. Conf. on Intelligent Networks (ICIN'96), Nov. 1996, Bordeaux (France).
5. V. Braun, T. Margaria, B. Steffen, H. Yoo: *Automatic Error Location for IN Service Definition*, Proc. AIN'97, 2nd Int. Workshop on Advanced Intelligent Networks, Cesena, 4.-5. Juli 1997, in "Services and Visualization: Towards User-Friendly Design', LNCS 1385, Springer Verlag, März 1998, pp.222-237.

6. V. Braun, T. Margaria, B. Steffen, H. Yoo, T. Rychly: *Safe Service Customization*, Proc. IN'97, IEEE Communication Soc. Workshop on Intelligent Network, Colorado Springs, CO (USA), 4-7 May 1997, IEEE Comm. Soc. Press.
7. J. Bredereke: *On Feature Orientation and Requirements Encapsulation*, in "Objects, Agents, and Features", pp. 26-44, Springer Verlag, LNCS 2975 (2004)
8. Chinnici, R., Gudgin, M., Moreau, J.-J., Schlimmer, J., and Weerawarana, S. (2004). Web Services Description Language (WSDL) version 2.0. http://www.w3.org/TR/wsdl20/.
9. B.E. Christensen, D. Underwood: "*Kommunikationsnetze werden intelligenter*," Telecom Report 14 (1991), Heft 5, pp. 262-265.
10. J. Garrahan, P. Russo, K. Kitami, R. Kung: "*Intelligent Network Overview*," IEEE Communications Magazine, March 1993, pp. 30-37.
11. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. F. (2003) SOAP Version 1.2 Part 1: Messaging Framework. http://www.w3.org/TR/soap12-part1/. W3C Recommendation 24 June 2003.
12. A. Hagerer, T. Margaria, O. Niese, B. Steffen, G. Brune, H.-D. Ide: *Efficient Regression Testing of CTI-Systems: Testing a Complex Call-Center Solution*, in *Annual Review of Communication*, Int. Engineering Consortium Chicago (USA), Vol. 55, pp.1033–1039, IEC, 2002.
13. H. Harris, M. Ryan: *Theoretical Foundations of Updating Systems*. ASE 2003, 18th IEEE Int. Conf. on Automated Software Engineering, IEEE-CS Press, 2003.
14. ITU: *General recommendations on telephone switching and signaling intelligent network: Introduction to intelligent network capability set 1*, Recommendation Q.1211, Telecommunication Standardization Sector of ITU, Geneva, Mar. 1993.
15. ITU-T: *Recommendation Q.1203. "Intelligent Network - Global Functional Plane Architecture"*, Oct. 1992.
16. ITU-T: *Recommendation Q.1204. "Distributed Functional Plane for Intelligent Network Capability Set 2: Parts 1-4"*, Sept. 1997.
17. M. Karusseit, T. Margaria: *Feature-based Modelling of a Complex, Online-Reconfigurable Decision Support Service*, WWV'05. 1st Int'l Workshop on Automated Specification and Verification of Web Sites, Valencia, Spain, March 14-15, 2005, – Post Workshop Proc. appear in ENTCS.
18. D. Kozen: "*Results on the Propositional μ-Calculus*", Theoretical Computer Science, Vol. 27, 1983, pp. 333-354.
19. Tiziana Margaria: *Components, Features, and Agents in the ABC*. In *Objects, Agents, and Features*, Revised and Invited Papers from the International Seminar on Objects, Agents, and Features, Dagstuhl Castle, Germany, February 2003, LNCS 2975, pp. 154-174, Springer Verlag, 2003
20. T. Margaria: Web Services-Based Tool-Integration in the ETI Platform, SoSyM, Int. Journal on Software and System Modelling, Springer Verlag, (available in Online First, DOI: 10.1007/s10270-004-0072-z).
21. T. Margaria, R. Nagel, B. Steffen: Remote Integration and Coordination of Verification Tools in jETI Proc. ECBS 2005, 12th IEEE Int. Conf. on the Engineering of Computer Based Systems, April 2005, Greenbelt (USA), IEEE Computer Soc. Press, pp. 431-436.
22. METEOR-S: see the project site at `lsdis.cs.uga.edu/projects/meteor-s/`
23. M. Müller-Olm, H.Yoo: *MetaGame: An Animation Tool for Model-Checking Games*, Proc. TACAS 2004, LNCS N. 2988, pp. 163-167.
24. J. Neises: *Benefit Evaluation of High-Availability Middleware*, Proc. ISAS 2004, 1st Int. Service Availability Symposium, LNCS N. 3335, pp.73-85, Springer Verlag, 2005.

25. O. Niese, B. Steffen, T. Margaria, A. Hagerer, G. Brune, H.-D. Ide: *Library-Based Design and Con-sistency Checking of System-Level Industrial Test Cases*, Proc. FASE 2001, Int. Conf. on Fundamental Approaches to Software Engineering, Genoa (I), April 2001, LNCS 2029, pp. 233-248, Springer-Verlag.
26. M. Reitenspieß: *High-Availability and Standards - The Way to Go!* Proc. ARCS Workshop 2004 - Organic and Pervasive Computing, Workshops Proceedings, March 26, 2004, Augsburg, Germany. LNI Volume 41, pp. 12-18 - Gesellschaft für Informatik.
27. The Service Availability Forum - http://www.saforum.org .
28. B. Steffen, A. Claßen, M. Klein, J. Knoop. T. Margaria: "*The Fixpoint Analysis Machine*", (*invited paper*) to CONCUR'95, Pittsburgh (USA), August 1995, LNCS 962, Springer Verlag.
29. B. Steffen, T. Margaria: *METAFrame in Practice: Intelligent Network Service Design*, In *Correct System Design – Issues, Methods and Perspectives*, LNCS 1710, Springer Verlag, 1999, pp. 390-415.
30. B. Steffen, T. Margaria, V. Braun: *The Electronic Tool Integration platform: concepts and design*, [36], pp. 9-30.
31. B. Steffen, T. Margaria, A. Claßen, V. Braun: "*Incremental Formalization: A Key to Industrial Success* ", In "SOFTWARE: Concepts and Tools", Vol. 17, No 2, pp. 78-91, Springer Verlag, July 1996.
32. B. Steffen, T. Margaria, A. Claßen, V. Braun, M. Reitenspieß: "*A Constraint-Oriented Service Creation Environment*," Proc. PACT'96, Int. Conf on Practical Applications of Constraint Technology, April 1996, London (UK), Ed. by The Practical Application Company, pp. 283-298.
33. B. Steffen, T. Margaria, A. Claßen, V. Braun, M. Reitenspieß: "*An Environment for the Creation of Intelligent Network Services*", invited contribution to the book "Intelligent Networks: IN/AIN Technologies, Operations, Services, and Applications – A Comprehensive Report" Int. Engineering Consortium, Chicago IL, 1996, pp. 287-300 – also invited to the *Annual Review of Communications*, IEC, 1996, pp. 919-935.
34. B. Steffen, T. Margaria, A. Claßen, V. Braun, M. Reitenspieß, H. Wendler: *Service Creation: Formal Verification and Abstract Views*, Proc. 4th Int. Conf. on Intelligent Networks (ICIN'96), Nov. 1996, Bordeaux (F).
35. B. Steffen, T. Margaria, V. Braun, N. Kalt: *Hierarchical Service Definition*, Annual Review of Communic., Int. Engineering Consortium, Chicago, 1997, pp.847-856.
36. *Special section on the Electronic Tool Integration Platform*, Int. Journal on *Software Tools for Technology Transfer*, Vol. 1, Springer Verlag, November 1997
37. Web Service Modeling Ontology (see `www.wsmo.org`).