Our daily life is characterized by an increasing number of process supporting IT services. With wireless and mobile networks, all kinds of devices are connected to the Internet and constantly access, collect and exchange data with various IT services to instantly provide information. The increasing volume and heterogeneity of computer devices requires changes in the architecture of IT services. At universities, students, researchers and decentralized service providers operate and work with such a heterogeneous and distributed IT service landscape. Existing monolithic process supporting systems are often highly specialized in locally defined processes and cover isolated business cases. If interfaces are available, they often use specific vocabularies and focus on technical details of the system, not the supported processes. Users, however, work in environments spanning across processes and often combine central and decentral IT services to achieve their goals. As digital competence is increasing, users seek to individualize and extend services according to their current needs. This extended degree of individualization requires a comprehensive digital service infrastructure.

To close the gap between the user’s perception of processes and the provided interfaces, this thesis proposes and evaluates a process-aware and service oriented reference architecture. By gradually hiding the technical complexity of back end systems, it introduces technical and semantic consistency across process and system boundaries. Taking into account the varying organizational, development and governance processes across back end systems, a web service landscape is implemented that uses technologies like HTTP, REST and the OAuth2 workflow to ensure consistency and compatibility with users’ environments. Adaptations of the OAuth2 workflow allow to cross boundaries of monolithic systems and integration into the landscape. Tiers of abstraction allow reuse of services by their technical specification or semantic modeling. Within various case studies from different areas of the university, the implementation is validated in real world scenarios. Semantic consistency allow users to adapt processes crossing system boundaries to their own needs. Technical consistency and focus on standardized technologies does not require technical knowledge of back end systems and enables individualized implementations in decentralized computing environments.

Marius Politze

A Reference Architecture and Implementation Enabling Data Protection in Distributed eLearning and eScience Processes
A Reference Architecture and Implementation
Enabling Data Protection in Distributed
eLearning and eScience Processes

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Marius Politze, M.Sc.
aus Neuss

Berichter:  Universitätsprofessor Dr. rer. nat. Matthias S. Müller
Universitätsprofessor Dr. rer. pol. Stefan Decker

Tag der mündlichen Prüfung: 05. Juni 2019

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.
Marius Politze:

A Reference Architecture and Implementation Enabling Data Protection in Distributed eLearning and eScience Processes

1. Auflage, 2019
Information technology and business are becoming inextricably interwoven. I don’t think anybody can talk meaningfully about one without talking about the other.

— Bill Gates, 1999 [1]
Contents

Acknowledgments v
Abstract vii
Kurzfassung ix
List of Figures xi
List of Tables xv
Nomenclature xvii

1 Introduction 1
  1.1 Motivation ............................................. 1
  1.2 Related Research ........................................... 3
    1.2.1 Ubiquitous Computing in University Processes ... 3
    1.2.2 Business Process Management ....................... 5
    1.2.3 Architectures for Process Supporting IT Systems .. 6
    1.2.4 Evaluation of Process Supporting IT Systems .... 9
    1.2.5 Security of Process Supporting IT Systems ...... 10
  1.3 Research Outline ......................................... 12
    1.3.1 Research Questions ................................... 12
    1.3.2 Contributions .......................................... 14
    1.3.3 Outline of the Thesis ................................ 15
    1.3.4 Course of Research ................................... 16

2 Reference Architecture for Process-Aware Services 19
  2.1 Current IT Service Landscape at RWTH Aachen University 19
    2.1.1 eLearning Services ................................. 21
    2.1.2 eScience Services .................................. 23
    2.1.3 Introduction of Case Studies .................... 25
  2.2 Conceptual Foundation .................................. 27
    2.2.1 Distributed Service Providers .................... 29
    2.2.2 Process-Aware Services ............................ 31
    2.2.3 Information Security ................................ 33
    2.2.4 Separation of Concerns ............................ 36
  2.3 pSTAIX: A Process-Aware Tier Architecture ............ 38
    2.3.1 n-Tier Architecture ............................... 39
3 Implementation of the Reference Architecture

3.1 Modeling and Methodologies
  3.1.1 Organizational Model
  3.1.2 Software Engineering Methodologies
  3.1.3 Proposed Key Technologies

3.2 Tier 0: Authorization and Security
  3.2.1 Deriving Requirements for Tier 0
  3.2.2 Local Authorization
  3.2.3 Cooperative Authorization
  3.2.4 Federative Authorization
  3.2.5 Monitoring and Auditing

3.3 Tier 1: General Services
  3.3.1 Deriving Requirements for Tier 1
  3.3.2 Caching Service
  3.3.3 Logging Service
  3.3.4 Private and Shared Persistent Storage Service
  3.3.5 User Profile Service
  3.3.6 Notifications Service

3.4 Tiers 2 to 3: Integration of Existing Services
  3.4.1 Existing eLearning Services and APIs
  3.4.2 Existing eScience Services and APIs

3.5 Tier 4: Process-Aware Services
  3.5.1 eLearning-Process-Aware Implementations
  3.5.2 eScience-Process-Aware Implementations

4 Case Studies

4.1 eLearning Processes
  4.1.1 RWTHApp
  4.1.2 Quizz2Go
  4.1.3 Audience Response System

4.2 eScience Processes
  4.2.1 HIApp
  4.2.2 Metadata Tool
Acknowledgments

During the past years, authoring this work took a lot of time, dedication and it was not always easy to keep going. After all, this would not have been possible without the support of family, friends and colleagues.

First of all I would like to thank my wife. Angela, your immense patience and support allowed me to dig into the research knowing that everything is taken care of: Sometimes reminding me that I need to sleep, eat or should retain a social life and sometimes pushing me to keep on working. That balance is what made me hang on when going through the next night shift. I love you and this would not have been possible without your support. I also appreciate my parents and in-laws for constantly reminding me to enjoy life to the full.

My friends, thank you for humor, sarcasm and joy; for hiking on first of May; for dining and feasting with Feuerzangenbowle, Weihnachtsvogel or on Water while sailing and for enjoying company at the Grill-Club.

Even though the topic I chose for this thesis was quite exceptional for the IT Center when I started my work, I received a lot of support in my work environment to peruse my ideas. The IT Center turned out to provide just the right relation between professional IT service provider and playing field to test innovative approaches. My gratitude therefore goes to colleagues at the IT Center of RWTH Aachen University for supporting this thesis with discussions as well as ideas and sourcing of project case studies and their implementations. In particular, I would like to point out Thomas and Basti for countless fruitful discussions to generate new ideas and Bernd for supporting me to put my ideas into practice. Of course, I would like to thank my supervisors, Prof. Müller and Prof. Decker for their support and the inspiration provided during the thesis. Special thanks go to Prof. Wehrle and Prof. Grohe for being part of the examination committee.
Abstract
Our daily life is characterized by an increasing number of process supporting IT services. With wireless and mobile networks, all kinds of devices are connected to the Internet and constantly access, collect and exchange data with various IT services to instantly provide information. The increasing volume and heterogeneity of computer devices requires changes in the architecture of IT services. At universities, students, researchers and decentralized service providers operate and work with such a heterogeneous and distributed IT service landscape. Existing monolithic process supporting systems are often highly specialized in locally defined processes and cover isolated business cases. If interfaces are available, they often use specific vocabularies and focus on technical details of the system, not the supported processes. Users, however, work in environments spanning across processes and often combine central and decentral IT services to achieve their goals. As digital competence is increasing, users seek to individualize and extend services according to their current needs. This extended degree of individualization requires a comprehensive digital service infrastructure.
To close the gap between the user’s perception of processes and the provided interfaces, this thesis proposes and evaluates a process-aware and service oriented reference architecture. By gradually hiding the technical complexity of back end systems, it introduces technical and semantic consistency across process and system boundaries. Taking into account the varying organizational, development and governance processes across back end systems, a web service landscape is implemented that uses technologies like HTTP, REST and the OAuth2 workflow to ensure consistency and compatibility with users’ environments. Adaptations of the OAuth2 workflow allow to cross boundaries of monolithic systems and integration into the landscape. Tiers of abstraction allow reuse of services by their technical specification or semantic modeling. Within various case studies from different areas of the university, the implementation is validated in real world scenarios. Semantic consistency allow users to adapt processes crossing system boundaries to their own needs. Technical consistency and focus on standardized technologies does not require technical knowledge of back end systems and enables individualized implementations in decentralized computing environments.
Kurzfassung


# List of Figures

1.1 Visualization of the course of research ................................. 18

2.1 RWTH Aachen University: Map of Processes [100] .......................... 20
2.2 Student-Life-Cycle of RWTH Aachen University [102] ............... 22
2.3 Connection of CAMPUS and L²P by Gebhard et al. [103] .......... 23
2.4 Research data-life-cycle of RWTH Aachen University [106] ....... 25
2.5 Distributed process elaboration methodology adapted from Mu et al. [63] ............................................................. 29
2.6 Aspects of security and usability by Feth [95] ............................. 35
2.7 Overview of the pSTAIX reference architecture ...................... 40
2.8 Tiers and their properties .................................................. 49

3.1 Distributed services at RWTH Aachen University ......................... 56
3.2 Schematic overview of the OAuth2 workflow ........................... 65
3.3 Schematic overview of the cooperative OAuth2 workflow .......... 68
3.4 Schematic of the federative OAuth2 workflow .......................... 74
3.5 Federative authorization process ....................................... 79
3.6 Federative token validation process .................................... 80
3.7 Token validation process extended by auditing steps .............. 82
3.8 Schematic architecture of the existing organizational and lecture directory back end system and Application Programming Interface (API) ....................................................... 99
3.9 Schematic architecture of the timetable back end system and API .......................................................... 101
3.10 Schematic architecture of the L²P back end system and API 102
3.11 Schematic architecture of the Moodle back end system and API 103
3.12 Schematic architecture of the audience response system service 104
3.13 Schematic architecture of the library service .......................... 105
3.14 Schematic architecture of the Persistent Identifier service .... 107
3.15 Schematic architecture of the metadata service .................... 108
3.16 Schematic architecture of the archive service ...................... 110
3.17 Schematic architecture of the SharePoint shared service .... 111

4.1 Application life-cycle for the development of RWTHApp ........ 122
4.2 Student centered lecture materials process ........................... 124
4.3 Student centered library loaned media process .................... 124
4.4 Student centered take quiz process .................................... 128
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Student centered answer question sub process</td>
<td>128</td>
</tr>
<tr>
<td>4.6</td>
<td>Supported lectures and total messages sent</td>
<td>133</td>
</tr>
<tr>
<td>4.7</td>
<td>Metadata tool integrated workflow</td>
<td>138</td>
</tr>
<tr>
<td>4.8</td>
<td>Metadata tool private workflow</td>
<td>139</td>
</tr>
<tr>
<td>4.9</td>
<td>Integration of the metadata tool architecture in a local workflow</td>
<td>140</td>
</tr>
<tr>
<td>4.10</td>
<td>Simplified overview of simpleArchive processes</td>
<td>141</td>
</tr>
<tr>
<td>4.11</td>
<td>Schematic of simpleArchive at RWTH Aachen University and TU Darmstadt</td>
<td>144</td>
</tr>
<tr>
<td>4.12</td>
<td>Deployment Diagram of eduroam Device Manager using the reference architecture</td>
<td>148</td>
</tr>
<tr>
<td>4.13</td>
<td>Adoption of eduroam device management process</td>
<td>149</td>
</tr>
<tr>
<td>4.14</td>
<td>Chat Support process</td>
<td>150</td>
</tr>
<tr>
<td>4.15</td>
<td>Number of messages sent in chat support</td>
<td>151</td>
</tr>
<tr>
<td>5.1</td>
<td>Separating technical and process concerns in the reference architecture</td>
<td>158</td>
</tr>
<tr>
<td>A.1</td>
<td>Overview of a lecture in RWTHApp</td>
<td>182</td>
</tr>
<tr>
<td>A.2</td>
<td>Answering questions right (left) and wrong (right)</td>
<td>182</td>
</tr>
<tr>
<td>A.3</td>
<td>Overview of all questions and answers (left) and aggregated results (right)</td>
<td>183</td>
</tr>
<tr>
<td>A.4</td>
<td>Teacher view during poll</td>
<td>183</td>
</tr>
<tr>
<td>A.5</td>
<td>Student view during poll</td>
<td>184</td>
</tr>
<tr>
<td>A.6</td>
<td>Add new device to eduroam device management</td>
<td>184</td>
</tr>
<tr>
<td>A.7</td>
<td>Manage existing device to eduroam device management</td>
<td>185</td>
</tr>
<tr>
<td>A.8</td>
<td>View device connection history</td>
<td>185</td>
</tr>
<tr>
<td>A.9</td>
<td>Chat Support integration into documentation web page</td>
<td>186</td>
</tr>
<tr>
<td>A.10</td>
<td>Chat Support integration into eduroam device management</td>
<td>186</td>
</tr>
<tr>
<td>A.11</td>
<td>User view starting a support chat</td>
<td>187</td>
</tr>
<tr>
<td>A.12</td>
<td>User view of support chat</td>
<td>187</td>
</tr>
<tr>
<td>A.13</td>
<td>Support agent chat overview</td>
<td>188</td>
</tr>
<tr>
<td>A.14</td>
<td>Support agent chat view</td>
<td>188</td>
</tr>
<tr>
<td>A.15</td>
<td>Contributor view of articles with coordinates</td>
<td>189</td>
</tr>
<tr>
<td>A.16</td>
<td>Rich content editor for articles</td>
<td>189</td>
</tr>
<tr>
<td>A.17</td>
<td>HIApp start page (left) and map view (right)</td>
<td>190</td>
</tr>
<tr>
<td>A.18</td>
<td>Metadata Tool start page</td>
<td>190</td>
</tr>
<tr>
<td>A.19</td>
<td>Editing a metadata set</td>
<td>191</td>
</tr>
<tr>
<td>A.20</td>
<td>simpleArchive home page</td>
<td>191</td>
</tr>
<tr>
<td>A.21</td>
<td>simpleArchive archive page</td>
<td>192</td>
</tr>
<tr>
<td>A.22</td>
<td>simpleArchive restore page</td>
<td>192</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>A.23 Moodle subsystem integrated into L²P</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>A.24 Moodle subsystem integrated into L²P</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>A.25 Exemplary assessment process supported by FLEX [168]</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>A.26 FLEX server architecture [168]</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>A.27 Learning room finder map (left) and details view (right)</td>
<td>196</td>
<td></td>
</tr>
</tbody>
</table>
# List of Tables

2.1 Centrally provided services supporting different phases of the Research data-life-cycle ........................................... 26

3.1 Attributes required by 347 service providers in the German DFN-AAI federation ........................................... 73

3.2 Comparison of different caching strategies. ....................... 89

4.1 User sessions counts within process areas of RWTHApp in winter term 2018 .................................................. 126

4.2 Number of quiz downloads in winter term 2018 ................. 129

4.3 Usage statistics of *simpleArchive* .................................. 143
Nomenclature

**ALM**  Application Lifecycle Management.

**API**  Application Programming Interface.

**BPEL**  Business Process Execution Language.

**BPM**  Business Process Management.

**BPMN**  Business Process Modeling Notation.

**BYOD**  Bring Your Own Device.

**DOI**  Digital Object Identifier.

**ePIC**  European Persistent Identifier Consortium.

**HTTP**  Hyper Text Transfer Protocol.

**HTTPS**  HTTP Over TLS.

**IaaS**  Infrastructure as a Service.

**IdM**  Identity Management.

**ITIL**  IT Infrastructure Library.

**JSC**  Jülisch Super Computing Centre.

**JSON**  JavaScript Object Notation.

**JWT**  JSON Web Token.

**KPI**  Key Performance Indicator.
**Nomenclature**

LMS  Learning Management System.

MOOC  Massive Open Online Course.

OTA  One Time Access Token.

PaaS  Platform as a Service.

PAIS  Process-Aware Information System.

PID  Persistent Identifier.

pSTAIX  Process-aware Software Tiers for Application Interfaces and eXtension.

RDF  Resource Description Framework.

RDM  Research Data Management.

RDMO  Research Data Management Organizer.

REST  Representational State Transfer.

SaaS  Software as a Service.

SAML  Security Assertion Markup Language.

SLM  Student Lifecycle Management.

SOA  Service Oriented Architecture.

SoC  Separation of Concerns.

SPARQL  SPARQL Protocol And Query Language.

UML  Universal Modelling Language.

URL  Uniform Resource Locator.

XML  eXtensible Markup Language.

xviii
1 Introduction

Globalization and digitization pose continuous challenges to already established processes in government and administration. The standardization of existing processes to improve cooperation between institutions, to reduce overall costs or to increase efficiency is a part of daily business on several levels: global (e.g. worldwide / EU), national (e.g. in Germany or North Rhine-Westphalia) or local (e.g. Aachen). Over the past years this led to IT infrastructures and applications that eventually cumulated into ubiquitous IT services that support a variety of processes. However, the supported business processes are subject to constant change and are regularly updated in accordance to new regulations or due to optimization. Each change to a previously defined process can therefore result in several changes within IT services. The resulting frequency, impact of change and variety of processes in turn require flexible IT services and applications supporting these processes.

1.1 Motivation

Due to increased mobility and the rising number of students and researchers, universities find themselves confronted with increased demands on IT services. However, the issue does not only affect the organizational level. Students and employees have already changed their requirements for IT service providers and thus also for universities in general. Not only IT services and applications are becoming increasingly important for the processes of universities but also the everyday life of students and employees. This in turn makes the quality of IT services a location advantage and leads to increased competition among the universities for the best and most attractive IT service portfolio. A development shown in the Horizon Report series, one of the most regarded studies concerning the development of education, for several years [2]–[4]. To meet these developments, these reports have consecutively identified upcoming major challenges in the higher education sector as follows:

- Bring Your Own Device (BYOD)
- The Internet-of-Things
In order for IT services of universities to remain operational, competitive and meet the expectations of governments, researchers and students, IT service providers are required to manage a growing, heterogeneous service landscape. While many standardized solutions exist, tailoring the services towards local needs and integrating the services with each other to create a consistent user experience requires an in depth knowledge of business processes of the university. For general process oriented management systems several methodologies and certifications, for example according to IT Infrastructure Library (ITIL) and ISO 20000 [5] or according to the ISO 9000 standards family [6], are available and are widely used in practice. Instead of focusing on the business processes to be supported, development processes for IT services and applications follow specific software engineering methodologies that rather aim for software quality than the quality of the supported business processes. Organizations that act as an IT service provider and that also tailor applications to fit to business processes, are therefore left in a tension field between their own processes for management, operation and engineering of services as well as the business processes they aim to support.

Increased digital literacy among the users additionally raises the need towards integration of centrally provided IT services within local and decentralized, environments of users. This trend becomes even more evident when considering that data driven and machine learning applications are gradually becoming affordable and more common in the local environment of the users. As press reports indicate there some prominent approaches towards integrating off the shelf machine learning with local business processes in the areas of research and education; some were more successful [7] than others [8]. Especially within environments that depend on business processes and data modeled in existing system landscapes application technologies for individualization and automation requires significant work to integrate data and make it accessible by applications.
1.2 Related Research

Considering processes within the university context there are two predominant areas of processes: supporting scientific research and supporting teaching and learning. IT systems of central and decentralized service providers that try to digitize and improve processes in any of these areas are generally referred to as eScience or eLearning systems respectively. Enabling this highly specialized system landscape to interact within a process oriented environment is the fundamental challenge to provide added value services for students and employees of the university.

1.2 Related Research

Even though the regarded processes have a specific implementation at the university, it is obvious that the mentioned complex of challenges is neither specific to RWTH Aachen University nor to a university’s context in general but generalizes to all kinds of processes oriented businesses. Various research groups and projects reduce complexity by modeling and try to find iterative or partial solutions to some facet of the aforementioned challenges.

The research area of Ubiquitous Computing focuses on technical and socio-technical challenges that arise from the decreasing size and increasing availability of computing services in general. Some research groups already focus how universities are influenced by this development. Supporting business processes with IT systems requires transferring to a consistent, often machine readable, description as discussed by the research area of Business Process Modeling. To successfully build IT systems supporting these processes, the research area of IT System Architectures defines and evaluates different variants of their implementation. These implementations of business processes potentially deal with personal or confidential information. Additionally to assessing their architectural properties, the research area IT Security evaluates the security of process supporting systems.

1.2.1 Ubiquitous Computing in University Processes

The rise of fully connected applications and always available and low-cost wireless connections have changed the view how computing is perceived
1 Introduction

in our daily lives. Processes in organizational and personal context are exceedingly drained by computing services. Weiser therefore coins the term *Ubiquitous Computing*; stating that eventually all computing processes are taking place in the background and without explicit interaction of the user [9]. This development is accompanied by an ever decreasing size of connected computing devices. While the Internet was initially intended for social interactions, currently devices and human users can freely participate in interactions. However, Ciortea *et al.* also require more specialized software architectures to account for this changed usage behavior [10]. One of these requirements is that information is available in a machine readable format that can be interpreted and accessed by devices easily as proposed by Gerber *et al.* and Han *et al.* [11], [12].

Within university context the research area *prevasive university* specializes on ubiquitous computing of research and educational processes. The fields of eLearning and eScience are therefore in focus of several research groups and projects. Juling describes that mobility and ubiquity of information technology is a sign of the times. Global accessibility, reachability and therefore mobility have become part of our daily life [13]. He demands that existing and future computing systems have to be designed in a way that they can actively use information in collaborative, flexible, reliable and secure ways. Transferred to universities he states that eScience requires a comprehensive digital infrastructure. He further requires to foster the creation, pooling and usability of IT systems, resources and infrastructure to provide additional value to researchers in different scientific disciplines. The use cases by Van Garderen and Kirsten *et al.* further show how such IT systems can support scientific processes at universities [14], [15].

Barkhuus *et al.* discuss the different needs of students related to ubiquitous services offered by their university. They focused on the role of technology in the classroom as well as the social arrangements in which the students were situated [16]. While the technological basis has severely changed since the study in 2004, the social roles, relationships and responsibilities of the students are mostly comparable. The most important issues of the pervasive university pointed out are organizational like reliability and ease of use. Engaging students when building process
supporting IT services can further increase satisfaction as shown by Kupila et al. [17]. Additionally, as pointed out by Gikas et al., matching IT services with students’ devices plays an important role within the supported processes [18].

During the past years mobile and web technology advanced drastically. Lucke et al. introduce a model for the pervasive university that takes the heterogeneous existing system landscape into account [19]. An example a prototype that is based on some university business processes in the field of eLearning is proposed and shown to be integrated into other systems. Lehsten et al. extend this model with a concept to integrate multiple context aware services [20]. The context of the user is aggregated by a central services and distributed to the attached services. This points out the issue of a heterogeneous system landscape that needs to be integrated to support ubiquitous processes. As shown by Schmees et al. and Haim many universities are facing this problem and had to develop various local solutions to handle it [21], [22]. Mincer-Daszkiewicz and Barata et al. show two example architectures for Campus Management as a supporting process of Education and Research [23], [24]. While Barata et al. show the requirements on a newly developed Student Lifecycle Management (SLM) system, Mincer-Daszkiewicz adopt an existing SLM system to changing processes and higher load due to increasing number of users. While their implementations do not follow any explicitly modeled processes they raise the issue of security of personal data when crossing process and system boundaries.

1.2.2 Business Process Management

Continuous improvement of organizations requires to model the ways in which they provide value to their customers or users. Business Process Management (BPM) therefore is used to build a common understanding on these operations and workflows within an organization in a way that can be heeded by employees of the organization. Resulting comprehensive models can then be applied by IT system architects building infrastructures to support the processes and therefore allow further digitization of the organization as described by van der Aalst et al. [25].
1 Introduction

Modeled business processes furthermore form the basis for change management and optimization of business processes. Schulte et al. further shows that the close connection to IT systems allows optimizing used resources which becomes explicitly beneficial in elastic cloud environments [26].

Apart from the technical description of a system BPM also reflects procedures carried out by employees. Giaglis et al. propose that modeled business processes should therefore fit two dimensions: on the one hand technical requirements like formal, quantitative and stochastic modeling or documentation, adaptability and objectivity; on the other hand social requirements like feasibility, communication or user friendliness [27]. Based on these requirements they show that BPM and its validation by simulation are valuable for modeling intra- and inter organizational processes. Wißotzki additionally points out that transferring business processes to IT systems often can only be performed by domain experts but is still an underestimated in many environments [28].

Aguilar-Savén shows that several languages with different properties have emerged in order to describe business process models [29]. The author shows that a multitude of BPM languages and supporting tools already exist. Additionally, a proposed classification framework provides some guidance for business analysts or IT system architects to chose among the available languages. Sandkuhl et al., however, point out that BPM needs to become a common practice providing additional value for the organization and not a purpose on its own [30]. They envision that also non-experts model processes, sometimes without knowing. Modeled processes are viewed as a kind of captured knowledge that can be shared within the organization, is used for decision making but has fewer requirements towards formality and completeness.

1.2.3 Architectures for Process Supporting IT Systems

Future distributed IT systems therefore need to consider that more devices are using the Internet to communicate. Within the research area of the so called Internet-of-Things Miorandi et al. identify heterogeneity,
1.2 Related Research

scalability, semantic interoperability and privacy preservation as some of the key challenges that IT systems need to support [31]. Facing some of these challenges Han et al. propose a full-IP ecosystem using the Representational State Transfer (REST) paradigm that is already predominant in modern Internet applications [12]. The authors further claim that composing services from existing workflows allow novel and creative approaches for future applications and devices. As shown by O’Reilly, the basic design pattern of Service Oriented Architectures (SOAs) that bases software on independent services that in turn base on business processes, is already part of many common Internet applications [32]. Commodity Internet standards like Hyper Text Transfer Protocol (HTTP) make these kinds of services widely applicable but also led to competing implementations as pointed out by Tihomirovs et al. [33].

Using standardized protocols and architectures distributed systems provide some features required by Miorandi et al. Nevertheless, Taheriyani et al. point out that semantic interoperability need particular attention [34]. The authors propose a method to integrate SOAs with each other using linked data. While manually lifting existing SOA is connected to high costs, they propose a semi-automated approach that infers semantic information by using the existing services. Especially when proving access to complex functionalities being understandable makes services reusable. To maximize reusability, Zhu claims that internal complexity and therefore the value of a service should be high while access to it should be easy and cheap [35]. A common understanding of services furthermore is required for the integration of services within an organization or across organization boundaries as shown by Ebert et al. [36]. Composing and integrating services from various IT systems may result in additional latencies as shown by Göb et al. reducing user experience with slow network connections or small compute devices [37]. Using prefetching middlewares as part of the SOA, the authors are able to actively work against the issue effectively shifting compute or network intensive workloads from users’ devices to a server infrastructure.

Gruber et al. points out that IT systems are becoming increasingly complex and state that this complexity needs to be properly managed to retain maintainability of the system [38]. Kecskemeti et al., Viennot
et al. and Salvatierra et al. show approaches of technically decomposing IT systems to an SOA to foster scalability of the infrastructures [39]–[41]. Microservice architecture as, for example, proposed by Namiot et al. is used to reduce dependencies and allow better scalability of SOAs [42]. By breaking the SOA into small, independent services each of which fulfills only few well defended tasks. Namiot et al. further show that traditional SOA are scaled by duplication or partitioning, the independence of separate modules instantiated as Microservices also offer new possibilities to scale the system by functional decomposition. This is especially important as more and more devices are accessing the services. Cloud technologies furthermore allow scaling of IT systems without direct dependency to physical hardware or on site resources.

Mircea et al. and Low et al. conclude migration of existing IT systems is especially an issue for complex process supporting systems within large organizations [43], [44]. As remarked by Serrano et al. existing legacy systems are usually not well documented and reengineering is likely expensive [45]. Instead, the authors claim it is desirable to wrap existing systems with a service layer to allow usage of modern technologies. Newly developed applications, however, often already adapt to new infrastructural and architectural requirements [46]–[48]. The introduced multitude of small services and abstraction layers additionally raise new challenges for software development processes in traditional environments. Schleicher et al. and Zimmermann et al. show in their works that an additional dependency is introduced as virtualization and cloud environments pose new technical and organizational requirements to the deployed software and development teams [49], [50]. To meet some of these technical challenges Toffetti et al. propose a service registry allowing self organization [51], while Balalaie et al. show how development and release processes have to be changed to fit to Microservice architectures [52].

A shown, breaking down complex systems into independently maintained units can ease the development process for the separated services. After all, services in their entirety need to provide value to their users. van der Aalst et al. and Schonenberg et al. therefore require that services need to be interoperable and thus can participate in the same organizational processes [53], [54]. To lift existing databases to an interoperable
state Wiederhold proposed a mediator architecture [55]. Mediators are services that collect information from these databases and convert the contained data set to a common semantics. Papakonstantinou et al. and Wiederhold et al. have proposed several computational operations that need to be supported on the datasets to perform this [56], [57]. Over the last years there were several systems that based on the mediator architecture that are specific to certain disciplines for example web crawling [58], bioinformatics [59] or learning repositories [60]. While these initial works focus on databases most results can be transferred to SOA: Barthe-Delanoë et al. show how a mediator information system can be used to achieve and evaluate interoperability of process supporting services [61]. Based on these findings, the research group around Bénaben et al. have developed and gradually refined a process model that allows managing orchestration of processes, data conversion and service selection within a mediator information system [62]–[64].

Apart from technical challenges, Aerts et al. claim that software architectures, modeled processes and business needs have to be aligned in order to provide the maximum value to the users [65]. The authors regard an IT system as a composition of layers decoupling technological, application and business models that need to be integrated. To overcome gaps between modeled processes and the supporting software Zimmermann et al. propose an integrated SOA that allows adoption of changing business processes and needs [66]. The authors translate architectural domains into a service oriented architecture ontology that allows automatic inference and therefore supports decision processes within a practical implementation. In subsequent works Zimmermann et al. and Kaidalova et al. show how their architecture adapts to the change of business needs induced by recent developments like the Internet-of-Things or micro-granular architectures like Microservices in various contexts [50], [67]–[69]. To practically achieve this alignment Montesi proposes an approach unifying implementation of SOA and BPM to create process-aware services [70].

1.2.4 Evaluation of Process Supporting IT Systems

Process supporting IT systems mostly result from a software engineering process. Tarr et al. introduce Separation of Concerns (SoC) as a method
1 Introduction

for evaluating software architectures in general [71]. The authors introduce three goals, Impact-of-Change, Reuse and Traceability, that should be pursued across four dimensions of concerns: Feature, Unit-of-Change, Customization and Data. Even though the concepts behind SOAs allow separating services according to these dimensions, Mateos et al. show that also applications using the SOA need an additional layer of abstraction to avoid coupling to specific service implementations [72]. Bianco et al. formulate fundamental design questions to be considered during the software engineering process and map them to potential business risks [73]. A catalog of detailed questions presents a guideline for technical evaluation of SOAs.

The model introduced by Rathfelder et al. does not only evaluate the technical maturity of an SOA but also considers non-technical domains like organizational structure, development and governance processes [74]. Based on five maturity levels within these domains the authors show various challenges, benefits, risks being encountered as SOA become more mature. Welke et al. propose a similar model and additionally point out that at the final maturity level business processes and services become equivalent [75]. While the assessment of some domains is subjective, both works point out the importance of governance and measurable quality indicators. For this kind of practical evaluation of SOAs Bogner et al. provide a measurable quality model that especially assesses the maintainability of Microservice architectures based on its structure [76], [77]. La Jung et al. measure the quality of an SOA by Quality-of-Service levels perceived by users of the SOA [78].

Independently of their architecture process supporting IT systems therefore need to be adaptable to business process models. Based on common patterns Russel defines a conceptual foundation using a catalog of criteria that Process-Aware Information System (PAIS) should meet to ensure they are able to align to changing business needs [79].

1.2.5 Security of Process Supporting IT Systems

Information security is a basic need and common business risk that relates to all kinds of information systems. The international standard ISO
27000 therefore defines goals that should be achieved by organizations and their IT systems [80]. The primary security goals of confidentiality, integrity and availability are subject in research areas regarding the security of process supporting IT systems. Karlapalem et al. apply these goals to distributed systems [81]. For their proposed security management system, the authors claim that controlling authorization, authentication and anonymity of all involved parties is fundamental to meet the security goals. In order to pursue security goals from the viewpoint of business processes Herrmann et al. propose extensions for BPM frameworks [82]. Basin et al. analyze how security restrictions effect business processes [83]. The authors further propose an algorithm balancing risk and cost of security policies [84]. Looking at different business risks, security related issues are the top concerns identified by González et al. when working with distributed service providers [85].

Within distributed system landscapes it therefore necessary to adhere common workflows and protocols for authentication and authorization. Memon et al. propose a centralized service that translates between different authentication flows and maps corresponding identities for other systems [86]. The SOA proposed by Lablans et al. provides centralized approach to consistent pseudonomization within a distributed system with high requirements towards anonymity [87]. Especially for large, decentralized system landscapes Grabatin et al. raise that data quality becomes an issue and therefore require a level of assurance management [88]. Several research groups analyzed attack schemes on authorization and authentication workflows for distributed systems [89]–[92]. Over the past years Li et al. and Yang et al. used their results to exploit several flaws within these workflows, allowing them to break into several applications at a time using the same vulnerabilities [93], [94].

Increased security often implies a reduction of user experience. Additional to the technical security of information systems therefore the users’ perceptions on security is also addressed by research groups. Feth therefore proposes Security Awareness, Trust and Perceived Security as user centric security goals that should additionally be considered [95]. Looking at common misconceptions, Hof defines practical guidelines for usable and secure systems and uses them to evaluate systems [96].
Usability of security mechanisms becomes even more important as users interact with them using a multitude of devices as pointed out by Montero et al. [97]. The approach of las Cuevas et al. uses machine learning techniques to create user profiles and detect changes in user behavior and potential security breaches [98].

1.3 Research Outline

Using the previously discussed motivation and research results as a basis, this work discusses the fundamentals to prepare the existing decentralized IT system landscape to meet the rising demands of students and researchers and to meet the requirements towards customizable process support. Apart from providing the theoretical basis for assessing and building this system landscape it is the goal to practically prove its functionality across several real world case studies supporting processes in the areas of eLearning and eScience.

1.3.1 Research Questions

To better cope with the latest developments and fast changing requirements, increased mobility and higher degree of customization SOAs are widely used. However due to the broad fields of application of SOAs during the past years a multitude of incompatible standards, formats and architectures have been developed and are now part of virtually any application framework. While choosing the right SOAs is a critical step for the short term success it is most important to build a comprehensive and consistent model of the data exchanged between the systems. To generate this common model several processes at the university need to be analyzed in how they use the available data.

**Research Question 1:** How can the architecture provide a comprehensive, process independent and reusable model to access existing data stored in heterogeneous distributed systems of the university?

In any software system if it is a SOA or a more traditional monolithic software a central asset is reliability. While in a monolithic environment
reliability may be controlled centrally, in a distributed SOA other measures have to be provided to assert this kind of service quality. Naturally these should base on the three key concepts from information security, confidentiality, integrity and availability. As the SOA will participate in a multitude of processes these need to be endorsed already by its design. Further security and reliability needs like authenticity, non-repudiation, accountability or anonymity should furthermore be attainable by an implementation the architecture.

**Research Question 2:** How can the architecture provide means to increase reliability of these distributed services for existing and new processes?

According to the need for reliability and security of the operators and users of the service, the protection of personal data needs to be enforced. By extending processes and sharing data across their boundaries, data will eventually be automatically accumulated to data sets and may become available to a broader audience. Especially for personal data the general right of personality needs to be protected. The resulting infrastructure needs to be designed in order to allow applications and processes to support regulations such as European and German data protection laws as well as organizational regulations.

**Research Question 3:** How can protection of personal or confidential data be supported by the architecture when sharing data across process and system boundaries?

Processes supported by the SOA were defined by stakeholders from several contexts. The architecture has to generalize and bridge between these contexts to enable spanning processes across organizational and technical borders. To compare the contexts of different processes three categories are regarded:

- *Local context* defined by a single or few closely related process, like a local Learning Management System (LMS) instance with local user accounts
- *Cooperative context* defined by processes that partially work across technical boundaries, for example by integrating instances of a library management system into an LMS using the same user accounts.
• *Federative context* defined by processes that work across technical and organizational boundaries, for example by integrating a third party storage provider into multiple LMS.

**Research Question 4:** How can the architecture generalize to support processes working across systems in distributed local, cooperative and federative service landscapes found in the university sector?

Answering the research questions will lay the foundation towards implementing process oriented services supporting eLearning and eScience processes on top of the existing decentralized IT system landscape. The resulting services and applications should furthermore grant a high degree of customization for students and employees of the university.

### 1.3.2 Contributions

By answering the research questions this thesis contributes to the areas of decentralized and service oriented system architectures. This work focuses on asserting security and integration of legacy systems within these systems. The architecture constructed and evaluated in this works lays the theoretical basis for cooperative and federative distributed services sharing users’ identities across process, system and organizational boundaries. The proposed and evaluated services allow managing of the flow of information and authorizations of systems within decentralized service infrastructures and especially consider user perceived security.

Additional to the proposed architecture, a framework to assess the quality of a process supporting SOA was developed. This framework brings together different evaluation criteria from BPM, SoC and IT Security and requires the compatibility with commodity Internet standards. This framework should form a basis for the design of upcoming IT systems supporting processes in distributed environments. It further provides guidelines on how to establish necessary technical and organizational structures to successfully implement and operate this kind of system.s

In conjunction with this framework a model that can be used to lift a legacy monolithic system landscape to allow microservice like interaction,
replacement and independence of systems. This is especially important since IT systems are subject to constant change. This model paves the way to gradually migrate from monolithic systems to SOA and then allows the migration of single responsibilities to other systems. By using widely available Internet technologies, it allows opening legacy systems towards use cases that were previously unsupported by systems themselves.

Apart from theoretical basis, the implementation of the system at RWTH Aachen University allows students and researchers provides a practical implementation of services that support university wide processes. The service infrastructure can further be used to customize these processes to integrate them into existing local or decentralized process supporting systems.

1.3.3 Outline of the Thesis

The following Chapter 2 presents a requirement analysis for a distributed service architecture to support processes using IT systems at RWTH Aachen University that bases on several case studies. These requirements are used to build a generalized architecture and derives several dimensions of evaluation criteria that can be used to assess how a distributed system can support decentralized processes. Based on these dimensions an n-tier architecture is proposed and evaluated, that should be put into practice at the university.

Chapter 3 will then introduce a distributed organizational model and key technologies used for an implementation of the architecture. Based on the case studies and derived requirements the chapter will commence to describe the implementation of the system that scales across local, cooperative and federative contexts. The resulting SOA will present an exemplary set of interfaces that can in turn be used to implement the various case studies.

After the introduction of the theoretical framework and its implementation Chapter 4 will discuss the previously introduces case studies featuring processes from eLearning, eScience and organizational support.
1 Introduction

The chapter will show how processes are modeled on top of the architecture and how their implementation is evaluated according to qualitative and quantitative usage metrics.

Finally Chapter 5 will conclude the thesis and sums up how the proposed architecture influences the implementation of process supporting applications at the university. The chapter also evaluates the implementation and the case studies according to dimensions of evaluation defined in Chapter 2. Based on this evaluation, the chapter answers concludes answers to the research questions and proposes some related future research areas.

1.3.4 Course of Research

An abstraction layer will be developed that is verified using case studies.

As opposed to the order in this work implementation of case studies and the theoretical foundation for the architecture were developed in parallel and were refined multiple times to gradually derive and answer the research questions. Different phases of the course of research can be roughly aligned to a PDCA-Cycle as described by Deming [99]. Using the phases as follows:

*Plan*

Basing on current results and new research ideas, the Plan-Phase is used to analyze and formalized the current state of the research. This generates new approaches and provides theoretical input that can be applied within case studies. Work in this phase formally contributes towards answering the research questions.

*Do*

The work during the Do-Phase focuses on applying insights gained during the research in order to advance the case studies and generating measurable results. Furthermore, this possibly arises issues that cannot be solved using the current architecture.
1.3 Research Outline

Check

During the Check-Phase presentations and articles are used to review the current state of research and present it to peer researchers. Discussions with peers allow identifications of gaps and future research goals and case studies.

Adjust

The Adjust-Phase then is used to integrate the current state of research into the implementation of the reference architecture. Basing on the feedback from use cases and publications, the implementation thus incrementally converges to reflect the current state of the research.

Figure 1.1 shows a selection of milestones from the course of research according to the PDCA-phases. The starting point was the introduction of a university mobile app at RWTH Aachen University: *RWTHApp*. The application required an infrastructure to access data from various systems and additionally itself became a platform supporting several eLearning processes. Various presentations showed positive feedback within the community and motivated formalization of the created infrastructures for eLearning process support. While eLearning processes primarily targeted students of the university, additional processes required supporting various groups of users within the organization and across organization boundaries. This scalability across organizational and process boundaries eventually allowed the architecture to capture other process areas like eScience.
Figure 1.1: Visualization of the course of research
2 Reference Architecture for Process-Aware Services

This chapter introduces the core concepts of the reference architecture, then it introduces a formalized model and discusses and highlights its properties. This discussion focusses on how the model supports process-awareness and security for distributed services. In accordance to the model a reference SOA is constructed.

By introducing tiers of independent services, the reference architecture can be used to model processes on top of a SOA. Higher level tiers base upon the lower levels but may also introduce new back end systems into the architecture. Central leverage point for (user centric) security is tier 0: Authorization. On top of that, tier 1 then allows storage of small amounts of data for each process and user creating a session store or cache. These two tiers are shared among all processes and form a basis for adoption of back end systems into processes. Higher level tiers are then used to step wise abstract and accumulate functionalities of back end systems. The top tier of the architecture then uses the available services and lifts them to a process level forming a process-aware services.

2.1 Current IT Service Landscape at RWTH Aachen University

Like many other universities, RWTH Aachen University operates a heterogeneous and decentralized landscape of IT services. The different services are operated, provided and supported by several organs of the university and used throughout various processes in the area of teaching, learning and research. Currently there are three central providers of IT services at the university: IT Center, University Library, and University Administration additionally many institutes operate their own IT services usually offered only to limited audiences like their employees or students taking a certain lecture. Centrally provided services likely use managed identities, whereas decentralized offers, usually avoid the efforts
necessary to integrate with centrally managed systems and therefore maintain their own user directories.

Generally it can be observed that there is a trend towards centralizing basic services, like email or web servers. Specialized solutions that are tightly integrated into processes at the institutes however remain to be operated and developed decentralized. Initiatives to formalize reporting and governance, however, continuously demand integration of information collected in decentralized systems into central infrastructure. Process supporting IT services should orient towards existing processes of the university. In the project for strategic development of teaching and learning, RWTH Aachen University developed a process map for the core processes of the university: *Teaching and Learning, Research, Transfer and Further Education*. As Figure 2.1 shows, the focus in this project is the core process Teaching and Learning.

The work in this section partially bases on previously published research. Most notably the different properties of eLearning and eScience services, their applications and processes was previously discussed by Politze *et al.* [101]. The current IT landscape of RWTH Aachen University is then analyzed based on the model presented in the paper.

![Figure 2.1: RWTH Aachen University: Map of Processes](image-url)
2.1 Current IT Service Landscape at RWTH Aachen University

2.1.1 eLearning Services

Obviously IT systems play a supporting role within the process carried out by teachers and students. IT services supporting the core process Learning and Teaching are referred to as eLearning services. These services can be roughly separated into two areas: SLM systems supporting planning and organization and LMS supporting the execution of lectures and examinations. Figure 2.2 further breaks down common areas of responsibility into a student-life-cycle. At RWTH Aachen University, systems supporting these processes then need to interact to provide a seamless integration and transition between organization and execution processes as shown in Figure 2.3. An analysis of existing eLearning services then allows deriving properties of current process supporting IT services at the university:

Users of the services

Users in eLearning applications consist of students and teachers, all associated with the university. Typical lectures have 20-100 participants, though sometimes number up to 1500 due to increased cooperation and new styles of teaching such as, Massive Open Online Courses (MOOCs) which require participation of external students.

Kind of data

Files that eLearning systems deal with are commonly small and standardized, like slides, scripts or homework. In addition, 1-2 GB video recordings have become common in recent years. Commodity software and hardware handle them quite well.

Kind of services and applications

Services like distribution of learning materials and contacting students are a key success factor of integrated eLearning systems. Available LMSs allow teachers to communicate with the right audience: students participating in a lecture. Individualized services and applications like online programming environments or audience response systems reuse these groups. Most of the services are web applications or mobile apps. The systems available for SLM walk the students from their matriculation through the enrollment for lectures and exams to their graduation.
Maturity of the processes

Teaching styles and didactic methods can vary, while student lifecycle and learning management processes themselves are well understood and clearly centralized. Most faculties and institutes have coordinators to develop teaching methodologies. Existing IT services enhance already existing processes.

At the time of writing, the core process of teaching and learning at RWTH Aachen University are supported by four specialized and centrally provided systems:

- **CAMPUS** supports management of lectures and examination rules,
- **CAMPUS Office** supports lecture registrations and as a front end to SOS and POS for students,
- **SOS and POS** supports management of admissions, registrations for exams and grades,
- **L²P** supports execution of lectures during the semester

Additionally, smaller applications integrate support several sub-processes and attach to the established infrastructure. This IT service landscape, however is not entirely static. For several years RWTH

![Student-Life-Cycle of RWTH Aachen University](Image)
2.1 Current IT Service Landscape at RWTH Aachen University

Aachen University is in the process of requirements engineering and migrating to replace the campus management systems mentioned above by RWTHOnline [104]. Also the eLearning system is subject to constant change and has been completely replaced while carrying out the research for this thesis [103], [105]. After all, this shows the high level of dynamics for each of the IT systems and therefore for the entire IT landscape.

2.1.2 eScience Services

Opposingly to eLearning services, eScience services support the Research core process. Reconsidering their illustration in Figure 2.1, from a centralistic point of view, this core process is less clearly defined as the previous example of Teaching and Learning. Even though not as clearly defined, processes are already supported by IT services, eScience services, as well. By considering the previously discussed properties it is possible to point out the difference and the common core of eScience and eLearning services:

Users of the services

National and international collaboration is mandatory for many research projects. While there are research groups with as few as ten members, regional, national and international collaborations may have far more than 100 researchers from various universities and research facilities. Number and diversity of users thus are equivalent to previous scenarios.
2 Reference Architecture for Process-Aware Services

Kind of data
File formats and software used by researchers is very heterogeneous and mostly specialized on the scientific disciplines. Research data varies greatly in file sizes and count. Nevertheless, research also involves documents like papers, grant applications, and dissertations that are well-standardized.

Kind of services and applications
Centralized services for researchers mostly focus on the archival of research data and publishing of results. Instead of the researchers themselves, technical personnel often use these services. There exist a lot of decentralized discipline-specific applications, services and implementations not connected to centralized systems. Offering APIs would allow the integration of discipline-specific tools into generic services.

Maturity of the processes
Research groups essentially follow their own research processes. Connections to university administration are rare and mostly related to funding. Developments of the research process remain within the research group, and few coordinate with other research groups. IT service providers should thus promote communities developing eScience methodology.

Existing eScience services can be further aligned to a research data-life-cycle shown in Figure 2.4. However, when compared to process supporting services in the area of eLearning, there is no continuous support for all phases. In many cases researchers are supported by an individual selection centralized and decentralized services. The available services often support only specific workflows within the phases and users typically have to transfer their data manually between systems.

The centrally provided services in Table 2.1 were mapped to the phases of the research data-life-cycle. It becomes clear that there are gaps in the overall continuity of process support. For example SharePoint provides rich document editing capabilities and detailed user rights management. Both features are viable during the Planing and Access phases. However the system is not capable of storing big quantities of files, making it
not suitable for Analysis or Storage. This gap can be filled with other services but this often requires researchers to manually copy their data artifacts from one system to the other. While integration of services can be achieved for centralized services, distributed local instances remain isolated.

Services in the eScience area also have a notably different level of process support and usability. While on the one hand some services, like archive or file servers, can be rightfully considered basic and apathetic to the processes that base on them, services like Research Data Management Organizer (RDMO) on the other hand are dedicated to the Planning phase of the research data-life-cycle. A similar ambivalence can be recognized in terms of user interfaces: file servers are typically integrated at operating system level, the archive service requires a proprietary command line tool to be operated and RDMO is a web application that is accessed using a web browser. Integrating these services therefore also needs to overcome technological gaps between services.

2.1.3 Introduction of Case Studies
To validate and show the effects of architectural properties several case studies were selected and are referred to throughout the following sections
Table 2.1: Centrally provided services supporting different phases of the Research data-life-cycle

<table>
<thead>
<tr>
<th>Service</th>
<th>Planning</th>
<th>Production</th>
<th>Analysis</th>
<th>Storage</th>
<th>Access</th>
<th>Re-Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDMO</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SharePoint</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RWTHPublications</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Archive</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>File servers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Databases</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GitLab</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

and chapters. These case studies were implemented to as supporting services for different core processes of the university. Therefore, a total of six eLearning and eScience services are considered. To additionally assess the flexibility of the architecture two case studies supporting more general processes are also part of the selection.

As a platform for mobile access RWTHApp integrates a rising number eLearning services in one application since 2014. Among access to core features of the SLM system and LMS, RWTHApp therefore provides access to eLearning services that take special advantage of the mobile and connected devices and their extensive spread student among students. The first case study is prototypical to RWTHApp and focuses on the integration of distributed information systems, namely the SLM system, the LMS and the library system on the mobile platform. The other two case studies in this area focus on the implementation of specific eLearning scenarios that use RWTHApp as a platform: Quizz2Go allows individual students to take quizzes that can contain various multi media content. The quizzes are saved to the device and can be directly evaluated. Using a back end system, students can compare their results. The Audience Response System is intended to support teaching in lecture halls for large
scale lectures with more than 500 participants. It provides a messaging interface that allows students and teachers to send texts, photos or polls before, during and after lectures.

Instead of integrating into one platform, eScience services consist of multiple small applications that focus on individual parts of the scientific workflow. Within the first cast study in this area HIApp models a discipline specific data management workflow that allows citizen scientists to access and create current research data from a centrally provided collaboration platform using a mobile application. In contrast to this individual approach, the Metadata Tool takes a more generic approach to data management. Using a graph database as a back end the application allows storing discipline specific metadata. While the Metadata Tool only hold references to the data, the third case study, simpleArchive allows storing of actual research data. Both case studies feature a basic application that provide access to users but focus on modeling the process in a way that can be integrated into the individual research infrastructure of the user. The case studies put an additional emphasis on consistency to allow integration of both services.

Based on the experiences in the previous case studies two more general processes are supported: Chat Support provides an application that allows users to contact support personnel of the university. The processes are shared by support agents of the IT Center and the University Library. Technically the case study reuses existing services from the Audience Response System. Last but not least Eduroam Device Manager provides an application that creates secure eduroam credentials. The case study focuses on how services can be integrated that are not web based applications.

2.2 Conceptual Foundation

As previously shown, the universities’ students and employees already use IT infrastructure to digitally enhance their processes. Thus, the starting point for a process supporting reference architecture is to analyze the existing IT landscape at RWTH Aachen University and derive a model that can later be enhanced using the reference architecture.
Current IT systems are situated in a heterogeneous landscape featuring a variety of independent processes, services, service providers and infrastructural components. Some decentralized services are offered by different institutes of the university as well as other universities or external contractors and may be available only to a limited number of users. Each service features various access modalities to regulate availability for the users.

When students or employees combine the available systems to follow their processes, data often needs to be transferred manually or is being entered multiple times. Interconnecting the systems to use the same identities and to automatically convey data from one system to another requires great effort for each system individually. Only university wide processes, having a critical number of users, are thus supported by such integrated solutions. Based on the requirements posed to the heterogeneous IT landscape at Universities a conceptual model can be derived. Apart from ready-to-use applications, this model allows users to extend and integrate processes in their own applications using the provided API. Exposing a programmable interface to all users, however, poses very strict requirements to the implementation:

1. The interface has to be comprehensive for users (especially external application developers),
2. it must not allow abusing the supported workflows within the process and
3. as application cannot be controled it has to be secure by design.

If the API is not specific, it may invite users to use it for other workflows than intended. While this may be beneficial for the short term interests of the user, IT service providers usually desire the usage only within the specified parameters, having to meet technical bounding conditions such as CPU load, disk sizes, and the like. Enforcing these boundaries within the interfaces is crucial for failure free operation of back end systems but may also limit the possible degree of individual adoptions. Allowing users programmatic access to services, potentially exposes personal and confidential data in a machine readable format. As strictly reviewing all applications implemented by users usually is considered infeasible, security becomes an important design principle of the API.
To close the gap between distributed services and spanning processes, Mu et al. have created a collaborative process elaboration methodology [63]. While their focus lies on collaboratively editing the processes, however, within the distributed system landscape the focus lies on actually achieving the cooperation and building a common understanding of the processes. The methodology in Figure 2.5 slightly adopts the approach of [63]: For each service provider the conceptual model needs to be extended looking at three viewpoints: organizational, functional, and informational. The main goal of the organizational view is to gather the goals users try to achieve using the system. Using this information the functional view makes necessary functionalities accessible. Finally, the informational view gathers the semantics of used information entities. The created conceptual model can then be integrated into process-aware services.

### 2.2.1 Distributed Service Providers

Due to emerging Internet technologies, on-site resources needed to operate commodity IT services are decreasing. This allows even small organizations that do not operate their own computing centers to run IT services. Leased, and mostly virtualized, infrastructures are available from numerous commercial service providers. Offering Infrastructure as...
a Service (IaaS) or Platform as a Service (PaaS) has become a major
business area for these companies operating large computing centers for
tenants all over the world. These companies furthermore offer Software
as a Service (SaaS) models: by taking care of operation of the whole IT
stack and offering ready to use software for their clients.

Transferring this development to the context of universities and other
research facilities it becomes clear that organizational units within these
bodies have a variety of options to obtain IT services not only from in-
ternal but also from external service partners. Universities traditionally
are operating independent and discipline specific IT services. Therefore,
IT services available to students and researchers are traditionally of-
fered by a variety of institutions. Also, regarding external, and often
commercial, offers available within this portfolio poses an additional
challenge on organizational level, especially if these kinds of services are
to be integrated with each other.

From the users’ point-of-view, it often does not make a difference who
is operating the service as long as their intended processes are well
supported. The proposed reference architecture therefore aims at being
able to integrate the services offered by different providers to create an
overall SOA that can be used across boundaries of existing services. It
should thus consider both, centrally provided and external services. This
also implies that there may be more than one service offering a certain
functionality to the user. The architecture should be able to deliver
the best possible business value for the user, by combining available
services.

The distributed nature of services has some further implications: Gen-
erally there is no central coordination of the services: Maintenance or
outage times are often not communicated to dependent parties. Not
because they are not announced by the provider but often because
dependent services are not known to them. Each service provider likely
maintains their own directory of users. While technical solutions and
organizational processes exist to provision user information from one
system to another, operators often rely on a local database.

Defining processes and supporting applications in a heterogeneous land-
scape of systems is a complex task. The proposed reference architecture
2.2 Conceptual Foundation

should take these characteristics of distributed service providers into account to help users and application developers defining process across system boundaries. The created SOA should thus be able to incorporate distributed service providers in a way that minimizes the changes needed to be done by external service providers.

2.2.2 Process-Aware Services

APIs offered by existing IT systems are often focused on the technology used for the implementation. When integrating multiple systems into business processes, naming conventions and technical details will often differ between systems. Users of an integrated SOAs then need to know the correct mapping of the technical APIs offered by the system to business processes defined by the organization. This is, however, often not trivial since it requires both: an understanding of the technical details of the IT system and knowledge about the organizations’ business processes. The notion of process-aware services is an extension that adds the missing semantics of existing business processes to the SOA. Process-aware services therefore decouple implementation of process supporting applications from the technical details of the underlying IT systems.

For PAIS, which are mostly engineered from scratch to bring the best possible value within the supported business processes, Russel introduces five dimensions to judge the maturity of modeling [79]. For the case of a process-aware distributed SOAs, they have to be adopted slightly to respect the specific field of application.

The proposed SOA should embrace formal definition of business processes, their semantics and workflows. Of course this requires process analysis and modeling of the supported processes prior to implementing respective parts of the SOA. The dimension of Formality orients towards existing processes and therefore requires that they can be technically described by the SOA. For example Ouyang et al. propose to transform abstract definitions of processes from Universal Modelling Language (UML) or Business Process Modeling Notation (BPMN) to a representation that can be better enacted technically such as Business Process
Reference Architecture for Process-Aware Services

Exceution Language (BPEL) [107]. In terms of process modeling it can therefore be assumed that formal definitions are available in an enactable representation. What cannot be assumed is, however, that this enactment can take place directly on back end systems as their semantics differs from those of the business process. To satisfy the dimension of Formality it is therefore required that APIs of back end systems are lifted to the same semantic level as the enactable descriptions of business processes.

The second dimension, Suitability considers the ability of the SOA to capture existing and new business process without significant conceptual reorganization. More general this means that the constructs used by the SOA should correlate with those present in the process domain. Reusing technological insights when modeling new or changing existing processes thus is crucial. While orienting towards the business process semantics, to satisfy the dimension of Suitability a process-aware interface of the SOA therefore needs to be independently reusable without interfering with other interfaces.

Focusing on concepts that are directly relevant to the business process domain and independent of technical or implementation related considerations is the key of Conceptuality. The SOA should therefore hide technical details from the user. Modeled business processes should be portable across back end systems and technical environments. At the same time they also embrace different technologies used for end user applications. To satisfy the dimension of Conceptuality interfaces are therefore required to be exchangeable without interfering with the supported business process.

The dimension of Enactability claims that in order to provide business value, the SOA has to be enacted within the organization. Often this can not be dictated but users have to be convinced that basing their processes onto the process-aware services also offers them short term benefits. If users and application developers more likely use process-aware services than the back end systems directly, it provides long term benefits for the whole organization. To satisfy the dimension of Enactability short and long term goals for users and the organization
thus need to overlap in order to ensure acceptance of the SOA within the organization.

Ultimately, the SOA needs users and application developers to integrate the services into their applications. They should be able to (1) capture the concepts of underlying business processes and (2) apply their understanding of the process when using the architecture. Comprehensibility is especially important when the SOA is used to support previously existing processes: users should be able to transfer their knowledge about processes to the offered services. The SOA satisfies the dimension of Comprehensibility if process-aware interfaces follow the same organizational and technical structure.

2.2.3 Information Security

Processes at universities often deal with sensitive data. Looking at different areas from research to teaching, sensitive data then range from unpublished research results to personal data of registered students. Data security thus is an important concern in designing the SOA. Key is that an SOA can increase its security by being transparent about how data is accessed by different applications. This concept is mostly user centric: by becoming more aware of who uses their personal data users may be more conscious when using applications.

Data security can be measured in various dimensions that should be embraced by the SOA. The ISO 27000 norm family defines three primary and four secondary dimensions of information security [80]. Again these need to be adopted from the general description of information security management systems to the frame of the reference architecture.

The primary dimension of Confidentiality requires that information can only be accessed by entities, individuals, or processes that are authorized to do so. The SOA should thus provide the means to transport and validate such authorizations across system and process boundaries. Different modules of the SOA should in turn be able to resolve the entity accessing information from the provided authorization.

According to the primary dimension of Integrity, information provided need to be accurate and complete. This especially implies that stored
or transferred information should not be changed if not initiated by an authorized entity. Adopting this requirement to the distributed nature of the regarded SOA shows that communication between cooperating systems needs to conform to supported processes. As in the case of Confidentiality authorizations need to be transported across processes and system boundaries and modules need the means to resolve entities from provided authorizations.

The primary dimension of Availability requires that information are accessible and usable when requested by an authorized entity. In a distributed SOA that spans processes across system boundaries this dimension also measures how sensitive the architecture reacts to failures in single systems. It needs to be embraced by the SOA that such failures have only limited effects on unrelated processes.

Apart from these primary dimensions, secondary dimensions define minor security goals: Authenticity, Accountability, Non-Repudiation and Reliability. While the primary dimensions remain quite broad, the secondary dimensions hint at cases that need to be considered when satisfying the primaries:

- **Authenticity** refers to entities and ensures that they are what they claim to be. Authenticity therefore demands that Integrity is ensured for the current state of an entity and that Confidentiality is ensured when changing entities.

- The secondary dimension of Accountability refer to distinguishing different entities and the effect of their authorizations and actions within the processes and systems. Accountability therefore demands Availability and Integrity of accounting information about authorizations and actions.

- Closely related is the secondary dimension of Non-Repudiation that demands that accountable entities cannot deny their actions. Additionally to the previous demands on accounting information, also it needs to be ensured that requirements towards Confidentiality are met.
2.2 Conceptual Foundation

- Last but not least, **Reliability** measures the determination of the processes and their outcome. This requires that processes and actions meet demands towards **Availability** and resulting states should meet demands of **Integrity**.

Even though the secondary dimensions raise the most important concerns within the area of information security, it becomes clear that they are subordinate to the primaries. For the design of the reference architecture it can thus be considered sufficient to evaluate based on the primary dimensions only.

Additional to the security goals of the ISO 27000 norm family, **user centric security** as proposed by Feth is evaluated as another dimension within the area of information security [95]. As pointed out, users often are the weakest link in security processes. Within a secured process **user centric security** therefore assesses how security measures influence and are influenced by users. As Figure 2.6 shows, these originate from the intersection of security goals and usability goals.

To satisfy the dimension of **user centric security** the reference architecture should therefore consider the secondary dimensions:

- If security is hidden deeply within an application users may not be aware of the relevance of their actions. **Awareness** therefore demands that processes highlight security relevant steps for the user.

![Figure 2.6: Aspects of security and usability by Feth [95]](image_url)
• Complicated security measures might often lead to reduced usability but so do systems that are obviously not secure. The secondary dimension of Trust demands that systems should present themselves as secure as necessary in order to be trustworthy for the user.

• Secure systems can be restrictive to the user and therefore reduce usability. Perception demands that systems should show users the security implications of their actions but allow them to override if necessary to achieve their goals.

2.2.4 Separation of Concerns

The distributed services regarded within the context of the reference architecture are specified and developed by independent experts within the university. While this bears the risk of incompatible interests, wherever possible this distribution of responsibilities should be beneficially used to increase the quality of the supported business processes. Achieving a common level of software quality and maintainability is therefore another goal of the reference architecture. For example Bogner et al. show that Size, Complexity, Coupling and Cohesion are common criteria from software engineering and how they can be applied to micro service architectures [77]. The authors define them as follows:

**Size**

The size of a service is the size aggregate of its dependencies. Provided all other criteria are similar, a large service is harder to maintain than a small one.

**Complexity**

The amount and variety of internal work carried out by a service as well as the degree of interaction between its dependencies necessary to achieve this. High complexity has a negative influence on maintainability.

**Coupling**

The degree or of dependencies of a service with other services. Services aggregating fewer and more loosely coupled services can be easier maintained.
2.2 Conceptual Foundation

Cohesion

The extent to which the operations of a service contribute to one and only one task or functionality. A high degree of cohesion positively impacts maintenance activities.

However, the authors also state that capturing metrics in distributed systems is especially challenging due to the lack of internal information about the systems. While process-aware functionalities derive from the combination and intersection of different services, each service typically forms a black box and only publishes an interface to be used by others hiding internal details about their actual implementation.

A more axiomatic approach, SoC, is proposed by Tarr et al. [71]. Generally SoC describes how responsibilities for different components within a system overlap or complement each other. From a point of software design, the SoC paradigm therefore prescribes abstraction and dependency injection to separate responsibilities and concerns to create independent modules. SoC actually is an inherent property of distributed systems as well as SOAs. However, Tarr et al. show that separation is not implicit but needs to be assured within the architecture. The authors therefore introduce three dimensions to measure SoC in a software system which can be adopted to the process-aware setting of the reference architecture. By following the principles of SoC it should thus be possible to satisfy also the criteria shown by Bogner et al.

The dimension of Impact of Change measures how additive changes in one service lead to invasive changes in others. This is especially the case whenever services use different kinds of abstraction and therefore have different units of change. Considering a process-aware SOA there are two kinds of changes to be considered when evaluating Impact of Change: (1) due to modification or adding of processes or (2) due to replacing or adding back end systems supporting the business process. If the architecture effectively controls the Impact of Change, in both cases adding new processes or back end systems should have little to no impact on the opposing side. In terms of the common criteria, Impact of Change assess the amount of work and dependencies between process-aware services and back end systems. Controlling Impact of Change therefore limits Coupling and Complexity.
Enabling and embracing reuse of modules, code, design decisions and requirements is concerned by the dimension of Reuse. As modules grow larger and incorporate more and more technical details and features their reusability often is reduced. Adopting the dimension of Reuse to process-aware SOA spreads this definition to reuse of processes, their concepts and semantics. The same is naturally valid for back end systems: their reuse is not only desirable from the point of software architecture, but also economically. Reuse therefore controls interconnections of services and back end systems and limits the number of implicit functionalities of a service and therefore reduces Coupling and increases Cohesion.

Taking into account the different levels of abstraction and technological choices within modules, the dimension of Traceability captures the correspondence of abstraction across different modules. It allows determining how many modules are effected by a single change by tracing changes through the system. As previously with Impact of Change, Traceability thus allows measuring the effect of changes in processes or back end systems. It is, however, not limited to additive changes. In this case it is especially important to limit effects of different technologies used by back end systems to reduce the risk of vendor lock in but also allow processes to change or evolve over time. Since Traceability demands transparency of effects of changes it makes Complexity auditable within the architecture. For back end systems integrated into the architecture it fosters Cohesion by demanding abstraction of specific technologies towards general tasks performed within the architecture.

2.3 pSTAIX: A Process-Aware Tier Architecture

Basing on the conceptual model, Process-aware Software Tiers for Application Interfaces and eXtension (pSTAIX) is constructed to be a reference SOA, abstracting decentralized system and technology depended interfaces to process-aware services.

Apart from technologically structuring services, it also formalizes the transitions of IT systems into process-aware services offered to external application developers and end users. As technology dependencies
decrease and services orient more towards defined business processes they can be easier understood and require less technology dependent knowledge about back end systems. Each software development and IT operation team responsible for a service then ensure the functioning of a defined set of interfaces for the architecture. They can therefore focus on development and implementation solely in their area of responsibility. Clearly defined interfaces between the tiers, as introduced by the microservice pattern, allow teams to use different technologies and even to exchange technology stacks while interfaces remain.

This architectural model was initially introduced by Politze et al. [108]. While the paper focuses on the application for eScience the following section introduces the pSTAIX architecture as a general concept. Additionally, the relationship to the different dimensions of the conceptual foundation shows the inherent properties of the reference architecture.

2.3.1 $n$-Tier Architecture

The pSTAIX reference architecture partition services into tiers to allow separation of concerns; especially in terms of separating technology and process dependent elements of the implementation. Each of the tiers then bases on the interfaces offered by the previous tiers and gradually combines them to process-aware services. Figure 2.7 shows an overview of the relationships of different tiers in the reference architecture. Technically the architecture makes use of the mediator pattern as presented by Wiederhold et al. [57]. Process-aware services are thus composed of more general, process independent services and back end systems from the lower tiers. Depending on their location within the architecture, services then take different responsibilities.

Tier 0: Authorization and Security

The authorization tier forms the basis for general security. It provides means to identify the current user, applications and their roles within the organization. By providing application level security as a service, services in this tier are the foundation for all other tiers. As almost
every process requires identifying their users. Once processes span across multiple systems boundaries, it needs to be assured that users can access all spanned systems if they are allowed to do so. A centralized Identity Management (IdM) is crucial to provision necessary user information to systems. To allow processes to span across distributed systems, users’ identities and sessions then need to be conveyed between the systems participating in the process.

This tier, however, only supplies and manages information about the users’ identity and current session. Systems are generally required to be capable of handling their own set of permissions. Whenever possible, tier 0 services should only provide enough context about the user such that services in the higher tiers can decide about their actual permissions.

Development and operation of this tier is crucial for other services. Aggregation of information about identities, groups and roles for higher level tiers makes tier 0 services essential for data security and data privacy. Services in this tier should focus on state-of-the-art technology to protect and govern usage of personal data. On the other hand this separation allows services in the other tiers to concentrate on their core duties as several aspects of security are already handled by tier 0. This allows decoupling of technical services in higher tiers from how users’ sessions are handled.
2.3 pSTAIX: A Process-Aware Tier Architecture

Tier 1: General Services

Services in tier 1 offer basic and process independent functionalities and therefore support several operations across process boundaries. The services may be used directly by applications but also by services from the higher tiers to offer more specialized and process-aware functionalities.

Process-aware services created in the higher tiers often require saving session states or small amounts of user data, like settings or session data, to be preserved between steps of the process. To reduce the need of individual storage solutions, tier 1 introduces persistent storage for small data quantities. Furthermore, this tier provides a centralized logging service that can be used for audit and error logging across system and process boundaries. Additionally, a temporary storage to cache repeated requests is crucial, since legacy back end systems are often not designed to be accessed interactively by users and may be slow when put under high load. Also, lifting services to process level may sacrifice efficiency such that requests to back end systems need to be done multiple times. Short term or temporary storage is then used for caching of repeated requests and therefore increases the performance of the overall architecture.

These three categories of data stores then operate either on a user or process basis for private or shared data respectively. Tier 1 services are likely to use different technologies to separate temporary from persistent data. If temporary storage needs the ability to cache files prior to processing, it may need to handle medium quantities of data in the range of a few gigabytes per user. It should however be kept in mind that services in this tier are generally intended for small data quantities, in the rage of several kilobytes per user. If back end systems need to store more complex or higher volume data these more specified services should be provided by upper level tiers.

Tier 2: Technology Dependent Services

Existing back end systems provide the core functionality of the process-aware services. To make these systems accessible for services within the
architecture experts operating, maintaining or developing the system use its internal interfaces. From the point-of-view of the other services, except for these interfaces, back end systems remain a black box. Opposed to tier 0 and 1, services in tier 2 are very specific as they orient towards existing systems and depend heavily on the technological details and interfaces of back end systems. However, they remain generic since they are not explicitly designed to support or endorse specific processes but simply allow programmatic access to single back end systems.

Especially in case of legacy systems, this tier often provides administrative or root access to the back end system or may allow impersonating arbitrary users. The interfaces therefore are commonly not available to end users but are used by technical personnel for administrative purposes. Vendors of the systems providing services in this tier likely deliver these interfaces, and they should only be individually extended if operators are missing necessary functionalities. Modern systems may additionally allow resolving the user’s session using tier 0 services by using plug-ins or specific extensions. If possible, this should be the preferred approach.

Apart from the semantic gap between these interfaces and the processes perceived by the users, APIs used in tier 2 are often low level and require expert knowledge to be operated appropriately. These issues, however should be addressed in the upper level tiers.

**Tier 3 to n − 1: Standardized Access to Backend Systems**

As technology advances, tier 2 services, are likely to be changed or may even be moved between internal or external organizations or contractors as operators as they depend on technical aspects of the back end systems. The tier architecture, however, limits the impact produced by this kind of frequent changes by introducing tier 3 services. While tier 2 services change abruptly when the back end system changes, services in tier 3 can change using more evolutionary processes, paving the way for more elaborate software engineering methodologies.

To form a basis for process-aware services, tier 3 introduces standardized protocols and semantics spreading across system boundaries. If not
supported by tier 2, these interfaces furthermore enforce personalized use and therefore do no longer offer the ability of impersonating arbitrary users, but work in a users’ context and thus enforce personal or role based access and permissions. By taking into account nomenclatures, identities, groups and their roles, this effectively integrates the back end systems into the organizational structure of the company or university.

Due to their dependency on tier 2 services it is very likely that tier 3 services also need to be engineered by experts having knowledge about technical details of the back end system. Abstracting these technology dependent interfaces makes the functionalities of the back end system available to a wider group of technicians and programmers. Additionally, tier 3 services should limit the capabilities of interfaces if necessary to ensure disruption free operation of the back end system. This definition of a service level makes sure that technical or bounding conditions like CPU load or number of operations are limited to defined thresholds. Quality of service assertions may further validate organizational thresholds like user limits for storage capacities.

In higher tiers intermediate services may depend on those in lower tiers. It is however important to note that cyclic dependencies of services need to be avoided to preserve maintainability of the system. As tiers further abstract technological dependencies and gradually allow more goal oriented implementations. Together with the defined service levels for lower tiers only little or even no expert knowledge is required for their implementation.

While the access protocols in these tiers are standardized, entities, nomenclatures and semantics may still orient towards the technical dependencies. If not limited by the service levels, tier 3 services thus should remain widely process independent. Lifting the semantics to business process level is addressed by tier \( n \).

**Tier \( n \): Process-Aware Services**

In higher tiers the specialization of services is gradually increasing from services delivering technical means of accessing a system towards processes of the organization. After standardizing the transport protocols
and technical semantics, this tier aims to combine services from previous tiers into process-aware services. It therefore entirely abstracts from previous technical solutions to organizational and business process levels. Most importantly, this tier introduces consistent naming and semantics across processes and system boundaries and bundles mandatory or dependent process steps from different back ends. Services from this tier are intended to be provided to end users or application developers respectively as they can be used for individualization, automation and integration of processes in their own applications.

End User Applications and Processes

Last but not least, end user applications are on top of the reference architecture. These applications are needed for end users to actually carry out the processes. The previous tiers focused on programmatic access to processes, services and back end systems. In turn, applications should thus use tier 4 services to provide actual business value for end users. Since services are process-aware, it is easy to implement applications guiding the user through a single or multiple processes. Furthermore, applications should make use of consistent semantics offered by tier 4 services to span across process boundaries.

End user applications therefore provide the specific model of the process that is tailored to their specific user group. Different applications may reuse the same services from tier 4 to present variants of processes to their users. Due to the standardized access and inter-compatibility of tier 4 services, applications can make use of advances process modeling end execution capabilities for example provided by BPEL.

2.3.2 Application Life Cycle Management

The general term Application Lifecycle Management (ALM) describes a set of processes supporting the idea to develop a software product, the initial deployment and finally the end-of-life when retiring the software product if it does no longer have business value. Additionally, Chappell further introduces three aspects that span these life cycle events: governance, development and operations. Within these areas
ALM processes guide through sequential, and often repeated, steps of requirements analysis, development and testing [109].

While pSTAIX provides an architecture for such a software product, it furthermore takes into account that (sometimes specialized) IT systems and hardware are required to support applications and services. Business processes are then needed to make these systems available within an organization. In the following ALM thus is meant to also cover the incorporation and operation of IT systems and the adoption of changed business processes into the software product.

Since the means provided by the different tiers are diverse, it is likely the software used for their implementation is selected based on local requirements. Therefore, the architectural model does not only provide high level guidance on structuring the system as a whole but also encourages software life cycles to be well-defined for all tiers. The distributed nature of services then results in distributed and independent services and therefore allows separating their life cycle. The SOA pattern used by the architecture applied to the tier structure defining the process-aware interfaces.

Within any one of the tiers, several independent services may exist. Therefore, the actual application life cycles also differ according to the requirements of the software and hardware products used for the service. Especially in technology dependent tiers, life cycles should be oriented towards the technical or vendor specific requirements of back end systems and thus might be defined rather by vendors than the service provider. Abstracting from technological to standardized services allows more independent and individualized ALM methods for services in higher tiers. In the top tiers, life cycle management should then orient towards the requirements of supported business processes. This allows different handling for static environments that require high reliability or changing demands and fast time to market. This takes into account the different stability and innovation requirements that can be found for example in administrative compared to eLearning context at universities.

In contrast to the software used, the life cycle of interfaces between the tiers is more critical. A change of an interface might induce changes
in the dependent services in the tier above. A propagation of changes across several tiers should thus be avoided. Using the abstraction levels between different tiers, the reference architecture mutes the effect of such changes. Replacement of a back end system therefore requires changes in the tier providing standardized access for the system only. Processes depending indirectly on the back end system, however, remain intact. Decoupling and reducing the number of effected systems, gives the flexibility needed to react to changing demands.

In fast changing and critical environments, ensuring the ability to independently test different parts of the system, is an important requirement demanded by application life cycle management processes. The separation of tiers delivering specific functionalities, allows forming separate test cases in order to validate proper functioning for each tier and service separately. Based on the specification of interfaces connecting the various services it is then possible to define mockups to replace services from lower tiers and specifically test and validate services in a single tier. Combined with tests confirming the compliance of interfaces with the specification this approach allows iterative validation of the available services and dependent processes.

2.3.3 Auditing and Governance

Implementations using the pSTAIX reference architecture likely support a variety of business processes. It is therefore only natural, that users and service providers need to gain insight on how well services within the architecture support their business needs and provide actual business value. Implementing centralized means for auditing and governance reduces the urge for every service provider to collect respective data. Among the obvious technical advantage this also allows central enforcement of data privacy and protection laws and best practices. From the process-aware setting two auditing levels can be identified: process and technical level. While the process level is likely more interesting for the users, raising awareness of technical impact of their usage might lead to more sophisticated use of the supplied services.

Obviously gathering of technical performance indicators is specific to the different services. As previously discussed for the case of ALM, services
need to define the extent of technical audit logging independently as it may be required for gathering specialized indicators. This also allows distinguishing between critical or confidential logging on a per service basis. For general purpose technical and process logging, the centralized authorization tier enables to capture several standard cases to record audit logs by making use of mechanisms for passing user authorizations between systems or process steps.

Processes supported by the architecture often base on personal data that is passed between different services. As the services are becoming more decentralized, the control of transferred, processed or saved data in the different services decreases. It is thus not entirely transparent when and by whom personal data is used in the processes. Especially for asynchronous or background processes, that are not invoked directly by the user, it remains opaque when and what kind of data is accessed. This is equally true for applications: The design principle of pSTAIX allows and endorses user defined, applications to access the process-aware services in order to adopt or integrate them in existing applications. Applications themselves, however, cannot be checked individually for security, the architecture has to provide basic security measures to protect personal data within the process-aware services.

Authorizations validated by tier 0 should thus convey at least information about user, process and application used. Whenever a service from any of the tiers verifies an authorization, it is thus possible to record the associated auditing information. This captures at least which processes are accessing services on behalf of a user and how application developers make use of the process-aware services. Based on this auditing process it is possible to build a user centric approach to application security. The audit data collected is therefore not only available to service providers but users are allowed to access their own audit data enabling each user to validate the actions taken by the applications within supported processes.

In contrast to general audit logging, error and diagnostic logging has to be performed by services themselves to support local error detection or enhancements of individual processes. If implementations of pSTAIX add a centralized logging service in one of the lower tiers this allows
service providers to gain insight into dependencies of processes and shows propagated errors of dependent services and provides the ability to trace effects of errors or other issues across system boundaries.

2.4 Evaluation of the Reference Architecture

Basing on the tier model presented in the previous section, it is possible to evaluate general properties of systems using the reference architecture with respect to the dimensions of the conceptual foundation. While the actual pSTAIX architecture is technology agnostic, some properties can only be assessed by making implementation specific decisions an implementation of the reference architecture therefore also needs to follow the dimensions to fully satisfy the posed requirements.

The higher a service resides in the architecture, the more specialized it is for certain processes depicted in Figure 2.8. In general \( n \) may be arbitrarily large: thus services may be reused in higher tiers as back end systems. In this section \( n = 4 \) will be considered as reasonable for most SOAs with:

- Tier 0: Authorization
- Tier 1: General services such as storage that are using tier 0
- Tier 2: Technology dependent back end systems that may use tier 0 and tier 1
- Tier 3: Standardized access to back end systems using tiers 0 \ldots 3
- Tier 4: Complex process-aware services using tiers 0 \ldots 3
- Applications using the services from tier 4

This minimal tier architecture furthermore follows the distributed process elaboration methodology by Mu et al. [63] as previously shown in Figure 2.5: Tier 0 and 1 provide technical support to allow inter-organizational cooperation. The distributed design principle of pSTAIX furthermore reduces organizational issues towards building a common understanding of user objectives. Tiers 2 then provides the functional view for back end systems supplied by the distributes service providers that are integrated into the architecture based on the needs of users’
2.4 Evaluation of the Reference Architecture

Figure 2.8: Tiers and their properties

business processes. Tier 3 then generalizes the functional dependencies and provides the informational view. Finally, tier 4 and implemented applications provide the process view using explicit nomenclatures and semantics defined by business processes and make the functionalities available to the users.

Properties discussed in this section are assessed on the mere design of the reference architecture and are thus independent of actual implementations. This is done to identify dimensions from the fields of process-awareness, information security and separation of concerns that are already covered by the architecture itself in contrast to those properties that additionally need to be addressed by implementations of the architecture.

2.4.1 Dimensions of Process-Awareness

In general, IT system providers have hardly any influence on the technologies provided by back end systems. Accessing these specific APIs is often not trivial, as they vary greatly based on the different types of targeted systems and vendors. Especially legacy software often does not document these interfaces very well and only few specialists within the organization know how to use them. The overall orientation of the reference architecture towards process-awareness tries to close this gap
of understanding since interfaces provided use state-of-the-art internet technologies and clearly defined semantics. Both of which, technologies and semantics, are then already known to a wide range of users and application developers and thus increases conceptuality and comprehensibility of the services offered.

The conceptual model combines existing back end systems into an overall SOA spanning across system boundaries and thus forms a basis for the development of more complex IT services. The top tier enables access by abstracting vendor specific or legacy APIs of back end systems to process-aware services. As required by the dimension of Conceptuality, the built-in abstraction forces the top tier to use the same semantic level like in the defined processes.

With every supported process in the interface, the complexity of an API increases. Explaining a complex interface to users thus becomes more challenging. Appropriate and consistent use of names, matching the definition and wording within the defined process, therefore is crucial and allows users to transfer their knowledge from taking part in the process to their implementation. Existing IT services, however, often do not offer such a process-aware view but provide a much more general, technical, interface and thus lack Comprehensibility. By lifting the technical services to consistent process semantics the complexity is hidden from the user.

The process-aware top tier mitigates the dependencies to technologies used in the back end systems. Using the temporary and persistent stores from tier 1 then allows to construct process workflows without directly depending on back end systems. As demanded by the dimensions of Formality and Suitability, combining services from these two tiers then allows to implement existing processes on top of the services from tier 2.

Especially in a distributed context the dimension of Enactability has two facets: from the users and service providers point of view. Combining multiple services into a consistent architecture makes adoption or integration of further processes easier. Users, and especially application developers, do not have to invest multiple times in initial training,
needed to get used to the supporting systems. For operators of distributed systems the threshold however is much higher. Incorporating a system into the architecture, requires analyzing the processes that the system currently is or will be part of. These processes then need to be matched with already existing processes in order to keep consistent semantics within the process aware tier. Technical integration into the tiers, however, is supported very well.

2.4.2 Dimensions of Information Security

Services in tier 0 aim to convey user contexts and their authorizations between process steps and involved systems. The mechanisms used, thus are crucial for the information security of the whole architecture. Enriching the user context with information about the current application and process, commonly required by auditing mechanisms, also fosters information security.

Passing user sessions between process steps and thus between different systems is one of the key concepts within pSTAIX. The aforementioned user context thus needs to enable back end systems to resolve the users identity to ensure that users can only access or change digital resources according to their duties in the supported process. However, this only provides the means to ensure Confidentiality within the processes supported by the architecture. It remains crucial that actual implementations take the user context into account. Most back end systems allow the notion of different users and offer practical ways to isolate actions of different users. This implies, however, that organizational roles from processes are mapped to the back end systems. Wherever possible, the back end systems should thus discriminate users in tier 2. Lifting user specific access to higher tiers gives room for errors in custom code used in tier 3, but may be necessary if back end systems do not support this kind of impersonation at interface level. Leaving impersonation to higher tiers in turn decreases re-usability of the services offered by tier 3 and should thus be avoided.

Ensuring integrity of provided information strictly depends on back end systems to handle the data correctly. The challenge thus is to ensure
that data passed between tiers and services within remains intact in the way the user described it. The dimension of *Integritey* thus has two facets: (1) providing technical means to protect from tampering and (2) ensuring that provided interfaces only allow valid access to the data and cannot be used for unintended purposes. The first facet can be tackled by using a technology for interfaces that validates the identities of communicating parties. Several standard protocols exist that secure interactions between systems and satisfy the needs. In turn, the latter facet has to be ensured by an implementation of the architecture. From a structural point-of-view, however, explicitly tailoring services in tier 4 to certain processes then allows ensuring correct usage and effectively limits possibilities to abuse services for other means. Additionally the audit logging service of tier 0 may allow tracing actions that alter resources using services of the architecture.

Many of the information services, especially those using legacy technologies, that are becoming part of the infrastructure are not designed to be accessed programmatically. This issue becomes even more important as more users access the infrastructure from their individualized applications. Certain methods of the API may be computationally intensive. While centrally provided applications often know the background information systems, this knowledge is usually missing for external users. Usage of caching services in tier 1 presumably reduces the load on information systems and allows faster handling of repeated requests. As demanded by the dimension of *Availability*. Intelligent cache implementations may even allow fetching chunks of information before the user requests them and thus provide even faster access to digital resources. An implementation additionally needs to ensure that failures in a single back end system do not lead to failures in processes that are functionally independent.

### 2.4.3 Dimensions of Separation of Concerns

The proposed reference architecture uses a service oriented pattern. As such, the expectations in dimensions of SoC are high. Following this pattern each service within the tiers is operated independently from one another and the different levels of abstraction further allow functional
independence of the services. Clearly defined interfaces allow replacing and reusing them within multiple processes. Authorization and general services for storage can be reused across multiple processes. Focusing on the interfaces between services within the different tiers further allows reusing them for a multitude of processes.

To assert these properties, the dimension Impact of Change demanded that effects of adding processes or back end systems are supposed to be minimized. To achieve this, the architecture introduces tier 3 services that build an abstraction layer between processes and the respective back end systems. Both tier 2 and 4 can incorporate new services without interfering with existing, already established, structures. Since cyclic dependencies are to be avoided for implementations, the tier architecture limits Coupling and Complexity only in terms of aggregation of lower tiers.

Back end systems are selected based on the process goals of the users. As such the architecture and the proposed decentralized process elaboration method lead to back end systems that have clearly defined responsibilities. To satisfy the dimension of Reuse, tier 3 additionally introduces defined service levels to make features of back end systems available in multiple processes. The same hold for process-aware services in tier 4: Applications can integrate process steps into existing workflows and therefore connect them with defined business processes to create short term benefits for their users. Making service levels and interfaces explicit in tier 3 thus increases Cohesion.

The distributed service oriented model allows additive changes without interference with existing services. Nevertheless this introduces additional complexity to the overall system. The different levels of abstraction within the tiers and clear definition of interfaces enable Traceability of changes. The general services in tiers 0 and 1 allow exchanging information across process boundaries. This is especially important for auditing and logging services proposed for tier 1. Provisioning this service, however, would need to be enforced by an implementation of the architecture. A minimal approach to Traceability is further provided by the user centric auditing services of tier 0.
3 Implementation of the Reference Architecture in a Process-Aware Service Landscape

Basing on the model described in Chapter 2 it is now possible to propose an implementation supporting real world processes. Transferring the pSTAIX reference architecture to recent internet technologies thus allows to define a set of supporting services with using commodity technologies. This lowers the threshold for developers and is essential to allow even externals to adapt services and, more importantly, the supported processes. Enabling service providers, application developers and end user to use well standardized and wide spread technologies furthermore reduces development costs across all tiers and also leads more sustainable software development. If infrastructures base on well-established standards, this additionally reduces the effects of vendor lock in.

Respecting the dimensions and metrics of Chapter 2, this chapter first introduces and discusses different technologies and implementation variants. Basing on this analysis and the pSTAIX reference architecture a set of technologies is selected for a consecutive reference implementation. The goal of this implementation then is to conduct case studies and to validate real world applicability of the reference architecture.

3.1 Modeling and Methodologies

Obviously the implementation should orient towards existing business processes but also take into account characteristics of the organization. After all, the implementation aims to support existing business cases rather than restructuring or establishing entirely new structures.

It is therefore necessary to review currently established organizational structures and processes. In turn, the implemented tiers that orient towards these processes are highly individual and base upon the organization’s demands. Nonetheless, the methodology used for modeling is applicable for arbitrary organization.
3 Implementation of the Reference Architecture

3.1.1 Organizational Model

Like supported processes, the implementation of a process supporting architecture using pSTAIX needs to consider existing organizational structures and functions. Hence, the technologies used as well as the distribution of responsibilities for operating and defining need to be assigned according to existing organizational models and structures. Due to the distributed nature, however, pSTAIX allows each participating organizational unit to define their own levels of commitment in supporting the whole organizations processes.

In the case of RWTH Aachen University there is a diverse IT service portfolio supporting various processes of students, teachers, researchers administrative personnel. While university has three centralized units (Department for internal Management and IT, IT Center and University Library) several institutes additionally supply their own IT services to their employees. Commonly intranet portals or discipline specific research infrastructures sometimes often also basic IT services like directory services, local storage systems or email servers. Figure 3.1 shows a partial overview of several IT services and their operators.

Also at the central service providers, the operation of IT services is organized using different models. The IT Center follows a stacked approach

Figure 3.1: Distributed services at RWTH Aachen University
in order to deliver IT services: The areas of Network, System Operation, Process Development and Support are each organized by one department of the IT Center. Within these departments several IT services are distributed. These services typically run across the stacked organizational structure and then involve multiple departments. This model, however, often separates the definition and requirements engineering of a service from the operation of the necessary technical infrastructure. In practice this often results in internal services that are reused internally and are given to the customers in different compositions. This requires, however, the definition of parameters in which the services can be re-purposed as well as clear definitions of interfaces of the services in order to be able to integrate them. Naturally, this organizational model distributes the different services across several teams and several technology stacks.

The same implications are valid for the other central service providers at the university. They chose their own models to operate the services, and make the service accessible to other members of the university under certain conditions using a defined interface. The advancement of the service generally is also defined by the providers. At RWTH Aachen University, this results in an extremely diverse system landscape provided at different service levels and modalities.

In many services users and their identities play an important role. Therefore, identity management processes often play an important role in the service design. As shown by Eifert et al., having established a central identity management system as it was done by RWTH Aachen University is one of the key factors for successfully integrating services [110]. This decentralized service landscape poses a significant challenge on a process level that is to be addressed by the implementation using pSTAIX. Instead of single decoupled systems, this allows users and application developers to spread their processes across IT systems of different providers and thus enables seamless transitioning between them. The ability to transition between systems furthermore generates directly notable, short term, value for users and in turn makes using the systems more attractive.

### 3.1.2 Software Engineering Methodologies

In general the proposed reference architecture does not pose any implications on the software engineering methodologies used in order to
implement services in the different tiers. In turn, it allows even de-
pendent services to use suitable methodologies required to operate the
service themselves without being influenced by its dependencies. Most
importantly, this is a building block to overcome the conceptual gap
between service operation and agile methodologies, that are often used
for individual software development. As such, the implementation of a
process aware service tier on top of an existing service infrastructure
should orient towards software engineering methodologies already in
place. At the same time this allows dependent services to define their
own methodologies to be followed.

The way IT services are built and operated is at constant change. New
software engineering and operational methods come up and vanish
together with the services they were supporting. Many methods like
DevOps, Scrum, Cloud or Serverless have become buzzwords for modern
methods without actually considering how they effect an organization
and the services provided.

Looking at the initial specification of many of these methods there
are three areas that they influence: Organization, Processes and Life-
Cycle:

Organizational Area describes how teams are organized and how they
share competences externally. It ranges from Unified to Stacked.

Unified
The full technology stack for one service or process is in
control of one person or team. This inherent responsibility for
a process and supporting systems within the team usually lead
to high commitment increasing Enactability of the processes
within the organization. While this is likely more efficient for
small teams, distinct teams tend to build redundant silos of
systems that can be hardly interconnected.

Stacked
The different technology layers are distributed to independent
teams, They are responsible for single systems or applications
rather than processes spreading across many systems. This
often leads to replication of processes in different systems as they are not connected or do not provide necessary interfaces. As only the top layer supports processes, it increases Conceptuality of the resulting service landscape.

**Process Area**

describes how teams determine the value of a process and how they distribute work internally. It ranges from Incremental to Linear.

*Incremental*

Processes are modeled and implemented partially. The selection of which part to implement next is value driven: parts that provide most value are preferred. The team likely works on multiple processes at the same time. This may lead to frequent changes of applications and systems supporting the process but can increase Suitability.

*Linear*

A process is modeled and implemented completely before adding it to a service. As such a team likely implements processes sequentially. These processes likely become more complex and more time-consuming to implement. This results in a longer time-to-market but often a better analysis of actual processes and thus increases Formality of the system.

**Life-Cycle Area** describes how teams add value to their services and how they provide services to their users. It ranges from Evolutionary to Bulk.

*Evolutionary*

Teams release small changes frequently to their user. Likely these small change sets only cause few dependencies to break that can easily be tested. Frequent changes, however, require automation of release and test processes. If errors are not encountered on first release, they may be hidden between several releases reducing Traceability.
Bulk

By accumulating changes to releases, teams work in fixed, longer cycles. This often gives more time for manual and integration testing. Longer release cycles, however, bear the risk of including many breaking changes resulting in higher Impact Of Change.

Across different services and processes within an organization each team is committed towards one or another characteristic of these methodologies. The Bringing together these different approaches obviously is a challenge that is faces when building an organizational SOA like it is done as part of the implementation.

Regarding the services available at the IT Center there are two kinds of methodologies in place: Stacked with Bulk, Sequential releases in operations tiers and Incremental, Evolutionary for supporting software; Unified with Bulk, Incremental releases in operations tiers and Incremental, Evolutionary for supporting software; Newly developed, process supporting tiers as part of pSTAIX (will) use incremental, evolutionary on top of these services.

Bulk releases in operations are often due to bulk software updates by vendors. As engineering becomes more independent of external software stacks it becomes more evolutionary. By defining interfaces between services.

3.1.3 Proposed Key Technologies

Before implementing actual processes using the pSTAIX reference architecture, some supporting key technologies were selected. Due to easy use and ubiquitous availability, web technologies have emerged to a defacto standard for process supporting services in many business areas and especially for internet and mobile applications. Even though technology agnostic, the discussed properties of the pSTAIX reference architecture thus relate to current web technologies as a basis for an implementation.
3.1 Modeling and Methodologies

REST as described by Fielding [111] is probably one of the most used paradigms in modern internet applications. As pointed out by Ti- homirovs et al., the support of many programming languages and frameworks is very mature [33]. Due to its widespread in internet applications, also many software developers know how to use REST APIs. In addition, various platforms, ranging from smart devices and sensors via smartphones and desktop computers to compute clusters, are able to access HTTP resources. HTTP and its secure variant HTTP Over TLS (HTTPS) [112] have thus emerged as a standard protocol in communication and is therefore used for standardized access in tiers 3 and 4.

Managing universities users, their roles and associations are key for many local services [110]. Identity federations are well established and widely used to secure academic internet applications. These federations offer single sign on capabilities and allow exchanging user information between participating services [88]. National federations organize hierarchically to inter-federations such as eduGAIN. This does not only provide a standardized way to authenticate users, but also allows easy sharing and reuse of services across universities and other organizations. The underlying Security Assertion Markup Language (SAML) [113], however, is technically restricted to interactive sessions in a web browser. Other applications, like apps installed on a smartphone, require a different set of protocols. Instead of authentication, the OAuth2 protocol allows users to authorize an application to use resources on their behalf [114]. It thus allows secured, personalized access to REST APIs and handles the user’s authorization without supplying credentials to the application itself. This also paves the way for third party developers accessing central IT services. The OAuth workflow has various extension points to integrate it further into existing infrastructures. This complex of identity management forms tier 0 in the reference architecture with OAuth as the main workflow to secure APIs for the users.

While not a technology in classical sense, programmers of the applications are crucial to the success of the conceptual model. Especially researchers from technical disciplines tend to re-implement existing functionalities. As digital literacy increases and APIs are easier to use, it is likely that
this trend becomes even more popular. Crotch et al. point out that many small supporting implementations may lead to issues in terms of software quality and reproducibility for future researchers [115]. Within the UK, the Software Sustainability Institute addresses these issues by trainings and further support for research oriented software. As IT specialists are a scarce resource, recruiting junior staff becomes even more important. Apart from handling basic web technologies, it is mandatory that they understand the business needs of the institutions. Integrated degree programs as presented by Küppers et al. join theory and practice from the very beginning of a study program in the field of computer science [116] and as such aim to solve this problem.

3.2 Tier 0: Authorization and Security

The most basic tier also requires the greatest level of standardization. Being also required by services in almost every other tier, demands that the implementation allows integration in processes supported by a variety of technologies. Additionally, scalability across local and federative contexts need to be considered for the different scenarios in which the services will be used.

This section will introduce the OAuth2 workflow to match these requirements. To comply with the organizational model, several extensions are proposed that allow extended scalability of the authorization service within a single organization on the cooperative level or in between organizations on the federative level.

The model for the authorization services introduced in this section originated from the setting introduces by Politze et al. to support IT services for mobile devices [117]. Later extensions to the model introduced the cooperative [118] and federative model [119] and described the basic properties and workflows for these settings.

3.2.1 Deriving Requirements for Tier 0

Micro services, as described for example by Gholami et al. [120] or by Di Francesco [121], are a recent architectural principle for internet
applications. From the software engineering point of view, this pattern reoccurs in current micro service architectures and has several advantages like easier maintainability, extensibility and replaceability of the components used. Due to the highly distributed nature of services and responsibilities, Universities face additional challenges in adopting these principles. Micro service architectures require that data is exchanged between the distributed components and to deliver a consistent user experience all services and processes using them essentially need to follow the same structure. Therefore, the goal is to profit from the advantages of micro service architectures while controlling the side effects within the distributed context.

The supported processes themselves are often based on personal data that has to be passed between the different services. As the coupling of services decreases, so does the control of which personal information is transferred, processed or saved in the different steps of the process. Workflows often need service providers and users to identify subsets of information to be passed manually between different parts of the process. However, how the personal data is actually used often remains nontransparent. To increase transparency services in tier 0 should thus allow to audit how processes handle personal data and also when systems are active to support users asynchronously. Especially the latter is becoming more relevant for ubiquitous computing systems that are always active even if the user is not.

Being a central service within the infrastructure poses specific requirements on the authorization service:

**The authorization service is the authority.**

The authorization service is clearly identified as the authority to issue validate authorizations. All relying parties, resource services, applications and users, make use of the same authorization service.

**The authorization service is trusted.**

Both users and service providers trust the authorization service to handle the authorizations. Therefore, authorizations should only be issued with explicit consent of the user and the association of authorizations to the user’s context needs to be correct.
Implementation of the Reference Architecture

Users are known.

The authorization service and the other services share the same set of users. Therefore, only minimal context information needs to be passed between the authorization service and the resource service. Also, applications do not need any explicit information about users as the user context is established between authorization and resource services.

Applications and resource services are separated.

Explicit consent from the users is required when accessing a resource. Therefore, an application should only gain access on behalf of a user. This is typically violated if the user gives credentials to an application, effectively granting the application access to any resource.

The OAuth2 workflow is therefore proposed as an integral part of most of these processes. The authorization that already grants access to the services as well as metadata, like the application which requested the authorization, are used to manage the authorizations. This role within the processes makes these services a leverage point to gain more transparency on how data is used for service providers and users.

As described in RFC6749 [114] the OAuth2 workflow allows secured, personalized access to web services or resources and handles the users’ authorization without supplying credentials to the application itself. This also paves the way for third party developers accessing central IT services and as such provices the needed scalability in terms of number of processes and services. Therefore, the Impact of Change from adding or removing processes can be controlled better. This makes the OAuth2 workflow an ideal starting point for the implementation of authorization service in tier 0.

Using a standardized workflow like OAuth2 allows external developers to easily incorporate supported processes in their own applications. While the specification of the OAuth2 workflow shows the interface for applications the functionality is described as a black box. It does not give guidance on how to implement it within a system architecture or how the internal communication between authorization and resource services is
organized. This allows the implementation of the authorization service to meet distributed and federative service landscape found in the university context.

### 3.2.2 Local Authorization

In the local scenario an authorization is typical for a single, local, service provider. This typically result in a tight coupling of authorization and resource services and are often merged into one system. This model is common at many commercial IT service providers to secure their services’ APIs.

Generally, the local authorization workflow follows the steps shown in Figure 3.2. At first the user accesses the application (1). To access the resource service on behalf of the user, the application needs to obtain an access token. Therefore, the user is directed to the authorization service where login credentials have to be provided to grant access for the application (2). The authorization service then issues the access token (3). The application can now use the token to access the resources provided by the service (4). The resource service now has to verify the authorization (5). In most implementations, resource services and the authorization service used are tightly coupled, so this step remains completely internal.

![Figure 3.2: Schematic overview of the OAuth2 workflow](image-url)
The OAuth2 implementations therefore need to offer four endpoints specified by RFC6749 [114] to support the local authorization workflow:

**Authorize**
The endpoint authorizes tokens for server side and web applications.

**Code**
The endpoint is used to request authorization codes that shown to the user used for installed applications.

**Token**
The endpoint manipulates access tokens during and after the authorization process. This includes extending the lifetime of the token or invalidating a token.

**TokenInfo**
The endpoint supplies information about a token. Application should verify that the token is valid and actually belongs to the application.

While this procedure is completely valid for local services a problem arises when scaling number of supported resource services and thus coupling of authorization and resource service need to decrease. In practice this is often done by adding an external service as an application and abusing the fact that the service provider also offers a user information service providing information to resolve the authorization’s context. OpenID Connect [122], which is an OAuth2 based workflow for authentication, implemented by many major service providers like Google [123], IBM [124] or Microsoft [125], specifies such a user information service to allow authentication.

Using OpenID Connect OAuth2 tokens for authorization of external services however violates the requirement, that applications and resource services need to be separated. This has led to major security vulnerabilities of applications and services using such a workflow as previously shown by Yang et al. [94].
3.2.3 Cooperative Authorization

Decreasing the coupling of services, and explicitly modeling the validation of the authorization for resource services is a key part of the cooperative workflow. This is especially important regarding the issue from the previous section: Not enforcing the disambiguation between a resource service validating a token and an application accessing the resources of a service makes it impossible to limit applications from accessing resources without explicit consent of the user and thus violates one of the requirements for tier 0.

This first extension to the elementary OAuth2 workflow allows more service providers to access authorizations on a single authorization server. However, all service providers are known to the authorization server a priori. Authorizations are managed by only this single authorization server instance.

To allow authorization within a cooperation of various service providers the OAuth2 workflow needs to offer the means to verify that a certain access token is valid for a certain resource, was issued to a certain application and to identify the context of the user that authorized the token. A 4-tuple $a$ containing validity $v$, client application $c$, identity $i$ and resource service $s$ then fully describes the authorization:

$$a = (v, c, i, s)$$

Figure 3.3 shows an overview of the cooperative workflow. Compared to the local scenario, the process changes as follows: before an application accesses resources, it needs to request access from the authorization service (2). The user explicitly grants access to a certain subset of resources (3). When accessing the resources, the application supplies the access token to the resource services (5). The resource services then independently validate if the token allows access to the requested resource (6). If and only if the authorization is valid, the application is able to access the resources.

The cooperative workflow therefore adds an endpoint to the authorization service:

**Context**

The endpoint validates tokens and resolves the user’s context for resource services.
Provided a token by the application, a resource service uses this endpoint to resolve the context of the token from the authorization server. New service providers can easily be added to the cooperation to extend the features of the interfaces offered to the users since services are explicitly authorized. It is however important that tokens issued before adding a service remain invalid for the new service until the user authorizes a token for the service.

This model is obviously related to the previously discussed workflow specified by OpenID Connect. However, there is a major difference: Performing authorization via OpenID Connect utilizes a information service to resolve the user’s context. Therefore, the application is authorized to access this user information service only. Resource services relying on the user information are only implicitly authorized. The cooperative model resolves these issues by explicitly authorizing specific services within the coopertaion and offering an additional endpoint for resource services to validate the authorizations.

Looking at the requirements, the main features of the cooperative model are:

The authorization service is the authority.

Within the OAuth2 workflow, the authorization service clearly identifies as an endpoint for the authorization workflow, for the user and backchannel communication of resource services.
3.2 Tier 0: Authorization and Security

The authorization service is trusted.
Both the users and resource services know and trust the authorization service. Users trust in the service to authorize applications only with their consent. Web services trust that the authorization information provided by the users is legitimate and cannot be manipulated.

Users are known.
Verifying the authorization only transfers minimal information about user context of the application. Resource services have to determine the user’s permissions to access certain resources. Within the cooperation, resource services have different means to obtain user information.

Applications and web services are separated.
Applications rely on resources provided by web services. Resource services however should only rely on their local data about the user rather than requesting other resources of other services. Users explicitly authorize an application to access a specific resource service.

3.2.4 Federative Authorization
The cooperative scenario required that only a single instance of the authorization exists. Considering the federative structures of universities this assumption, however, does no longer hold. Therefore, the second extension allows for multiple authorization servers, each of which, manages the authorizations for its local users. A service provider again uses the local instance to query the authorizations. Authorization servers have an explicit trust relationship with each other and can mediate between the parties.

Considering the application in the university context, the adoption of OAuth2 should of course consider already established national and international AAI federations. Two of which are widely established:

EduGAIN introduced by López [126] is an inter-federation that uses SAML [113] to authenticate users for web applications. Several national chapters such as the German DFN AAI exist.
Eduroam introduced by Florio et al. [127] uses RADIUS [128] to authenticate users for (Wi-Fi) network access.

Both federations base on entirely different technical frameworks and are widely independent. The OAuth2 workflow should not replace nor duplicate already existing parts of the federations, but extend them and to enable authorization of applications. Subsequently, before granting access to an application, the user needs to supply credentials in order to log in. Since most of the applications and services considered by the implementation, the orientation is clearly towards the authentication services of eduGAIN. Integrating the authorization service as a service provider into the federation allows using sign on capabilities of the existing federation.

Looking at a simplified scenario, a federation could still set up a single authorization server. The federation thus degenerates to a cooperation. However, considering the key implications from the previous section, even this simple scenario reveals challenges of federative authorization:

The authorization service is the authority.
Since only a single instance of the authorization service exists, it remains the authority for all interactions with the user and backchannel communication.

The authorization service is trusted.
One organization that is part of the federation provides the authorization service. Local implementations may be trusted more. In addition, there may be issues with someone else managing authorizations for local users and services.

Users are known.
In federative scenarios, this is generally not the case. The authorization needs to convey additional user information. While this can be achieved using the existing context method additional configuration per resource service is necessary to define the extent of additional user information.

Applications and web services are separated.
The security considerations from the cooperative implementation remain the same. The federation has to endorse service providers and application developers to use the workflows correctly.
Even though this solution is simple, several problems arise:

1. Setting up only a single authorization server, subverts the idea of many federations and their members to remain responsible for their own set of users. Consequently, the infrastructure should remain decentralized and allow each member of the federation to provide their own token service.

2. The Authorization service needs to collect all information for all users within the federation in order to be able to convey them during token validation.

Providing Decentralization

In a second approach, several authorization servers exist in the federation. Either Users or web services need to establish a trust relationship between themselves and every single authorization server. In a case where users use different token services based on the resources they want to access applications might require authorizations from multiple servers and usage of web services across organization boundaries is hardly possible. In case of web services, this would require to keep a list of trusted authorization servers. A federation then is required to keep an additional directory of authorization servers. Many commercial services implement the one or the other pattern using accounts and token services of Google, Facebook, GitHub or the like by maintaining their own list of trusted services. Considering that the German federation DFN-AAI-Basic already consists of 230 identity providers this hardly seems possible.

In commercial applications this can be often seen using specialized authentication or identity management applications that allow to sign in from numerous identity providers as offered by Google, GitHub, Facebook and many more. Again abusing authentication workflows for authorization remain problematic. In the university context this would imply that each application and service provider need to agree on the same identity providers for their authorization services. In this highly decentralized approach it hardly seems possible that the trust relationship between the participants can actually be established.
The third approach tries to make better use of the existing federative infrastructure. The general idea is that users and web services will always interact with their local authorization server. The authorization server then handles the communication and redirection to other peers to start or verify an authorization. Users and web services keep the local authorization server as the authority; in turn, the authorization server then establishes the trust relationship to the remote server.

Either way, the federation has to maintain a directory of authorization servers, possibly by extending existing metadata directories by token service endpoints.

**Providing User Information**

As shown in the previous approaches, providing user information is the key challenge in the federative scenario. Based on the existing federations, it is, however, possible to provide a strategy to resolve these issues.

For most resource services, it is most likely sufficient to provide basic user information. The existing federations participating in eduGAIN already address this second issue. Table 3.1 shows the results of a survey on the published metadata of the German chapter of eduGAIN, the DFN-AAI federation. The table shows the number of times a service provider requires a certain attribute. Being able to serve the top attributes generally should suffice for services to be able to establish the user context. Apart from user information like `displayName` or `mail`, attributes from the `eduPerson` schema obviously have a high relevance in academic contexts.

Existing federations clearly define these attributes and how service providers can access them. Technically conveying the information is also trivial. It should however be noted that some of these attributes like `mail`, `sn` (surname) or `givenName` are personal data which should not be shared with relaying parties without the users consent. Furthermore, operators of identity providers often verify that requested attributes are legitimate and sometimes enable them upon request for single service providers. Nevertheless, federations and local operators already acquire
Table 3.1: Attributes required by 347 service providers in the German DFN-AAI federation

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Count</th>
<th>Attribute</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>eduPersonPrincipalName</td>
<td>167</td>
<td>persistentId</td>
<td>10</td>
</tr>
<tr>
<td>mail</td>
<td>117</td>
<td>eduPersonUniqueId</td>
<td>7</td>
</tr>
<tr>
<td>eduPersonScopedAffiliation</td>
<td>80</td>
<td>schacHomeOrganization</td>
<td>6</td>
</tr>
<tr>
<td>sn</td>
<td>73</td>
<td>schacHomeOrganizationType</td>
<td>3</td>
</tr>
<tr>
<td>givenName</td>
<td>67</td>
<td>schacPersonalUniqueCode</td>
<td>3</td>
</tr>
<tr>
<td>eduPersonEntitlement</td>
<td>63</td>
<td>bwidmOrgId</td>
<td>1</td>
</tr>
<tr>
<td>eduPersonTargetedID</td>
<td>37</td>
<td>eduPersonOrgUnitDN</td>
<td>1</td>
</tr>
<tr>
<td>cn</td>
<td>18</td>
<td>eduPersonPrimaryAffiliation</td>
<td>1</td>
</tr>
<tr>
<td>uid</td>
<td>18</td>
<td>ou</td>
<td>1</td>
</tr>
<tr>
<td>displayName</td>
<td>14</td>
<td>dfnEduPersonTermsOfStudy</td>
<td>1</td>
</tr>
<tr>
<td>eduPersonScopedAffiliation</td>
<td>13</td>
<td>uniqueIdentifier</td>
<td>1</td>
</tr>
<tr>
<td>o</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This information and it is readily available. Embedding the resource service as service providers within the existing federation would allow profiting from the already established infrastructures like attribute quality assertions defined by Grabatin et al. [88].

**Implementation of Federative OAuth**

Based on the findings specified in the previous sections, a model for an OAuth2 federation is proposed and implemented, that to complies with the existing identity federations. Federation authorities are therefore required to keep directories of participating authorization and resource services, but can reuse existing infrastructures. However, the presented findings are merely a first step towards actually establishing a federation of OAuth2 service providers. Starting from the presented model it is now possible to deduce a proposal for implementation.

Figure 3.4 shows an overview of the participating parties and necessary communication in this workflow. Bold arrows show parts additionally
needed by the federative workflow when compared to the initial workflow. These three steps, token service resolution during authorization and validation, token issuing and remote token validation therefore need closer investigation for the implementation.

Using the previous requirements for tier 0, this solution is evaluated:

_The authorization service is the authority._

Every local authorization service is the authority. They are the only authorization service known to local users, applications and services. Users use this single instance to manage their authorizations.

_The token service is trusted._

Users and resource services only trust their local token service. A federation metadata directory establishes trust between local and remote token services.

_Users are known._

A small set of attributes from existing federations is contemplable. Transferring them during validation provides user information.

![Figure 3.4: Schematic of the federative OAuth2 workflow](image-url)
Applications and web services are separated.

The federation has participating authorization services, web services distributes metadata needed to connect the services. Applications and services remain subject to local administration and can further be added to the local authorization service. This effectively separates services and applications.

Token service resolution

When a user authorizes an application, the application redirects to the authorization service within its organization. This authorization service can then validate the legitimacy of the application and then accesses the authorization service directory offered by the federation and let the user select their trusted home institution. For each token service in the federation, this directory should at least contain:

Name of the token service

Much like in current federative scenarios, during authentication, the user has to determine a home organization. This name should serve as a hint to find the correct organization.

Namespace of the token service

The eduPerson scheme offers several attributes, which are unique within a namespace, so called scoped attributes. This namespace determines valid scopes of a token service. Comparing eduPersonTargetedID and eduPersonPrincipalName it is obvious that different kinds of techniques are used. While eduPersonTargetedID uses the entity ID of the identity provider, the service provider and a unique id to form a three tuple, eduPersonPrincipalName and other scoped attributes make use of a postfix in the form user@namespace as described by the eduPerson Object Class Specification [129].

In the OAuth2 federation, this namespace consequently applies to available applications and web services connected to the different token services.
3 Implementation of the Reference Architecture

**Signing key of the token service**

While local services and their respective token services generally have a strong trust relationship, public key cryptography leverages trust among token services. By signing messages with these keys, token services confirm the authenticity of requests.

**Endpoints of the token service**

The five endpoints already used by the cooperative OAuth2 workflow support various authorization tasks:

- The **Authorize** endpoint authorizes tokens for server side and web applications.
- The **Code** endpoint to request authorization codes that shown to the user used for installed applications.
- The **Token** endpoint to manipulate access tokens during and after the authorization process. This includes extending the lifetime of the token or invalidating a token.
- The **TokenInfo** endpoint supplies information about a token. Application should verify that the token is valid and actually belongs to the application.
- The **Context** endpoint for other token services validates tokens and resolves user attributes.

According to the already existing conventions in the OAuth2 workflow, this directory could be encoded as a JSON file hosted by the federation. A single file therefore may include all token services in the federation. The format leaves room for later additions and metadata added by the federation:

```json
1 {
  2   "token_services": {
  3     "https://oauth.example.com": {
  4       "displayName": "Example University",
  5       "namespace": "example.com",
  6       "key": "-----BEGIN PUBLIC KEY-----
MIGfM...
```
3.2 Tier 0: Authorization and Security

```
3.2 Tier 0: Authorization and Security

8     "endpoints": {
9     "authorize": "https://oauth.example.com/authorize",
10    "code": "https://oauth.example.com/code",
11    "token_info": "https://oauth.example.com/token_info",
12    "context": "https://oauth.example.com/context"
13  },
14 },
```

**Token Issuing**

After the user selected a home organization, the token service signs the authorization request and, depending on the authorization flow, redirects the user to the home token service or relays the response to the application that in turn redirects the user. The users’ home token service validates the signature and starts its authentication workflow. The user may now log in and grant access for the application. Based on the currently used workflows for authorizing installed applications (e.g. apps on a smartphone) and web applications additions to the workflow arise:

In the device workflow, the token service issues a temporary device code and a user code as in the sample below. The user then has to enter the code on a web page. For devices that do not have rich input abilities the user has to copy the verification URL and user code manually to a computer. Many devices, like smartphones, offer to open a web browser directly. If the user still has to copy a code, this results in very bad usability.

```
1  {
2       "device_code": "BaUAJHPFYFi6wKU0WY5xLC",
3       "user_code": "SFW7WZK7G",
4       "verification_url": "https://oauth.example.com/verify",
5       "expires_in": 1800,
6       "interval": 5
7  }
```
In order not to undermine cases where the user has to type the verification URL, it should remain as short as possible. Allowing user-friendly transmission of the code, token services should however also accept URLs of the form:

1 https://oauth.example.com/verify?user_code=SFW7WZXK7G

In contrast to the device workflow, the web application authorization obviously requires a web browser. It relies on HTTP redirects and ends by directing to a page of the application. For the security of the OAuth2 workflow, it is very important that this last redirection URL actually is valid for the application since authorization tokens could otherwise be hijacked. The user’s home organization however does not know the application in this case. The token service in the application’s organization therefore has to verify that the redirect URL actually belongs to the application prior to signing and redirecting the authorization request. In order to identify the token service that issued the token, it should also include the namespace of the token service in the form \{token\}@namespace.

As depicted by Figure 3.4 this results in a redirection workflow shown by Figure 3.5 in more detail: An application directs the user to the application-local authorization service. This authorization service validates that the application is legitimately requesting the authorization, then accesses the authorization service directory and lets the user chose the user-local and therefore trusted authorization service instance. The user-local instance then requires the user to authenticate and explicitly authorize the application. The authorization service therefore generates a token and assigns it with the 4-tuple of the authorization. The token information are signed and transferred to the application-local authorization service, which, in turn, validates the signature. The token is then transferred to the application to access resource services.

**Remote Token Validation**

When accessing resource services, applications pass the received authorization token. As in the cooperative workflow, the web service validates
the token against the authorization service in its own organization. The authorization service then validates the originating resource service. Using the token namespace, the authorization service can now resolve the user’s home organization and the associated token service. The authorization service signs and then forwards the request to the remote token service at the user’s home organization. The token validation follows the steps shown in Figure 3.6.

Using this signature, the authorization service in the user’s home organization is able to validate that the request is legitimate and originates from another peer in the federation. The service can then actually validate the token and resolve the associated 4-tuple of the user’s context. Doing so also retrieves additional attributes necessary to establish the user context at the request service on completing the response. By default, there should be at least one attribute uniquely identifying the user in the context of the resource service. The aforementioned \texttt{eduPersonPrincipalName} is a viable choice. Many services however need additional attributes, commonly \texttt{mail} or \texttt{givenName}. If resource services require further attributes, operators of the organization can add them on demand. Based on the findings in Table 3.1 and if compatible with local privacy conventions a \textit{zero configuration} mode for locally unknown web services should include these three attributes plus \texttt{eduPersonScoped}. 

![Figure 3.5: Federative authorization process](image-url)
Affiliation as a trade-off between the need for manual intervention and data privacy.

A valid response from the authorization server of the user’s home organization in JavaScript Object Notation (JSON) format could look like this:

```json
{
  "isValid": true,
  "application": "ahcndwlsaajcnalfjejalzd@example.com",
  "mail": "max.power@example.com",
  "displayName": "Max Power",
  "eduPersonPrincipalName": "anpqr7d@example.com",
  "eduPersonScopedAffiliation": "student@example.com"
}
```

The authorization service receiving the validation of the token then needs to verify that all scoped attributes actually belong to the same namespace the token suggested. Only then, it should forward the validation response to the web service.

**Request Signing in the Federative Workflow**

In several steps withing the workflow it is necessary to sign different messages in requests or responses. The prerequisites needed is provided by the key entry in the federation’s authorization service directory.
3.2 Tier 0: Authorization and Security

JSON Web Tokens (JWTs), as proposed by Jones et al. [130], provide a simplistic and platform independent technology to signing these messages.

A JWT contains three segments: header, payload and signature. The header specifies the properties of the token most importantly the type of signature. The specification contains three classes of signatures each using a hash-based message authentication code: with a pre shared key (HS), with an RSA [131] key pair (RS) or with an elliptic curve [132] key pair (ES). The classes are combined with a SHA2 [133] hash length. Generally the variants RS and ES support private/public key pairs and therefore are suitable for signing in this context.

The payload of the JWT can then be used to securely transport the contents of the message. In the case of a token response from the users-local authorizations server the payload uses sub to contain the token. All signed requests and responses should have an expiry date set to reduce risks for replay attacks. An example payload of an authorization service at https://oauth.somewhere.net requesting a context from the authorization service from the previous examples located at https://oauth.example.com is formatted as follows:

```
1 {
2     "iss": "https://oauth.somewhere.net",
3     "sub": "2pdMihRsq10Z3eqAGibdSFnndRnCbT2w...@example.com",
4     "iat": 1600000000,
5     "exp": 1600000060
6 }
```

Equivalently the other responses and requests can be embedded into the JWT format. If needed the Payload can be further extended by custom properties. This could for example be used to sign a token validation response.

3.2.5 Monitoring and Auditing

As previously stated transparency is key to the infrastructures security. Monitoring and auditing capabilities presented here will show how users
are able to gain insight in how their data is being used by different applications and processes.

In order to achieve more transparency for the user, the cooperative OAuth2 workflow is extended. When verifying the context of the token, the service providers may add auditing information to their request. This information is then processed and aggregated by the authorization server. While most service providers are very well capable of collecting such data, they rarely make it available to users or application developers. This workflow present a user centric way to provide more transparency of the usage of personal data.

Resource services have to call the context endpoint to resolve the context associated with the token. Essentially this has to be done every time a resource is requested by an application. At this point it is possible for the authorization server to log the 4-tuples that describe the context as shown by Figure 3.7. Using this audit log it is then possible to produce an overview for users on which applications were active due to their authorization and which services were used. Both, the service local and the user local authorization services can obtain the same tuples.

Auditing information may also be provided to the service providers and application developers. This does not only allow the deduplication of

![Figure 3.7: Token validation process extended by auditing steps](image)
information but also removes the necessity to save personal data on remote locations. Laws and best practices concerning personal data and privacy are more easily controlled on the central systems serving the OAuth2 workflow.

In the initial model, the context does not include the actual resource accessed by the user. To make this auditing more attractive to service providers the context endpoint is extended. If service providers append additional auditing data to context requests issued the 4-tuple of the context is extended by this information. While it is generally possible to extend the 4-tuple to an n-tuple of arbitrary length, the additional fields to be logged for auditing are limited to preserve performance of the actual authorization workflow. Additionally, services in a federation should agree on the reported auditing data and their semantics. The auditing system allows appending the following auditing parameters, all of which are entirely optional:

**Resource**

The actual resource requested by the user. For most RESTful web services this is equivalent to the URL requested by the application.

**Operation**

The kind of operation performed on the resource. For most RESTful web services this is equivalent to the HTTP method used by the application.

**Cost**

It is considered best practice that the authorization is checked before any actions are taken by the service provider. Therefore, actual costs caused usually cannot be taken into account. However, this allows the service provider to supply an estimate based on the resource and the operation.

This extension of the context tuple allows the service providers and users to gain more detailed insight in how their resources and personal data are used. Again all auditing information collected by the authorization servers are subject to personal data and privacy laws which can be strictly enforced at this point.
As the auditing relies on the `context` endpoint to collect the additional information, it needs to be further extended. Basing on the `token` and `service` parameter as specified by the cooperative and federative workflows additional the fields `resource`, `operation` and `cost` are added to the validation requests. In JSON notation this yields the following format:

```
1 {
2   "token": "2pdMihRsq10Z3eqAGibdSFnndRnCbT2w...@example.com",
3   "service": "https://resources.example.com",
4   "resource": "https://resources.example.com/resourceA",
5   "operation": "GET",
6   "cost": 5,
7 }
```

To ensure that only valid requests are included in the audit logs in the federative workflow, these also need to be signed. As previously shown in Section 3.2.4, JWT can be used to convey the signing information.

Even though they are not the primary target audience for the auditing, service providers need to be convinced to provide the additional information. Using the extended workflow additional value can be added for the service provider by partitioning the information further. Based on the introduced auditing parameters it is now possible to present access per resource, giving service providers more insight in how their services are used. Furthermore, the cost parameter allows to identify applications that put more excessive load on the services than others and therefore may provide an indication if applications are abusing the services offered.

For app developers, on the other hand, it is common to retrieve information on how their applications are actually used. Application platforms like Android and iOS already offer detailed reports on the kind and number of devices running the application. Also, most platforms offer usage reports that allow advanced analysis of user behavior. Even though these tools are sophisticated, their use is generally critical due to privacy concerns by the users. To overcome the need to include such tools in their application and protect the users’ privacy, some insights
from the auditing can therefore be offered to the application developers. This especially includes the number of users who used the application as well as the number of requests issued to the different endpoints and service providers.

In the federative workflows this poses an additional challenge since the application local authentication service is not included in the extended process as seen in Figure 3.7. In order to effectively include applications, the auditing information would have to be forwarded to the application local authorization server. Unfortunately this further duplicates auditing data to a third service provider. However, again this can be reasonable to provide auditing information to application developers complying to data privacy regulations. Leaving the distribution of the auditing information to the user local instance furthermore could provide the user an opt-in workflow to determine applications that receive auditing information.

It should be kept in mind, that the focus group of the auditing are users of the applications and resource services. This is extremely important as they most certainly have no other means of collecting and auditing how applications access their personal information. Users can therefore gain insight into their full profile, they see when a certain resource was requested and, most importantly, by which application it was requested. To make the information collected for the auditing even more transparent the users can further be provided access to their raw n-tuples saved for the auditing process. For concerned users this allows an in depth analysis on how their data is used. Of course this data is not stored permanently: Generally, it is a good practice that all personalized data is anonymized after 14 days due to the enforced general privacy regulations. This in turn limits the user to view their history only of the last two weeks.

3.3 Tier 1: General Services

As previously shown in tier 0, services in tier 1 provide process independent functionalities. The main purpose of these services is to allow a more seamless integration of legacy systems into a process aware service
landscape. While the services in tier 0 need to be highly interoperable, tier 1 bridges the gap between legacy systems and process aware services and supporting applications. Thus, they likely dependent on the requirements by existing local systems and processes and require less standardized interoperability. Their definition therefore is more pragmatic than the authorization services of tier 0.

The endpoints listed in this section provide an example of the services forming tier 1 of the implementation at RWTH Aachen University. They can serve as a line of reference for implementations in other contexts.

### 3.3.1 Deriving Requirements for Tier 1

Build a set of process and system independent services that allow better fulfillment of the dimensions. Especially considering the nature of incorporating distributed and legacy systems in the infrastructure.

Again by considering the organizational model, tier 1 needs to support a highly distributed and decentralized infrastructure. The services in this integration stage are, however, not directly accessible by users or application developers outside the organization but leverage the decentralized components towards an integrated infrastructure. This allows an incremental and more evolutionary approach for the software development workflows used in this tier. The services designed should be light weight as such they can be individually integrated into the existing services.

While tier 0 focused on the ability to join distributed services, tier 1 further aims at the overall reliability and basic user experience, both for end users and (external) application developers. The implementation, again, uses standardized web technologies but the conceptual foundation in Section 2.3.1 can be applied to other distributed environments.

This leads to the following requirements for services in tier 1:

**Functional Independence**

Services in this tier serve a and specific purpose. There should be no duplication of functionalities.
3.3 Tier 1: General Services

Process Independence
Services in this tier make no assumption about the processes they are integrated into. This allows service providers and developers in the higher tiers to integrate them into processes as needed.

Technological Independence
Services in this tier should function independently of the technologies and existence of the other services in this tier. This allows services to be set up or teared down individually without side effects.

3.3.2 Caching Service

To enhance the overall reliability of the infrastructure, a cache service is introduced. This service acts as a process independent short term data store for small quantities of data in form of independent cache items. In general the cache should be non-persistent and follows some sort of eviction strategy to remove stale cached items when more space is needed. The general principle behind the cache service works like a dictionary, each cache item is identified by a unique key that allows read and write access to the item. The cache is intended to speed up resource requests that otherwise would be time-consuming to compute or transfer within the distributed infrastructure. By default, all responses generated by resource services should be cached at least a short time. This effectively reduces the load on back end systems and safeguards against load produced by repeated requests. However, services should also consider other resources or intermediate results that are worth caching. According to Nah [134] response times for interactive applications like smartphone apps, services should aim at about one second. Additionally, considering latency in mobile networks and rendering leaves about 700 milliseconds for back end processing.

The cache is a shared medium for all services in the infrastructure. As such all services theoretically can access the entire cached content if cache keys are shared between processes. For confidential information, the cache keys thus should not be shared and organized in a way that can not be easily enumerated. Looking at the OAuth2 authorization
3 Implementation of the Reference Architecture

service specified in tier 0, the authorization token used in the process can be used to easily pseudonomize keys and protect cache items from unauthorized access.

Different eviction strategies have been developed and thus can be reused for the implementation. Additionally, a maximum age can be associated with each cache item after which the cache will automatically evict the item. This maximum age, however, only sets an upper bound on the cache time and eventually the cache decides to evict items before. The different strategies were evaluated in detail by Schaffert [135] to show the effectiveness for some evictions strategies. The results composed in Table 3.2 show performance of the cache in three configurations:

**First In First Out (FIFO)**
The cache evicts items only based on their maximum age and their insertion time. When additional space is needed the oldest items will be evicted first.

**Rule Based Fetching and Invalidation (RBFI)**
Additionally to the FIFO strategy rules are manually formulated that allow fetching of information before they are requested and invalidation of cache keys that certainly have changed after a request.

**Predictive Prefetching (PF)**
Based on historic user interactions a set of rules is derived in an offline learning phase, using association rule mining. Then, based on the current interactions resources fetched before they are actually requested.

Except for the FIFO caching strategy, all advances strategies, however, need additional support from the resource services in the higher tiers as prefetching is otherwise not possible. This is especially important for caching user specific information. In most applications of the cache these advanced strategies are therefore left to be implemented in the higher tiers of the architecture.

- The **Get** endpoint allows to retrieve an item from the cache. The item is specified using the cache **key**. To separate different cache areas an
optional cache secret that serves as a kind of password can be used. The response is either the cached item or empty if no item was found with the given key.

- The Store endpoint allows to insert an item into the cache. The item is identified using the cache key. If an item with the key already exists it is replaced. The item is retained for a given max-age. Again separation of cache areas can be done using a cache secret.

- The Invalidate endpoint allows to specifically evict a cache item identified by its key as needed for advanced applications of the cache. Again separation of cache areas can be done using a cache secret.

- The Clear endpoint allows to clear all currently cached items of a cache area identified by its secret.

### 3.3.3 Logging Service

The logging service aims at increasing traceability of accessed resources within the distributed architecture. Like the auditing extension of the authorization service, the logging should collect information about the way resources are accessed. However, instead of aiming at user centric security, the logging service targets system operators as well as technical and support personnel. Neither users nor support personnel are often not aware of the different distributed services and their connections within the supported processes. The information collected should thus help to tracing errors and user requests within the connected systems.
The service therefore collects information about informational and critical events from their origin. Again, session and authorization information from the authorization service can play an important role in identifying requests of a specific user and forms a common thread shared by all connected resource services. The logging service then allows support personnel to gather information from all connected resource services at one place and thus enables tracing the flow of information among the different services and applications. Apart from reactive management of errors, this allows proactive analysis scanning of log information to identify occurring errors before users report them.

Apart from internal, technical information from the resource services, the logging service contains highly confidential and personal information that needs to be handled according to privacy laws and best practices. Considering the distributed service landscape it is likely more practical to enforce these regulations in a centralized system rather than in all the connected systems. Insights into the raw data should however be given only to selected personnel.

In order to engage service providers nonetheless into using the centralized infrastructure instead of their own logging mechanisms a clear value must be provided. This can be done for example by offering a central first level support that, based on the log information, can process user requests. Also, service providers should gain access to their own log information and effectively allow them to reduce their management overhead to the bare minimum.

Additionally to technical informational and error logs the service also allows specifying Key Performance Indicator (KPI) logs. The latter are derived indicators for the performance of a service. Centrally reporting KPIs allows efficient distribution to stakeholders. The definition and necessity of KPIs is done individually by the service providers and considered to be out of scope of the definition of the logging service. The logging service merely focusses on collecting and storing the necessary information. Compared to technical logging information, KPI logs are stored for an extended time frame, often several years.
3.3 Tier 1: General Services

The logging service therefore provides the following endpoints for informational logging:

• The Append Log endpoint allows appending a log entry. A secret separates different logging areas from each other.

• The Get Logs endpoint allows retrieving log entries from the service. It will retrieve a maximum count and skip entries if respective parameters are supplied. The log entries are returned in the format specified by the Append endpoint.

Both endpoints use the same specification of a log entry that contains several fields:

- **level**
  An integer containing the error level where 0 is purely informational and 7 is critical.

- **service**
  A string containing a unique name or identifier of the resource service.

- **message**
  A string containing log message that describes the information or error to be logged.

- **token**
  Optional. A string containing the current OAuth2 access token to identify the session of the user.

- **timestamp**
  Optional. An integer containing a unix timestamp of the originating system.

- **resource**
  Optional. A string containing the accessed resource.

- **parameters**
  Optional. A dictionary containing additional parameters and their values.

- **exception**

- **stacktrace**
  Optional. A string describing the origin of an error.
The endpoints for KPI logging provided by the service are:

- The **Append KPI** endpoint allows to append a KPI to the log database. A *secret* separates different logging areas.

- The **Get KPIs** endpoint allows reading KPIs from the log database. It returns all entries between a *start* and an *end* date. The response is formatted as specified by the **Append KPI** endpoint.

This time both endpoints use a KPI log entry contains the following fields:

- **id**: A string uniquely identifying the KPI.
- **value**: An integer representing the current value of the KPI.
- **affiliation**: Optional. A string specifying the target audience.
- **resource**: Optional. A string specifying a resource that was assessed.
- **note**: Optional. A string containing an internal note.
- **timestamp**: Optional. An integer containing a unix timestamp of the time the KPI was assessed.

### 3.3.4 Private and Shared Persistent Storage Service

Obviously most system in the higher tiers will include their own, specialized, data stores. The persistent storage service should by no means replace these kinds of infrastructures. In turn, it is intended to store small quantities of data in the superordinate context of a process or a user. This should allow persistently saving of session states or user settings required for individualization of supported processes.

In the current implementation, the storage service uses a document model with a simple access control layer to allow the exchange of information
3.3 Tier 1: General Services

between otherwise independent processes. The service therefore allows
saving JSON documents. At the root, the stored document have fixed
properties that cannot be changed by the users:

ReadSecret
A string. Holding a secret needed to retrieve the document. Cannot
be overwritten by the user.

WriteSecret
A string. Holding a secret needed to write to the document.
Cannot be overwritten by the user.

Payload
Any type. Holding the actual stored information. Can be changed
by the user.

Permissions
An array of strings. If it contains remoteWrite other processes that
obtained the secret can write to the document. If it contains
remoteRead other processes that obtained the secret can read to the
document. Can be set by the user.

As discussed in detail in the thesis of Drenckberg [136] the storage
service distinguishes between user specific and process or application
specific data stores. It therefore offers different endpoints for their
manipulation. On process and application level multiple documents
exists. These documents can be shared by multiple users and can be
used to connect otherwise independent processes.

- The Create endpoint allows to create a new document. When a
document is created it is issued a readSecret and a writeSecret that
are used to access the document. Both secrets are unique in the
context of the storage service. The created document is associated
with the process, identified by a token from the authorization service,
by which it was created.

- The GetAll endpoint allows to list all documents, including their
readSecrets and writeSecrets, created in the context of the process
identified by a token from the authorization service.
3 Implementation of the Reference Architecture

- The Get endpoint uses a readSecret and retrieves the content of a document. A process identified by the token can read all documents within its context and any document that has remoteRead set in the permissions array.

- The Store endpoint uses a writeSecret and overwrites the existing document. The document root must only contain a payload field and an optional permissions array. A process identified by the token can write all documents within its context and any document that has remoteWrite set in the permissions array.

On user level, the storage service offers two endpoints. Both use the token from the authorization service to identify the current user and application. For each application and user there exists only one document. The endpoints are:

- The Get endpoint to retrieve the document.
- The Store endpoint to modify the document.

For compatibility reasons, the document has the same root structure as in the previous case: The payload field can be modified by the user. The permissions array, however, is ignored.

Additional Operations like storing or retrieving multiple documents or querying could be added. However, this service is intended for storing small amounts of data in the range of several KB per user and application. This kind of data stores are not entirely secure and should be handled with care. Applications should check for integrity or encrypt confidential data if necessary. If processes have additional requirements like security concerns, complex data sets or sophisticated queries, they should provide methods that are more elaborated and better fit the actual process.

3.3.5 User Profile Service

Some basic information about the user is often needed by applications and other services. Apart from first and last name the service provides general organizational information like employment or student status.
3.3 Tier 1: General Services

Additionally, processes that require notifications via email are supplied the user’s email address by the service. However it is preferrable that processes make use of the notification service to allow more flexible management of notifications as described in Section 3.3.6.

The information provided by the user profile service are non-authorative. If possible, Services should therefore obtain user-ids for uniquely identifying users by using tier 0 services. Applications may request an id uniquely identifying users within the context of the application (e.g. for local multi user applications). This is an important main difference to the Context endpoint provided by the authorization service that serves authoritative information but focusses on services to establish the user’s context.

The user information service should never include a user Id that is unique across applications as this impairs security. Wherever possible services or processes in higher tiers should implement their own user information functions supplying additional details like roles enriched with information from the OAuth context.

The user profile service offers only a single endpoint: The Profile endpoint retrieves the current user’s profile information. The user is identified using a token from the authorization service. The response contains at least the following fields:

**displayName**
- A string. The full name of the user.

**email**
- A string. The contact email address of the user.

**affiliation**
- An array of strings. An array containing the student and employment status of the user. Contains one or both of employee for employees and student for students.

**userId**
- A string. An identifier that is unique for the combination of user and the current process.
3 Implementation of the Reference Architecture

3.3.6 Notifications Service

The notification service allows informing users and applications about the results of asynchronous processes. The actual technical form of notification can differ and greatly depends on the supported process and preferences of the user. The notification service therefore abstracts sending notifications using different mechanisms and combines them using a consistent interface. The current implementation focuses email and smartphone push messages. Emails have a wide acceptance and are often required by when modeling a process as they offer a low threshold for sending and receiving notifications. However, they are often indirect and usually do not allow direct interaction. Smartphone push notifications, however, can directly interfere with an application but in turn require that the user installs a specific application.

Depending on the process and on user preferences the one or the other notification channel may be more adequate. Centralizing the delivery of notifications allows users to manage their preferences more easily. Also adding different notification channels in future implementations is less redundant. The notification service therefore lowers the impact of change if additional channels need to be served.

All endpoints of the notification service use a token from the authorization service to resolve the user context. The endpoints defined by the notification service are:

- The RegisterChannel endpoint allows to add a notification channel for the user. Adding a channel may require additional actions by the user such as clicking a link in a verification email or installing a smartphone application. Each channel requires a name displayed to the user. Channels support several transports: email, and webservice callbacks and various mobile platforms, most importantly, ios for iOS based devices using Apple Push Notification Service and android for Android based devices using Google Firebase. Additionally, a listener determines the specific identifier for the transport. For email this is the email address, for web service endpoints the respective URL. Platform specific push notification services use special device tokens. The endpoint returns a channelId that needs to be known in order to send notifications to the channel.
3.4 Tiers 2 to 3: Integration of Existing Services

- The `RemoveChannel` Endpoint removes a channel based on the `channelId` obtained during registration.

- The `GetAllChannels` endpoint retrieves all channels for the current process and user context. It returns their names, transports, and `channelIds`.

- The `Send` endpoint allows to send a notification to one of the channels identified by a `channelId`. Notifications consist of a title, a message, and a data field that is not displayed to the user. Transports may convey notifications differently or may have length limitations. Within all fields `$$notificationId$$` will be replaced by the `notificationId` needed to get the notification.

- The `Get` endpoint retrieves a single notification based on a `notificationId` that was transferred as part of the actual notification.

- The `GetAll` endpoint retrieves all current notifications within a user’s context. It allows giving a maximum count and a number of entries to skip.

3.4 Tiers 2 to 3: Integration of Existing Services

As introduced by Section 2.1.3 use cases were selected to support the university value processes using `eLearning` and `eScience` services. To enable these case studies it is therefore necessary to integrate the already existing, heterogeneous system landscape into the architecture. These back end systems that form tier 2 thus are extended by standardized interfaces within tier 3. Even though back end systems and services in tier 3 are not specific to processes, the existing back end systems are already used within the processes and serve their specific purpose for now they are therefore regarded exclusive to their respective process area.
3 Implementation of the Reference Architecture

3.4.1 Existing eLearning Services and APIs

As shown in Section 2.1.1 there exist various systems supporting different parts of the teaching and learning process. The implementation of pSTAIX therefore focuses on incorporating the existing LMS and SLM system into the process aware architecture. During the implementation the Looking at the architecture there are three main systems supporting the learning process: CAMPUS for planning and completion and L²P for execution. Besides these, other systems are supporting teaching and learning and need to be connected using existing and newly designed processes. The implementation of tier 3 is therefore building upon these existing systems. A technical review of the existing systems then allows building tailored and standardized interfaces that conform to specific service levels. While some back end systems already exist, some tier 3 services are developed and introduced in parallel with newly introduced back end systems aiming for specific processes. However, alike existing systems, it is the goal to create process independent layer within tier 3 to allow supporting a variety of processes with these systems.

The selected back end systems show various areas of application within the heterogeneous system landscape. Additionally, these systems offer different degrees of freedom for extensions and changes according to followed life-cycle models and system maturity.

Organizational and Lecture Directory Service

The organizational and lecture directory are part of the SLM back end system CAMPUS. The system has been in place for several years and is accessible by a web application for students and most employees. For some actions, mostly for changing information an installed application for employees is necessary.

Information from these directories form the basis for various processes: All employees, their affiliations and respective organizational units and contact information are stored in the organizational directory. Closely connected is the data stored in the lecture directory: every lecture, their appointments and associated employees that function as teaching personnel. It is therefore only natural that these directories are already
accessible using a widely standardized model that is being reused in all kinds of connected applications.

As shown by Figure 3.8 the *CAMPUS* system, uses data sources for the directories and for logins. To access the system it offers a web and an installed application that communicate with an application server. These different components form the back end system in tier 2. With deep knowledge of the systems internal functioning and reverse engineering this, mostly undocumented, protocol was well as direct manipulation of the database allowed building a more accessible API. Only few specialists know the internals of the system and can securely write applications accessing the data.

In the past this API allowed to connect serveral applications that needed data from the organizational and lecture directory. The Interfaces were developed using commodity technology: an eXtensible Markup Language (XML) Export engine for bulk exports from the database and a SOAP API to query the directory using remote procedure calls. The API was also partitioned into two areas: public serving information that is also publicly available using the web interface and private with more detailed

Figure 3.8: Schematic architecture of the existing organizational and lecture directory back end system and API
and bulk information. As with the web interface, the API provides only read access. Both, the public and read only access are sufficient for the majority of supported processes. Based on the access capabilities, the orientation towards machine accessibility and its maturity, this existing API provides the necessary service level to be directly included as a tier 3 service within the architecture.

**Personal Timetable Service**

The setting of the timetable service is closely related to the organizational and lecture directory services as the back end systems are provided by the same vendor. Additionally, the timetable services uses some information from the lecture directory, namely the appointments. As in the previous case there is already a web application available for students to manage their schedules. Unlike the previous case, however, the access to schedules is personalized as such that every student has their own, private, schedule. The general architecture is mostly resembled: databases for identities and saved schedule data are abstracted using application servers that provide a web interface for the users as shown in Figure 3.9. An internal synchronization mechanism copies lecture appointments from the lecture directory to the timetable database.

While the back end system is very mature, the timetable service was newly designed to be included into the tier 3 of the architecture. It allows reading as well as adding and removing events within schedules of the students. The connection of synchronized appointments requires an additional internal mapping to resolve the copies made during the synchronization that is performed using internal database queries. The service initiates a session on behalf of the user and then uses automation of the web user interface to manipulate the timetables. This newly provided API however does not provide truly personalized access but allows administrative access for all users. To connect with tier 4 services a trust relationship is established using credentials and network address filters. The timetable service therefore relies on services in tier 4 to provide impersonation. Considering re-usability and confidentiality this is not a preferred mode of implementation but clearly shows the flexibility needed when influence on back end systems is limited.
3.4 Tiers 2 to 3: Integration of Existing Services

Figure 3.9: Schematic architecture of the timetable back end system and API

eLearning Portal Service

The system L^2^P is mostly used for the execution of lectures during the semester. Typically, this includes teachers sharing materials, like slides or readings, with their students but also several didactic scenarios like home work assessment or quizzes. The current version of L^2^P is developed in parallel to the reference architecture and the resulting process-aware services. This allows a high degree of integration of the provided services.

The back end provided by L^2^P bases on SharePoint Server. An extensive object model shown in Figure 3.10 allows customization and adoption for supported processes that make use of the basic functionalities of SharePoint as described in [137]. As in the previous cases, the service can be accessed through a website allowing students and teachers to interact with the various functionalities of the system. The system uses the already available API from the lecture directory, to synchronize with planned lectures.

During a reconception phase the tier 3 API was build offering access to many interactions and resources previously only available via the
website. This API allows personalized access to the system and already uses authorization services from tier 0. It therefore provides a good example of a modern integrated system, that fully benefits from the other services within reference architecture. Even though the lecture directory and L²P are closely connected, in many cases this API is however missing the connection to events or lecture entities and employees described in the previous services. These connections need to be obtained by consolidation of the distributed systems by services in tier 4. Nevertheless, L²P and its API already feature several applications written by external developers, mostly students. Even though they are using a tier 3 service at this point, because they are already using tier 0 for authorization these applications can benefit from additional services integrated into tier 4 and can eventually migrate towards process-aware interfaces.

**Quiz Service**

Quizzes are a part of a Moodle subsystem that is contained within L²P. From the user’s point-of-view the Moodle web application is part of the L²P website, forming another, already established, connection between
existing systems that has to be considered when integrating *Moodle* as a back end system into the architecture. As shown in Figure 3.11 this connection already utilizes an early implementation of the authorization service in tier 0 in order to convey user sessions between the two systems. This allows users to access *Moodle* directly through L²P and allows *Moodle* omit a separate user and lecture management.

The existing quiz service is an extension to the *Moodle* sub system that was developed for a mobile application prototype called *Quiz2Go* [138]. It provides a REST interface to download questionnaires called eTests as quiz packages from *Moodle*. These quiz packages consist of an XML description and the multimedia assets needed to display the quiz. Even though *Moodle* will store results of eTests taken in the browser, *Quiz2Go* did not offer such an interface to submit answers. REST endpoints were additionally defined for this purpose and thus form the quiz service in tier 3. As in the case of the moodle website, the quiz service also uses the authorization service of tier 0 to establish the user context.

Even though the degree of integration between L²P and *Moodle* is high, both systems and especially the quiz service are not consistent. Regarding the provided APIs there are no connections or shared identifiers

Figure 3.11: Schematic architecture of the *Moodle* back end system and API
between the systems requiring the user to always query both interfaces. If it was not for the authorization workflow from tier 0, the APIs of the systems appear completely distinct.

**Audience Response System Service**

The services supporting the audience response system are specifically built to satisfy the case study. The general idea is to allow communication between students and teachers in large scale lectures with more than 500 participants. In contrast to the services that are readily available to the users, the audience response system is an entirely new application build within the proposed tier model. As shown in Figure 3.12 there is no dedicated back end system but only a generic database to store user data.

In order to access the database a tier 3 service encapsulates the required queries and supplies an API basing on low level actions for sending and receiving messages, needed for the realization of the audience response system. Since this API was implemented as part of the case study it also already uses the authorization service to ensure only authorized users can trigger respective actions.

![Figure 3.12: Schematic architecture of the audience response system service](image)
3.4 Tiers 2 to 3: Integration of Existing Services

Library Service

A web application allows users to search library collections and manage accounts and loaned media. To access information from the library management system and make it available to the users through a browser, the library service uses TouchPoint. Both, this library management system and TouchPoint are operated by the university library. It is thus an example of how third party service providers within the same cooperative context can be integrated within the infrastructure.

Through an internal API TouchPoint provides access to the functionality and allows impersonation of users. The tier 3 services makes use of these internal interfaces to provide personalized access to the available information. Figure 3.13 shows from an architectural point of view, that the internals of the system are hidden. External services as such are therefore assumed to be a black box. However, as with the previous example, this API was developed by the library from scratch to be integrated into the architecture. It therefore makes use of the authorization services of tier 0. Naturally this leaves control over the impersonation functions with the operators of the system. To conform to the requirements of the use case, the provided API allows viewing and extending the loaning periods of loaned items and provides access to fees for overdue items but can be extended to capture more of the back end systems’ features.

![Figure 3.13: Schematic architecture of the library service](image-url)
3 Implementation of the Reference Architecture

3.4.2 Existing eScience Services and APIs

Services supporting the research generally range from project management and controlling to Research Data Management (RDM). Regarding current developments in data science and artificial intelligence and ever-increasing amounts of research data generated, especially RDM is moving into the focus of researchers from various scientific disciplines [15], [139], [140].

Corresponding to the use cases targeting this area, the implementations focus on RDM services. However, extending the supported services towards more general eScience workflows is endorsed from an architectural and organizational point-of-view. Generally, centralized support for eScience processes has been established only quite recently. Therefore, the systems, available services and processes are less mature and have yet to be fully developed.

Persistent Identifier Service

Commonly Persistent Identifiers (PIDs) are used in RDM processes to identify individual data sets. They may however be used to identify arbitrary digital resources of varying granularity. Ranging from collections of data sets to data points therein. While there are several approaches for creation and management of PIDs two major realizations base on the Handle system [141]: Digital Object Identifier (DOI) [142] and European Persistent Identifier Consortium (ePIC) [143]. The back end system regarded in this case is an instance of the ePIC service.

Within the Handle system PIDs are represented as strings with of two parts: a prefix identifying the service instance and a suffix identifying the data set. The combination of prefix and suffix resembles a globally unique identifier. For each identifier a set of key value pairs is stored in the system. Keys and values can be arbitrarily set by the user of the system, however, when using HTTP based resolvers, users are redirected to the value stored for the url key.

The instance of the ePIC system provides an API that allows manipulation of PIDs using a REST interface. As Figure 3.14 suggests, this
service is hosted and operated by an external service provider, leaving no insight to its internals and is viewed as a black box. While the provided API generally allows personalized access, the impersonation mechanism uses HTTP authentication with user name and password opposed to the federative OAuth2 workflow provided by tier 0. This implies that impersonation in this case is handled as part of the tier 3 implementation. Other than that the already standardized interface of the tier 2 service is widely passed through.

**Metadata Service**

The metadata service uses a triple manage metadata of research data. It aims to provide a flexible and searchable data store for heterogeneous metadata sets. The back end system uses Virtuoso as a database system using its triple store to provide semantic storage and search capabilities. The metadata service was specifically established to serve as a part of the use cases. Apart from the storage of metadata the service also allows validation of metedata sets according to metadata schema definitions also stored in the database.

Virtuoso offers a SPARQL Protocol And Query Language (SPARQL) interface to insert and query triples from the database. Generally this

Figure 3.14: Schematic architecture of the Persistent Identifier service
3 Implementation of the Reference Architecture

interface follows REST principles. However, as a language SPARQL is neither a commodity technology nor does the implementation offer state-of-the-art data protection mechanisms [144]. As shown by Figure 3.15, impersonation as well as rights and role management are left to the implementation of the service in tier 3. The underlying triple data model, Resource Description Framework (RDF), however can be flexibly used to store the permission information needed and therefore qualifies Virtuoso as a back end system for the service.

Basing on the provided SPARQL interface an impersonated API is constructed within tier 3. This API provides access to metadata and metadata schemas stored in the database. This newly created interface also defines the required operations for inserting, updating and searching triples representing metadata sets or metadata schemas. The tier 3 service provides these as simplified REST endpoints to hide the increased complexity of the underlying SPARQL interfaces. Impersonation in this case allows definition access rights based on the users’ context and additionally stored triples within the metadata sets.

Archive Service

Long term storage of research data is one of the challenges faced within the context of RDM. The archive service aims to solve this challenge by making a tape library accessible as a service. The tape library uses

Figure 3.15: Schematic architecture of the metadata service
**3.4 Tiers 2 to 3: Integration of Existing Services**

*IBM Spectrum Protect*, a proprietary client and server software suite for archival and backup of data, as a back end system.

The existing system is already widely used by administrative personnel. Users can obtain virtual storage spaces called nodes that retain specific credentials. Nodes are managed by a web portal that gathers information, like their respective owners and saves them in a database. A limited set of actions, like node creation or resetting credentials, are available through a REST interface provided by the portal. A command line interface then allows ingestion of data into the tape library. The functionalities of the command line client largely depend on the operating system: as such different operating systems cannot interact with the same node. Even though the underlying data model is comparatively simple these technical dependencies require expert knowledge to use and integrate the current service within the service landscape.

While the archiving processes in place need to remain, a newly created RESTful tier 3 interface as shown in Figure 3.16 allows access to basic operations on file level. To reduce technical issues, nodes are partitioned into distinct modes of operation: those that are managed by the users themselves and those managed by the interface. Both modes of operation remain side by side. To allow storing and retrieving of files within a user’s context, the services in this tier again handle impersonation utilizing the authorization service from tier 0 and allow users to interact with nodes without needing specific credentials. Additionally, since the tape archival process is asynchronous, the interface uses temporary storage to buffer files before ingestion or after restoring. This greatly simplifies the existing interface by offering only few well-defined operations on the back end service.

**SharePoint Shared Service**

Initially *SharePoint* is a web portal for collaboration and document management. Instances, called Sites, are supplied to projects or institutes and customized to their needs. Sites brings an extensive web interface that allows storing documents and other information in lists that can be compared to database tables. Additionally, *SharePoint*
allows on line viewing and editing of common office documents. This makes *SharePoint* an integration platform that can be adapted to local processes.

Each instance is separated and can be individually configured and customized. This allows the service to be running on a shared instance even though some sites make use of custom extensions. Data stored in a Database can be accessed through server and client side object models. The server side interfaces are fully trusted and therefore allow administrative and impersonation of arbitrary users and therefore make it very easy to implement individual workflows within extensions. The client side model always runs in the context of a user authenticated using traditional credentials.

To allow more standardized access, the available interfaces are further abstracted from the *SharePoint* technology stack. The different information entities within lists, their metadata and stored documents are mapped to an object storage access model where a REST endpoint for
each list enables management of the contained information or files. As shown in Figure 3.17 the API is yet another extension to the SharePoint service. This interface uses the services from tier 0 for authorization and impersonation and thus allows standardized and technology independent access to information stored in sites.

3.5 Tier 4: Process-Aware Services

With the standardized interfaces available for the back end systems, process-aware services in tier 4 can now create interfaces that focus on defined processes using common nomenclature and common identifiers for shared resources. For users of tier 4 back end systems are no longer visible and can be replaced by any other system providing the same capabilities. The actual integration into a process supporting applications is then shown separately as part of the respective case studies.

As previously in Section 3.4 the areas of eLearning and eScience are considered separately. However, since both areas effect widely the same group of people, used technologies and, more importantly, nomenclatures

Figure 3.17: Schematic architecture of the SharePoint shared service
remain consistent across both areas. While the services as a whole are required to be consistent, several endpoints supporting related processes form a scope within the areas.

The presented service endpoints in this section mostly present only a digest of all available service endpoints within the scopes. For the sake of brevity only those endpoints that are directly relevant to the presented use cases are fully explained.

### 3.5.1 eLearning-Process-Aware Implementations

Respecting the pSTAIX reference architecture, the process-aware implementation of eLearning services forms the first part of tier 4. Technically the different APIs from the previous section are consolidated to use a common standardized RESTful implementation. The creation of common identifiers and workflows is especially important for services that are commonly used coherently like L²P with Moodle or the organizational and lecture directories.

#### Lecture Scope

The Lecture scope integrates information from all services mentioned above to provide consistent access to aggregated lecture information. This is the entry point for most eLearning processes if they operate within a lecture scope.

- The **Find** endpoint uses the lecture directory service to retrieve a list of lectures based on a query string using a full text search.

- The **GetMy** endpoint uses the lecture directory service to retrieve a list of lectures that the user is enrolled in the current semester. It allows setting a termOffset to navigate through semesters.

- The **Get** endpoint retrieves more details on one lecture identified by a lectureId. It combines the information from the lecture directory with information from the eLearning Portal if a course room is available. Information about lecturers are integrated from the organizational directory.
3.5 Tier 4: Process-Aware Services

- The GetAppointments endpoint retrieves lecture dates from the lecture directory. Information about rooms and addresses are integrated from the organizational directory. Appointments are uniquely identified by an appointmentId.

Basing on the lectures from the lecture directory, learning materials and resources supplied by the lecturer in the learning portal are also supplied within the Lecture scope. Their management, however, has to be done by lecturers directly in the eLearning portal since no endpoints have been abstracted. However, this is mainly because the focus group within the use cases were students not teachers. All endpoints use the lectureId from above to identify a specific lecture.

- The GetAnnouncements endpoint retrieves a list of announcements for a given lecture.

- The GetAssignments endpoint retrieves a list of assignments for a given lecture.

- The GetMaterials endpoint retrieves a list of materials like slides for a given lecture. The endpoint constructs a downloadUrl to allow downloading the actual file.

- The GetLiterature endpoint retrieves a list of literature supplied by the researcher.

**Quiz Scope**

Since the quiz service was initially designed as a standalone application, in order to integrate the endpoints into tier 4, information from the eLearning service and the quiz service need to be consolidated. This forms a common semantics and understanding for the retrieved resources in both systems within the Quiz scope.

The GetAll endpoint allows retrieving a list of all quizzes for a specific lecture from the quiz service. It uses the lectureId to identify the lecture. Like in the case of the Materials endpoint above, an Uniform Resource Locator (URL) is constructed that allows downloading the actual quiz file. A quizId is included in the response to identify the quiz when using the other endpoints.
The StoreResult endpoint allows to submit a quiz result to the quiz service. The quiz is identified by the quizId.

The GetResults endpoint aggregated the submitted results and provides averages of the previous responses for each question in a quiz identified by the quizId.

Timetable Scope

Based on their lectures, students can add lecture appointments to their personal timetables. The timetable scope therefore defines tier 4 endpoints that integrate the information from the lecture directory and allow lightweight access to basic timetable functionalities.

- Based on the lecture appointments, the AddAppointment endpoint allows adding a lecture appointment to the personal timetable of the user. Timetables are stored using features of the lecture directory. The RemoveAppointment reverses this action.

- The GetWeek endpoint allows retrieving a schedule containing all appointments added for the current week. To navigate between weeks a weekOffset can be specified. The appointments returned by the endpoints are equivalent to those from the lecture scope.

Library Scope

The library scope in this case remains independent the other process-aware scopes. It allows interaction with items that the user loaned from the library. From a process perspective the information provided should allow the users to get an overview of loaned items and extend the loaning period if offered by the library.

- The GetAccount endpoint retrieves the items currently loaned by the user. Items are identified by a mediaId. The endpoint also returns the title, the returnDate and a property canExtendLoan denoting if the items loaning period can be extended.

- The ExtendLoan endpoint allows to extend the loaning period of a loaned item.
3.5 Tier 4: Process-Aware Services

Audience Response System Scope

Even though the audience response system is clearly an application that is intended to support lectures, it was not integrated in the lecture scope on purpose. Instead, the audienceResponseSystem scope is independently modeled to allow using the services in other scenarios than just lectures. In turn the different channels are only tied to the organizational unit rather than a certain lecture.

- To obtain a list of all channels for a user, the GetChannels is used. The entries in the list include a channelId and a title. Channel descriptions are available to all users. If used by a lecturer the list additionally includes a secret that allows access to all messages for channels owned by their organization. The endpoint uses information from the organizational directory to merge channels and respective users.

- Using the SendMessage allows sending messages. The body and type need to be supplied. The type property separates different message types, commonly used to separate text and image messages. By default, messages are sent to the teacher. If a secret is supplied, messages can be sent to arbitrary recipients using the pseudonomized recipient.

- The GetMessages endpoint retrieves a list of all messages in a channel identified by its channelId. The endpoint accepts a skip parameter to retrieve only the latest messages. The returned list contains an messageId, a pseudonymized sender, a timestamp, the message’s body and its type. The full body is only supplied if the type is not image and smaller than 1KB. Users can only retrieve their own messages and those sent by a teacher. If a secret is supplied messages from all students can be retrieved.

- Using the messageId, the GetMessage endpoint allows retrieving the raw body of a single message in order to download images or bigger messages. As previously, if no secret is supplied, users can only retrieve their own messages and those sent by a teacher.
3.5.2 eScience-Process-Aware Implementations

As previously discussed for the case of eLearning services, the existing and newly built process supporting infrastructures for eScience are integrated into one common RESTful interface. With respect to the pSTAIX reference architecture, this interface forms the process-aware tier 4. Obviously the implementations in this tier also profits from the existing services in tiers 0 and 1 like authorization, caching and logging.

Generally the process-awareness is achieved by a combination of existing the services from tier 3. For example: upon creation of a metadata set, it is assigned a PID that can be used for externally references to the dataset but also for identification within other supported operations. Generally the PID service is widely used in most of the scenarios supported by the implementation as it provides the means to assign globally unique and persistent identifiers to digital resources and in turn provides a way to establish a common understanding of a dataset across provided services.

Metadata Scope

The endpoints in the metadata scope provide access to the metadata service of tier 3. Instead of allowing storage of arbitrary triples, however, the process-aware implementation additionally incorporates the PID service to create only metadata sets for data sets identified by a PID.

Metadata that is created within the metadata scope has to conform to metadata schemas and is validated by the metadata services in tier 3 before it is saved. The process-aware service also allows translation between the RDF format internally used for storage and key value pairs that allow a simplified access to stored metadata.

- The `Get` endpoint retrieves a single metadata set associated with a pid. Private metadata sets can only be obtained by their creator or using an ota. Public metadata sets can be obtained by every user. If the metadata set is retrieved by its creator, the response also includes an One Time Access Token (OTA), that can be passed to other users in order to view or make changes to a private metadata set.
3.5 Tier 4: Process-Aware Services

- The `Store` endpoint allows to create a new or update an existing metadata set. These are associated with a `pid` that references the actual data. If no PID is supplied a new one is created for the user. Changes to existing data sets or can either be done by the owner or another user has obtained an `ota`. Generally metadata sets are private to the user, they can however be published for other users using a `publish` parameter. The metadata sets are specified as an RDF graph and need to conform to the metadata schema identified by the `schemaId`. Adding metadata to a PID adds a `meta` key value pair to the respective PID.

- The `GetAll` endpoint retrieves a list of metadata sets created by the user. The list contains the `pid` and a `title` if available in the metadata sets.

- The `Find` endpoint retrieves a list of available metadata sets. It queries both, the metadata sets created by the user and public metadata sets. The list contains the `pid` and `title` if available in the metadata sets. The `query` can either be a full text search or conform to a query syntax that allows targeting single metadata fields.

- The `GetAllShemas` endpoint retrieves a list of metadata schemas available for the user. Apart from the `schemaId` used to identify the actual schema, the endpoint returns a `title` of the schemas.

- The `GetSchema` endpoint retrieves a single schema identified by a `schemaId`. Metadata schemas define a several of metadata fields and their range of values in a linked data model. Metadata schemas are transferred in a RDF data model.

**Archive Scope**

In order to be able to connect archived files and respective metadata sets, the `archive` scope also basis on PIDs to identify the stored data. This makes both scoped interoperable as such that they use the PID service to exchange information about the location and the owner of the data.
and metadata. In this context, the **archive** scope supports long-term storage processes where datasets are not altered after archival.

- Using the **Store** endpoint, allows to store a file in the long tail archive. The file is associated with the user who uploaded it. Upon uploading, a **pid** parameter specifying a PID owned by the user, can be supplied. This also associates the PID with the stored file by adding a **DATA** key value pair to the respective PID. If no PID is supplied the endpoint will issue a new one and return it to the user. The archival process works by adding the file to a processing queue and then uses the archive service to store the file for long term. This processing is asynchronous and uses the notification service to inform the user once arching is completed. In order to retain the list of all files for a user, the storage service is used.

- The **GetAll** endpoint retrieves a list of all files stored on behalf of the user in the archive. The list contains the stored **fileName**s and **pids**.

- With the **Retrieve** endpoint the user can queue the retrieval process of a file identified by the associated PID. Again this step is asynchronous. When the request is processed, the file is made available using a temporary URL, transferred to the user using the notification service.

### Collaboration Scope

The **collaboration** scope focuses on short tail data storage. It allows more interactive workflows than the endpoints of the previously introduced **archive** scope. The endpoints feature a lean version of metadata management that is compatible to the **metadata** scope. Even though the scopes base on different back end services this allows transferring the stored information between them.

- The **Store** endpoint uses the *SharePoint* service to store the file on a server. In this case the file is associated with a **fileName** rather than a PID. This results from the volatile nature of collaborative files and from the constraint that the PID service does not allow deleting PIDs. If needed, a PID can be created optionally. The backend system is organized in a hierarchical structure that is encoded as folders in the file name.
• The `SetProperties` endpoint is used to set additional properties of the file. The SharePoint service used as the backend supports a metadata model that can be accessed using a dictionary of key value pairs. These properties also allow setting access permissions for the file.

• Retrieval of a stored file is done using the `Get` endpoint. The file is identified using its `fileName`. Only files that are accessible by the user can be retrieved.

• In order to obtain a list of files accessible by the user, the `GetAll` endpoint retrieves all files starting with a `prefix`. This allows listing contents of different hierarchical levels or folders. The list includes the `fileName` and the `properties` dictionary. For each file the method also constructs an URL that can be used to obtain the binary file.
4 Case Studies

To validate the properties of the architecture designed in Chapters 2 and 3 several case studies using different services were conducted and evaluated. Each of these presented cases further investigates real world processes and applications at RWTH Aachen University in three overall areas: eLearning to support student’s and teacher’s learning processes, eScience to support scholar’s research process and genereal University IT processes.

Apart from considering different processes, the case studies also aim at asserting the maturity of the architecture in different dimensions as introduced in Section 2.2.

4.1 eLearning Processes

The area of eLearning processes naturally focuses on the services of tier 4, provided on top of the back end systems of this area as described in Section 3.5.1. As all the services use a single implementation of the architecture, there is no technical necessity to look at the services in this isolated manner. There is, however, a strict separation of these concepts in the minds of employees at the university, making the separation necessary in order to comply with the expectations on Suitability posed on the architecture.

Processes in this area are mostly established and well-known to employees at the university. The case studies in the eLearning area will therefore focus on how existing processes can be enhanced by introducing the process-aware implementation. These processes are furthermore enacted by students, often neither knowing about the formal definitions nor the technical systems driving the services they are using. This need for employees and students to adapt the supported processes within their applications, underlines the importance of Comprehensibility and Enactability within the dimension of process-awareness. Additionally, it is important to regard students not only as passive users, but also as potential developers of processes and applications supported by the
architecture and the implementation. This decentralized development of applications cannot be as closely controlled and validates as if they were developed by organizations of the university. Generally these developments cannot be trusted. If they provide short term values for users they will however be used. This clearly underlines the need to contain Integrity and Confidentiality within the services offered.

4.1.1 RWTHApp

The aim of RWTHApp is to form a single point that accumulates all information needed by the students of the university. Using an incremental development model and an evolutionary application life-cycle the application was gradually extended and enhanced. The whole development process was based on the requirements directly posed by the students. The ambition to fully understand the needs of the students lead to a development cycle of four weeks as displayed in Figure 4.1. Students could continuously provide input on the current state of the application. Based on this input, the ideas and remarks were evaluated prioritized and checked for practicability based on the system landscape and finally emerged in the implementation of the software. The application and its programming model were initially introduced by Politze et al. [145] and the model was later refined to the presented version [146].

Condensing all functionalities in a single application posed a significant challenge both for front end design as well as back end systems collecting
and consolidating the required information. The initial feature set included: a timetable to display when and where the next classes will take place, recent activities on the eLearning platform such as new documents or announcements, University wide announcements and news, a room finder and university staff search. Also navigational capabilities give the students the possibility to find their way through the different buildings of the university spread all over the city of Aachen. All these features base on the needs of the students to receive information almost instantly.

The information needed already exists in the current eLearning and campus management systems of the university. However, it needs to be integrated and made available in a secure and personalized manner in order to be accessible by the application. This setting makes RWTHApp an ideal use case showing the benefits of the pSTAIX architecture applied in the eLearning context.

**Supported Processes**

RWTHApp aims at supporting a variety of processes, serving as the single point of information for the students. Regarding this case study two general features are selected to be discussed in particular: obtaining current lecture materials and loaning in the library. However, also the other case studies, namely Quiz2Go and the Audience Response System conducted in the context of eLearning are eventually integrated into the application.

The supported lecture materials process is depicted in Figure 4.2. Student start by using RWTHApp to list all lectures they currently are enrolled into. They can now view lecture details and teacher information, add the appointments of the lecture to their personal schedule and list and download different learning materials made available via the eLearning system. Based on this information an asynchronous task is started that continuously checks for materials added or changed. If such a change is detected, the student is notified and can directly access the materials in question.
The second process that is in focus of this case study is the extension of loaned media from the university library as shown in Figure 4.3. Again this processes is modeled using a student centered point-of-view: The student uses RWTHApp to sign up for a notification schedule. This allows a back-end software checking for current due dates and will notify the student right before loaned media need to be returned. Additionally, the student can check the current status at any time within the application. Either way, the process then allows the user extending the loaning period if the media is still required.

Essentially both processes deliver a similar user experience. As presented they show how interactive and asynchronous process steps can be combined using the reference architecture and its implementation.

**System Architecture**

The presented processes as part of RWTHApp run on top of the tiers specified in Section 3.5.1. RWTHApp itself is a hybrid mobile application.
leverage JavaScript and HTML5 technologies to obtain maximum compatibility with different mobile platforms using only one code base. Apart from the application the presented use case also introduced a server side component for asynchronous processing and initiating the user notifications. The installed application then serves as a target for platform specific push notifications sent by the notification service. Even though both parts of the applications run on entirely different hardware, their communication paths with the other components of the infrastructure are identical.

RWTHApp uses the different back end systems in order to allow high level modelling of the process tier. In order to achieve this, RWTHApp directly uses the process-aware eLearning, lecture directory and library services made available by tier 4. These services mediate between the different legacy back end systems using the standardized interfaces of tier 3 and the general services from tier 2. Tier 0 is used to transfer the user’s context between the different participating systems and thus is also accessed directly by the application in order to initially obtain the authorization token.

**Results**

After the initial release with only a limited feature set, RWTHApp quickly evolved to spread among students and employees of RWTH Aachen University alike. At the time of writing, the application was downloaded 70,000 times and has about 9,000 individual daily active users over all platforms. As shown by the values gathered from the KPI logs in Table 4.1, in the current term there were on average 2.8 million sessions per month. In this context a session is counted whenever a process is used within the area in a 30 minute time frame. The numbers clearly show the importance of the eLearning processes being available for the students.

In absolute numbers, the RWTHApp thus presents the most widespread use case. However, being also the first real world application of the reference architecture, the modeled processes are rather simplistic.
4 Case Studies

Table 4.1: User sessions counts within process areas of RWTHApp in winter term 2018

<table>
<thead>
<tr>
<th>Month</th>
<th>Lecture</th>
<th>Library</th>
<th>Organizational</th>
<th>Timetable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 17</td>
<td>1,195,106</td>
<td>186,776</td>
<td>127,180</td>
<td>44,802</td>
</tr>
<tr>
<td>Oct 17</td>
<td>2,403,493</td>
<td>260,654</td>
<td>460,433</td>
<td>776,784</td>
</tr>
<tr>
<td>Nov 17</td>
<td>2,811,573</td>
<td>295,125</td>
<td>297,847</td>
<td>456,766</td>
</tr>
<tr>
<td>Dec 17</td>
<td>1,758,875</td>
<td>266,250</td>
<td>219,090</td>
<td>170,400</td>
</tr>
<tr>
<td>Jan 18</td>
<td>2,101,089</td>
<td>209,030</td>
<td>268,354</td>
<td>178,731</td>
</tr>
<tr>
<td>Feb 18</td>
<td>2,166,382</td>
<td>201,574</td>
<td>233,758</td>
<td>90,697</td>
</tr>
<tr>
<td>Mar 18</td>
<td>2,667,249</td>
<td>187,730</td>
<td>291,110</td>
<td>69,056</td>
</tr>
<tr>
<td>Avg.</td>
<td>2,157,681</td>
<td>229,591</td>
<td>271,110</td>
<td>255,319</td>
</tr>
</tbody>
</table>

4.1.2 Quizz2Go

Initially, Quizz2Go was developed as a standalone mobile application by the Chair for Engineering Hydrology as part of their institutional eLearning activities [138]. The principle behind the application can be compared to learning with flashcards. Teachers prepare a set of questions that can be automatically evaluated like multiple choice or association questions. Students then take the quiz to check their progress on the topic. They receive immediate feedback whether they answered the question correctly and can re-take the quiz at any time.

Apart from the mobile application, a Moodle plug-in allows packaging of tests edited in Moodle and necessary assets, like images or even short videos, as a single downloadable file. These tests can used within Moodle or in the client application. The client application offers the advantage of storing the quiz package and thus also works with limited or no network connection. To allow all teachers of RWTH Aachen University to profit from the software, the aim of this case study is that Quizz2Go becomes a feature of RWTHApp.
4.1 eLearning Processes

 Supported Process

The existing Quizz2Go application allows teachers to edit eTests in Moodle and then allows students to download those eTests as an off-line quiz. While the editing process vastly remains the same, the integration into RWTHApp and the existing LMS is a significant change.

Based on the existing implementation, the quiz allows different types of questions: Single Choice, Multi Choice, Association and different types of assets like images, videos or formulas. The students should then be able to download quiz packages for their lectures and take the quiz with the mobile application. Moreover, as a new feature not supported by the initial prototype, after taking the quiz, they should be able to compare their performance to other students taking the class. Therefore, their answers are also collected pseudonomized on server, that in turn also validates them and stores the results of each question. As the system is intended as a kind of self assessment for the students, they can re-take the quiz at any time to check their learning progress.

From a student perspective, the process is depicted in Figure 4.4. Students use the application to access their lectures and download quiz packages. Once these packages are downloaded, the student can take the quiz. Results are automatically stored on the server to compare to other students taking the class. If the quiz was not taken yet, however, the accumulated results of the class cannot be accessed. If the students are not satisfied with their performance, for example after a learning session, they can re-take the quiz. The answering process, as shown in Figure 4.5 furthermore allows directly checking the current answer and an optional extended explanation and thus can also be used by students to learn about the topics.

System Architecture

In the supported processes, three existing backend systems are combined: Two different Moodle systems, hosted by the IT Center and by the Chair for Engineering Hydrology. Each system uses the Quizz2Go plugin. Additionally, $L^2P$, the general LMS of the university. As part of the implementation of the case study both Moodle systems
were integrated with L²P in order to build a comprehensible model of eLearning materials as a whole. Moodle remains the user interface for teachers authoring the questions while the client side functionality for the students is included into RWTHApp. And therefore can make use of already existing user base. Additionally, necessary services are therefore integrated into the architecture with the existing services. This allowed effectively reusing already existing eLearning functionalities as discussed in Section 4.1.1.

**Results**

The Quizz2Go feature was first implemented as a prototype that only allowed downloading and taking quizzes to gain some insight on how students make use of the system. After a pilot phase the feature was optically refreshed and extended in a way that allows students to anonymously compare their results as shown previously.

Authoring quizzes poses an additional workload on teachers preparing their lecture. Therefore, it is very likely that there are only few lectures where quizzes are available to the students. Table 4.2 shows the KPIs gathered from the respective endpoints and confirms this assumption. This is especially obvious when compared to the far more widespread eLearning process from Table 4.1.

128
4.1 eLearning Processes

<table>
<thead>
<tr>
<th>Month</th>
<th>Quiz Downloads</th>
<th>Result Uploads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 17</td>
<td>71</td>
<td>9</td>
</tr>
<tr>
<td>Oct 17</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td>Nov 17</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>Dec 17</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Jan 18</td>
<td>36</td>
<td>7</td>
</tr>
<tr>
<td>Feb 18</td>
<td>45</td>
<td>4</td>
</tr>
<tr>
<td>Mar 18</td>
<td>49</td>
<td>17</td>
</tr>
</tbody>
</table>

### 4.1.3 Audience Response System

As a starting point the instant messaging client WhatsApp was installed on an Android emulator running on the teacher’s notebook. Once the phone number of the notebook is published to the class the students are able to send text messages to the teacher who, in turn, can see the incoming messages and react to them during the lecture. In one lecture with about 1000 students this setup was used as an initial prototype for one semester [147]. Due to the extensive spread of WhatsApp on the students’ smartphones it was adopted very fast. While convenient for the students, the teacher needed technical skills to install and run WhatsApp in an Android emulator. Also, the usability to navigate through the app (e.g. to view a photo) during a lecture is quite limited. This results in a high threshold that needs to be overcome before WhatsApp can be offered by a teacher to the students. WhatsApp is bound to phone numbers, therefore, for every teacher an additional phone number and most probably extra hardware is needed. While suitable for one or few lectures, scaling to many lectures, or even the whole university, with different teachers is an issue.

The developed audience response system, that was developed in this case study, is intended to be lightweight and easy to use and offers a low threshold to adopt it in the lecture. Of course each lecture and teacher have their own didactic style. Based on the experiences with the
4 Case Studies

prototype scenario it became clear that the limitations in usability, the high initiation threshold for the teachers and also the scalability are the main issues that need to be considered. These problems are addressed by creating a web application for the teachers and therefore granting them access using mobile as well as desktop devices. The teachers need to be able to set up their own communication channels so the application remains scalable even when many teachers are using it. Additionally, it was important to reduce preparation time needed to use the application. Politze et al. gives a detailed overview of different didactic scenarios supported by the application [148]. Later Drenckberg et al. showed how the reference architecture supported migration to a different database back end [149].

Supported Processes

The experiences with the prototype served as a reference during the development of the feature in RWTHApp: For the teacher it was very important not to share the messages directly among the students. This makes it easier for the students to overcome their inhibition threshold to ask allegedly stupid questions that are actually relevant. Also distributing messages among students may lead to escalate bragging or inappropriate behavior. Additionally, the review of existing systems made clear that spontaneous polls, which were not covered by the prototype scenario, are a use case that needs to be covered by the system. It should furthermore target the presence learning scenario where teachers and students are in the same room or at least in proximity. Also, the teacher can communicate with the students by addressing the audience directly. Apart from the usage in lectures it should be feasible to use the system in events with enrolled students and additionally other participants are present.

Especially the possibility of sending images is a feature that caught attention after it was used by the students to send photos from their devices. It showed to be convenient for the students sending photos of handwritten equations and drawings to the teacher instead of typing them on small screen devices.
4.1 eLearning Processes

To reduce the distraction of reading the messages while holding the lecture, some teachers have assistants scanning through and pre-process the messages to only interrupt the lecture when an important question comes up. To achieve this all messages send by the students are presented to the assistant in the web application. The assistant can then categorize the messages. While this approach proved useful for audiences with 300 or more students, this procedure requires an assistant being present during the lecture. Apart from receiving messages teachers or assistants can also send messages to the students. Two ways are offered to send a message: in response to a single student but also as broadcasted messages to all students in the audience. When pre-processing messages, this allows providing feedback to students even though their question is not directly answered by the teacher.

In order to get a more condensed view of the students’ opinions, the teacher can send a poll request to the students giving them the possibility to select between one of four choices. The choices are presented as four buttons to the student, labeled with the letters A to D. To give these choices a meaning teachers have to present a question and a description according to these four choices using another medium like presentation slides or the blackboard. The general idea is that the teacher should stick with the medium that is used for the majority of the lecture to formulate the question and therefore distraction by starting the poll is kept to a minimum.

System Architecture

The audience response system consists of three parts: two clients, for students and teachers and the server side back end. The client for students is integrated into RWTHApp, reusing existing installations on their devices for maximum availability. It is therefore implemented as a hybrid mobile application that can be installed on most modern platforms, most commonly Android and iOS. The client for teachers is a stand-alone web application that can be accessed via a web browser. The application can handle multiple users at the same time allowing teachers and their assistants to access the system simultaneously.
Both client side applications communicate via the REST interface offered by the tier 4 services with the server back end. The back end processes the requests and persistently stores messages in a database. This database also holds metadata about the channels such as lecture times and ownership information.

## Results

The feedback feature is generally available to all organizations at RWTH Aachen University. Every member of an organization can create new feedback channels supporting their lectures and other events. To allow access for visiting teachers a private URL is generated for every channel that allows access to the teachers view without prior login. This link can be passed, e.g. via email, to make accessing the channel easier by not having to handle login credentials during the lecture. This self-service channel creation allows teachers and other members of the staff to independently organize their events and thus increases scalability of the system.

Currently the feedback feature is supporting more than 45 lectures and other events. The number of participants ranges from as few as 50 to more than 1000. While most events are traditional lectures in a single lecture hall some events are broadcasted live to a remote lecture room or even to multiple locations like the students’ homes. These and other aspects, make each lecture unique in some way. The usage statistics in Figure 4.6 show the number of messages sent by the students and the number of lectures supported on a monthly basis. The chart shows typical fluctuations at the start of semesters where usage drastically increases and then gradually decreases during the semester.

Since the interface offers machine readable access to the messages sent during the lecture this can also be used for offline analytics. These analyses include the number of messages sent during the lecture or some time frame within. This effectively gives teachers the ability to perform their own specific analysis of the messages on their channels.
4.2 eScience Processes

The area that got into focus for the application of the reference architecture more recently aims to support the digitalization of the research process. Supported processes again span across different departments and groups of users within the organization. Typically, the previously regarded teachers are also active researchers at a university. It is therefore only natural to supply a common infrastructure to support digitalization of teaching and research processes.

At the time being, the research process, however, did not undergo the standardization and generalization procedures that have been observed in the area of eLearning. The case studies considering processes in the area of eScience therefore need to focus more on supporting future changes. This places the dimensions of Impact of Change and Comprehensibility into focus for the evaluation of case studies in this area.

4.2.1 HIApp

During summer term 2015 the chair of ancient history conducted a student project *The Roman Surroundings of Aachen* as part of their lecture. The goal was to look at the wider area of Aachen in ancient roman times and to interpret the regional role within the historical context. Based on on-site research and interviews with citizen scientists, results of several
smaller research projects, carried out by student groups, should then be published as an interactive tour guide mobile application.

As part of this scenario the students and citizen scientist create a content for a content driven mobile application that displays the results of their findings and makes it available to their peers. After a review, these results are then published and will be available to other researchers, a wider public and also to the students of following semesters who could in turn continue refining the work that was previously done. This way the collected content is growing over time and new features can gradually be added to the application. HIApp thus is an example how an individual research project is supported by the general infrastructure. The case study and its application within the university and research context were previously discussed by Politze et al. this section primarily highlights details relevant for the implementation [150].

**Supported Processes**

The media files and articles gathered by the students are stored in an online repository. It combines the files with technical metadata such as image resolution or video length as well as organizational metadata such as a written description or information about the authors. This is especially necessary when data is going to be reused in future lectures. In this special case a metadata field for GPS coordinates was introduced. The GPS coordinates are then used to browse the repository using an interactive map and as displayed in figure 1 this is increasing the overall interactivity working with the stored files. Retrievability is also the key to reuse of the data in future lectures or other projects. This is supported by further search and tagging abilities in the repository. Apart from a full text search, the repository can also be explored using different kinds of filters and views.

When authoring content, the users only interact with the repository software as a front end. It offers rich editing capabilities, automatically generated forms based on the defined metadata fields as well as upload and management of media files. The repository also features a mobile preview that shows the content as it would appear on a mobile phone. This allows authors with only fundamental technical experience to use the system and create content based on their current research.
4.2 eScience Processes

System Architecture

Technically the system consists of these parts: The online repository uses Microsoft SharePoint to store the media files and metadata gathered by the students and includes an online editor to edit their articles and arrange the media to be presented to the user. Secondly, a mobile application accesses the content. To reach all the different mobile platforms that are already used by the students, the application uses platform independent HTML5 and JavaScript technologies. Apart from displaying the content created, in the first Version of the application two main features are planned: A hierarchical view of all the articles that can be used as a table of contents and an interactive map that displays all content that was tagged with GPS coordinates.

From the articles that were edited with the editor in the repository the application renders a mobile view. Apart from textual information written by the students these articles also display images and videos that were inserted using the editor. The videos are converted into formats that can be decoded by smartphones with less computing power and thus are feasible to be viewed on-line while the user is on site. To allow navigation between related articles they can be interlinked with each other as well as with external web resources such as YouTube videos or web pages.

The reference architecture is used to define the access layer for the mobile application that allows a wider public access to the data. As depicted in The tier 4 services for collaboration that were implemented on the SharePoint platform are used by the mobile application in order to access data and metadata. The application then and displays the articles to the users. Additionally, the GPS coordinates in the metadata and the access to GPS sensors of the smartphone allow to locate the points of interest described in the articles.

Results

The process of gathering all resources and the results of this interdisciplinary lecture with a strong practical relevance is now done. The students’ overall feedback was very positive. The created infrastructure,
however, can support a multitude of different research projects and makes it easier for researchers from non-technical fields to create and publish content driven mobile and web applications. This makes the system essentially a headless content management system as described by Schürmann [151] that based on centrally provided services and was successfully integrated into local research processes.

4.2.2 Metadata Tool

Registering metadata is the prerequisite for re-discovering and re-using data, but often requires extra effort from researchers. However, this task is the easier the earlier it is performed within the research process since the context of the data generation is more present to the researcher. The Metadata Tool therefore allows registration of metadata independently from the local IT environment of the researcher, allowing every researcher to document metadata as soon as research data is produced.

The Metadata Tool allows researchers to define their own storage system for their research data as well as discipline specific metadata schemas since many institutes have their own IT infrastructure and sometimes strong privacy and confidentiality concerns regarding even meta information about their research. However, structure and format of metadata need to be standardized allowing a seamless transition to a long term archive or publication repository.

During the development process of the Metadata Tool there were several prototypes that eventually emerged to the software system presented in this section. The intermediate stages were previously described by Politze et al. [152]–[154]. The thesis of [155] discusses the capabilities of the presented version in great detail [155].

Supported Process

Researchers log on to a web interface where they can register their data and assigning it a PID that is used to link data and corresponding metadata. To describe their data, they can choose from a number of predefined metadata schemas. By conventions these use Dublin Core
[156] as a basis. Depending on the institutional context of the user, it is also possible to preset a metadata schema. To achieve maximum reusability the metadata schemas and presets are defined in RDF format. Researchers can create an register data originating from some kind of pipeline aia the REST API provided by the reference architecture. This allows to automatically collect and register metadata from local systems such as test stations or measurement devices. Storage of the data and metadata can be realized in two different workflows depending on the preferences of the researcher.

The Metadata Tool therefore supports two workflows: private and integrated as shown in Figure 4.7 and Figure 4.8 respectively. In the private Workflow formalized metadata is stored together with the research data in local archive provided by the researcher. Researchers then can use RDF formatted metadata files generated by the metadata tool to store the metadata. The Integrated workflow, however, stores the metadata in a centralized database. With the number of users and disciplines the available metadata schemas will be continually expanded. The researcher also has the option to transfer the research data into the central archive. In both cases a PID is issued to identify the dataset and link data and corresponding metadata, regardless of where the files are actually stored.

When the centralized infrastructure of the integrated workflow is used, the values of the PID can be automatically maintained and updated if their location changes during the life cycle of underlying storage systems. In the private workflow, however, researchers have to take care of maintain these links.

Even though the workflows store resulting information on other servers both essentially follow the same steps: When inserting a metadata set into the repository a PID is created to globally identify and link the dataset and corresponding metadata. Key value pairs of the PID then hold information about the location of the actual artifacts. While the PID is globally resolvable, artifacts can only be accessed by the researchers having access to their storage location.

Especially in the integrated workflow it is necessary to additionally store information about the author of the metadata. This allows basing future
workflows on the authorship such as role based access control to limit access to the metadata to certain groups of the university like members of the same research group.

**System Architecture**

This case study combines the PID and metadata services of tier 4 of the reference architecture to implement the previously defined processes. Among the already described metadata creation process, in the future, the system should also support the metadata retrieval process. Both processes use the PID and metadata services defined by the implementation of tier 4 that, in turn, base on the *epic* system and a triple store in tiers 3 and 2. For service stability and to allow monitoring the tier 1 services for logging and caching are used. The user profile service from tier 0 supplies the information needed to fill preset values in the metadata files.

As a proof of concept the system uses a HTML web application to make the processes accessible by users. It should however be noted that the processes defined in the case study aim to be implemented as part of a discipline specific research process automating the creation of metadata and PIDs as research data artifacts are created. The final architecture thus greatly depends on the implementation of the discipline specific research process as shown in Figure 4.9.
4.2 eScience Processes

Results

As a minimum set of metadata Dublin Core is used. Finding the right discipline specific metadata schemas has turned out to be difficult. Discipline specific metadata schemas have yet to be analyzed and evaluated together with data curators from the university library and researchers from the discipline.

The HTML web application that implements the described process is only intended for administrative or experimental access. It therefore is rather in a prototype stage to evaluate the overall concept and the process design. The current implementation was therefore evaluated by researchers from different disciplines. According to the user feedback it will be adapted and further extended to the more specialized requirements by the researchers.

4.2.3 simpleArchive

Long term data storage is becoming more and more important as research funding organizations require researchers to make their data available
Figure 4.9: Integration of the metadata tool architecture in a local workflow and re-usable. For example this can be seen by the recommendations of *Deutsche Forschungsgemeinschaft (DFG)*:

Primary data as the basis for publications shall be securely stored for ten years in a durable form in the institution of their origin.

— DFG, 2013 [157]

Also the *German Council for Scientific Information Infrastructures (RfII)* sees long tail archival of research data as a substantial task for providers of research infrastructures:

Long-term archiving and ensuring long-term availability are two of the most important tasks of a national research data infrastructure.

— RfII, 2016 [158]

The current offer for long tail preservation of research data at RWTH Aachen University uses a tape archive and a respective proprietary client that requires technical expertise not necessarily available to every
researcher. Therefore, the goal of simpleArchive is to allow researchers to access long tail preservation capabilities without the need of acquiring technical knowledge by offering a SaaS that uses state-of-the-art web technologies. Archiving a file for the user is made as simple as uploading a file on a web page. This case study has been previously presented by Politze et al. showing the archival process that is also presented in this section [159]. Later Politze showed how the process can be scaled [160] across organizational boundaries.

**Supported Process**

This case study features two sub processes: archival and retrieval of research data. For the archival process as shown in Figure 4.10 a data artifact is uploaded by the researcher. The data is then temporarily stored on a server before being transferred into a tape archive where the data can be stored for long terms at relatively low costs. The actual archival process is transparent to the user. Immediately after the user has completed the upload of the data a PID is issued which in turn may be used for referencing data artifact in a text publication or on a web page. According to requirements e.g. posed by funding or publishing organizations, simpleArchive retains a copy of the data for at least ten years.

Figure 4.10: Simplified overview of simpleArchive processes
4 Case Studies

If researchers want to access or share the data at a later point in time, simpleArchive allows stored data to be retrieved by researchers using the retrieval process shown in Figure 4.10. To restore data from the archive the researchers use the PID issued to identify their data. Researchers can request to restore artifacts which are then copied from the tape archive to a temporary file store. Based on the temporary file a download URL is generated that researchers can use to access the archived data or to share it with their peers. The generated URL is only accessible temporarily and the restored artifact is deleted after two weeks and the restoring process needs to be triggered again. Furthermore, if the PID is obtained by other researchers a landing page allows them contacting the researcher who archived the data. Using the restore process described above the download URL can then be passed to allow access to the data within a community of researchers.

Within both processes the interaction with the archive is asynchronous and can take several hours to complete. Upon completion of each of these steps, the researchers are notified with the relevant information like PID or download URL respectively.

System Architecture

An existing tape archive provides a cost-efficient solution for long tail archival of research data. Access of the stored artifacts, however, requires using specialized software and is generally not suitable to be used by non-technical personnel. This specialized software and a C++ API provided are therefore technology dependent services from tier 2. Notifications for the completion of asynchronous process steps utilize the notification service defined in tier 1.

The actual process defined by simpleArchive was implemented as a HTML web application and can be used by researchers to upload research artifacts that are then archived and restored according to the previously described process. The system uses the archival service of tier 4 to interact with the actual archive. The identification of research artifacts uses the PID service.
4.2 eScience Processes

Results

Even though it was a requirement posed by different organizational units within RWTH Aachen University, *simpleArchive* is adopted quite slowly by the researchers. Within the first six months of operation 8 different organizations have used the system to archive a total of 61 files. Table 4.3 shows the progress of adoption during the prototype phase of the system. Even though the application was highly demanded by researchers, the observed rate of adoption is quite low. The reasons can only be speculated: AS there is some continuity of returning organizations and *simpleArchive* aims towards archiving research data sets at the end of the research data life cycle possibly only few data sets of the early adapters reach this sate at a time.

<table>
<thead>
<tr>
<th>Month</th>
<th>Organizations</th>
<th>Files</th>
<th>Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>new</td>
<td>returning</td>
<td></td>
</tr>
<tr>
<td>Oct 17</td>
<td>3</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>Nov 17</td>
<td>1</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Dec 17</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Jan 18</td>
<td>1</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>Feb 18</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Mar 18</td>
<td>1</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>61</td>
<td>8.23</td>
</tr>
</tbody>
</table>

As part of the case study, *simpleArchive* was not only implemented at RWTH Aachen University but also at TU Darmstadt. Both systems make use of federative OAuth infrastructure to share a single temporary file system instance at Aachen University to generate download URLs. This ability of reusing services across organization boundaries depicted in Figure 4.11 especially shows the reusability of the services implemented as part of the reference architecture.
4 Case Studies

Figure 4.11: Schematic of *simpleArchive* at RWTH Aachen University and TU Darmstadt

### 4.3 University Processes

This third area of application for the services of tier 4 focuses on general university processes. It reuses the already introduced backend systems from Section 3.5.1 to support a different set of processes. Processes spanning the university and affecting the mainly administrative tasks usually are centralized within the organization. Additionally, these processes convey authoritative information. Thus, identification of the users is crucial in this area.

Even though these case studies were initially not in the focus areas of eLearning or eScience they provide valuable insights on the properties of the architecture. It is thus necessary to assess how the architecture supports *Impact of Change* and *Reuse* in the dimension of SoC, as additional processes are added. Compared to the area of eLearning, Processes in this area are usually well-defined and follow widely standardized patterns. This effectively reduces the need for individualization,
but shifts the focus towards *Formality*. It is thus important that the architecture allows supporting previously defined processes and considers organizational and technical constraints. The central and authoritative character of the supported processes furthermore poses requirements towards *Availability* and *Integrity*.

### 4.3.1 Eduroam Device Management

The creation of device based credentials is a step to reduce the impact of security vulnerabilities of *eduroam* as discussed by Brenza *et al.* [161]. Since the actual vulnerability originates from implementation errors in the operating systems it cannot be addressed directly. However, this risk mainly originates from the fact that *eduroam* credentials, that can be retrieved using a man in the middle attack, are mostly the same credentials as for other university services. To reduce the risk of identity theft, *eduroam* credentials are randomly generated per device and are therefore not usable to access other university services. While this does not actually solve the initial problem, it reduces the impact and thus allows managing the risk.

Based on the general idea presented by Decker *et al.* it took several years to finish the implementation of the case study due to various organizational issues [162]. Later Decker *et al.* showed the successful implementation and roll out of the application [163].

**Supported Processes**

A web application allows managing the device credentials and provides access to additional information like the MAC address of the device and its last logins. Even though the accounts are randomly generated, the back end systems associate them to the initial user for administrative purposes or to be able to identify users in case of infringement.

The *eduroam* device management processes therefore mediate between the different federative infrastructures *eduroam*, using the *RADIUS* protocol, and the federative OAuth2 as introduced by tier 0. The processes and their implementations defined by this case study can
therefore be reused by other institutions already offering eduroam services to their users. Apart from RWTH Aachen University, the eduroam device management was also rolled out at Jülich Super Computing Centre (JSC) as part of this case study. JSC uses the implementation in a SaaS model, thus several extensions to the device management process described above have to be defined to allow handling of accounts created by their users. The main issues addressed are the life-cycle of created logins and the ability for local administrators or help desks to support their users when supplying eduroam device management as a service.

The device credential life-cycle management process is defined in three sub-processes: (1) when the user logs in, the identity provider presents a unique user id and a flag telling the service if the user is eligible to use the device manager. This is usually the case for students or employees who are able to login but not for alumni who may retain their login after graduation. (2) After six months without activity, the created credentials are automatically removed. This is done to comply with the usual semester based organization that many academic institutions follow. (3) If more fine granular control is desired passwords have to be kept alive by the participating organization. This is performed using a white list containing all user ids that are currently eligible to access the network.

To enable local support for the users, IT support departments need access to some information logged when using eduroam. From the experiences of the IT-ServiceDesk at RWTH Aachen University it was clear that especially the information logged during authentication of the user in the eduroam network are crucial for support. Participating organizations therefore retain access to current authorization logs of their users, allowing them to trace login problems for example caused by misspelled passwords, wrong settings on the client or network abuse. The login information for each device also provide additional value to the users. They may therefore individually see why their device is currently not connecting to eduroam or can check where their device has been used. In case of irregularities that would also occur when the password was hijacked, the users are able to react by resetting the compromised credentials.
System Architecture

Generally eduroam uses a federative network of trusted RADIUS servers to authenticate users. Upon connecting to the wireless network, users supply their credentials, a scoped user name and password, that allow resolving the RADIUS server of their home institution which in turn validates the password. The user records are stored in some kind of directory service at the home institution. Considering the existing architecture, the implementation of the device manager is integrated deeply into the existing infrastructure supporting eduroam and using the RADIUS protocol to authenticate device credentials. Figure 4.12 shows the different parts of the device management aligned to the tiers defined by the pSTAIX reference architecture. The three supported processes: device management, access control and local support base on the service endpoints of the eduroam service. The RADIUS backend is separated in two logical systems, a directory service holding the device credentials and a database containing the authentication logs needed for local support and users. The logging service allows error and KPI monitoring and the caching service reduces the load on the backend systems for repeated requests.

The implementation of device management process for the user and the local support process is done using an HTML5 web application that can be accessed by any browser. For the user access control no user interface was constructed but the services are used directly by local service providers using the service as the process is deeply integrated into local identity management processes. This local integration into existing processes is especially easy since the eduroam service already abstracts the interfaces to directory and authentication log services.

Results

As a part of the case study the resulting process and its implementation were also established at JSC. In this case JSC supplied a local instance of the authorization service that integrates with the remaining infrastructure on a federative level and thus reuses the software instances at RWTH Aachen University. It is currently planneed that th service
will be made available for other cooperating institutions in the same manner.

In the current implementation as a HTML5 web application, generation of credentials and set up of eduroam on the device requires several manual steps. In order to enhance usability on selected platforms the processes could be re-implemented to allows automating the setup process using a native application directly on the device. Especially smartphone platforms already offer an initial internet connection using the mobile network. Using the current SOA, an eduroam setup application would need to transfer only few kilobytes of data to create a new account and conveniently set it up on the device. This would make it easier to connect new devices to eduroam.

The process was gradually adopted by the users of RWTH Aachen University since existing accounts continued to function. As shown by Figure 4.13 about 18,000 users have used the process to create one or
more accounts for their devices during a pilot phase in 2016. As of December 2016 RWTH Aachen had about 60,000 eligible users.

### 4.3.2 Chat Support

A department of the IT Center, the IT-ServiceDesk, serves as a single point of contact for all users of IT services at the university. Based on the omni channel support concept as described by Hengstebeck et al. [164] the IT-ServiceDesk operates several channels that users can utilize in order to contact IT support. The Chat Support thus is one of many channels monitored and needed to be integrated into the existing infrastructures and processes.

The Chat Support therefore reuses the infrastructure already established for the audience response system to connect users with support agents of the IT-ServiceDesk of the IT Center. Additionally to the integration into RWTHApp, the Chat Support is also available as a web application integrated into several web pages.

**Supported Processes**

In general the Chat Support should enable users to interactively communicate with support agents of the IT-ServiceDesk. The support agents on the other hand should be supplied with the necessary technical and

![Figure 4.13: Adoption of eduroam device management process](image-url)
organizational information in order to support the user. This information include: the originating web application, the user’s name and affiliation with the university and email address. If users are logged in, the system is able to automatically collect this information from the tier 2 user profile services already provided and used within the other case studies. Anonymous users, however, have to supply the necessary when starting the chat.

The chat follows a three state life-cycle process depicted in Figure 4.14: first the user opens the request which is then displayed to the chat agents. One of the support agents then takes ownership of the chat and processes it. After the user’s request has been answered, the chat is closed by the support agent. Finally, it is forwarded to the existing ticket system for archival and statistical purposes.

System Architecture

This case study clearly shows the reusability of services defined in tier 4 for other processes. Except for gathering user information from the user profile service in tier 2 all critical parts base on the services already provided by the audience response system. Additional to the already introduced message types text, image, poll and pollresult Chat Support uses the meta type to transfer status messages that are not directly visible to the user but are handled by the application.

Figure 4.14: Chat Support process
Meta messages are stored in a machine readable JSON format and are required to define the state of the current chat, transfer the user’s name and contact information as well as count the number of support agents currently online.

Additionally to the already existing services in tier 4, the process is supported by a web application and a mobile implementation within RWTHApp. Both, the web application and RWTHApp serve as an entry point for users to start chats. The web application, however, is also used by support agents to manage and respond to chats. The application automatically recognizes the available support agents and adjusts the maximum number of parallel chats accordingly.

**Results**

Users and service agents perceived the introduction of *Chat Support* as an additional communication channel for as positive. As shown by Figure 4.15 the application is regularly used and the number of users is growing as the *Chat Support* is integrated into more applications like the *eduroam Device Management*. According to an internal report of the IT Center in 2017 there were a total of 59,021 requests 1,322 of which were chats [165].

Together with the introduction at the IT-ServiceDesk, the Information Center of the University Library also introduced the system for individual consultation of students. Since 2017 *Chat Support* is also used by the student council of RWTH Aachen University.

![Figure 4.15: Number of messages sent in chat support](image-url)

[165]
5 Conclusion

The preceding chapters outlined the construction of a reference architecture to support implementation and data protection of processes in distributed system landscape. A proposed implementation used state-of-the-art technologies to support several case studies within the university context. This infrastructure further aims to be an integration platform allowing users to connect their own, locally defined processes to centrally provided services. In order to validate whether these goals were achieved by the implementation it is validated according to the previously defined conceptual foundation and its dimensions of evaluation: process-awareness, information security and separation of concerns. Based on this final evaluation it is then possible to answer the research questions posed in Chapter 1.

5.1 Evaluation of the Implementation

Apart from assessing the implementation using the successful support of case studies presented in Chapter 4, a conceptual evaluation of the way the architecture was put into place is required. It is especially important to validate how design decisions for specific implementation of workflows influence the ability of the infrastructure to support processes in distributed system landscapes. Therefore, the implementations of the different defined tiers are evaluated individually and how their interaction influences the different dimensions of evaluation. This is especially important for those tiers that generalize across processes, namely tiers 0 and 1, and the interfaces used between the other tiers.

Within tier 0 the OAuth2 workflow and its extensions for cooperative and federative contexts, as introduced in Chapter 3, serve as a basis for a centralized auditing structure. Apart from providing reports to service providers this approach allows making information available to the user and therefore leads to a more user centric and transparent auditing. By centrally collecting the auditing information the enforcement of data privacy laws and best practices can be controlled in a better way. This behavior allows to centrally control data protection and governance
within the distributed environment. The services in tier 1 then support interconnection of distributed services and integration with commodity technologies. On the one hand providing central caching and logging services the services in this tier increase general reliability of the connected services of tier 3. On the other hand storage user profiles and notifications services allow lifting legacy services to a common technological level and reduce disruptions of processes. Tier 4 finally provides necessary integration according to defined business processes and introduces the ability of individualization and integration of processes within local environments.

5.1.1 Evaluation of Process-Awareness

The different properties of the tier-architecture aim to provide a basis to create process-aware services. To assess to what extend the implementation satisfies process-awareness it is validated according to the dimensions of Formality, Suitability, Conceptuality, Enactability and Comprehensability.

In order to satisfy Formality, processes have been modeled in advance and independently of the implementation of the reference architecture. Within each case study, UML was used for technology independent modeling of processes. Based on these models, the specific processes were implemented using commodity imperative programming languages. The endpoints specified by tier 4 that are used within the processes utilize standardized methodologies and technologies like REST and HTTP to allow usage of the process-aware implementations from a variety of programming or runtime environments that can be chosen to be suitable for implementation of the processes. Additionally to modeling languages like UML, this abstraction layer provided by process-aware interfaces of tier 4 can also be used by approaches combining modeling and implementation of processes using specialized workflow engines.

The services provided by tier 4 are basing on the nomenclatures and semantics of modeled processes. In order to satisfy the dimension of Suitability, it is therefore necessary to translate from process-aware to technical terminologies and nomenclatures used by the back end systems.
of tier 2. This translation is obviously dependent on the process domain and has to be adopted for new processes or changing back end systems. Tier 3 hence created an intermediate step to abstract specialized or even proprietary interfaces offered by tier 2 and therefore establishes process independent interfaces using standardized and widely used protocols. This primarily reduces the effects of vendor lock-in. This allows the translation into process-aware services to be independent of back end technologies and to reuse knowledge about standard processes and technologies when implementing process-aware services in tier 4.

Based on this principle of gradually hiding the technical complexity of the back end systems and providing a well-defined and consistent interface. Users generally only interact with the process-aware interfaces offered by tier 4. Additionally using the abstraction introduced by tier 3 allows transferring modeled processes from one back end system to another and therefore satisfying the dimension of Conceptuality. By using the provided interfaces, modeled processes therefore rather focus on the supported processes instead of technological requirements of back end systems and therefore allow directly transferring concepts from modeling to the implementation of processes. Additionally, tier 3 introduces defined service levels for backend interfaces and thus defines the purpose for which they can be used. As processes evolve over time, this additionally allows migrating them to the most suitable back end systems.

Since the implementation uses widely available technologies like REST, HTTP and the OAuth2 workflow it presents a low threshold to be integrated into local research environments and processes. Additionally, in the cooperative and federative settings, the OAuth2 workflow allows scaling processes across organizational boundaries. Sharing common understanding of central processes and the ability of customization and adoption to local needs further increases the acceptance of the implementation. The cooperative and federative workflows defined by tier 0 furthermore allow gradually integrate more systems into the architecture effectively extending its reach to decentralized systems that can be included in central processes. By lowering both, the technical as well as organizational thresholds this satisfies the dimension of Enactability.
Consistency according to defined processes is key for the REST interfaces exposed by tier 4. Combining both, the consistent use of technologies and consistent nomenclature across processes allows users to transfer their knowledge about real world processes to implementations and local adoptions. While the technological consistency is mostly convention and supported by a variety of runtime environments the challenge is consistent modeling, nomenclature and semantics especially when adding new or extending already existing processes and process supporting interfaces to tier 4. To satisfy the dimension of Comprehensability it is thus necessary that the process-aware interfaces offered by tier 4 are actually modeled with processes in mind. Again tier 3 serves as a mediator to reduce the influences of back end specific implementations towards modeled interfaces and processes. Additionally, the organizational model proposed by the reference architecture, allows internal separation of development teams to further reduces these effects and only work towards defined interfaces.

5.1.2 Evaluation of Information Security

Information security is another design principle for the reference architecture. Instead of being defined by the relationships of the different tiers, measures to meet the requirements in these dimensions are mostly provided by single tiers within the reference architecture. Generally the transport protocols, especially HTTP, used internally and by processes accessing REST interfaces, need to be properly secured. Being a commodity technology HTTP offers widely spread and well standardized transport security using HTTPS that is at this point assumed to be used extensively for all REST interfaces.

On process level, the implementation of tier 0 uses the OAuth2 workflow in order to secure and personalize access to the offered services and convey necessary context information to identify users and their permissions wherever required by other tiers. Hence, within the architecture the authorization service in tier 0 is the single point of authoritative identity information. The authorization $n$-tuple $(v, c, i, s)$ describes the user and process context of an operation and, in principle, is made available at in each of the tiers to identify the context within a current
operation. While the authorizations are managed by a trusted organization, the newly introduces cooperative and federative settings allow the authorization tuple to be passed across organization boundaries if they provide a trusted endpoint and if the user consents. The integrated monitoring and auditing capabilities allow users, application developers and service providers to verify that processes are operating within their limits. As part of the architecture, the OAuth2 workflow in tier 0 offers user-centric security as a service and therefore satisfies the requirements posed by the dimension of Confidentiality.

Centralization of the management of authorizations allows process-aware services in tier 4 to focus on the modeled process and validity of translations between process-aware and general services rather than security concerns. Especially abstractions in tier 3 typically require to establish the user context. Therefore, the OAuth2 workflow as a service allows implementations to request authorization contexts as needed. The same independence of abstraction layers is used for the data models accessing different back end systems. Each service in tier 3 individually controls how data is processed and passed to the back end systems. The central leverage point to satisfy the dimension of Integrity therefore is validation of the correctness of the implementation tier 3. This set of interfaces therefore is validated once by specialists rather than repeating this conversion for each process-aware service in tier 4.

To satisfy the dimension of Availability tier 1 introduces several services that are used by the higher tiers: Most importantly the Caching and Logging services. On the one hand the caching service reduces the load on back end systems. This is especially important to allow the inclusion of systems that are not intended for interactive use and that otherwise would decrease the overall stability. On the other hand the logging service allows implementations in tiers 3 and 4 making their actions transparent and allows resolving hidden dependencies. This is especially important in the distributed setting since processes using several back end systems only function properly if all of these systems are operating correctly. While this is one of the greatest benefits of the architecture, this issue also presents one of the greatest challenge. By decoupling the back end systems, standardized abstractions and process-aware services
into functional independent tiers, they can be maintained separately according to the locally selected ALM processes these effects are further reduced.

### 5.1.3 Evaluation of Separations of Concerns

In order for the architecture and its implementation to remain maintainable the dimensions of evaluation for SoC assess the coupling of the services. As shown by Figure 5.1, these dimensions have to be regarded from two viewing angles: technical (horizontal) and service (vertical). As previously discussed in the dimension of Availability horizontal separation is widely accounted by the tier model. Hence, it is necessary to assess how the architecture also ensures SoC along the vertical axis in more detail.

Each back end system and service that is being integrated into the architecture is independently engineered. This is especially important when additional distributed back end systems are integrated and made available for services within the architecture. Also, processes that are supported by the process-aware services of tier 4 can be freely added according to defined service levels and processes. For each service the suitable back end systems are selected, based on their availability and

![Figure 5.1: Separating technical and process concerns in the reference architecture.](image-url)
service level defined by tier 3. Therefore, when performing additive changes within the tiers 0 to 3 existing services and processes are not affected. The same holds for the introduction of new process-aware services in tier 4 and therefore satisfies the requirements of the dimension of \textit{Impact of Change}. Due to the abstraction of technology dependent interfaces in tier 3 even exchanging entire back end systems can be masked and if required concepts are correctly mapped existing implementations of processes using process-aware services do not need to be changed. This separation not only considers technical dependencies but also organizational as the architecture explicitly allows different ALM processes that orient towards the requirements of integrated services or back end systems.

In order to satisfy the dimension of \textit{Reuse} the architecture allows combining services from tier 3 into process-aware services in tier 4 by using the abstractions as consistent building blocks for newly added and adopted processes. In turn, the services provided by tier 3 are not bound to specific processes, so they can be reused wherever needed according to their specific service level agreements. This is especially important in the distributed scenario as it implies that back end systems that are integrated into the architecture can be reused in processes in a way that provides the most value for the users without their operators specifically knowing about the process. The abstraction of proprietary protocols in tier 3 and common semantics and terminologies according established processes in tier 4 further allows external developers to grasp concepts of the service and focus on providing additional value with an implementation of process in the local setting.

Allowing processes to cross system and even organizational boundaries introduces a high degree of complexity into implementations and likely makes processes error prone if single services do not function properly. To allow identification of error prone systems and to take actions to reduce the impacts of errors, the architecture generally does not allow circular dependencies of the services as services should only be composed by services of lower tiers. Additionally, the implementation of the architecture introduces two specific services to satisfy the dimension of \textit{Tracability}: the logging service in tier 1 and the auditing service as part
of the OAuth2 workflow in tier 0. The logging service allows centralized, internal monitoring of the integrated services and therefore allows tracing of errors or other events across system boundaries. Regarding organizational boundaries this is often more problematic if separate instances of logging services exist at each of the distributed organization. In this case the auditing service provides essential information of each service that requested the user’s context. The more detailed extensions of the service should allow reconstruction of events that lead to erroneous situations. Being provided their own raw data this can even be performed by application developers and users themselves further fostering the distributed scenario.

5.2 Answering Research Questions

Based on the findings from the previous chapters, this section answers the research questions posed in chapter one.

Research Question 1: How can the architecture provide a comprehensive, process independent and reusable model to access existing data stored in heterogeneous distributed systems of the university?

Abstraction of interfaces defined by proprietary or vendor specific implementations and independent description of the process. Translating between process and technical levels is necessary. In order to be reusable technical infrastructures need to commit towards specific service levels only demanding to retain certain technical bounding conditions like number of elements or total load. While processes remain within these limits back end systems can technically be reused. As presented in the case studies, back end systems then become general purpose infrastructures. Based on the available back end systems implementations of processes are then able to chose the most suitable offer. This, however, makes the abstraction layer is especially important. Access to these services need to adhere standard protocols and semantics in order to alternate back end systems and to limit the impact of these abrupt changes. After all mediating between the different release, deployment and operation life-cycles can only be done if abstraction layers guarantee certain sustainability.
Research Question 2: How can the architecture provide means to increase reliability of these distributed services for existing and new processes?

A distributed process bases on several heterogeneous services. The operational reliability of the process then is determined by the least reliable service. In the university context this is especially important in distributed systems that include services that initially were not intended to be part of a distributed infrastructure. It is once more important to remain within limits of each of the back end systems in order to keep all processes operational. Explicitly defining interfaces that are intended to be used by other services and specifying technical conditions under which they can be used not only impacts reusability of services but also increases their reliability when used within new use cases. Additionally, the abstraction needed to obtain process and cross process nomenclature and semantics may pose additional load on the services. Caching strategies have proven to be essential to reduce computational load by storing already computed responses for a limited time and therefore reducing the effects of repetitive requests on the back end systems. Apart from the technical support for existing systems, using the centralized logging and error tracking abilities provided by the architecture additionally allows analysis of cases that do not meet the reliability requirements. Defining and measuring KPIs as part of this logging systems then allows checking measures taken to increase reliability using measurable goals.

Research Question 3: How can protection of personal or confidential data be supported by the architecture when sharing data across process and system boundaries?

Sharing back end systems between processes is a central concept of the pSTAIX reference architecture. For providers of these services, the authorization service provides the n-tuple \((v, c, i, s)\) as fundamental context information allowing the back end system not only to establish the user’s context but also identification of the application or process that is requesting a resource for the user. Allowing the context resolution within all tiers provides exhaustive information for all integrated services. Additionally, the user centric security concept introduced by the reference
architecture allows increasing trust in the supported processes: Providing
users with full information about the actions invoked interactively by
themselves or by automated processes on their behalf enables users
to review how their personal or confidential data is accessed. Using
the authorization service, users can furthermore view all processes that
potentially access their data and can revoke access rights. This form
of explicit consent from the users hence allows users to decide which
processes are allowed to access resources on their behalf and therefore.
This provides the user the responsibility and the opportunity to decide
if a certain implementation process is trustworthy or not and therefore
enables external developers that are trusted by the users to target
specific user groups with individual implementations.

Research Question 4: How can the architecture generalize to support
processes working across systems in distributed local, cooperative and
federative service landscapes found in the university sector?

The necessity of scaling infrastructures across organizational bound-
daries is a common requirement in the university context. Especially for
web applications, existing approaches already provide implementations
that are limited towards providing single services across organizations.
Within a cooperation, user groups are shared among integrated systems.
The $n$-tuple of fundamental context information introduced as an ab-
straction layer for the cooperative scenario and serves to convey the user
context within the loosely coupled landscape. Processes implemented in
federative system landscape, however, an additional challenge is posed
by the need for the distribution users’ context information to the in-
tegrated systems. By extending the information transferred as part
of the users’ context the extended OAuth2 workflows and the already
existing federate SAML infrastructures can be effectively combined to
retain a further extended variant of the OAuth2 workflow supporting
federative processes with multiple authorization services. Both variants
are integrated with standardized OAuth2 and for back end systems
as well as implementations of processes the workflow remains. This
effectively allows scaling back end systems and processes basing on the
reference architecture generating a federative service landscape trans-
parently without technical changes to these services. Apart from the
provided technical scalability the previously discussed data security concepts are also scaled. Back end systems, processes and users always interact only with their local authorization service. Especially for user centric security this entails that authorizations are always managed by the users’ home institutions and reporting and auditing features thus are consistent across processes.

5.3 Outlook

Obviously the presented processes and use cases are only a first step towards process support in its entirety. Future extensions of the reference architecture and its implementation are necessary in order to support more processes. Especially the area of eScience automation of scientific workflows can potentially provide additional value to researchers. This requires, however, that tool support and process models in this area need to be further advanced.

As a result of the case studies spanning across organizations it became clear that scaling the architecture across multiple universities will continue to pose additional challenges. The reference architecture gives a basis for a future distributed SOA but especially tier 1 services need more fine-grained specification to allow interoperability and portability of services in higher tiers provided by different organizations. With the current implementation and specifications, however, it was possible to build a federative SOA at RWTH Aachen University and two partners. While the conceptual model provided the expected scalability, it gave valuable insights for future enhancements of distributed deployment and operation of the infrastructure.

To increase technical maintainability of the implementation, especially when multiple, connected instances need to be managed in a federative SOA a future enhancement should be to automate orchestration of the services. While tier 4 focused on common nomenclatures and semantics, tier 3 only lifts interfaces to standardized communication protocols and service levels. In the current implementation, services from the tiers are selected using expert knowledge and then connected using the defined interfaces. In order to truly scale across organizations this process can be
automated further. Services from tiers 3, 4 and respective applications could then be automatically connected based on their service levels or capabilities as it is for example proposed by Oberhauser [166]. As stated, this, however, will require to explicitly model these capabilities of services which in turn allows automated service discovery and composition.

Apart from the technical perspective, the processes implemented on top of the infrastructure should be assessed further. While the auditing capabilities of the authorization services already supply user centric reports, the general logging service in tier 1 is limited for administrative workflows. Both sources of information, however can provide in depth insights into the way process-aware services are integrated into existing workflows. Apart from validating conformity of modeled processes and real world applications this also allows to align process-aware services with user expectations and enhance user experience. The described level of audit and informational logging then enables tracking of user behavior and thus allows aligning processes modeled by the implementation with actual processes as proposed by Bolt et al. [167]. This underlines the importance of the extended cooperative workflow as a standard, to lower the hurdles for distributed service providers in order to generate the required information and therefore enables in depth analysis even within distributed SOAs.
Appendix

A.1 Federative OAuth2 Endpoint Definitions

Endpoint: Authorize
URL: Authorize
Query: ?client_id=[string]&response_type=[string]&scope=[string]
&state=[string]&redirect_uri=[string]
Method: GET

Endpoint: Code
URL: Code
Query: client_id=[string]&scope=[string]
Method: POST
Returns: {
    userCode: [numeric],
    deviceCode: [string],
    verificationUrl: [string],
    expiresIn: [numeric],
    interval: [numeric]
}

Endpoint: Token
URL: Token
Query: ?client_id=[string]&code=[string]&refresh_token=[string]
&grant_type=[string]&device_name=[string]
Method: POST
Returns: {
    expiresIn: [numeric],
    accessToken: [string],
    refreshToken: [string]
}

Endpoint: TokenInfo
URL: TokenInfo
Appendix

Query: ?client_id=[string]
Headers: Authorization: Bearer {token}
Method: GET
Returns: {
    isValid: [boolean],
    expiresIn: [numeric]
}

Endpoint: Context
URL: Context
Headers: Authorization: Bearer {token}
Method: POST
Body: {
    service: [string],
    resource: [string],
    operation: [string],
    cost: [numeric]
}
Returns: {
    isValid: [boolean],
    application: [string],
    mail: [string],
    displayName: [string],
    eduPersonPrincipalName: [string],
    eduPersonScopedAffiliation: [string]
}
A.2 REST Endpoint Definitions for Tier 1

Caching Service

Endpoint: Add cache entry
URL: Caching/{key}
Query: ?max-age=[numeric]&secret=[string]
Method: POST
Body: [binary]

Endpoint: Get cache entry
URL: Caching/{key}
Query: ?secret=[string]
Method: GET
Returns: [binary]

Endpoint: Remove cache entry
URL: Caching/{key}
Query: ?secret=[string]
Method: DELETE

Endpoint: Clear cache
URL: Caching
Query: ?secret=[string]
Method: DELETE

Logging Service

Endpoint: Append log
URL: Logging/Logs
Query: ?secret=[string]
Method: POST
Body: 

    

    service: [string],
    message: [string],
Appendix

stacktrace: [string],
resource: [string],
parameters: [string],
exception: [string],
level: [string],
timestamp: [string],
token: [string]

Endpoint: Get log
URL: Logging/Logs
Query: ?secret=[string]&count=[numeric]&skip=[numeric]
Method: POST
Returns: [{'
  service: [string],
  message: [string],
  stacktrace: [string],
  resource: [string],
  parameters: [string],
  exception: [string],
  level: [string],
  timestamp: [string],
  token: [string]
}, ...]

Endpoint: Append KPI
URL: Logging/Kpis
Query: ?secret=[string]
Method: POST
Body: 
  {
    id: [string],
    value: [numeric],
    affiliation: [string],
    resource: [string],
    note: [string],
    timestamp: [numeric]
  }
Endpoint: Get KPIs
URL: Logging/Kpis
Query: ?secret=[string]&start=[numeric]&end=[numeric]
Method: POST
Returns: [{
    id: [string],
    value: [numeric],
    affiliation: [string],
    resource: [string],
    note: [string],
    timestamp: [numeric]
}, ...]

Persistent Storage Service

Endpoint: Create storage space
URL: DataStore
Headers: Authorization: Bearer {token}
Method: POST
Returns: {
    readSecret: [string],
    writeSecret: [string]
}

Endpoint: Get all storage spaces
URL: DataStore
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    readSecret: [string],
    writeSecret: [string],
    readUrl: [string],
    writeUrl: [string]
}, ...]

Endpoint: Store data
URL: DataStore/{writeSecret}
Appendix

Headers: Authorization: Bearer {token}
Method: POST
Body: {
   payload: [any],
   permissions: {
      remoteWrite: [boolean],
      remoteRead: [boolean]
   }
}

Endpoint: Store data
URL: DataStore/{readSecret}
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
   payload: [any],
   permissions: {
      remoteWrite: [boolean],
      remoteRead: [boolean]
   }
}, ...]

User Profile Service

Endpoint: Get user profile
URL: User/Profile
Headers: Authorization: Bearer {token}
Method: GET
Returns: {
   displayName: [string],
   email: [string],
   affiliation: [string],
   userId: [string]
}

Notification Service

Endpoint: Register notification channel
REST Endpoint Definitions for Tier 1

URL: Notifications/Channels
Headers: Authorization: Bearer {token}
Method: POST
Body: {
    name: [string],
    transports: [[string], ...],
    listener: [string]
}
Returns: {
    channelId: [string]
}

Endpoint: Remove notification registration
URL: Notifications/Channels/{channelId}
Headers: Authorization: Bearer {token}
Method: DELETE

Endpoint: Get all notification channels
URL: Notifications/Channels
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    channelId: [string],
    transports: [[string], ...],
    name: [string]
}, ...]

Endpoint: Send notification
URL: Notifications/{channelId}
Headers: Authorization: Bearer {token}
Method: POST
Body: {
    title: [string],
    message: [string],
    data: [any]
}
Appendix

Endpoint: Get notification
URL: Notifications/{notificationId}
Headers: Authorization: Bearer {token}
Method: GET
Returns: {
    title: [string],
    message: [string],
    data: [any]
}

Endpoint: Get all notifications
URL: Notifications/{channelId}
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    notificationId: [string],
    notificationUrl: [string],
    title: [string]
}, ...]

A.3 REST Endpoint Definitions for Tier 4

Lecture Scope

Endpoint: Find lectures
URL: Lectures
Query: ?query=[string]
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    lectureId: [string],
    eventUrl: [string],
    title: [string]
}, ...]
Endpoint: Get my lectures
URL: Lectures/My
Query: ?termOffset=[numeric]
Headers: Authorization: Bearer {token}
Method: GET
Returns: 

```json
[{
lectureId: [string],
lectureUrl: [string],
title: [string]
}, ...]
```

Endpoint: Get lecture
URL: Lectures/{lectureId}
Headers: Authorization: Bearer {token}
Method: GET
Returns: 

```json
{
eventId: [string],
name: [string],
appointmentsUrl: [string],
announcementsUrl: [string],
assignmentsUrl: [string],
materialsUrl: [string],
literatureUrl: [string]
}
```

Endpoint: Get lecture appointments
URL: Lectures/{lectureId}/Appointments
Headers: Authorization: Bearer {token}
Method: GET
Returns: 

```json
[{
appointmentId: [string],
lectureId: [string],
lectureUrl: [string],
title: [string],
start: [numeric],
end: [numeric]
}, ...]
```
Appendix

Endpoint: Get lecture announcements
URL: Lectures/{lectureId}/Announcements
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  title: [string],
  message: [string],
  attachmentUrl: [string]
}, ...]

Endpoint: Get lecture assignments
URL: Lectures/{lectureId}/Assignments
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  title: [string],
  description: [string],
  attachmentUrl: [string]
}, ...]

Endpoint: Get lecture materials
URL: Lectures/{lectureId}/Materials
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  title: [string],
  description: [string],
  downloadUrl: [string]
}, ...]

Endpoint: Get literature
URL: Lectures/{lectureId}/Literature
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  title: [string],
}
REST Endpoint Definitions for Tier 4

citation: [string],
attachmentUrl: [string]
}, ...

Quiz Scope

Endpoint: Get all quizzes for lecture
URL: Lectures/{lectureId}/Quiz
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  quizId: [string],
  downloadUrl: [string],
  title: [string]
}, ...]

Endpoint: Store quiz result
URL: Lectures/{lectureId}/Quiz/{quizId}/Answers
Headers: Authorization: Bearer {token}
Method: POST
Body: {
  answers: [[string],...]
}

Endpoint: Get quiz results
URL: Lectures/{lectureId}/Quiz/{quizId}/Answers
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  mine: [string],
  totalResponses: [numeric],
  correctResponses: [numeric]
}, ...]

Timetable Scope

Endpoint: Add appointment
Appendix

URL: Timetable/{appointmentId}
Headers: Authorization: Bearer {token}
Method: POST

Endpoint: Remove appointment
URL: Timetable/{appointmentId}
Headers: Authorization: Bearer {token}
Method: DELETE

Endpoint: Get timetable week
Query: ?weekOffset=[numeric]&count=[numeric]
URL: Timetable/{appointmentId}
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    appointmentId: [string],
    lectureId: [string],
    lectureUrl: [string],
    title: [string],
    start: [numeric],
    end: [numeric]
}, ...]

Library Scope

Endpoint: Get library account
URL: Library/Media
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    mediaId: [string],
    extendUrl: [string],
    title: [string],
    canExtendLoan: [boolean],
    returnDate: [numeric]
}, ...]
Endpoint: Extend loan
URL: Library/Media/{mediaId}/ExtendLoan
Headers: Authorization: Bearer {token}
Method: POST

**Audience Response System Scope**

Endpoint: Get audience response system channels
URL: AudienceResponse
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    channelId: [string],
    channelUrl: [string],
    title: [string],
    secret: [string]
}, ...]

Endpoint: Add audience response system channel
URL: AudienceResponse
Headers: Authorization: Bearer {token}
Method: POST
Body: {
    title: [string]
}
Returns: {
    channelId: [string],
    secret: [string]
}

Endpoint: Send audience response system message
URL: AudienceResponse/{channelId/Messages}
Query: ?secret=[string]&type=[string]&recipient=[string]
Headers: Authorization: Bearer {token}
Method: POST
Body: [binary]
Appendix

Endpoint: Get all audience response system messages in a channel
URL: \( \text{AudienceResponse/\{channelId\}/Messages} \)
Query: ?secret=[string]&skip=[numeric]
Headers: Authorization: Bearer \{token\}
Method: GET
Returns: [{ messageId: [string], messageUrl: [string], sender: [string], body: [string], timestamp: [numeric] }, ...]

Endpoint: Get audience response system messages
URL: \( \text{AudienceResponse/\{channelId\}/Messages/\{messageId\}} \)
Query: ?secret=[string]
Headers: Authorization: Bearer \{token\}
Method: GET
Returns: [binary]

Metadata Scope

Endpoint: Get metadata set
URL: \( \text{Metadata/\{pid\}} \)
Query: ?ota=[string]
Headers: Authorization: Bearer \{token\}
Method: GET
Returns: [jsonld]

Endpoint: Store metadata set
URL: \( \text{Metadata/\{pid\}} \)
Query: ?schemaId=[string]&ota=[string]
Headers: Authorization: Bearer \{token\}
Method: POST
Body: [jsonld]
Returns: [{ pid: [string], }]

178
Endpoint: Get All metadata sets of users
URL: Metadata/My
Headers: Authorization: Bearer {token}
Method: Get
Returns: [{
  pid: [string],
  metadataSetUrl: [string],
  title: [string]
}, ...]

Endpoint: Find in public and own metadata sets
URL: Metadata
Query ?query={string}
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  pid: [string],
  metadataSetUrl: [string],
  title: [string]
}, ...]

Endpoint: Get all metadata schemas
URL: Metadata/Schemas
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
  schemaId: [string],
  schemaUrl: [string],
  title: [string]
}, ...]

Endpoint: Get metadata schema
URL: Metadata/Schemas/{schemaId}
Appendix

Headers: Authorization: Bearer {token}
Method: GET
Returns: [jsonld]

**Archive Scope**

Endpoint: Store file
URL: Archive/{pid*}
Query ?fileName=[string]&ota=[string]
Headers: Authorization: Bearer {token}
Method: POST
Body [binary]
Returns: {
    pid: [string]
}

Endpoint: Get all files
URL: Archive
Headers: Authorization: Bearer {token}
Method: GET
Returns: [{
    pid: [string],
    fileName: [string]
}, ...]

Endpoint: Retrieve file
URL: Archive/{pid}/Restore
Headers: Authorization: Bearer {token}
Method: POST

**Collaboration Scope**

Endpoint: Store file
URL: Collaboration/Files/{filename}
Headers: Authorization: Bearer {token}
Method: POST
Body: [binary]

Endpoint: Store file properties
URL: Collaboration/Properties/{filename}
Headers: Authorization: Bearer {token}
Method: POST
Body: {
    [string]: [string],
    ...
}

Endpoint: Get file
URL: Collaboration/Files/{filename}
Headers: Authorization: Bearer {token}
Method: GET
Returns: [binary]

Endpoint: Get file properties
URL: Collaboration/Properties/{filename}
Headers: Authorization: Bearer {token}
Method: GET
Returns: {
    [string]: [string],
    ...
}
A.4 Screenshots of Case Studies

The discussed case studies were successfully implemented and are available to users at RWTH Aachen University. The screenshots presented here provide a visual impression of the implemented processes.

**RWTHApp**

![Figure A.1: Overview of a lecture in RWTHApp](image)

*Figure A.1: Overview of a lecture in RWTHApp*

**Quizz2Go**

![Figure A.2: Answering questions right (left) and wrong (right)](image)

*Figure A.2: Answering questions right (left) and wrong (right)
Screenshots of Case Studies

Figure A.3: Overview of all questions and answers (left) and aggregated results (right)

Audience Response System

Figure A.4: Teacher view during poll
Appendix

Figure A.5: Student view during poll

Eduroam Device Management

Figure A.6: Add new device to eduroam device management
Screenshots of Case Studies

Figure A.7: Manage existing device to eduroam device management

Figure A.8: View device connection history
Appendix

Chat Support

Figure A.9: Chat Support integration into documentation web page

Figure A.10: Chat Support integration into eduroam device management
Screenshots of Case Studies

Figure A.11: User view starting a support chat

Figure A.12: User view of support chat
Figure A.13: Support agent chat overview

Figure A.14: Support agent chat view
HIApp

Figure A.15: Contributor view of articles with coordinates

Figure A.16: Rich content editor for articles
Appendix

Figure A.17: HIApp start page (left) and map view (right)

Metadata Tool

Figure A.18: Metadata Tool start page
Screenshots of Case Studies

Figure A.19: Editing a metadata set

simpleArchive

Figure A.20: simpleArchive home page
Figure A.21: simpleArchive archive page

Figure A.22: simpleArchive restore page
A.5 Further Use Cases and Case Studies

Apart from the case studies presented in Chapter 4, the pSTAIX reference architecture was used in the design of other process supporting applications.

A.5.1 L²P Third Party System Integration

Even though L²P is already part of the reference architecture, it also implements a variation of previously presented case studies. Instead of supporting stand alone implementations of processes, the third party system integration of L²P uses tier 0 and tier 1 services to authorize applications that are embedded within the eLearning portal itself. The integrated services then use tier 4 services to interact with the portal. Figure A.23 shows the interactions of the participating systems. Figure A.24 shows an impression of the integrated Moodle subsystem in L²P.

Other existing portals can base on this model and integration strategy. This could allow them to integrate the standalone applications developed within the case studies without the need of implementing entirely new user interfaces.

![Figure A.23: Moodle subsystem integrated into L²P](image-url)

Figure A.23: Moodle subsystem integrated into L²P
Appendix

Figure A.24: Moodle subsystem integrated into L²P

A.5.2 FLEX eLearning Framework

FLEX is an eLearning framework for electronic examinations on students’ devices developed by Küppers et al. [168]. Using a client-server model, the assessment framework meets issues of electronic examinations by establishing a trusted platform on students’ devices and uses digital signatures and asymmetric encryption to identify the students as shown by Figure A.25. Even though the services defined by FLEX are not part of the distributed architecture, the server architecture resembles the tier model as shown by Figure A.26.

A.5.3 Learning Room Finder

The learning room finder is another extension of RWTHApp. It incorporates usage statistics of WiFi access points as a back end system into the architecture. By correlating the location of access points and available learning rooms, a tier 3 service aggregates the data and uses an adaptive prediction model presented by Selzer et al. [169]. A tier 4 service consolidates the data with additional information about the facilities. The application then presents an interactive map showing nearby rooms and a predicted filling level as shown in Figure A.27.
Further Use Cases and Case Studies

Figure A.25: Exemplary assessment process supported by FLEX [168]

Figure A.26: FLEX server architecture [168]
Figure A.27: Learning room finder map (left) and details view (right)
Curriculum Vitae

Marius Politze
eMail: m@politze.net
ORCID: 0000-0003-3175-0659

About Me

I am a data scientist with a strong programming background. Since 2008 I am working with recent web and mobile applications using a variety of technologies, platforms and programming languages.

My research is focused on analysis and modelling of educational and scientific processes, secure service oriented infrastructures as well as web and mobile applications for eLearning, eAdministration and eScience.

Academic Career

RWTH Aachen University, Germany
Doctorate in Computer Science
Thesis: A Reference Architecture and Implementation Enabling Data Protection in Distributed eLearning and eScience Processes
Final Grade: magna cum laude

Universiteit Maastricht, The Netherlands
Master of Science in Artificial Intelligence, cum laude
Internship: Enhancement of a web based student self service application
Thesis: Automatic Ontology Mapping in a Pandisciplinary Repository for Research Projects
Final Grade: zeer goed (8.55)

Aug. 2008 - Feb. 2011
FH Aachen University of Applied Sciences, Germany
Curriculum Vitae

*Bachelor of Science in Scientific Programming*
Thesis: Ein Komponenten-Framework zur Bildbasierten Steuerung von Robotern
Final Grade: *sehr gut (1.5)*

Sep. 2006 - Aug. 2008  
RWTH Aachen University, Germany  
*Computational Engineering Science*  
without certificate

Gymnasium Norf, Neuss, Germany  
*Abitur*  
Average grade: *gut (2.2)*

**Professional Career**

Aug. 2012 - present  
IT Center, RWTH Aachen University  
*Research Associate*  
Team Lead eScience services  
Development of web and mobile applications, data security and consolidation for service oriented architectures supporting eLearning and eScience processes

Center for Computing and Communication of RWTH Aachen University  
*Mathematical and Technical Software Developer*  
Development and Product Owner of web portals for administration and student self service  
Assistant Professor for C#, C/C++ and scripting languages

Sep. 2008 - Feb. 2011  
Center for Computing and Communication of RWTH Aachen University  
*Mathematical and Technical Software Developer, Trainee (IHK)*
Development of web portals for administration and student self service
Image processing for robotics with the NAO Robot
Final Grade: *gut (82/100)*

P3 Solutions GmbH
*Student Assistant*
Software and usability tests for mobile applications
Quality assurance

Publications & Talks

The publications and talks presented here highlight the evolution and preliminary results of the thesis but also other research results during my scientific career.

2019


2018


199
Curriculum Vitae


2017


**2016**


Curriculum Vitae


2015


2014


Talks

M. Politze and A. Schwarz, “Metadatentools in FDM Prozessen: Bausteine und Schnittstellen für die dezentrale und hochschulübergreifende FDM Infrastruktur”, in *Kick-Off Workshop für die Entwicklung von hochschulübergreifenden FDM-Basisdienstleistungen in NRW*, 2019

M. Politze, “simpleArchive as a Service”, in *Kolloquium der Betreiber von Forschungsdateninfrastrukturen in NRW*, Aachen, Germany, 2017

M. Politze and T. Eifert, “Rahmenbedingungen und Voraussetzungen für den Einsatz elektronischer Laborbücher”, in *Workshop Elektronische Laborbücher*, Aachen, Germany, 2017


F. Krämer, M. Politze, and D. Schmitz, *Empowering the Usage of Persistent Identifiers (PID) in Local Research Processes by Providing a Service and Integration Infrastructure*, Garching, Germany, 2016


M. Politze and B. Decker, “Shibboleth & OAuth2 für die Autorisierung von Apps: Als Infrastruktur für die RWTHApp”, in *60. DFN Betriebstagung*, Berlin, Germany, 2014
Statement of Originality

This work would not have been possible without the close collaboration with colleagues of the IT Center and especially the department IT process support for research and teaching (IT-PFL). The research carried out in the context of this thesis was part of several projects in the context of the IT Center and was subsequently established to support processes of the university.

Within this collaboration several preliminary results have been published that highlight the evolution of this thesis. For these foregoing works, this section highlights the authors’ contributions in two areas related to this work: The conceptual foundation of the reference architecture and its implementation as described in Chapters 2 and 3 and the application of the architecture to support several case studies as shown in Chapter 4.

Conceptual Foundation of the Architecture

- “RWTHApp: from a requirements analysis to a service oriented architecture for secure mobile access to personalized data” (M. Politze et al., 2014 [145]), “Internship Report” (S. Schaffert, 2015 [135]) and “A secure infrastructure for mobile blended learning applications” (M. Politze et al., 2016 [117]): These works describe the previous analysis of eLearning services and the resulting architecture. They focus on RWTHApp and other mobile applications as target platforms for eLearning processes. Changes in Student behavior from pure consumers to application creators motivated the creation of an infrastructure for secure access to student data. My key contributions were the consultancy on the choice of technologies as well as the general software architecture to support the use cases. In the later work I further formalized the architecture and proposed explicit modeling of the authorization service within the architecture. Most of the architectural ideas were then implemented during my time
as a technical project lead for RWTHApp. Within this context I co-supervised the internship project of S. Schaffert who evaluated different caching mechanisms within the reference architecture and that eventually lead to the implementation of the mechanisms. During the design and implementation I greatly profited from the experience of Decker in supporting eLearning processes.

- “From eLearning to eScience: Building a Service Oriented Architecture to Support Research” (M. Politze et al., 2017 [101]), “pSTAIX - A Process-Aware Architecture to Support Research Processes” (M. Politze et al., 2017 [108]) and “Forschungsdaten managen – Bausteine für eine dezentrale, forschungsnahe Unterstützung” (D. Schmitz et al., 2018 [106]): To close this gap between the newly set up process-aware eLearning infrastructures and eScience, these works show that the architecture is generally suitable for also supporting eScience processes. An insight that came from discussions with Eifert and Schmitz is that while there are several (inter-)national services in development, the architecture needs to support researchers daily work already now. In consequence, I did not only investigate how technical bounding conditions influence process-aware services, but also included organizational, operational and application lifecycle issues. Schmitz showed how decentralization is a basic requirement for supporting eScience processes at universities which allowed me to derive a more formalized architecture that categorizes the services in different tiers based on their role within the supported process.

- Co-Supervision of “Entwurf und Implementierung eines Monitoring-Systems für das OAuth-basierte Autorisierungssystem des IT-Centers der RWTH Aachen” (M. Ernst, 2015 [170]) and “Extending the OAuth2 Workflow to Audit Data Usage for Users and Service Providers in a Cooperative Scenario” (M. Politze et al., 2017 [118]): Scaling across organizational boundaries is one of the core properties to achieve decentralization within a process supporting infrastructure. My contributions to these works show how the explicitly modeled authorization service allows this kind of scaling to cross organizational boundaries within a cooperative scenario. From the discussions with Decker about data privacy followed that the dedicated authorization
service also allowed modeling user-centric auditing capabilities that I was able to include within the existing authorization flows. I also co-supervised the bachelor thesis of M. Ernst who implemented a proof of concept for gathering monitoring data within the authorization service. I later extended this prototype to provide user-centric auditing capabilities for the cooperative scenario as a service.

- “Extending OAuth2 to Join Local Services into a Federative SOA” (M. Politze, 2017 [119]): In order to extend and decentralize the architecture even further, the publication proposes authorization flows for federative scenarios. Therefore, I compared current federation services for single sign on and their capabilities with the requirements of the proposed process-aware architecture. With these insights I could then propose workflows that allow the authorization service to support federative scenarios without changing existing processes and applications. As a result of the study, I implemented the workflows within the process-supporting infrastructure.

Case Studies

- “Facilitating Teacher–Student Communication and Interaction in Large-Scale Lectures With Smartphones and RWTHApp” (M. Politze et al., 2015 [148]) Co-Supervision of “Von einer relationalen zu einer dokumentenorientierten Datenbank: Migration und Erweiterung eines Webservice Backend” (S. Drenckberg, 2017 [136]) and “Migration of a web service back-end from a relational to a document-oriented database” (S. Drenckberg et al., 2018 [149]): These works present the Audience Response System as a feature of RWTHApp. We have shown different usage workflows and how they are handled by teachers in their classes. My main contributions to this case study were the requirements engineering and consultancy on the software architecture during my time as technical project lead for RWTHApp. Later, I co-supervised S. Drenckberg during his bachelor thesis. He showed a practical example how a back end system can be exchanged without effecting processes and applications within the architecture. During this migration I especially observed how different application and back end lifecycles effect each other within the architecture, and we were able to publish the joint results.
• “Device specific credentials to protect from identity theft in Eduroam” (B. Decker et al., 2017 [163]): Based on a project idea of mine several years before, we presented a process in this case study that allows users to create device specific Eduroam credentials. While at that time the reference architecture was not yet formalized, the case study greatly profited from its advances. Based on the user-centric concepts already shown as part of the authorization service, this process also emphasizes providing audit information for user. Renner evaluated the process from the users’ point-of-view and designed comprehensible user interfaces to operate the process. My main contributions were the design and implementation of the process and inclusion of back end systems according to the reference architecture. This cast study was also the first process that made use of the federative scenario of the authorization services. While B. Decker evaluated the organizational bounding conditions for scaling the process across universities, this allowed me to further validate federative authorizations with a real world application.

• Co-Supervision of “Framework zum Datenaustausch zwischen Mobilien Applikationen und Microsoft SharePoint” (S. Consoir, 2016 [171]) and “A general architecture for content driven mobile applications: building an interactive tour guide for historical sites” (M. Politze et al., 2016 [150]): In these works we introduced HIApp, a content driven mobile application. During his bachelor thesis S. Consoir implemented the necessary processes and connections to back end systems. Among co-supervising his work, my main contributions to the case study were the requirements engineering process and providing architectural guidance when putting the services into place. During the requirements engineering phase this case study made especially clear to me that eScience processes can be practically supported with technologies known from the eLearning context, such that we evaluated how this work integrates into the existing architecture. As part of this case study, I was also able to show how the proprietary interfaces of the back end can be translated into standardized interfaces that in turn can be used to integrate services without expert knowledge of the back end system.
• “Towards a distributed research data management system” (M. Politze et al., 2016 [153]), “Ontology Based Semantic Data Management for Pandisciplinary Research Projects” (M. Politze et al., 2016 [152]) and Co-Supervision of “Metadatenmanagement mithilfe eines Triple-Stores als Komponente des Forschungsdatenmanagements an der RWTH Aachen” (S. Bensberg, 2017 [155]): During the development process of the metadata tool we built several intermediate prototypes that were evaluated independently and led to step wise refinement of used semantic data models platforms. While initially intended as an add-on to the collaboration features of SharePoint, the tool advanced to an independent application supporting data management processes by assigning persistent identifiers and heterogeneous, discipline specific metadata to data sets stored in distributed systems. During her bachelor Thesis S. Bensberg did the implementation work of the current version of the case study corresponding to the reference architecture. As a co-supervisor I also provided architectural advises needed to comply with the reference architecture. Starting with the first works for this case studies I further contributed to the project by creating the semantic data models for metadata, metadata schemas and the overall processes supported by the tool. While in the early stages of the case study B. Decker did the requirements engineering, after moving away from SharePoint I took over this task from him. The project is now continued and supervised by me in my role as technical project lead for eScience services.

• “simpleArchive – Making an Archive Accessible to the User” (M. Politze et al., 2017 [159]) and “simpleArchive as a Service” (M. Politze, 2017 [160]): The case study simpleArchive aimed to provide an easy-to-use archival solution for researchers at RWTH Aachen University. In the works we described a process that re-uses existing services and combines them to create additional value for the user. Based on the designed process, I then integrated the required back end systems into the implementation of the reference architecture. Since these back end technologies are common among universities, in this case my focus was on scalability across university boundaries. Only short after the release at RWTH Aachen University we were therefore able to scale the process to TU Darmstadt within a federative szenario.
References


References


References


References


References


References


References


References


References


References


References
