Service-Oriented Development
- Whitepaper -

Broy, Diernhofer, Grünbauer, Meisinger, Rappl,

Rittmann, Schätz, Schoenmakers, Spanfelner

Introduction

Software systems are rapidly increasing in complexity. This is caused by their growing size, multi-functionality, multimodal user interaction, higher quality requirements, degree of distribution, dynamic execution environments, mobility, etc. In particular the distribution across heterogeneous networks over different processes, nodes, control units or components makes reactive systems difficult to develop and maintain. Multi-functionality increases this challenge – the purpose of the systems magnifies and in combination with distribution leads to complex functional dependencies, feature interactions and inefficient architectures.

A promising paradigm that offers solutions to several of the mentioned problems is service-oriented development. Most generally, a service can be understood as a coherent, purpose oriented unit of behavior. Thus, service orientation provides a method to modularize and structure the functionality of a system, dividing its behavior from a purpose-oriented point of view.

Service-orientation relies on two basic principles: By *modularization* the overall functionality of the system is split up to obtain separate modules of behavior, thus reducing the overall complexity of the system under development; by *composition* these separate modules are combined to obtain the overall behavior of the system, thus supporting a modular development process.

There are two main paradigms of application: the notion of *service as an analysis and design concept* is used to structure the description of the system, focusing on issues like modular description of functionalities and their combination; *service as a runtime concept* is used to support the implementation of the dynamic allocation, activation, orchestration, and termination of behavioral modules.

From a methodical point of view, service-orientation provides simplified views on complex reactive systems. Services can be understood as projections of such systems on partial functional aspects. Services, as independent modular views of functional behavior can later be combined to form an integrated system behavior view. Irrelevant details are abstracted away at the level of services which leads to a reduction of complexity. Thus, service-orientation emphasizes interface behavior and functionality over structure and can provide a more abstract view on component-based architectures.

The purpose of this white paper is to emphasize the opportunities that a service-oriented development approach offers and the benefits that go along with the concept of a "service". It points to applications of these approaches in research and development projects and ongoing dissertations at the research lab software & systems engineering at the TUM.

Service-Orientation throughout the development process

In the following we show how the service-oriented view can be applied at different phases in the development process at different levels of abstraction. Depending on the stage in the development process, the term "service" has specialized uses and applications.

Services in Requirements Engineering

Services can be seen as projections of system requirements onto partial pieces of behavior along the lines of use cases. Services are partial views of the functionality of systems. Synonyms for services in this case are system function, feature, use case etc. This view of a system is also called the usage view. Applications of services in requirements engineering are to structure the functional syntactic and semantic interface of the system, to identify any functional dependencies and to help in detecting and resolving conflicts.

Projects:

The project *mobilSoft* [mobilSoft] uses services during the model-based requirements engineering phase to formalize functional requirements and thus bridge the gap to formal models of the design phase.

The project MEwaDis [MEwaDis] uses services to identify the features/services, which are used in the system under consideration.

Dissertations:

The dissertations of Martin Deubler and Johannes Grünbauer cover the problem of modeling and specifying (multifunctional) systems using services. The services are connected via *relations* having a formal semantics and thus it is possible to discover unwanted feature interactions or even missing relations between services.

The dissertation of Wassiou Sitou is about the special challenges for requirements engineering for adaptive systems which heavily rely on services as a representation of certain functionality.

The dissertation of Sabine Rittmann addresses questions like how to methodologically build up and pragmatically specify service hierarchies (based on formal models).

Services in Design

Designing a reactive system entails designing effective architectures that can support the multiple functions of such systems. Services can be seen as projections/views on the system architecture that help to structure the entire design model. Services as prime elements in design models are pieces of functionality that are independent from certain elements of the software, such as components. Services can later be associated with these elements to form an integrated view. One way to specify services in design models is by means of interaction patterns between system entities.

Projects:

The InServe project [InServe] uses service-oriented development to design effective architectures for reactive systems. InServe's main focus is to research and establish a service-based development methodology based on a solid formal foundation. This includes research on description techniques, process, service-oriented architectures, as well as foundational research for a comprehensive service notion and service composition. Furthermore, InServe provides an implementation to support the concepts and conducts several case studies to evaluate the results.

In the MEwaDis project [MEwaDis] the system design is heavily supported by using services. In a first step, Use Cases are specified. Based on these Use Cases, the services are arranged within activity- and sequence diagrams. The message flow between services can be identified thereby.

From a formal perspective the Cawar framework [cawar] uses services as filters cutting out pieces of observable behavior out of a total system behavior. By changing the filter the section of the observable behavior can be moved. The service architecture is used do define the filter. With this formalization it is possible do describe adaptive systems.

The research of Bernhard Schätz views services as a notion of a configuration describing which sub-services can be active simultaneously. Services represent modes of operation.

Dissertations:

Michael Meisinger researches systematic architecture design using service-oriented development.

Michael Fahrmair described in his dissertation the use of services to define a system filter. This filter consists of services (in fact four service types) and has the feature of self description. Latter one is used to reason about the filter and to enable calibration.

Maurice Schoenmakers researches the design principles needed for modeling services and service descriptions to ensure flexible service composition capabilities at runtime.

Services in Implementation

Services help to realize systems by structuring an implementation in units that can be specified, developed and deployed independently. Composing services to realize application behavior realizes loose coupling between the system components, because of existence of black-box service interfaces. Thus, services are functional interfaces of components which can provide a protocol for service access. In some approaches services describe the activation and deactivation by configurations of services.

Projects:

The implementation of services is aided in MEwaDis [MEwaDis] by using the tool AutoFocus [AutoFOCUS]. Services can be modeled according to the sequence diagrams from the design phase. The data flow between services is realized via channels, which can transmit certain data. On adding state transition diagrams to services, the system can be simulated and tested. Finally, it is now possible to generate code out of the model.

The Cawar Framework [cawar] uses the filter specification via services to implement the system. Filters are treated as proxies that are bound to components at runtime. Through alternation of the service architecture (i.e. altering the proxy relationships) and underspecification of services the formally as filter modeled adaptive behavior is implemented.

Services in Deployment and Runtime

Services as modular, executable, deployable entities can be instantiated, initialized, registered, published, and looked up. Describing services, dependencies between them and providing a model integrating the runtime infrastructure provides a comprehensive model, showing all the different layers of service provisioning. Infrastructures can be compared to each other and changes in the infrastructure simulated, restrictions to a deyployment based on the infrastructure analyzed before real deployment. The effects of a change on the service and other dependant services can be analyzed, before the change is implemented.

Projects:

In MEwaDis tool supported development process, the modeled services can be exported from AutoFocus into the OSGi-service-framework directly. The user can group services to so-called *bundles*, and thus it's possible to run the exported services in most OSGi-environments.

The Cawar Framework [cawar] enables automatic deployment of services via the model activator. The model activator gets a K-Model (Service architecture) as input and finds and binds the corresponding components to the specified Services (Proxies)

The CKI project Dynamic Value Webs for IT-Services [CKI] focuses on on-Demand services and changes in the provisioning of services by new technologies like virtual infrastructures.

Dissertations:

The research of Maurice Schoenmakers defines the semantic meaning of common technical service infrastructure concepts like service registration, discovery, lookup, binding and resource handling as well as the semantic meaning of finding a matching service by using service descriptions for a dynamic service composition at runtime.

The research of Bernd Spanfelner focuses on decomposition of services at runtime. Since the K-Model provides an integrated view on services (abstract and technical view) it is possible to manage (de)composition on the model level and to deploy the changes at runtime via the model activator.

The research of Norbert Diernhofer focuses on business applications providing an integrated view of services consisting of the business processes they implement, their software components and the infrastructure they are provided with. Integrating models of services developed during development improves such a model.

Applications of Service-Oriented Development

Service-orientation has several beneficial applications in software development. The following sections briefly illuminate some of the most prominent ones.

Verification and Testing

Services can be used to generate test cases from models at all levels of abstraction.

Dissertations:

The research of Christian Pfaller deals with these issues.

Development of Product Lines

Services are used to structure requirements. New products in product lines can be configured from a network of services by selecting a consistent subset. The functional dependencies remain without any restriction to particular component architectures that are implementing them.

Projects:

The project VEIA addresses these issues.

Tool support of Service-Orientation

The InServe tool-chain

The project InServe developed parts of a tool chain with contributions from the TUM and the UCSD. The service-oriented modeling tool SODA governs the development process and provides a platform for modeling, specification, analysis, verification, refinement and code-generation.

References

[AutoFOCUS] Homepage of the tool AutoFOCUS http://autofocus.informatik.tu-muenchen.de/

[cawar] Homepage of the project cawar https://www.cawar.de

[CKI] Homepage of the project TUM CKI (in cooperation with Siemens Business Services) http://portal.tum.de/cki/index_html

[InServe] Homepage of the project InServe http://www4.in.tum.de/proj/inserve/

[mobilSoft] Homepage of the project mobilSoft (funded by the Bavarian government) https://www.mobilsoft.info

[MEwaDis] Homepage of the project MEwaDis (funded by the Bavarian government) https://www4.in.tum.de/~mewadis