Service Orchestration with Generic Service Elements

Ioannis Fikouras
Bremen Institute of Industrial Technology and
Applied Work Science (BIBA),
Hochschulring 20,
D-28359 Bremen,
Germany
fks@biba.uni-bremen.de
http://www.biba.uni-bremen.de

Friedhelm Ramme
Ericsson Eurolab Deutschland GmbH,
Service Networks &
Applications Technologies Research
Ericsson Allee 1, D-52134 Herzogenrath,
Germany
Friedhelm.Ramme@ericsson.com
http://www.ericsson.com

Abstract
The advent of the mobile Internet depends on a corresponding evolution of new type of services vital for providing concrete added-value to users. In spite of the success of mobile telephony and Internet applications, as well as the steadily increasing popularity of mobile Internet access, the take-up of the Mobile Internet vision is still behind expectations. The authors identify the steep learning curve and the high level of technical requirements associated with mobile environments as one of the main issues preventing developers from producing mobile services and applications at the same rate that applications are made available in the Internet. This paper introduces the novel concept of “Generic Service Elements” (GSEs) in an effort to boost the rapid development of system aware applications and services. The development of high-value composite services, utilizing a specific form of Variant Configuration, is described. An example how such a method can be applied in the mobile context is given.

Keywords
Service Composition, Orchestration, Distributed Middleware, Value-added Services, Knowledge-based Variant Configuration

1. Introduction

Significant technological advances in recent years in the areas of mobile devices and wireless communications were accompanied by infiltration of all aspects of our lives by all sorts of new Internet based services. Mobile communications and the Internet have been the two major drivers of consumer demand for telecommunication services in the last decade of the twentieth century [1]. Mobile phones are already pervasive in all major developed economies as well as in an increasing number of developing ones. The average mobile penetration in Europe in 2002 reached 72.4 per cent [2] and internet penetration in EU homes reached 38 per cent in December 2001 [3]. In November 2001, almost 50 per cent of the population over 15 years used the Internet either stationary or mobile. The rate of Internet take up by businesses is far higher at almost 90 per cent of enterprises with more than ten employees.

Furthermore it is forecasted that by 2005 an increasing portion of Internet users will be using wireless devices such as web-enabled cell phones and PDAs to go online and the number of worldwide Internet users will nearly triple to 1.17 billion [4]. By that time, a variety of different wireless network platforms with different properties, capable of transporting Internet traffic will be available [5]. In addition, the turn of operators towards license-free frequencies [6] and their eventual congestion will lead to the realisation of alternative dynamic network structures, namely Internet compatible, multi-hop, ad-hoc networks. This development will be assisted by the rise of new mobile devices capable of maintaining various access interfaces that will allow simultaneous connectivity over a range of providers and technologies [7] [17].

In spite of repeated past forecasts of the contrary, mobile Internet access today accounts for less than 10 per cent of those online globally even though the number of mobile users greatly exceeds the number of Internet-users [8]. This clearly indicates that the advent of the mobile Internet is depending on an adequate evolution of new types of services, vital for providing concrete added value to users [9] and thereby resulting in motivation to adopt the new technologies. In an environment with the potential for true global user mobility, we will experience a paradigm shift from stationary to mobility aware services and from a provider or operator-centric service model to a user-centric view [10]. Main characteristics of the new paradigm are:

• User centricity: Use of all available means to free the user from established restricting structures in order to offer the best possible service.

• Mobility awareness: Use of all data concerning the user’s position, movement and direct environment or context for the provision of services.

The watchword of this evolution is "The customer is sovereign and he knows what he wants: It is not your product; it is his/her time." [11] Hence, providers of any product or service will try to approach users by providing solutions based on new user-centric business models built on the premise of transparent service provision. The main characteristic of this new generation of services will be the empowerment of the user achieved through two opposite trends. On the one hand technical complexity is shifted towards the mobile devices (more resources on the terminal enable greater functionality), while on the other organisational complexity is transferred from the user to the provider. That is, users will be in the position to express their wish through an appropriate interface and receive the product or service needed without having to cope with the actual details involved in locating, choosing, composing, organising, ordering or delivering it.
Due to considerations related to advanced business models (i.e., the user as service developer/provider) an abstraction and interworking layer integrating such technologies in a manner tailored to the creation of end-user services and applications is required. In contrast to traditional approaches the authors expect that a future mobile Internet middleware layer will consist of fully distributed and self-organizing system-and application services. These services will be composed of coexisting and loosely coupled components.

This paper introduces the concept of “Generic Service Elements” (GSEs) as instantiation of corresponding loosely coupled service components. The paper details a service composition approach for the orchestration of Generic Service Elements for Mobile Services. It is envisioned that together with advanced service orchestration methods the introduction of the GSE system design concept will lead to the required paradigm shift in the Mobile Services area.

2. GSE Composition in Mobile Environments

GSEs are reusable service building blocks plugged into (hosted by) an arbitrary Service Execution Environment (SEE) and composed internally of two parts, (a) a core implementing the actual service logic required to provide the service in question, (b) a set of mandatory functionality allowing for deployment, discovery, orchestration & management of GSEs through a standardised interface. GSEs are organised in a highly distributed architecture without any inherent need for centralized technical or business control. This architecture cuts across the whole service platform and even goes beyond that, spreading across many physical and logical components like networks, terminals, moving vehicles or independent business entities. Consequently new technologies that will enable providers to make elementary and composite services transparently available are necessary.

Value-added composite services are built leveraging elementary assets available in telecommunication networks. Fig.8 illustrates how three sources of location information (Cell ID, MPS and GPS) are gathered independently and provided as unified location data with related error vector by a Generic Service Element aggregating these services and providing a unified positioning service called “Location GSE”. The Location GSE can deliver this information directly to an end-user application or feed it to a “Profile Data GSE” which once again can provide access to a single or multiple sources of information. Together with further profile data, like the subscriber data provided by the HSS/HLR Elementary service, the Profile Data GSE provides its information to two further applications.

The Location GSE illustrates how reliable services can be provided to applications even in case of major failures in one of its elementary services.

3. Knowledge-Based Variant Configuration

The proposed scheme for the creation of Composite Services is based on the concepts of Knowledge-based Variant Configuration and more specifically the “Lean Configuration” variation of the Object Oriented approach. Both are briefly illustrated in the following sections.

Knowledge-based Variant Configuration [13] is a process where complex products are composed out of elementary components. A Configurator is an expert system that supports this process and thereby uses predefined goals as well as expert knowledge. Design goals can be constraints, functional requirements, predetermined components or various quality criteria [14]. Such systems do not follow a single predefined method, but rather a strategy based on a series of small steps, each step representing a certain aspect or assumption leading to the configuration of the composite service. Configuration is therefore considered as the solution to a single exercise and not the solution to a whole problem or problem class that has first to be methodically analysed (see Fig.1). This implies the following:

- The set of all possible solutions is finite.
- The solution sought is not innovative, but rather is a subset of the available parts.
- The configuration problem is known and well defined.

3.1. Object-Oriented Variant Configuration

Object-oriented Variant Configuration is based on the concept of iterative composition of the final product out of a set of elementary components that have been previously organised according to a product data model into a structure, known as the object hierarchy that contains all knowledge related to the product in question. The relationships between components and how they fit together are described with the help of constraints.

Constraints are constructs connecting two unknown or variable components and their respective attributes, which have predefined values (taken from a specific knowledge domain). The constraint defines the values the variables are allowed to have, but also connects variables, and more importantly, defines the relationship between the two values [15]. In other words, constraints contain general rules that can
be applied to make sure that specific components are put together in a correct fashion without having to specify any component-related rules or calculations [15]. The constraint satisfaction problem is defined as follows [16]:

- There is a finite set of variables \( X = \{ x_1, \ldots, x_n \} \).
- For each variable \( x_i \), there exists a finite set \( D_i \) of possible values (its domain).
- There is also a set of constraints, which restrict the possible values that these variables are allowed to take at the same time.

The object hierarchy contains all relevant objects and the relationships between them in an “is-a” relationship that defines types of objects, object classes and subclasses, and their properties. The configuration process creates objects on the basis of this information according to the products being configured. In one specific hierarchy (as depicted in the following figure for the configuration of automobiles, classes for specific car types (i.e. coupé, minivan, etc.) are connected by “is-a” relationships to the main “car” class. This hierarchy also allows the breakdown of a product into components with the help of further “has-parts” relationships. These “has-parts” relationships are the basis for the decision-making process employed to create new configurations. An example of such a relationship would be the relationship between a chassis and a wheel. A chassis can be connected to up to four wheels in a passenger car, but the wheels are represented only once, with an appropriate cardinality (see Fig.2).

![Figure 2. Object hierarchy of a product domain](attachment:image)

The configuration process itself is composed of three phases [18]:

- Analysis of the product in order to define possible actions.
- Specification of further configuration actions.
- Execution of specified actions.

These actions are:

- **Disassembly of the product into its components.** This is meant to reduce the complexity of the problem and create a large number of smaller objectives in the manner of conventional top-down specification.
- **Assembly of components, integration and aggregation.** This step creates a product out of its components in a bottom-up manner.
- **Creation of specialised objects.** Object classes are specialised through the definition of subclasses.
- **Parameterise objects.** Define attributes and parameters for the specified objects that can be used for the application of constraints or other configuration mechanisms.

The greatest hurdle to be resolved when creating new configurations is the fact that the software is required to make decisions that are not based on available information. Such an action can possibly lead to a dysfunctional composition or simply to a combination that does not conform to user requirements. In this case all related configuration steps have to be undone (an activity called backtracking) in order to return to a valid state. The longer it takes for the configuration to detect that a mistake has been made, the more difficult it is to correct the error in question [14].

Object-oriented configuration is a modern approach to variant configuration suitable for complex structures in arbitrary product domains. Furthermore this approach allows for simplified maintenance of established service repositories through clear hierarchical structures.

### 3.2. Lean Configuration

The “Lean Configuration” [19] approach (developed in the course of the INTELLECT IST-1999-10375 Project) to variant configuration is object oriented, but reduces the configuration process to a search problem by eliminating the complex, computationally intensive and error-prone first two steps of object oriented configuration thereby eliminating the need for back-tracking.

The reduction in complexity is realised by the usage of a hierarchically structured product data model containing all the components available for composition. These components are categorised primarily according to their functionality, thus making the search for compatible components relatively simple and efficient. Additional categorization criteria can be implemented for supporting more complex configuration requirements. Furthermore a number of correctly configured, complete compositions are offered as the basis for interactive configuration. As long as the user uses a pre-configured composition as a template for the new variant, the configuration process can be transformed into a search problem and, specifically, a search for the next component to be exchanged. The Configurator supplies the user with lists of components that (a) are comparable to the service being exchanged and can be safely used in place of the component to be removed or (b) are compatible to existing services and can be safely added to the configuration. This mechanism ensures that the configuration is constantly in a correct state.

### 4. Service Orchestration

Based on the variant configuration principles presented in the previous sections the need for a hierarchically structured data model describing all the components involved in the service composition process is identified.

#### 4.1. Services Data Model

The proposed Composite Services Data Model divides services conceptually into two categories, Elementary Services and Composite Services. Elementary Services represent a specific instantiation of a service and contain all data needed to describe it. They inherit a set of general attributes available to all services from the Abstract Service data-type. Composite Services consist of groups of Composite Service Components derived individually from Elementary Services. Composite Service Components inherit their attributes from Elementary Services, as well as attributes related to workflow management from the Abstract Service Component datatype. The purpose
of these components is to describe the exact composition of the service, including data on which components are connected, by what Interfaces and in what order. Connections between such components are described additionally with the help of Connection components (see Fig.3). Service Categories descend from the Abstract Category datatype and implement a means of grouping elementary services into sets according to functional criteria. These sets are meant to simplify and optimise the composite service configuration process, by providing predefined groups of components that can be used to reduce the amount of services the Configurator has to process in his/her search for suitable components.

Interfaces between components implement constraints and as such offer mechanisms for determining whether Elementary Services are suitable for integration into a composite service (see Fig.4). The requirements that need to be fulfilled for a successful composition are derived both from Components connected to the Interface, as well as from user preferences or Connection components. Interfaces can be defined between Elementary Services, Composite Services, Service Categories and Service Providers. Interfaces are mainly used to determine whether two Elementary Services fit, whereas Connection components describe a specific bond between two Composite Service Components.

The relationship between interfaces and elementary services matched by the filters contained in an interface resembles the one between plugs and sockets, whereby interfaces as sockets match multiple plugs. Henceforth, connections to Elementary Components that have a direct reference to an interface via its unique identifier will be referred as “sockets” and components that are matched by a socket will be referred to as “plugs”. An interface object is not restricted in its scope to use by only one pair of Service Components, but rather implements a generic rule (constraint) that can be used by multiple components for describing their interfaces.

When modifying existing Composite Services (i.e. when exchanging one Service Component for another) the Configurator needs to know exactly how components are connected to each other, in order to avoid altering the structure of the composition by using a different interface for connecting the new component with the rest of the composition. Connection components are therefore used for storing information on the connections between composite service and the corresponding interfaces (see Fig.7).

An elementary component can be connected to multiple interfaces and can act as a plug and or a socket (see Fig.5) depending on whether it is executing a search for a compatible component (through one of its interfaces) or being the target of such a search. Provided that multiple interfaces are available, all plugs corresponding to the filters in the interface objects are possible composition candidates. A positive match between two components however requires matching interfaces in both directions. In turn, this causes the Configurator to backtrack the connection between the plug and the socket in order to make sure that the plug also fits the socket.

Figure 5. Elementary Service “socket” (left) with multiple "plugs"(right)

When modifying existing Composite Services (i.e. when exchanging one Service Component for another) the Configurator needs to know exactly how components are connected to each other, in order to avoid altering the structure of the composition by using a different interface for connecting the new component with the rest of the composition. Connection components are therefore used for storing information on the connections between composite service and the corresponding interfaces (see Fig.7).

Figure 6. Global Workflow Model

Composite Services are assembled as a set of Composite Service Components arranged according to a specific type of workflow. Workflow functionality is introduced based on mechanisms specified by WSFL [12] (Web Services Flow Language) in order to ensure Web Service compatibility. Composite Services can thus have one of the following types of workflow: Flow Model, Global Model and Recursive Composition.

- A Flow Model is a linear workflow where each service has to be executed in a specific sequence. The correct...
sequence of execution is stored within the Composite Service Components.

• Global Models provide a description of how the composed services interact with each other. This type of workflow requires no additional considerations.

• Recursive composition of services provides scalability to the composition language and support for top-down progressive refinement design as well as for bottom-up aggregation. Recursive composition of services is made possible by the loosely coupled nature of Composite Services. New Composite Services can be composed out of existing compositions by merging the existing groups of components into new bigger compositions.

5. Conclusions

This paper has shown the need for transparent orchestration of composite services out of elementary services supporting a fully distributed and loosely coupled middleware layer comprised of co-operating components named “Generic Service Elements”. Furthermore an approach was illustrated for Composite Services Orchestration based on Variant Configuration theory. An example how such a method can be applied in the mobile context is given. The authors propose the implementation of a distributed middleware, based on GSE service elements, to cope with the steadily increasing system complexity whilst answering the demands on business flexibility and system integration speed.

6. Acknowledgements

The authors wish to express their gratitude and appreciation to the European Commission, the partners of the NOMAD project and the participants of the Wireless World Research forum (WWRF) WG2 for their strong support and valuable contribution during the various activities presented in this paper. Project NOMAD (IST-2001-33292) is funded by the European Commission within the IST Programme of the FP5.

7. References