Highlevel Service Handover through a Contextual Framework

Thomas Strang (1), Claudia Linnhoff-Popien (2) and Matthias Roeckl (2)
(1) German Aerospace Center (DLR), Institute for Communications and Navigation, Muenchener Str. 20, D-82234 Wessling/Oberpfaffenhofen, Germany, Email: thomas.strang@dlr.de
(2) Ludwig Maximilians University (LMU), Institute for Computer Science, Oettingenstr. 67, D-80538 Munich, Germany, Email: {linnhoff | roeckl}@informatik.uni-muenchen.de

Abstract

This paper presents an architecture for context driven handovers between highlevel (non-carrier) services. One of our key assumptions is that any change in context can cause a handover, and we discuss the several requirements to our architecture deriving from this assumption. We introduce a contextual extension to a general service model, which is the basis for our contextual framework. To demonstrate its applicability, our architecture is applied to a web service middleware exemplarily.

1 Introduction

Context is widely viewed to be very important in service discovery and service usage scenarios, especially in ubiquitous computing environments (Samulowitz et al., 2001) (Samulowitz, 2002) (Satyanarayanan, 2001). In our definition, a *context information* is any information which can be used to characterize the state of an entity concerning a specific aspect. A system is *context aware*, if it uses any kind of context information before or during service provisioning. A more detailed definition and discussion of context can be found in (Strang & Linnhoff-Popien, 2003). In (Strang, 2003) we have shown how context information can be used to reduce the amount of required user interaction with, as well as to improve the user interface of small mobile devices such as mobile phones. Even though the contextual framework described in the latter is not restricted to wireless networks, it significantly benefits from the mobility of the user and the change in location as a prime context aspect derived from the movement.

A contextual framework is required for observation, processing, delivery and management of context information to enable a system to be context aware. As part of a service provisioning architecture (middleware, platform), a contextual framework is a precondition for interoperability checks on the context level and deriving actions like handover initiations in the architecture.

In the latter we deduce the requirements on a contextual framework from some general reflections on handovers as applications of interoperability (section 2). We model them as an extension to a well established general service model in section 3. The applicability of the extended model for context driven handovers is shown in section 4, before we discuss a mediation architecture in section 5, which is demonstrated using an example implementation based on Web Services. We will summarize our paper with a conclusion in section 6.

2 Handover Procedures, Types and Conditions

Handover procedures, sometimes also called *hand-off* procedures, have been designed for and successfully deployed in several architectures. They are for instance a key element in mobile phone networks such as GSM, where the mobile terminal is “handed over” from one base station to another base station while the user is moving because of the cellular architecture of the network. Another area where handovers are common is Mobile IP, where a portable networked computing device such as a notebook may change IP domains without loosing data. This means the architecture in the background cares for the mobile device by providing inter-domain handover mechanisms.

In connection oriented systems any handover mechanism means a procedure to substitute an intermediate or even edge component of the underlying communication system while maintaining a logical connection between two interacting parties. The most important classification of handover procedures may be expressed by the scope determining the time of initiation, the effects on the logical connection in terms of transparency to the user, as well as by the amount of functional equivalence during a handover (see (Manner et al., 2001), (Kammann & Blachnitzky, 2002), (Perkins, 2000)).

2.1 Scope determining Handover Initiation

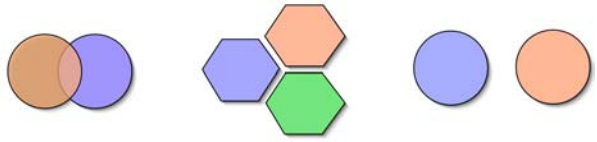
The point in time when the handover initiation is possible or must be enforced strongly depends on the topology of the underlying infrastructure.

If a simultaneous interaction with the current peer and a future peer is possible (e.g. in overlapping coverage areas of a wireless network, we call this *intersection of scopes*, see Figure 1a), a *proactive* handover is possible. Proactive means the ability of using the period in time of scope overlap to determine the optimal point of handover initiation by a given optimization function (“make before break”), e.g. by monitoring the signal strength of the current and a future peer candidate to estimate the point of initiation. Thus, a proactive handover is an expected handover (that is why (Perkins, 2000) calls this type a *predictive* handover) where some signalling can be done in advance to prepare the handover, e.g. initiation of a re-routing of data packets.

In contrast to this, an unexpected or *reactive* handover has to be performed if the intersection of scope of the current and any future peer is empty (“break before make”, see Figure 1c). Reactive handovers are usually less desirable because they require more complex algorithms to ensure continuous data delivery and minimum delay.

Kammann introduces in (Kammann & Blachnitzky, 2002) also a virtual *ideal* handover based on continuous but not overlapping scopes (*adjacent scopes*) of the current and the future peer (see Figure 1b). This kind of handover is optimal in the sense of minimum resource provisioning from

the perspective of the infrastructure as well as minimum delay and data loss rate caused by the handover.



a) overlapping b) adjacent c) disjunctive

Figure 1: Service Scopes

If a handover can be expected, a system may decide to perform a handover at time t_0 and initiate the handover at time $t_1>t_0$, by using the time between t_0 and t_1 to prepare the handover step-by-step (e.g. by establishing a parallel logical connection to the future peer), which is called a *soft* handover. Opposed to that a *hard* handover is required where a component is not able to interact with two peers simultaneously. In order to move the logical connection from the old to the new peer the component abruptly changes the settings to new values associated with the new peer. Obviously a soft handover is strongly coupled to a proactive scope scenario, whereas a hard handover is suitable for proactive and reactive scope scenarios.

(De Carolis, 2000) came up with three reasons for a handover. The first is the loss of QoS (e.g. unpredicted link drop), which s/he calls a *forced* handover. This reason requires a *fast* handover, and relates to a *reactive* scope scenario. The second reason is a possible improvement of QoS (e.g. better quality at the same cost or lower cost for the same quality), thus s/he calls this a *QoS-aware* handover. This reason relates to a *proactive* scope scenario. The last reason identified by De Carolis is based on a change of the spatial location of a component, which is thus called *location aware* handover.

2.2 Degree of Transparency

Any handover may also be characterized by the degree of transparency to the involved components. If a handover causes only minimal or even no loss of data during the handover procedure, this is called a *smooth* handover. If a handover causes only minimal or even no delay during the handover procedure, this is called a *fast* handover. A handover which is *smooth* and *fast* is called a *seamless* handover. Up to which amount of loss data a handover is called smooth, and up to which amount of additional delay a handover is called smooth, is not specified in a general purpose manner and thus system dependent.

2.3 Degree of Equivalence

Another classification of handover is done by comparing the substituted peer with the replacing peer. This is done either by comparing the technology (Tirla et al., 2002) or more general by comparing the facilities of the old and the new peer. If the technology or the facilities of the peers are similar or equal (e.g. from one GSM base station to another), this kind of handover is called *horizontal* handover. If the service offered or technology used by the new peer is different from the old one (e.g. reduced bandwidth or less detailed information), this is called a *vertical* handover.
A vertical handover is usually not a *fast* handover. A specialized version of a vertical handover is a handover where either the old or the new peer is a *parked* state. If a logical connection is vertically handed over from a peer to the parked state, all relevant session information must be stored to be able to resume the session when this logical connection is vertically handed over from the parked state to the new peer. A good example of a vertical handover is a handover between wide area communication network like GSM and short range networks like Bluetooth. Vertical handovers between these two kinds of networks have usually significant differences in bandwith, transmission costs and reliability.

2.4 Handover Protocol Options

A handover usually consists of a predefined sequence of steps which are under control of a specific instance. Thus the knowledge and the position in the network topology of the entity having the role of the handover controller (see Figure 2) has an non-negligible influence of the handover procedure.

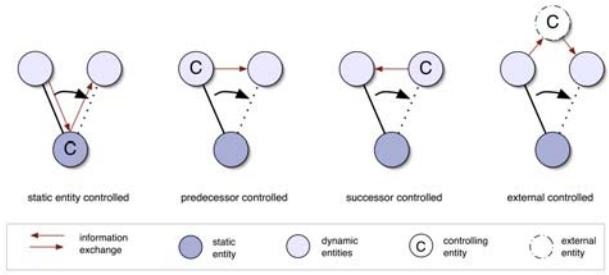


Figure 2: Handover Controller Role

The sequence of procedure steps of a handover protocol depends on the type of handover (e.g. proactive or reactive), the entity selecting the substitute candidate, the entity being the initiator of the handover procedure and other options.

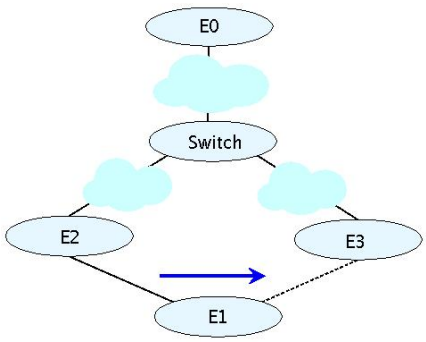


Figure 3: Handover Scenario

If we look at an example scenario such as shown in Figure 3, we can identify multiple entities involved in the handover procedure.

In the latter we assume an entity E1 (e.g. mobile network terminal) having an interaction (e.g. phone call) with entity E0 (e.g. fixed network terminal), which is currently active via entity E2 (e.g. base station). If E1 moves out of scope of E2, the infrastructure is responsible to switch the interaction in a way that after a certain point in time the interaction is active via E3.

A protocol in use during the handover procedure is outlined

in Figure 4, which shows exemplarily the protocol steps for a client initiated, predecessor controlled, hard handover protocol.

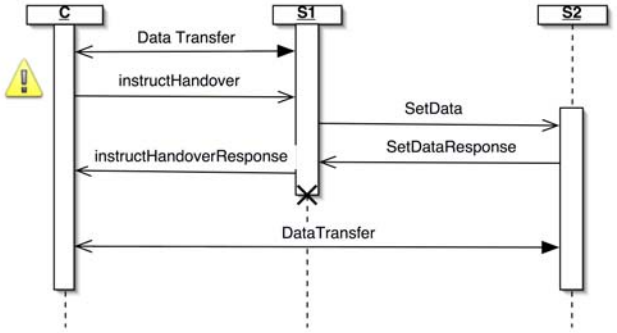


Figure 4: Client initiated, Predecessor Controlled, Hard Handover Protocol

Several constellations of the different characterizing options of a handover protocol do exist. Each constellation has certain individual requirements on the participating entities within a handover procedure. Some constellations like a reactive, successor initiated procedure do not make sense (or are simply impossible) in real client-server architectures and are therefore omitted in Figure 5, which gives an overview of the available handover options.

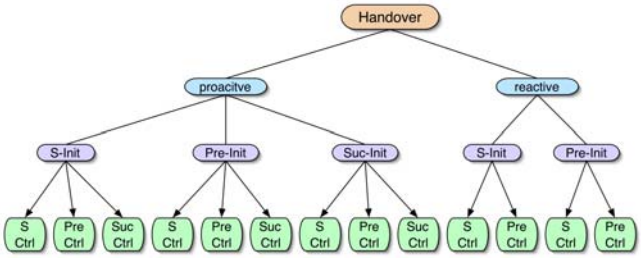


Figure 5: Handover Option Tree

For instance, a handover in GSM is a proactive, predecessor initiated (*Pre-Init*), predecessor controlled (*Pre-Ctrl*) handover (GSM, 1999), whereas a MobileIPv6 handover is either a predecessor or a successor controlled, static entity initiated (*S-Init*), proactive handover procedure (Khalil et al., 2000).

3 Highlevel Service Handover

Handover procedures are well known from (usually mobile) communication networks as an “automatic rerouting of the radio portion of a call for signal quality, traffic management, or other reasons” (ANSI, 2000). A component or series of components is transferred permanently or temporarily to another application process to “stay connected”. The service provided by a platform implementing this kind of handover is called a *connection* or *carrier* service.

All the experiences gained by conducting research on handovers for carrier services can also be very helpful when applied to general purpose services like a ticket reservation service of a travel agency or a web service of an online shop. In this sense a service may be defined as “a namable entity being responsible for providing information or performing actions with specific characteristics” (Strang, 2003). For a clear distinction from carrier services, this kind of services is called *non-carrier services* (NCS) or *highlevel services* in the latter.

3.1 Model and Framework

Garschhammer et al. introduced in (Garschhammer et al., 2001) a generic model of commonly needed service-related terms, concepts and structuring rules to describe a service from different perspectives (e.g. service view vs. implementation view). Their model is intended to analyze, identify and structure the necessary actors and the corresponding inter- and intra-organizational associations between these actors. In this model the actors are grouped in either the *customer domain*, or the *service provider domain*. Structural elements which cannot be associated to any of these two domains are called *side independent*. In the model's *service view* these elements “build” the service in an abstract manner, thus we prefer to call this set of elements the *abstract service*.

The MNM service model has been designed primarily with network management tasks and carrier services in mind. But due to the level of abstraction this model can be perfectly applied to highlevel (non-carrier) services. Moreover, the model fits direct service usage approaches (client ↔ server) as well as intermediate service usage approaches (client ↔ middleware ↔ server). In the latter one a middleware component “fulfills” an abstract service, i.e. it behaves like a client towards the service provider domain, and behaves like a service provider towards the service customer domain by providing at least a service access point.

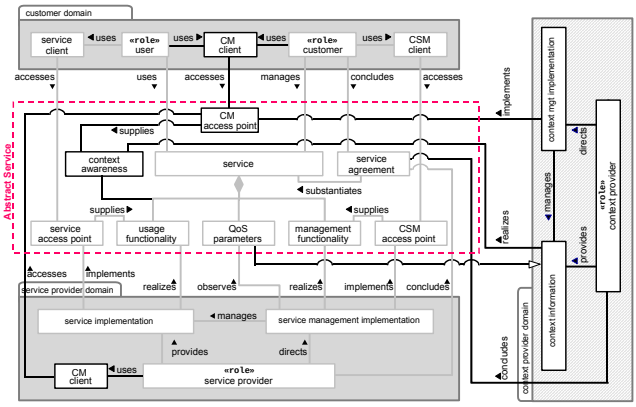


Figure 6: MNM+CE Model with emphasized Context Extension

To be able to describe contextual dependencies and issues of context provisioning with the model in a similiar and consistent way, we extended the MNM service model with a *context provider domain* (see Figure 6 on the right). This domain groups the actors responsible for managing the context observation, context processing and distribution as context information.

The context provider is not yet another service provider. Its extraordinary position is caused by, among others, the fact of being involved as a third party service provider in an interaction between the customer domain and a set of service provider domains simultaneously. The context management implementation offers a *context management access point (CMAP)* to give access to the context to both other domains in the model (customer domain and service provider domain) to enable context-aware services and context-aware service usage. A service client, a service

provider or even a middleware component may determine the entities relevant for a specific task by an interface provided by the CMAP, which causes the associated context management implementation to deliver context information in an adequate manner.

In the latter we refer to this contextual extended MNM service model as *MNM+CE service model*. In this model, carrier services are a specialized derivative of highlevel services, and the Quality-of-Service (QoS) parameters describing a specific service instance are derived from context information in the model. By modelling QoS parameters as a specialization of context information, handover procedures of carrier services can be modelled as a subset of handover procedures of highlevel services.

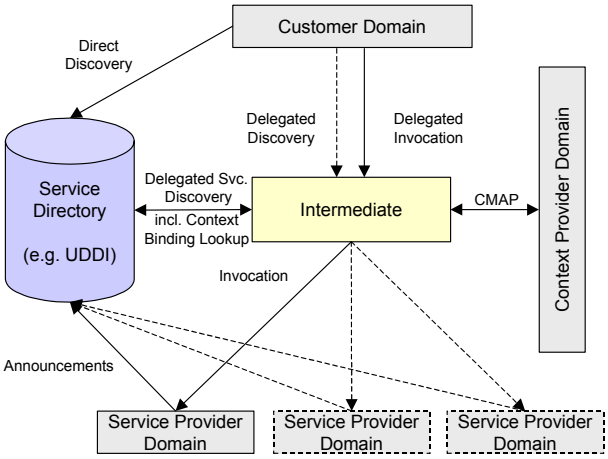


Figure 7: Middleware hosting Intermediate

This MNM+CE model has been the background when designing our framework's architecture as outlined in Figure 7. A central component of this architecture is the middleware hosting a intermediate which is able to run the abstract service introduced with the model. The middleware is, among others, responsible for determination of interoperability of any two interacting components, which is described in the following section.

3.2 Interoperability and Service Handover

Service interoperability can be determined on *signature*, *protocol*, *semantic* and *context* level. On any of these levels, interoperability has two perspectives: *compatibility* and *substitutability* (see Figure 8). One of any service platform's main tasks is to provide mechanisms to determine interoperability in terms of compatibility and substitutability.

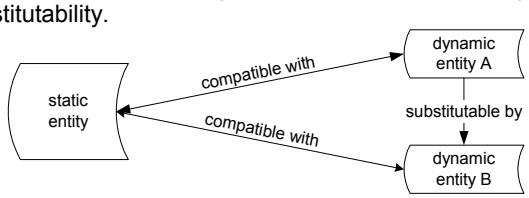


Figure 8: Two Perspectives: Compatibility and Substitutability

Compatibility is required for any communication between two interacting entities (user/service or service/service) and means a *shared understanding* of the communication primitives exchanged including their expected sequence, the semantic meaning of the single primitives and the service as a whole, as well as the context of the interaction. Compatibility is a symmetric relation, meaning if A is compatible with S, then S is also compatible with A. Usually the platform enables compatibility by supporting or even providing definition languages for all four levels to create specification documents making the shared understanding explicit and comprehensible for any entity involved in an interaction.

Substitutability refers to a relation between two entities (A and B) concerning an interaction with a third entity (S), and means the ability of entity B to replace entity A by providing equivalent functionality and impact with respect to the interaction with S. In contrast to compatibility, substitutability is an asymmetric relation, meaning substitutability of A by B does not imply substitutability of B by A. Substitutability of A by B requires compatibility between A and S as well as B and S, but not necessarily compatibility between A and B – this would be required only when A and B have direct communication (e.g. for exchanging data, which has been previously processed by A and shall be continued to be processed by B). Moreover, the shared knowledge between A and S may differ from the one between B and S (e.g. specification of an email service in the first case and a SMS service in the latter). A service framework determines substitutability usually in a two-step procedure during a up-and-running or interrupted communication between a static entity S and a dynamic entity A after detecting the necessity of a substitution: First, search for a candidate dynamic entity B being compatible to S, and second, check for substitutability of any found candidate entity B. A handover in a service framework is the activity possibly performed by the platform to substitute one service entity by another service entity in a third step after determination of substitutability. Again, the service platform usually provides a definition language to specify the shared understanding about substitutable services w.r.t. semantics and, in our case, contextual facts.

In analogy to what has been described on carrier services in section 2.3, a horizontal inter-provider handover may be performed by the platform, if and only if two non-carrier services are interoperable on all four levels. If a NCS is not interoperable to another service on at least one of the four levels, the platform may perform a vertical inter-provider handover, if there is such an option available. Intra-provider handovers are provider-dependent and are usually not handled by the platform (but may be supported by and delegated to the platform).

4 Context Driven Handover

Assume the following example: A user currently being in a city like Munich is using an electronic timetable service of the public transport provider of that city. If the user moves to another city, the platform should detect this and perform a (horizontal) handover to a proper equivalent service for the new city. In other words, if the user moves out of the scope of a service (predecessor), the platform must search for a service which is compatible to the client and may be a substitute (successor) of the previous service. In this case the new service may be a substitute of the old one, if the scope of the new service does cover the new city. One of the requirements to be able to perform a horizontal handover from the predecessor service to the successor service are *congruent contexts* as part of the equivalence conditions. If the platform is unable to identify another

service with a matching scope, it may perform a vertical handover (e.g. pause the service usage or switch to a service with less accuracy, larger refresh periods or static content). The scope condition in the example above is a context condition, because it depends on the current location of the user, which is a context information. Determination of compatibility between a service and the client is usually done by the platform during service discovery primarily, whereas determination of substitutability between a service and another service is primarily done by the platform during service execution.

If a change in context results in a handover initiation we call this a *context driven handover*. One of our key assumption is that

any change in context can cause a handover.

From this perspective the three reasons for a handover described by De Carolis (see section 2.1) - drop of QoS, increased QoS and location awareness - are covered by the same model, making this distinction unnecessary.
A key accessor to context information in any context-aware system is a well designed model to describe contextual facts and contextual interrelationships. In the following we will wrap up our model (section 4.1) which facilitates the understanding of the concepts used to determine contextual interoperability and in particular enabling us to describe the contextual conditions (section 5) which leads to a handover. These conditions are evaluated in our testbed by the use of ontology based inferencing (Barners-Lee et al., 2001), which requires a projection of the model to a specific ontological syntax (Uschold & Grüninger, 1996).

4.1 ASC Model

Our *Aspect-Scale-Context (ASC)* model is named after the core concepts of the model, which are *aspect*, *scale* and *context information*, see Figure 9. Think of an *aspect* as a concept of semantically individual classification. As such, an aspect aggregates related, into one another mappable *scales* of discrete or continous values or symbols. A scale is an unordered set of objects defining the range of valid *context information*. In other words, a valid context information with respect to an aspect is one of the elements of the aspect's scales. For instance the aspect “GeographicCoordinateAspect” may have two scales, “WGS84Scale” and “GaussKruegerScale”, and a valid context information may be an object instance created with `new GaussKruegerCoordinate ("367032", "533074")` in an object oriented programming language like Java. Scales based on primitive datatypes like scalars instead of objects are captured by corresponding wrapper classes. Thus a valid context information of the aspect “SpatialDistanceAspect” with the given scales “MeterScale” and “KilometerScale” may be an object instance created with `new Integer(10)`.
Each scale is *constructedBy* one class of context information. All scales within one aspect are constrained by the ASC model in a way, that there must exist a mapping function called *IntraOperation* from one scale to at least one other of the already existing scales of the same aspect. Like that, it is possible to access every scale from every other scale of the same aspect by a series of IntraOperations. In other words, a new scale of an aspect may be virtually constructed by providing an IntraOperation from an existing scale. This allows to build multiple related scales by providing different IntraOperations representing different scaling factors (“nautical miles”, “km” or “m” for a “SpatialDistanceAspect” aspect). Scales which require access to scales of one or more other aspects can be defined using *InterOperations*. An example for such a scale would be “KilometerPerHourScale” of a “SpeedAs-pect” aspect. This scale can be defined using an InterOpe-ration with two *Parameter*, delta_s and delta_t, where the parameter delta_s is from an aspect “SpatialDistance-Aspect” and delta_t is from an aspect “DurationAspect”.

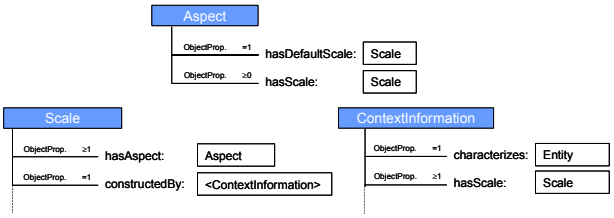


Figure 9: Toplevel Concepts of ASC Model

Due to the fact that a scale is an *unordered set* of context information instance objects, there may be no relative sort order between the context information inherently given. Therefore we introduced the *MetricOperation* which may be used to compare two context information instance objects of the same scale in an implementation-defined manner to see if they match or what their relative sort order is by returning either the first or the second parameter. Thus the return value indicates the ordering of the two objects.
Information about the signature of any InterOperation, IntraOperation or MetricOperation is available in the signature specification pointed to with the property *identifiedBy*, e.g. an operation within a WSDL file or an AtomicProcess within a DAML-S grounding.
Each context information has got an associated scale defining the range of valid instances of that type of context information. Context information characterizing the content of another context information is a meta information and thus a context information of higher order and expresses the quality of the lower order context information. Our Context Ontology Language (see section 4.2) includes already a set of standard quality aspects such as a *minimumError*, a *meanError* and a *timestamp*, but any other kind of context information characterizing the quality of another context information may be assigned to the context information of interest using the *hasQuality* property of the ASC model.

4.2 HoT and CoOL

Our Context Ontology Language (CoOL) is mainly a projection of our ASC model into a XML based markup language, supplemented by a catalog of instance documents (*ontology artifacts*) to be used when describing contextual facts. The language is defined using either DAML+OIL or OWL, which are both ontology languages based on XML. Thus, the correctness of statements defined in CoOL may be validated using any appropriate DAML+OIL or OWL validator. In our Web Service based prototype system, the *handover testbed* (HoT, see section 5), these documents are converted to F-Logic documents after validation of any CoOL document in the expressed manner, loosing some of the expressiveness of DAML+OIL respectively OWL, but enabling a much more efficient use towards a backend component of the context provider domain: the OntoBroker reasoner (Decker et al., 1999). F-Logic, for instance, is much more appropriate for specifying

relevance conditions (what are the conditions for considering an entity and/or specific context information to be relevant).

5 Applied Reasoning in Handover Testbed

Our prototype system is based on a Web Service architecture, where we extended the existing protocols and language modules to describe the shared understanding on the context level with elements of CoOL, as well as added architecture elements to evaluate interoperability as described in the previous sections, enabling us to perform context driven highlevel service handovers.

Several different handover options discussed in section 2 are addressed within the prototype. For instance, as you can see in Figure 10, we implemented the role of the Handover Controller (see section 2.4, which e.g. estimates an optimal time of handover initiation, as well as performs e.g. a session management during the handover protocol) at the intermediate, which is an implementation of the abstract service view handler of the MNM+CE model (see section 3.1) itself.

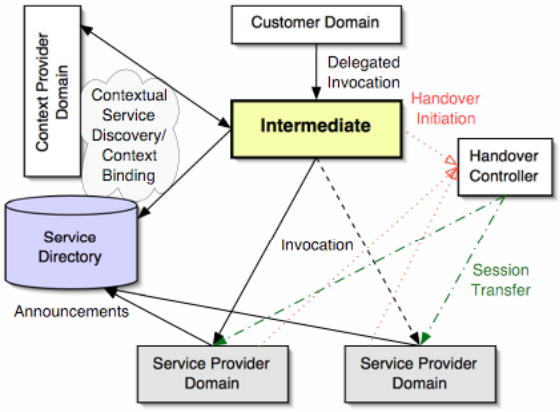


Figure 10: Handover Controller Implementation

Our implementation evaluates guarantees (which we call *context obligations*) of a service provider to maintain its state w.r.t. a specific aspect within some limits expressed by the obligation. As such, an obligation is a scale for a context information characterizing an entity which is a service. The limits defined within an obligation constrain the context information to a certain subset of a scale of a specific aspect. To be able to constrain a scale with predicates like “LessOrEqual”, a MetricOperation must be provided within the obligation, which defines the relative sort order on a scale. The context information may be further constrained by an arbitrary amount of context information of higher order, each based on a quality aspect. We use a range of obligations to characterize a service from a context perspective. Among them are for instance *GeographicScopeObligation*, *TimeScopeObligation* and *LegalScopeObligation*, which we used for testing purposes. The necessity of a handover is detected by the intermediate in cooperation with the ontology reasoner: The intermediate searches for a successor service if the reasoner indicates a near-end-of-scope or out-of-scope context condition. If a congruent (see section 4) candidate is found, it initiates a horizontal proactive, client controlled hard handover of our test service, using the reasoner again to identify a proper data adaptation path. If no candidate is found, a vertical handover to a parked state simulating service is performed alternatively.

6 Conclusion

In the previous sections we introduced an architecture enabling context driven handovers between services. This architecture is based on a contextually extended general service model. An analysis of different handover protocol options has shown the implications to the customer and service provider domains. Our approach uses ontology based reasoning to determine the necessity of a service handover as well as additional adaptation information from contextual facts based on the ASC model. The applicability of our architecture is demonstrated within a Web Service based Handover Testbed.

7 References

ANSI (2000), *American National Standard for Telecommunications - Telecom Glossary 2000*, 2000.

Barners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. *Scientific American*, Vol. 284, No. 5, pp. 34-43.

De Carolis, A. (2000), *QoS-Aware Handover for Mobile IP: Secondary Home Agent*, November 2000.

Decker S. et al., 1999. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. *Semantic Issues in Multimedia Systems*, pp. 351-369, Boston, USA.

Manner, J., Kojo, M., Suihko, T., Eardley, P., Wisely, D., Hancock, R. and Georganopoulos, N. (2001), *Mobility related terminology*, Juli 2001.

Garschhammer, M., Hauck, R., Kempter, B., Radisic, I., Roelle, H. and Schmidt, H. (2001), *The MNM Service Model - Refined Views on Generic Service Management*, in Journal of Communications and Networks, Vol. 3, December 2001, pp. 297 – 306.

GSM (1999), *Handover Procedures*, GSM 03.09, 1999.

Kammann, J. and Blachnitzky, T. (2002), *Split-Proxy Concept for Application Layer Handover in Mobile Communication Systems*, in Proceedings of the 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm, September 2002.

Khalil, M., Akhtar, H., Pillai, K. and Qaddoura, E. (2000), *AAA Interface for IPv6 Handoff*, October 2000.

Samulowitz, M. (2002), *Contextadaptive Service Usage in Ubiquitous Computing Environments*, PhD thesis, LMU Munich, Germany, June 2002.

Samulowitz, M., Michahelles, F. and Linnhoff-Popien C. (2001*), Capeus: An architecture for context-aware selection and execution of services*, in “New developments in distributed applications and interoperable systems”, Kluwer Academic Publishers, Krakow/Poland, September 2001, pp. 23-39

Satyanarayanan M. (2001), *Pervasive Computing: Vision and Challenges*. IEEE Personal Communications, Vol. 8, pp. 10-17.

Strang, T. (2003), *Towards Autonomous Context-Aware Services for Smart Mobile Devices*,' in LNCS 2574: Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003), Springer, Melbourne/Australia, January 2003, pp. 279-293

Strang, T. and Linnhoff-Popien, C. (2003), *Service Interoperability on Context Level in Ubiquitous Computing Environments*, in Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w), L'Aquila/Italy, January 2003.

Tirla, O., Munoz, A.B., Schoo, P. and Tessier, S. (2002), *Accounting management in heterogenous mobile access networks: the mind approach*, 2002.

Perkins, C.E. (2000), *Fast Handovers for Mobile IPv6*, November 2000.

Uschold M. and Grüninger M., 1996. Ontologies: Principles, methods, and applications. *Knowledge Engineering Review*, Vol. 11, No. 2, pp. 93-155.