Design and Implementation of a Test Tool for PSD2 Compliant Interfaces

Gloria Bondel[1], Josef Kamysek[1] and Markus Kraft[2] and Florian Matthes[1]
[1]*Technical University of Munich, Faculty of Informatics, Garching, Germany*
[2]*msgGillardon AG, Munich, Germany*

Keywords: PSD2, XS2A, Test Tool.

Abstract: The Revised Payment Services Directive (PSD2) forces retail banks to make customer accounts accessible to TPPs via standardized and secure ”Access to Account” (XS2A) interfaces. Furthermore, banks have to ensure that these interfaces continuously meet functional and performance requirements, hence testing is very important. A known challenge in software testing is the design of test cases. While standardized specifications and derived test cases exist, the actual implementations of XS2A interfaces often deviate, leading to the need to adapt existing or create new test cases. We apply a design science approach, including five expert interviews, to iteratively generate a concept of a test tool that enables testing of several XS2A interface implementations with the same set of test cases. The concept makes use of files mapping deviations between the standardized specification and the implemented interfaces. We demonstrate the concept's feasibility by implementing a prototype and testing its functionality in a sandbox setting.

1 INTRODUCTION

The European Union published the Revised Payment Services Directive (Directive (EU) 2015/2366, abbreviated PSD2) to address risks resulting from new online and mobile services that emerged in retail banking. One measure prescribed by the PSD2 is the realization of standardized and secure ”Access to Account” (XS2A) interfaces enabling TPPs to access customer accounts, given the customers' provided their consent (Scheja and Machielse, 2019; BG, 2018). However, while the PSD2 and the additionally published regulatory technical standards (RTS) require banks to implement an XS2A interface, they do not specify a technical solution (Scheja and Machielse, 2019).

The prescription for implementing secure XS2A interfaces does not only lead to additional effort for the bank implementing the interface but also for TPPs who consume it. The TPPs need sufficient time to adapt their systems and continuously well performing interfaces to prevent disruption of their business. Hence, the regulations define strict testing requirements for XS2A interfaces. Even now, after the transition phase (14.09.2019) has passed, the banks have to ensure that the interface continuously meets functional and performance requirements.

A major challenge in software testing is test case generation, which is perceived as being a tedious and erroneous process (Arcuri, 2019). In the context of the PSD2 regulation, industry standardization institutions design XS2A interface specifications and derive test cases from these specifications. Thus, it would be intuitive that banks can leverage the existing test cases if they adapt the standard. However, banks usually implement XS2A interface standards with deviations. These deviations lead to the need to create new test cases or adapt the existing test cases for each bank. To address this issue, we aim to answer the following research question: *”How can we design and implement a test tool for XS2A interfaces exploiting existing sets of test cases?”*

We apply a design science research approach (Hevner et al., 2004) to answer this research question and conducted five expert interviews to iteratively design the artifact. As a result, we present a test tool concept that enables testing interfaces that deviate form standards with one set of test cases. We show our concept's feasibility by implementing a prototype and testing its functionality in a sandbox setting. The test tool enables banks to save time on the design of test cases and provides automatic testing facilities to TPPs. Furthermore, we contribute to the scientific and practical community by presenting a concept for efficiently testing RESTful Web APIs based on a standard that could also be transferred to other fields with emerging interface standards.

Bondel, G., Kamysek, J., Kraft, M. and Matthes, F.
Design and Implementation of a Test Tool for PSD2 Compliant Interfaces.
DOI: 10.5220/0010439502490256
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 2*, pages 249-256
ISBN: 978-989-758-509-8

2 FOUNDATIONS

We will introduce the background of the PSD1 and PSD2, the importance of testing in the context of PSD2, and the challenges of testing Web APIs.

2.1 Regulation of the European Payment Market

The European Union introduced the first Payment Service Directive (Directive 2007/64/EC, abbreviated PSD1) in 2007 with the goal to increase competition between banks and payment service providers, leading to a broader choice of convenient, efficient, and secure cross-border payment services for end-users within Europe. However, after the PSD1 became effective, Third-Party Payment Service Providers (TPPs) emerged and started offering new online and mobile payment services (Cortet et al., 2016; Bramberger, 2019). More specifically, the regulators identified three services which are (1) the creation of an integrated view of customer accounts even if different banks manage these accounts, (2) the initiation of a payment from an existing account, and (3) the confirmation that a certain amount of funds are available (EU, 2015). These new services have in common that the TPPs do not operate customer accounts themselves. Instead, they access the customer's account operated by a bank to enable their new services. Due to missing regulations and standards, only fragmented interfaces to securely access these accounts existed (Scheja and Machielse, 2019). Hence, TPPs often resorted to screen-scraping online banking websites using the end-users online banking login credentials (Scheja and Machielse, 2019; Cortet et al., 2016).

Addressing the resulting security risks in the evolving online payment services industry, in 2015, the European Union published the Revised Payment Services Directive (Directive (EU) 2015/2366, abbreviated PSD2) replacing the PSD1. The overall goal of the PSD2 is to foster innovative online and mobile payment services while at the same time ensuring customer protection (Bramberger, 2019) (Zachariadis and Ozcan, 2017). The PSD2 defines several measures of which we focus on the PSD2's prescription to allow TPPs standardized and secure access to customer payment accounts (BG, 2018).

The PSD2 requires that banks and TPPs implement secure communication standards to enable TPPs to access customer accounts securely (EU, 2015). However, the PSD2 does not detail such standards or any specific technical solutions (Scheja and Machielse, 2019). Instead, the PSD2 commissioned the European Banking Authority (EBA) to issue regulatory technical standards (RTS) (Art. 98 PSD2). The RTS, published in the final version in March 2018, mandates that banks that provide online banking have to provide an "Access to Account" (XS2A) interface. An XS2A interface enables TPPs to request information on payment accounts and to initiate payment orders securely (Art. 30 RTS). Although more specific than the PSD2, the RTS still only provide a high-level definition of the interface functionality and do not detail any technical specification (Scheja and Machielse, 2019). Nevertheless, all regulated entities have to comply with the RTS provisions and provide an XS2A interface 18 months after the RTS was published, i.e., latest on the 14.09.2019.

The development of detailed technical standards is thus left to the banking industry (Scheja and Machielse, 2019). Several standardization bodies formed and have since provided technical specifications of the XS2A interface, e.g., the Berlin Group[1] or the Open Banking Initiative[2]. Among these, the Berlin Group is one of the major standardization bodies since it created an XS2A interface specification in cooperation with 52 European banking entities, including banks, banking associations, payment associations, payment schemes, and interbank processors active in the EU (Scheja and Machielse, 2019). The XS2A interface specification of the Berlin Group is named the Berlin Group Next-GenPSD2 Framework which defines the XS2A interface as a RESTful Web API (BG, 2018).

2.2 Testing of XS2A Interfaces

Testing plays an essential role during the transition to XS2A interfaces as well as afterwards. The RTS prescribe functional and performance testing. Functional requirements are mentioned in the RTS but not specified in detail (Art. 30 RTS). Banks are instead encouraged to follow standards to realize these functional requirements (EBA, 2018b). Regarding performance testing, banks need to prove that the XS2A interfaces perform as good as the old interfaces, i.e., online banking websites. Performance testing is necessary since latency in the TPP's product due to inferior interface performance could lead to dissatisfaction and loss of TPPs' and banks' customers.

During the transition to the standardized XS2A interfaces, testing played an essential role. Timely access to XS2A testing and production environments allowed TPPs to explore the new interfaces and to adapt their systems before access to old interfaces was revoked. Also, the RTS introduced a mandatory contin-

[1]https://www.berlin-group.org/

[2]https://www.openbanking.org.uk/

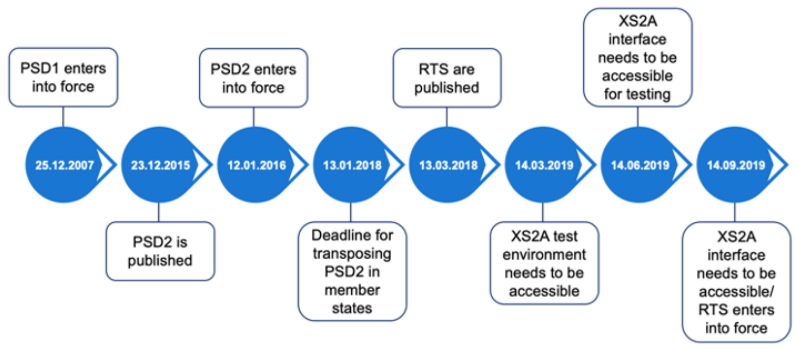
PSD1 enters into force
25.12.2007
PSD2 is published
23.12.2015
PSD2 enters into force
12.01.2016
Deadline for transposing PSD2 in member states
13.01.2018
RTS are published
13.03.2018
XS2A test environment needs to be accessible
14.03.2019
XS2A interface needs to be accessible for testing
14.06.2019
XS2A interface needs to be accessible/ RTS enters into force
14.09.2019

Figure 1: Timeline of the PSD2 regulation.

gency measure, often referred to as fall back rule (Art. 33 RTS) (EBA, 2018b; EBA, 2018a) to enforce the compliance with functional and performance requirements. The fall back rule prescribes that banks have to enable TPPs to access customer payment accounts not just through the XS2A interface, but also through the customer-facing interface, i.e., the online banking website. However, the banks have to adjust the customer-facing interfaces to comply with the same security requirements as the XS2A interface, leading to additional implementation effort for banks. An exemption of the fall back rule is possible, if a bank meets strict testing and reporting requirements (Art. 33 RTS). These requirements entail that banks have to provide a test sandbox for connection and functionality testing with mock data to certified TPPs as well as appropriate documentation of the XS2A interface half a year before the API goes into production (14.03.2019) (Art. 30(5), 30(3) RTS). The productive API has to be accessible for testing three months before the RTS becomes effective (14.06.2019). During that time, the bank has to promote the testing of the XS2A interface by TPPs and address raised issues (EBA, 2018b). After the testing period, the banks have to report the number of TPPs that tested the API, the issues raised, and measures to address them and certain performance KPIs (EBA, 2018b). An overview of the timeline is provided in Fig. 1.

After the transition period the essential role of testing is not diminished. A bank has to continuously ensure that the interface still meets the functional and performance requirements, especially if the interface implementation evolves. A failure to meet the requirements, even after the end of the transition phase, leads to the need to implement the fall back solution.

2.3 Challenges of Testing

Software development is a complex and error-prone process that can lead to defects or unexpected behavior in software systems or components. Thus, testing activities are an essential part of software development. Testing aims at evaluating the properties of software and ensuring they meet the user's needs (ISO, 2013). Therefore, testing can prevent negative impacts of releasing erroneous software, e.g., security threats, losses, or a bad reputation (ISO, 2013).

As part of the test design and implementation, a tester creates a test case for a test item. The test item is the *"work product that is an object of testing"* (ISO, 2013), e.g., the software component to be tested. The test case is the *"set of test case preconditions, inputs [...] and expected results, developed to drive the execution of a test item to meet test objectives"* (ISO, 2013). The test execution can be performed manually or automated with the help of a test tool.

In this research paper, the test item is a XS2A interface of a bank. According to the Berlin Group specification, the XS2A interface is a RESTful Web API. A Web API makes functionality or data available via endpoints, that can be accessed over a network using the HTTP protocol (Bermbach and Wittern, 2016). For each endpoint, the API designers define operations and parameters of a successful API call. REST is an architectural pattern that prescribes certain constraints to Web API design (Fielding, 2000). In this context, a test case is one HTTP request to a RESTful Web API.

A major challenge of testing is test case generation. Test case generation is tedious and error-prone (Arcuri, 2019). Several approaches to enable automated test case generation for RESTful Web APIs have been previously presented, e.g., search-based testing (Arcuri, 2019) or test case generation based on a formal model (Fertig and Braun, 2015).

The approach presented in this research paper circumvents the challenge of test case generation by enabling banks to test an XS2A interface by exploiting preexisting test cases provided by standardization bodies. Thus, banks do not have to create and maintain their test cases. To the best of the authors knowledge, no scientific paper addressing the testing of XS2A interfaces have been previously published.

3 RESEARCH APPROACH

We apply a design science research approach since it ensures relevance and rigor while focusing *"on creating and evaluating innovative IT artifacts that enable organizations to address important information-related tasks"* (Hevner et al., 2004). We describe the research approach following the seven guidelines for design science as presented by (Hevner et al., 2004).

Design as an Artifact. The goal of a design science research approach is to create a purposeful artifacts that address an organizational problem (Hevner et al., 2004). The artifact presented in this research paper is a concept of a prototypical implementation of a test tool that enables the testing of several standardized XS2A interfaces with a single set of test cases, even if the XS2A implementations deviate from the standard. The organizational issue that the prototype addresses is the regulatory need to test XS2A interfaces properly. The need to test XS2A interfaces is not limited to the PSD2 transition phase in 2019, but regression testing is also valuable afterwards since the interface implementation is continuously evolving. Furthermore, the concept could be applied to other fields with emerging interface standards, e.g., in the health sector with the Fast Healthcare Interoperability Resources (FHIR) standard (Kasthurirathne et al., 2015).

Problem Relevance. In design science, the developed artifact has to solve a significant business problem. Our attention has been brought to testing of XS2A interfaces through discussions with industry experts specializing in XS2A implementation projects. The implementation and testing of XS2A interfaces is mandatory for retail banks but the ambitious timeline for implementing and testing the interfaces is a challenge. Also, when evolving the interface, regression testing is necessary. Our artifact enables banks to save time on the design of test cases and to provide automatic testing facilities to TPPs. The relevance of the problem is further highlighted by the emergence of several start-ups and consulting companies that specialize in providing testing solutions for XS2A interfaces, e.g., adorsys[3], ForgeRock[4], and Tieto[5].

Design Evaluation. We use two evaluation methods to show the *"utility, quality, and efficacy"* (Hevner et al., 2004) of our design artifact for addressing the problem of testing XS2A interfaces. First, we conducted five interviews with experts for XS2A interfaces and API testing to iteratively evaluate and advance the artifacts. The experts have different roles including a product owner, a head of department, a test manager, a software architect, and a start-up founder and are employed at different types of organizations including a retail bank, banking service providers, and IT service providers. Secondly, we show the feasibility of our concept by implementing a prototype and testing its functionality in a sandbox setting.

Research Contributions. The research contribution is the artifact itself, i.e., a concept and a proof of construction of a test tool for testing XS2A interfaces using a predefined set of test cases. The contribution is novel since the current practice is that banks design their own test cases and build their own test tools, which is error-prone and costly. Thus, with the test tool, testing XS2A interfaces can be realized more efficiently. Furthermore, the concept can be transferred to other areas with standardized interfaces.

Research Rigor. Research rigor means that *"The artifact itself must be rigorously defined, formally represented, coherent, and internally consistent."* (Hevner et al., 2004). Thus, we describe the artifact as well as any assumptions and limitations in sections 4 and 5.

Design as a Search. The design science process is *"essentially a search process to discover an effective solution to a problem"* (Hevner et al., 2004). We designed the test tool concept and prototype between December 2019 and April 2020. During that time, we conducted expert interviews to evaluate and advance the artifacts. Furthermore, we regularly presented our intermediate results to our main industry partner and discussed improvement potentials.

Communication of Research. The results of design science need to be communicated to a technical and managerial audience (Hevner et al., 2004), hence we motivate the research endeavor understandably and provide information to reconstruct the artifact in appropriate settings.

[3]https://adorsys.com/de/produkte/xs2a-compatibility-test-kit/

[4]https://www.forgerock.com/open-banking-sandbox

[5]https://openbanking.api.tieto.com/

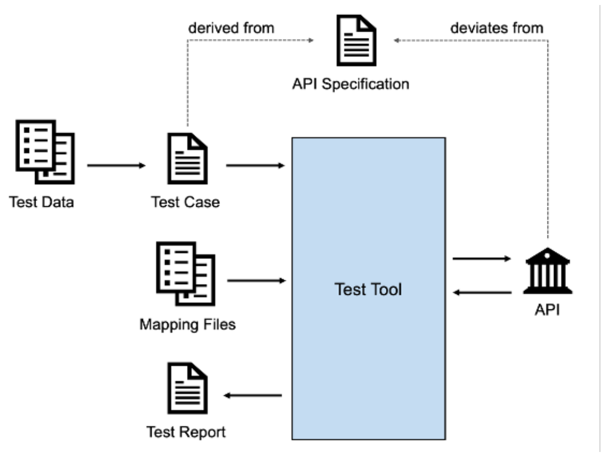

Figure 2: Test tool concept.

4 CONCEPT OF A XS2A INTERFACE TEST TOOL

In this section we present the test tool concept and the roles involved in operating the test tool.

4.1 Test Tool Concept

We assume that standardized interface specifications and associated test cases exist. One or several banks have implemented an XS2A interface guided by the standardized interface specification, but the actual implementations diverge from the specification. The goal is to nevertheless test the API implementations with the predefined set of test cases. Thus, two challenges arise that have to be addressed.

First, the structure of valid API calls can differ between the implemented API and the test cases. Our concept addresses this issue with mapping files, that map the structure of each implemented API endpoint with the structure defined in the standardized specification. A mapping file describes how a standardized test case is modified by the test tool to generate a valid API request to a specific interface implementation. The mapping files are an input to the test tool and can easily be adapted. Hence, the test tool itself does not have to be modified if a bank changes its API implementation or a new API is tested.

The second challenge is that the banks' systems store different data. If a tester executes a predefined test case without considering the deviation in (test) data, a test case will fail even if the bank implemented the API functionality correctly. We address this issue by designing the test cases in a data-driven way, i.e., the data is separated from the test case structure and can easily be adapted for each API implementation.

In summary, we present a test tool concept for testing APIs which use the same standardized specification as a starting point but deviate with regards to the implemented endpoint structure and the test data (see Fig. 2). A standardization body prescribes the API specification and the derived test cases. The test cases with the corresponding test data and the mapping files are inputs to the test tool. The test tool uses the inputs to generate an API call taking into account the specifics of the actually implemented API. The test tool then makes the API call to the implemented API and receives a response. The tool evaluates if the response matches the expected response and creates a test report. The concept addresses the differences in the endpoint structure with a mapping file and the differences in the test data with a data-driven testing approach. Thus, the test tool itself does not have to be adapted if an API implementation and the test data changes or if a new API is tested.

4.2 Involved Roles

In this subsection, we present five relevant roles for operating the test tool. The first role is the **standards provider**, who creates a standard, usually in cooperation with several interested parties, e.g., the Berlin Group. The second role is the **test case provider**, who derives a set of test cases from an XS2A standard. This role can be taken by an industry standardization body with expertise in testing, e.g., the NISP.

The **test tool provider** maintains the test tool. The test tool provider's tasks are to adapt the test tool in case standards change and to add new standards if they emerge. This role should be taken by an independent party.

The **mapping file provider** needs to know the XS2A interface standard and the implemented XS2A interface. This role could be taken by an external party as long as the bank provides good API documentation. However, since a banks XS2A development team knows the actual implementation best and has studied the standardized specification, this team is particularly suitable to create the mapping file.

Finally, the **test tool user** utilizes the tool to test an XS2A interface. The test tool user needs to have the right test data for a specific XS2A interface. Thus, the test tool user could be a bank that wants to test their own interfaces. Also, a TPP with an eIDAS certificate and access to test data could use the test tool to verify an XS2A interface functionality.

5 PROTOTYPICAL IMPLEMENTATION

In this section, we present a prototypical implementation of the concept of a test tool that enables testing of different XS2A interfaces. Furthermore, we report on the challenges that we encountered.

5.1 Test Tool Implementation

Assumptions. First, we want to disclose the underlying assumptions of the prototype. We choose the Berlin Group Next-GenPSD2 Framework as the XS2A interface standard since it is a dominant standard used by many banks in Europe. Also, it is freely accessible. The Berlin Group provides a detailed XS2A interface specification, including the paths of all endpoints and the methods, parameters, and possible responses for each endpoint. The Berlin Group standard defines JSON or XML as the data exchange format in the API request and response bodies. In our prototype, we choose the data exchange format JSON, since it is the most common media type used in Web APIs (Arcuri, 2019). We use the well-recognized library REST-assured[6] to execute test cases.

Inputs of the Test Tool. A test tool user has to provide three types of inputs for the test tool to work. These inputs are the test cases, the mapping files, and a certificate. We will describe each of these inputs.

Test Cases. The test tool user has to provide the test cases derived from the standardized XS2A interface specification. The NISP[7] derived test specifications from the Berlin Group XS2A specification[8], however, these test specifications are not publicly accessible. Also, no information regarding the format of these test cases is available. Hence, we define test specifications using a JSON schema.

The test case contains test metadata, all data necessary for generating the API request to be tested, and the expected API response. The test metadata includes a unique test ID, a test name, and a description. Each test case holds all information for one API call, but the tester can indicate that several test cases belong together in the metadata. The test tool then executes these test cases sequentially. Being able to execute test cases sequentially can be necessary for XS2A interfaces since an interaction often consists of first verifying if a client authorized an action and then performing said action, e.g., triggering a payment.

Additionally, the test case holds all information to create a valid API call according to the Berlin Group specification, including the endpoint, method and parameters structure and names. Also, the test case contains the expected response to the API request, i.e., a specific status code. This information enables the test tool to evaluate if the test was successful or not.

Mapping Files. The test service user has to provide two mapping files for each test case, a header and a body payload mapping file. The mapping files have two purposes. First, they provide specifics for accessing a specific bank's interface, e.g., the host and port. Secondly, they account for the differences between the standardized API specification and the actual API implementation. The mapping files are JSON files that use a specific notation to map variations in the naming or structure of the implemented API and the standardized API specification. Thus, the mapping files are basically JSON API adapters.

At this point, we will shortly describe the notation used to map two JSON structures. JSON defines two data structures, which are objects and arrays. In our notation, we characterize an object with its name and a subsequent dot (.). The string after the dot is the name of the object, array or key of a key/value pair located inside the object. An exception is the highest-level object, the JSON object, that is indicated by a dot preceding the first element inside the object. An array is described with a subsequent colon (:) followed by a number indicating the position of the relevant value inside the array. The number is followed by a dot and the name of the object, array, or key of a key/value pair located at that position in the array. To make the notation clear, we present the following example.

```
".customer.accounts:0.balance"
```

In this example, the object ”customer” is located in the JSON object. The object ”customer” contains

[6]http://rest-assured.io/

[7]https://nisp.online/

[8]https://berlingroup.stackstorage.com/s/1FBrOlC7Iquz G35B

an array ”accounts”, and a key/value pair with the key “balance” is located at position 0 within the array.

We also provide an example of the mapping between two JSON schemas. The schema reflecting the standardize API specification is placed on the left of the colon with whitespaces around it. The schema reflecting the actual implementation of the API is on the right side of the colon.

```
".customer.acc:0.balance" :
".customer.acclist:2.balance"
```

The mapping in this example accounts for differences in the naming within the JSON schemas, i.e., the array is named ”acclist” instead of ”acc”. In the prototype, this would reflect a difference in the naming of endpoints or parameters. Also, the key/value pair with the key “balance” is stored in the third instead of the first position in the array, which accounts for a change in the structure of the JSON schemas.

```
".customer.acc:0.balance" :
".creditcard.customer.acc"
```

Also, the notation can handle more severe differences in the structures of the JSON schema as shown in this last example. Instead of a nested structure with an object, an array, and a key, the implemented API is reflected in a JSON schema nesting two objects and a key.

Certificate. Finally, the test tool user needs to provide her ”Electronic Identification, Authentication and Trust Services (eIDAS)” certificate in every API request. The eIDAS certificate verifies a tester’s identity and ensures that the respective national banking regulation institution approved the tester. A request to an XS2A interface without an eIDAS certificate will invariably fail. In Germany, an organization or tester has to undergo a complicated registration process with the Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) to receive the eIDAS certificate (EBA, 2018c).

Test Tool Execution. Finally, we describe the process of the test tool execution. First, the test tool checks if the test service user provided all necessary inputs, i.e., the test case, the mapping files, and a certificate. Secondly, the test tool transforms the generic test case specification into a test case that accounts for a bank’s API’s specifics using the mapping files. Thirdly, the tool generates an API request from the transformed test case and routes it to the respective banks’ API. Once the test tool receives the response, the response is modified back from the specific bank API response to the generic Berlin Group specification. Finally, the test tool compares the API response message to the expected response in the test case derived from the Berlin Group specification and determines if the test was successful.

5.2 Limitations of the Test Tool

Overall, we implemented and tested the prototype successfully, thus providing a proof of construction. However, we also encountered some challenges which lead to limitations. First of all, the test tool can currently only test XS2A interfaces based on the Berlin Group standard. Furthermore, the test tool is limited to testing the eight API endpoints necessary for the account information service (AIS) use case. The AIS use case entails the creation of account holder consent and retrieval of information on the balance and transactions of the account, e.g., to create a service providing an integrated view of the account holder’s accounts operated by different banks.

Furthermore, since we did not have access to the NISP test cases, we derived test cases for the Berlin Group specification ourselves.

Another challenge is that we could not access XS2A test sandboxes or productive interfaces of banks. The access to these sandboxes and productive interfaces is limited to TPPs with a valid eIDAS certificate, i.e., TPPs approved by the respective national banking regulation institution. Since we could not obtain such a certificate, we resort to testing the test tool using a publicly available sandbox of a financial services provider[9]. The sandbox is based on the Berlin Group standard but slightly differs in some points.

Finally, the last challenge that we want to report is the handling of the customer authentication process. Except for a few exceptions, most use cases consist of two steps. First, an account owner has to provide her consent on an action performed on her account. Afterward, the respective operation can be execute, e.g., a payment is triggered. As part of giving consent, the customer has to be authenticated by a strong customer authentication (SCA) approach (EU, 2015). Even though many different approaches are defined, in most cases, a combination of a login and a One Time Password (OTP), e.g. a TAN, is used. Automating this process in a test tool is complex.

In summary, we had to make some assumptions and limit the test tool’s scope, mainly due to a lack of access to specific resources.

[9]https://adorsys.com/de/produkte/xs2a-sandbox/

6 CONCLUSION

The PSD2 prescribes banks to provide secure access to customer accounts via XS2A interfaces that continuously meet predefined functional and performance requirements. Thus, testing is essential. A known challenge in software testing is creating test cases since it is tedious and error-prone(Scheja and Machielse, 2019). For XS2A interfaces, standardization bodies have designed specifications and already derived test cases from these specifications. However, the standards are not legally binding, and the actual implementations of the XS2A interfaces often deviate from the standardized specification. Thus, banks have to adapt their test cases or create new ones. To address this issue, we address the research question *"How can we design and implement a test tool for XS2A interfaces exploiting existing sets of test cases?"*

We answered the research question by applying a design science research approach including interviews with five experts to create a concept and a prototypical implementation of a test tool for XS2A interfaces. We addressed the issue of interface implementations diverging from standardized implementation using mapping files. We use a simple notation to map changes between the structure and naming of the specification and the implementation. Furthermore, we introduce relevant roles for operating the test tool. A bank benefits from the artifact since it can use a predefined and collaboratively developed set of test cases for testing its XS2A interface instead of having to design test cases, which is an effortful and error-prone task. Also, the bank can use an existing test tool and provide access to the test tool to TPPs. Finally, the concept can be applied to other fields with emerging interface standards.

Several limitations of the tool have been mentioned in section 5.2. Thus, our future work will focus on evaluating the test tool in a real-world setting, i.e., using the test cases created by the NISP and testing the tool with a productive XS2A interface. Furthermore, we will extract general guidelines and findings on testing standardized interfaces and evaluate them by transferring them to other industries with interface standards.

REFERENCES

Arcuri, A. (2019). Restful api automated test case generation with evomaster. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(1):1–37.

Bermbach, D. and Wittern, E. (2016). Benchmarking web api quality. In *International Conference on Web Engineering*, pages 188–206. Springer.

BG (2018). Joint initiative on a psd2 compliant xs2a interface - nextgenpsd2 xs2a framework - operational rules. Technical report, Berling Group. Technical Report.

Bramberger, M. (2019). *Open Banking: Neupositionierung europäischer Finanzinstitute*. Springer.

Cortet, M., Rijks, T., and Nijland, S. (2016). Psd2: The digital transformation accelerator for banks. *Journal of Payments Strategy & Systems*, 10(1):13–27.

EBA (2018a). Commission delegated regulation (eu) 2018/389 of 27 november 2017 supplementing directive (eu) 2015/2366 of the european parliament and of the council with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication (text with eea relevance). *Official Journal of the European Union*.

EBA (2018b). Final report - guidelines on the conditions to benefit from an exemption from the contingency mechanism under article 33(6) of regulation (eu) 2018/389 (rts on sca & csc). Technical report, EBA. Technical Report.

EBA (2018c). Opinion of the european banking authority on the use of eidas certificates under the rts on sca and csc. Technical report, European Banking Authority. Opinion.

EU (2015). Directive (eu) 2015/2366 of the european parliament and of the council of 25 november 2015 on payment services in the internal market, amending directives 2002/65/ec, 2009/110/ec and 2013/36/eu and regulation (eu) no 1093/2010, and repealing directive 2007/64/ec (text with eea relevance). *Official Journal of the European Union*.

Fertig, T. and Braun, P. (2015). Model-driven testing of restful apis. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1497–1502.

Fielding, R. T. (2000). Rest: architectural styles and the design of network-based software architectures. Technical report, University of California. Doctoral dissertation.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

ISO (2013). Iso/iec/ieee 29119-1: Software and systems engineering — software testing — part 1: Concepts and definitions — first edition 2013-09-01. Technical report, ISO.

Kasthurirathne, S. N., Mamlin, B., Kumara, H., Grieve, G., and Biondich, P. (2015). Enabling better interoperability for healthcare: lessons in developing a standards based application programing interface for electronic medical record systems. *Journal of medical systems*, 39(11):182.

Scheja, O. and Machielse, W. (2019). The nextgenpsd2 framework in a pan-european psd2 account access context. *Journal of Payments Strategy & Systems*, 13(1):54–65.

Zachariadis, M. and Ozcan, P. (2017). The api economy and digital transformation in financial services: The case of open banking. Technical report, SWIFT Institute. Working Paper.