# Web Migration Revisited: Lowering the initial hurdle to commence a Web Migration

# Introduction

Situation

* Web Applications widespread & familiar for users
  + Web Application definition
    - Client-Server Architecture
    - HTTP(S) as main communication protocol
    - Client is Web Browser/uses HTML Rendering
    - TODO:CITE
  + One paragraph on the history of Web Applications (TBL, Web 2.0, SOA, Mobile, Cloud/SaaS)
  + One paragraph on advantages of Web Applications over traditional desktop applications
    - Instant Deployment, Faster Development Cycles, Faster Reaction on Changing/New Requirements > reduced time-to-market [1]
    - One target development platform based on open standards HTML, CSS, JS > lower development workload, single infrastructure & tools, lower risk of technology deprecation, staff less training and larger base for hiring
    - Low access requirements (“any system with a browser”), Ubiquity allows for home office etc., lower complexity and effort for end users, potentially larger user base
    - High level of Interactivity, Social Interactions in potentially large user bases, Community-building > URLs, APIs, protocols like OAuth
  + Statista SaaS statistics <https://www.statista.com/statistics/510333/worldwide-public-cloud-software-as-a-service/>
    - Total public cloud SaaS market size 91.75 billion USD worldwide in 2016, increasing (already big and increasing)
  + SaaS market is growing <https://www.bettercloud.com/monitor/wp-content/uploads/sites/3/2017/05/2017stateofthesaaspoweredworkplace-report-1.pdf>
    - Vendors are no longer building on-prem software.
    - SaaS growing faster than PaaS IaaS (cloud shift rate 37%)
    - The number of SaaS apps organizations use is rising, Companies will be running purely on SaaS soon.
    - 86% of end users say SaaS applications help them succeed more than desktop alternatives
  + <https://content.digitalmediastream.co.uk/blog/60-saas-statistics-that-will-change-the-way-you-think>
  + Legacy Systems replaced by SaaS solutions <https://www.gartner.com/newsroom/id/2253215>
  + Digital Banking established in Germany for internet users <https://www.bitkom.org/NP-Marktdaten/Konsum-und-Nutzungsverhalten/E-M-Commerce/Studienbericht-Digital-Banking-2014-bis-2016-161104.pdf>
  + Arbeit 3.0 Verbreitung von mobilen Geräten, Homeoffice, Share-Economy, Einsatz von sozialen Netzwerken, Blogs, Collaboration Tools etc. <https://www.bitkom.org/noindex/Publikationen/2013/Studien/Studie-Arbeit-3-0/Studie-Arbeit-30.pdf>
* Still many non-web legacy systems existing, modernization is still an important topic in industry
  + Legacy System Definition
    - Reasons for their existence [1] (business-critical, perceived as reliable/stable/proven – never touch a running system, optimized performance)
  + TODO: find statistics (<https://docs.google.com/document/d/1iZpumRpQrOuquwLusJk8VzgOS_T1p9oOn1v5nYKWrCo/edit>)
  + Gartner: legacy modernization at position 5 of top ten buisness priorities, 2013

<https://www.gartner.com/newsroom/id/2304615>, Previous years: <http://www.gartner.com/imagesrv/cio/pdf/cio_agenda_insights2013.pdf>

* + NASCIO. State CIO Top Ten Policy and Technology Priorities for 2017, position 5: <https://www.nascio.org/Portals/0/Publications/Documents/2017/NASCIO-TopTen-2017.pdf>
  + Forrester: Application Modernization: Procrastinate At Your Peril! 2011
  + SaaS requires a Web Application first
* Technical Debt topic attracts attention in academia and industry
  + Technical debt definition
  + CAST Report on Application Software Health 2011
  + Estimating the Principal of an Application's Technical Debt, IEEE Software.
  + A systematic mapping study on technical debt and its management, Journal of Systems and Software
  + How do software development teams manage technical debt? – An empirical study, Journal of Systems and Software
* Web Migration in relation to three levels (domains) in ADM [2] (changes on all three levels) > makes it difficult

Problem statement

* Making a transition from non-web legacy systems to web is desirable for companies
* BUT: Commencing web migration is hard
  + Effort
    - High effort [@﻿ForresterResearch2011Modernization] (TODO: Distinguish Big Bang/Incremental [3]reports 12+PM for one project, [4]18+ PM for incremental approach, [5] 8PM, [6] reports about 14 PM for two projects), not only including direct development, but also training for users etc.
    - Hard to estimate, cf. Corrective Maintenance Estimation [7] and difficult to predict the ROI [1]
  + Risk
    - Uncertainty of success/risk of failure due to feasibility threats (monolithic architectures [@﻿ForresterResearch2011Modernization], staff not trained in migration, not trained in web development, LS typically complex)
    - Uncertainty of desirability [@﻿ForresterResearch2011Modernization] (acceptance by customers, usability of web-based solution)
    - Resistance from organization [8]
* (Existing migration approaches assume decision to migrate is taken already)
* Existing migration approaches do not support initial phase
  + Cf. phase support from [9]

Question

* How to support companies (SMEs?) to commence web migration

Solution

* Define web migration process addressing initial fears/resistance
* Risk minimization through migration pilots (prototypes) recommended to identify problems early [@﻿Sneed2010SoftwareMigration] ([@﻿ForresterResearch2011Modernization] accelerates plans to modernize)
* Demonstrate desirability and feasibility of web version of legacy system
  + Reasons/consequences of modernization [@﻿Khadka2014ProfessionalsModernization]
* Identify and maintain existing valuable knowledge through concept assignment platform (annotation platform?)
* Apply rapid prototyping to web migration
* Control impact of web migration on customers through measuring visible changes

Contributions

* Decision Support System for SMEs
* Concept Assignment Process integrated into development and Platform
* Rapid Migration Prototyping
* Similarity Metrics for non-web and web UIs

# State of the Art

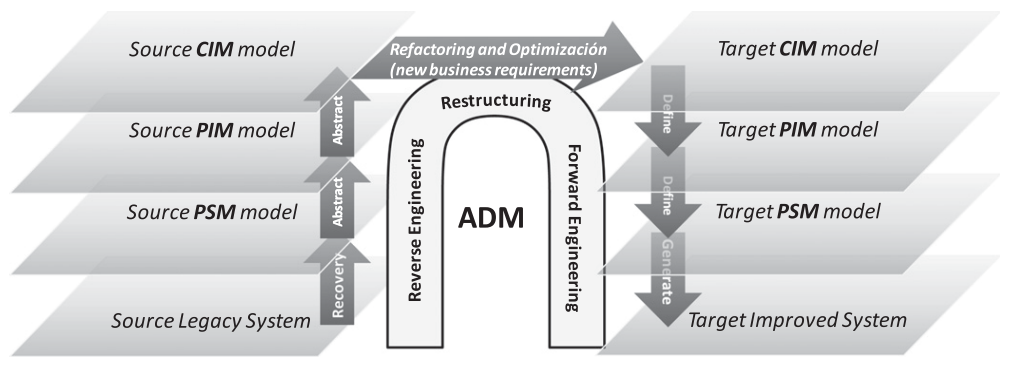
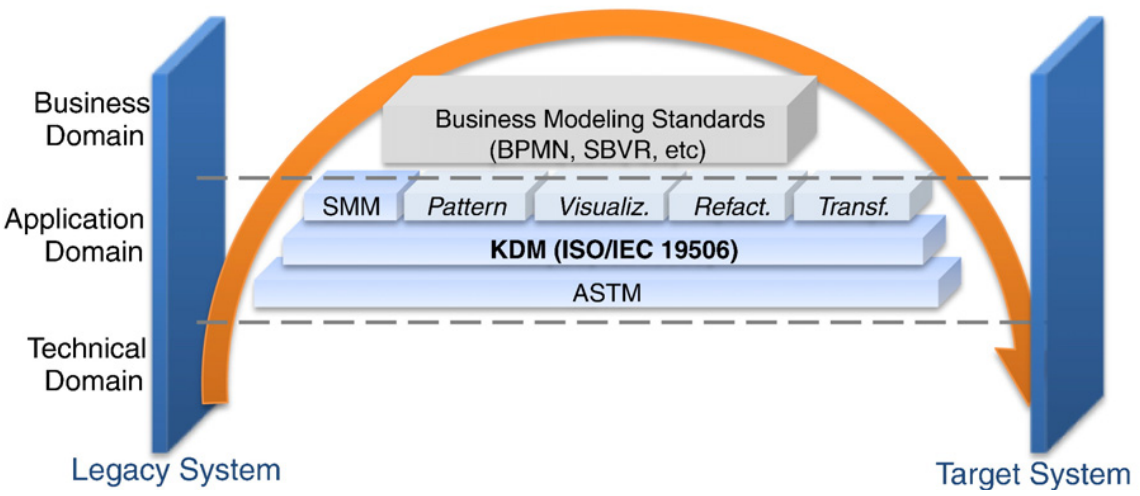
Scenario

* Scenario purpose
  + Real world example of the problems described in Introduction
  + Basis for problem analysis and abstractions of source system, requirements and constraints for web migration process
* Describe Medatixx Company and Software Development
  + Sector and overall products, sector regulations, release cycles
  + Size (SME)
  + Staff and qualifications
  + Development Process, Agility
  + Merger from several former companies and corresponding products with running contracts, impact on software quality
* Describe x.concept legacy codebase
  + LOC and language analysis from cloc
  + classification as large non-web legacy system (compare to Appmarq Repo)
  + Table from Rewamp Paper for abstracting characteristics
* Describe migration objectives

Requirements

* (original Medatixx process requirements 2015)
  + Existing staff, expertise
  + Integration into development process
  + Incremental [@﻿Sneed2010SoftwareMigration] required for large migration projects
  + Cost and Time efficient
  + Tool support
* Scope requirements
  + **S1 Initial phase (Phase Support)**
  + **S2 Web Application Target (Target Environment)**
* Stakeholder (company) requirements
  + **C1 Risk**
    - Migration process failure (feasibility-related)
      * Incremental vs Big Bang/Cold Turkey
      * Resources (Staff Expertise) **TODO**
      * Technical feasibility
    - Migration result failure (desirability-related)
      * Loss of knowledge
      * Pilots/Prototyping
    - Strategies
      * Portfolio management and business case
  + **C2 Re-use of existing artifacts**
    - No re-development from scratch (risk)
      * Knowledge
      * Continuity in UI(X)
    - Existing artifacts
      * Source code
      * Running software including UI
      * Missing: documentation, requirements, models, but source code contains them
    - Re-use can speed up the start (effort)
    - New System should be web version of old one, balancing new technology with continuity
  + **C3 Experience & Tool Support for SMEs**
    - 4 Experience levels:
      * Experience in legacy technology high
      * Experience in legacy system potentially low (erosion of soft knowledge)
      * Experience in migration low
      * Experience in target environment low
    - SMEs, no additional staff, no outsourcing
    - Requires tools support/partial automation and good visualization/guidance in the migration process
  + **C4 Process Integration**
    - Effort, organization structures/teams with responsibilities, ongoing dev and maintenenance activities
    - Integration required
  + Development activities easy to integrate (cf. to ReMiP disciplines)
  + Migration specific ones harder, especially reverse engineering
  + Integration of process, artifacts

Situation

* Standards and Reference Models
  + OMG modernization standards for ADM (architecture driven modernization)
    - ADM
      * ADMTF Mission and goals
      * ADM Horseshoe
      *  [@﻿Perez-Castillo2011MARBLE] [@﻿Perez-Castillo2011KDM]
      * ADM standards
    - KDM (above procedure level) ISO/IEC 19506:2012
    - ASTM (below procedure level)
    - SMM (metrics)
    - QVT (transformations)
  + ReMiP + Figure
  + SOA-MF + Figure
* Existing Migration approaches [9]
* Existing Web Engineering approaches (not addressed in detail, since they lack the migration aspect)
  + MDWE
    - WebML 🡪 IFML (OMG Standard)
    - UWE [10]
    - SHDM
  + Agile WE Approaches
    - AWE [11]
    - (MockAPI [12]/ELECTRA [13] /) MockupDD [14]

Problem

* Migration approaches do not address initial phase [9]
* Migration approaches do not consider SMEs? [9] (TODO:check Software System Migration to Cloud-Native Architectures for SME-Sized Software Vendors)
* Web Engineering approaches start from scratch, not from legacy
  + Requirements Elicitation vs Re-Discovery of Requirements, cf. “Brownfield Software Engineering”
* Lack of consideration of impact on existing users

Current Solutions

* TODO: mention Decision Support System for SMEs as contribution resulting from [9]
* TODO: Check Iterative Migration Approach by Noah Spahn
* Web Migration Approaches
  + to SOA
    - Replacement, Wrapping, Redevelopment, Migration – Advantages/Disadvantages cf. ﻿[@Almonaies2010SOAStrategies]
  + to Cloud
  + Web Systems Evolution
  + to Web
    - wrapper-approaches like [@﻿Colosimo2007ControlledExperiments]
    - ﻿[@Distante2006] UWAT+ redesign process
    - ﻿[@Horowitz1998]



Answer – clear stakeholder, scenario/problem

* Stakeholder: Software Companies (SMEs?) with legacy, non-web, desktop applications and large existing user bases
* Stakeholder Problem: initial hurdle to commence a web migration due to doubts about feasibility and desirability

# Solution

Stakeholder-specific situation

Cf. LFA Problem Tree Analysis from EHRL



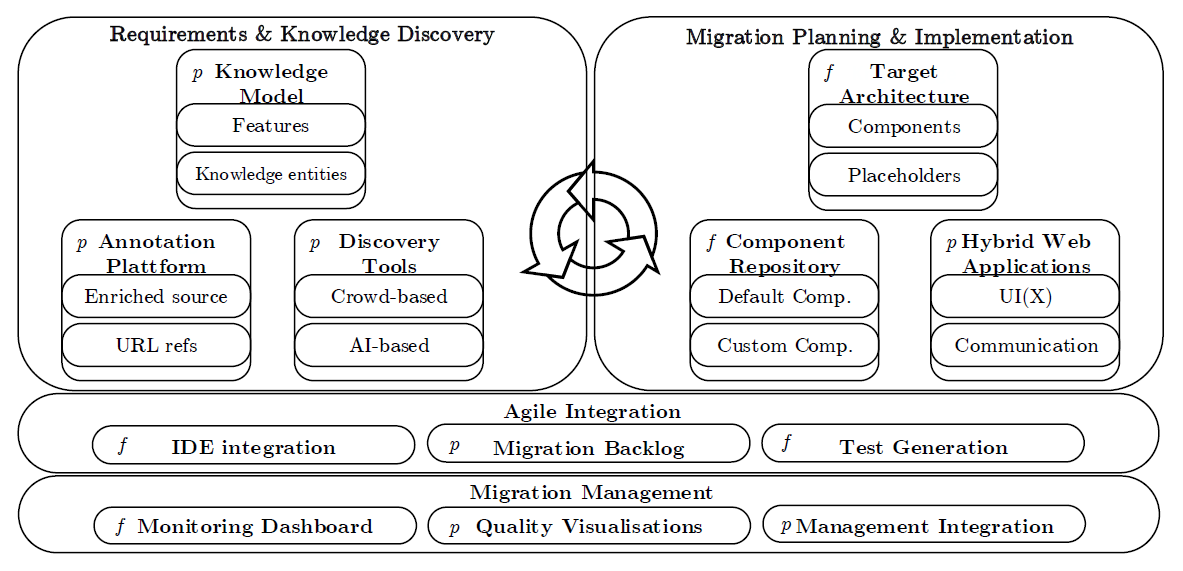
* TODO: refer back to research question in introduction, interpret as HMW Question [@﻿HCD2015] for ideation
* Software Companies (SMEs) are hesitant to commence a web migration because of doubts about feasibility and desirability
* Take into account business case from risk management (risk requirement) [@﻿Seacord2003ModernizingLS]
* Feasibility Doubts
  + Technical: possibility of implementation as web application
  + Human-Resources: lack of staff with web engineering expertise (important for success, to use all possibilities of new technology [@﻿Sneed2010SoftwareMigration])
  + Process Integration: limited human resources and problems of integrating with ongoing software development process
* Desirability
  + Loss of knowledge: risk of losing knowledge
  + Customers: risk of losing customers because of changed UIX (cf. resistance from organizations in [1])
  + Advantages: lack of concrete evidence of advantages of a web based version

Existing Technologies

* HCD, Agile Development: Rapid Prototyping
* Reverse Engineering: Concept Assignment [15], [16]
* HCI: Usability Metrics

Solution overview

* (Decision Support System from Survey)
* AWSM Process & Platform [17]
  + Web-based reverse engineering platform combined with IDE Integration, similar to ARTIST Methodology Process Tool (MPT) [@﻿ARTIST2015ProcessFramework;@﻿Menychtas2014ARTISTJournal] to support business and technical stakeholders and allow for easy integration
* Package-oriented incremental (cf. chicken little Brodie et al. 1995), longterm migration Borchers et al. 1997 (only migrate those parts when they require maintenance anyway), feature-driven incremental ([@﻿Menychtas2014ARTISTJournal]) (ref state of the art)



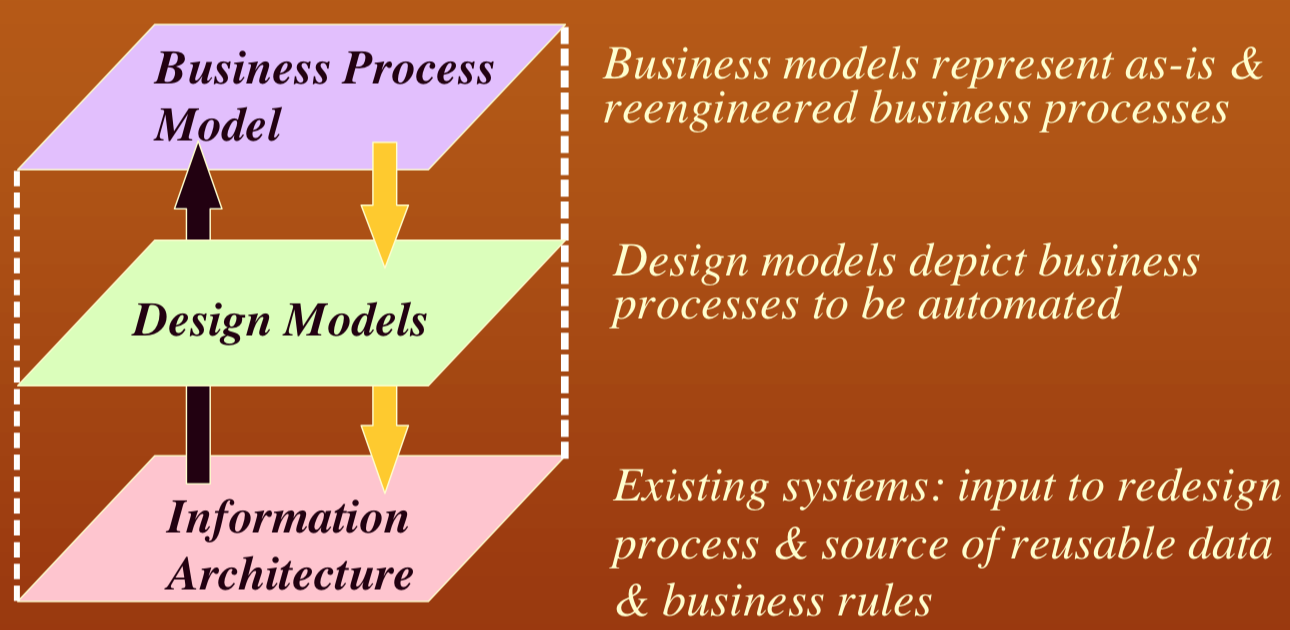
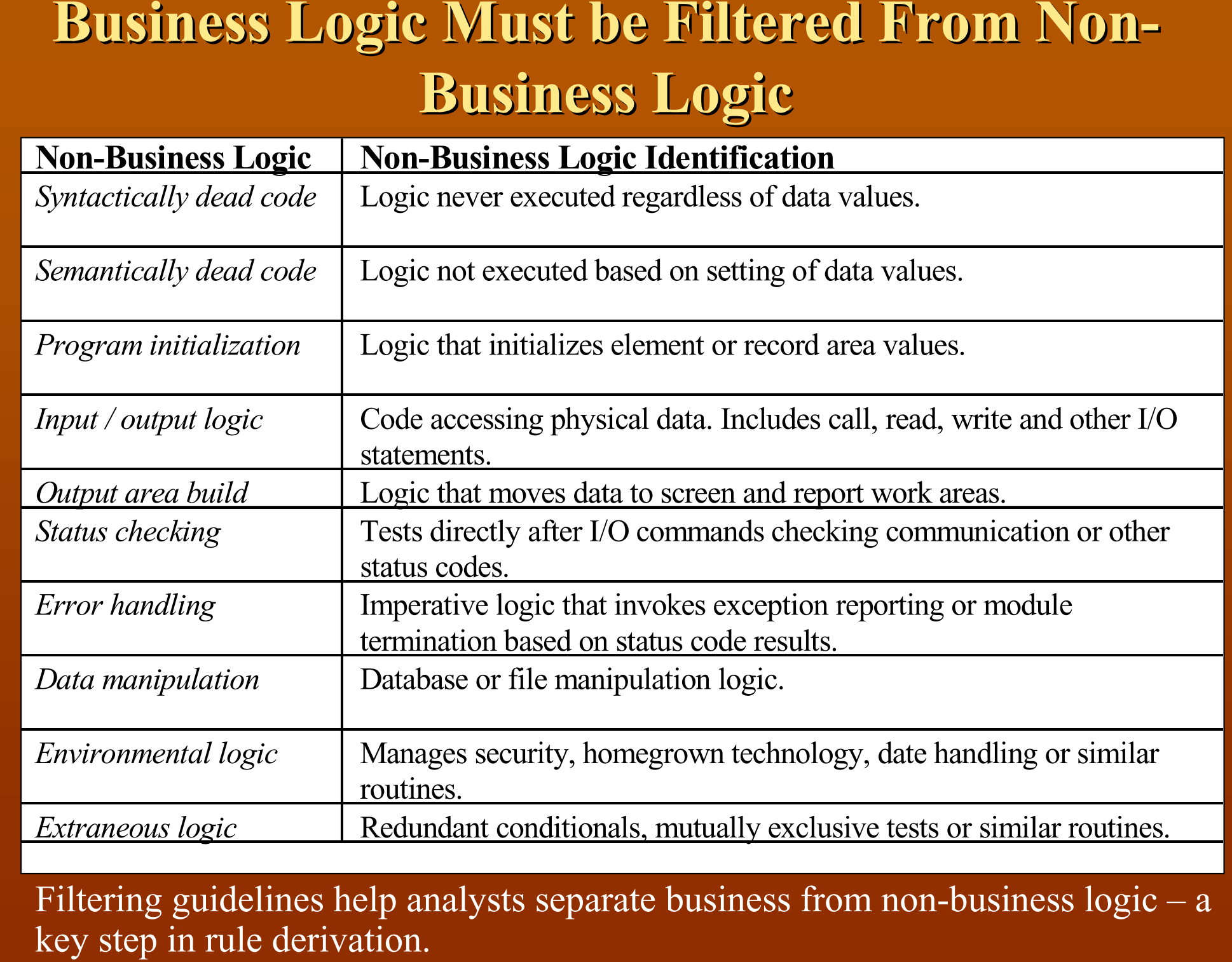
3 Partial Problems/Questions of Architecture

Define Research Goal and Objectives

* How to identify and manage problem and solution domain knowledge in legacy systems? (one of the main obstacles for modernization [8], it is not possible to migrate without understanding the domain [@﻿Masak2006])
* How to apply Rapid Prototyping in Web Migration
* How to measure the similarity of non-web and web versions of a user interface

# Identification and Management of knowledge in legacy systems

Situation

* **SITUATION:** Legacy systems contain valuable problem and solution domain knowledge
  +  [@﻿Ulrich2011]
  + [@﻿Ulrich2011]
  + the legacy source is often "the only source of domain knowledge [@Bodhuin2002DesktopWebMVC]
* Losing this knowledge is a big risk [1]
  + Modernization poses the risk of loss of knowledge. Existing legacy systems represent valuable knowledge about business processes, rules etc. [@Aversano2001;@Sneed2010SoftwareMigration;@Wagner2014Fundamentals;@Bodhuin2002DesktopWebMVC] often resulting from years of requirements elicitation and experience. However, similar to tacit knowledge in organizations is not expressed explicitly but guides human behaviour [@Nonaka2008TacitKnowledge], the knowledge legacy systems is not explicitly documented but governs how they operate. Thus, modernization bears the risk of losing this valuable knowledge.
  + Codification [@Hansen1999KnowledgeManagement]
* Re-gaining this knowledge is important for successful migration and requires reverse engineering techniques and tools [﻿@Sneed2010SoftwareMigration]
* Migration aims at retaining functionality of legacy system in new environment [@﻿Bisbal1999LegacyInformationSystems]
  + Maintaining functionality means maintenance of [@﻿Wagner2014]
    - Domain knowledge
    - Business logic
* Modeling business processes is also an opportunity to renew them/extend them with new functionality, representing a target state with extended scope [@Sosa2014MigraSOA]
* Importance of knowledge and knowledge management in migration has recently been acknowledged [@﻿Razavian2010SAPIENSA;@﻿Razavian2013PHD]
* Decomposability is vital for migration
  + Depends on separation of UI, business logic, Data layer [@﻿Lucia2008]
  + ﻿inseparable in non-decomposable systems, whereas in semi-decomposable systems the user interfaces are separated from the application logic and the database.
  + Cf. also ﻿[@Canfora2000Decomposing]

Existing Technologies/Related Work

* **CONTEXT:** Reverse Engineering
  + Cf. [@﻿Tilley1996ProgramUnderstanding]: canonical process, types of knowledge
  + Re-documentation
  + Recover Activity Area in REMICS
    - KDM and UML
    - KDM extension EKDM [@﻿Remics2013RecoverToolkit]
    - Focuses too much on technical perspective and structure of the legacy system
  + MARBLE [@﻿Perez-Castillo2011]
    - Business process recovery
  + ﻿Some migration strategies like [@Bodhuin2002DesktopWebMVC] use static anlysis to identify the fundamental concepts of the application domain in the legacy source, but do not further specify how, do not provide support or automation
* **SPECIFIC METHOD:** Concept Assignment
  + - “will never be completely automated”, but “some useful automation is possible” [15]
    - Two step process: 1 “Identify which entities and relations … are really important” 2 “Assign them to known (or newly discovered) domain concepts and relations” [15]
    - Similar to reverse engineering in MIGRASOA, where services are manually labeled [@﻿Sosa2014MIGRARIA;@Sosa2013MIGRARIA]
* Knowledge management in Software Engineering [@﻿Razavian2013PHD]
  + Codification [@﻿Hansen1999KnowledgeManagement]
* **ENABLER:** Crowdsourcing

Problem

* **PROBLEM:** No/poor legacy artifacts: documentation, models, requirements
  + ﻿“lack of modelling steps, these frameworks do not provide a systematic and uniform way to deal with changes in the organization” [@﻿Sosa2014MIGRARIA]
* Ad-hoc migration approaches adopted in industry do not consider systematic reverse engineering, focus on forward engineering, knowledge “remains tacit in stakeholders minds” ﻿[@Razavian2012]
  + But stakeholders are not necessarily available for old legacy systems
  + Tacit knowledge [@﻿Nonaka2008TacitKnowledge] requires “person-to-person knowledge transfer” ﻿[@Razavian2012], which can be disadvantageous in the light of organizational resistance [@Khadka2014ProfessionalsModernization] hindering knowledge sharing
* Existing re-documentation approaches focus on solution domain knowledge, e.g. discovery tools in [@AmazonWebServices2018Migration] on systems level (databases, servers, OS, …), OMG Standards KDM and ASTM focus only syntactic level
  + ﻿“Knowledge discovery is often limited to reverse engineering of legacy code. The business process and rules recovery is poorly addressed” [﻿@Mohagheghi2011REMICS]
* Or they focus on specific Meta-Models like UWA in RE-UWA [@﻿Bernardi2009Re-UWA]
* Manual re-documentation is time-consuming & cannot be easily integrated into daily software development and maintenance
* Hard to extract [8] (Survey with professionals: Difficult to extract business knowledge, cf. participants quotes in [1])
* Very difficult to automate for large, complex information systems [@﻿Canfora2007], risk is low precision/recall
  + In line with results from our own experiments using ML technologies

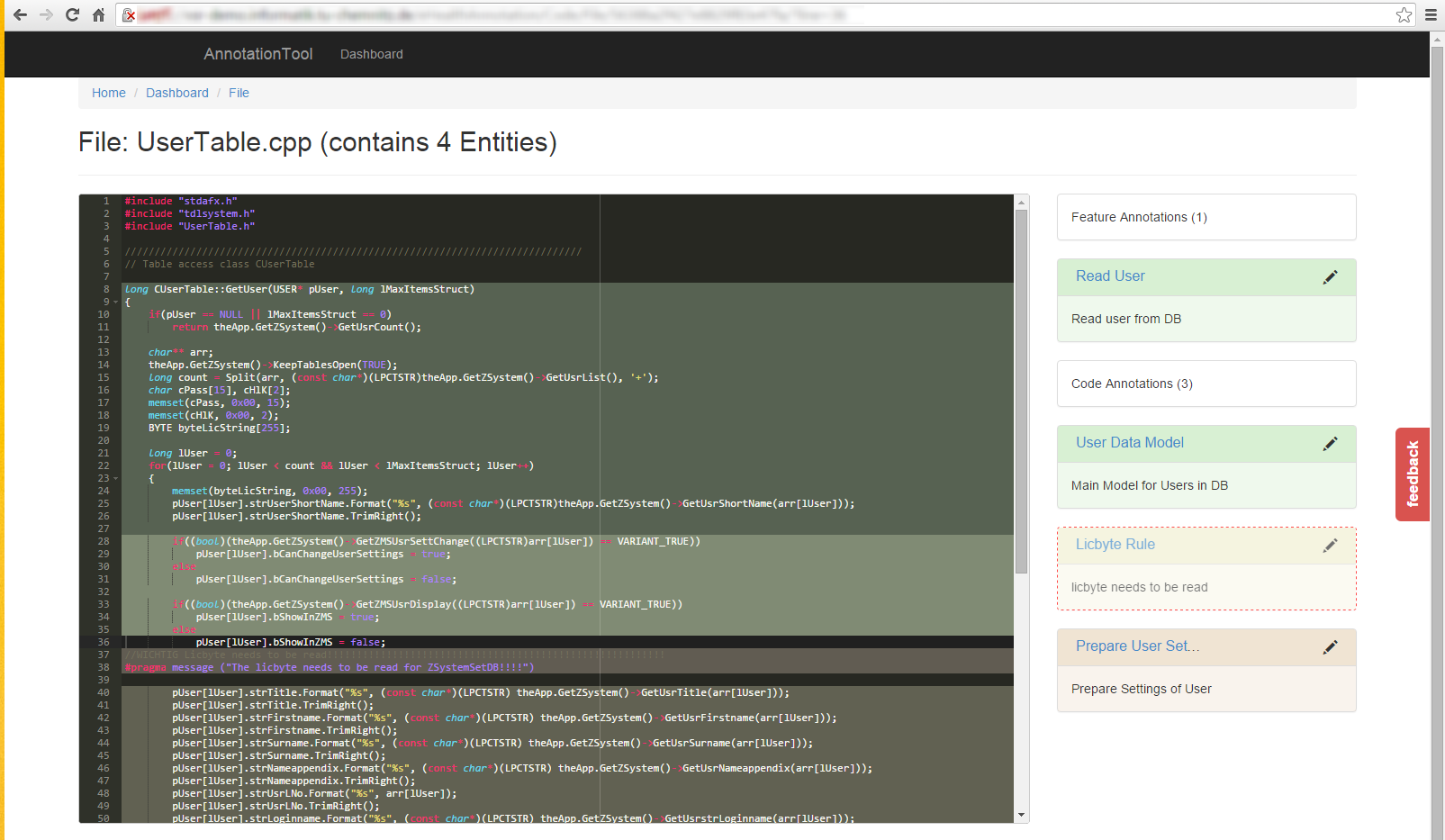
Question

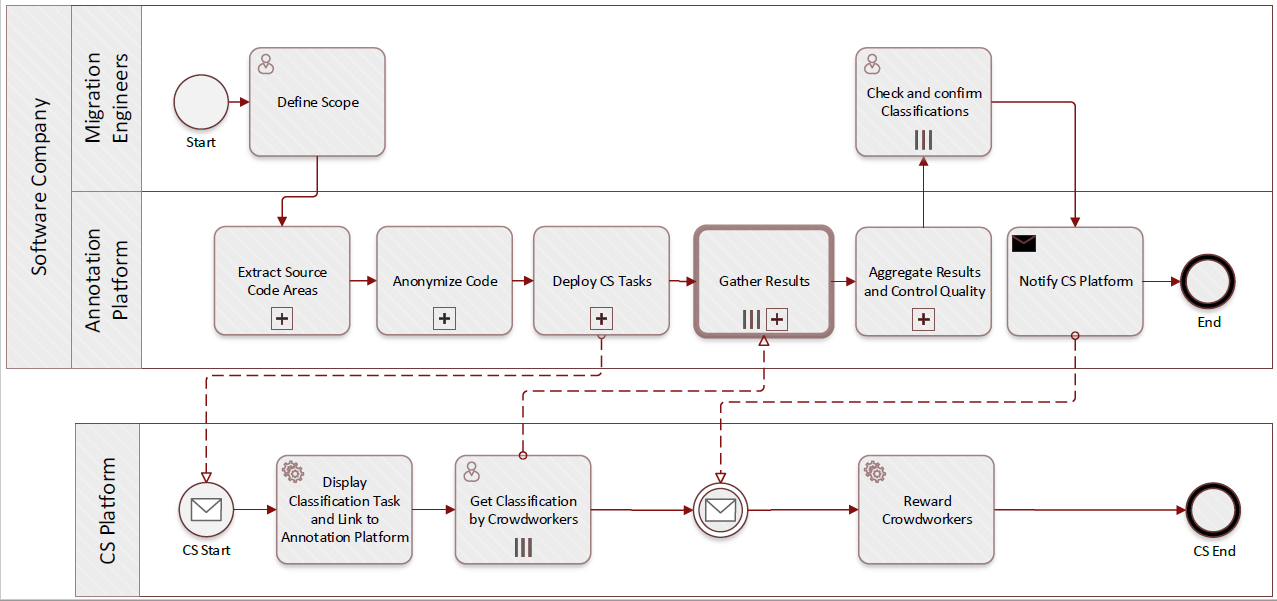
* **CHALLENGE:** How to identify and manage problem and solution domain knowledge in legacy systems?

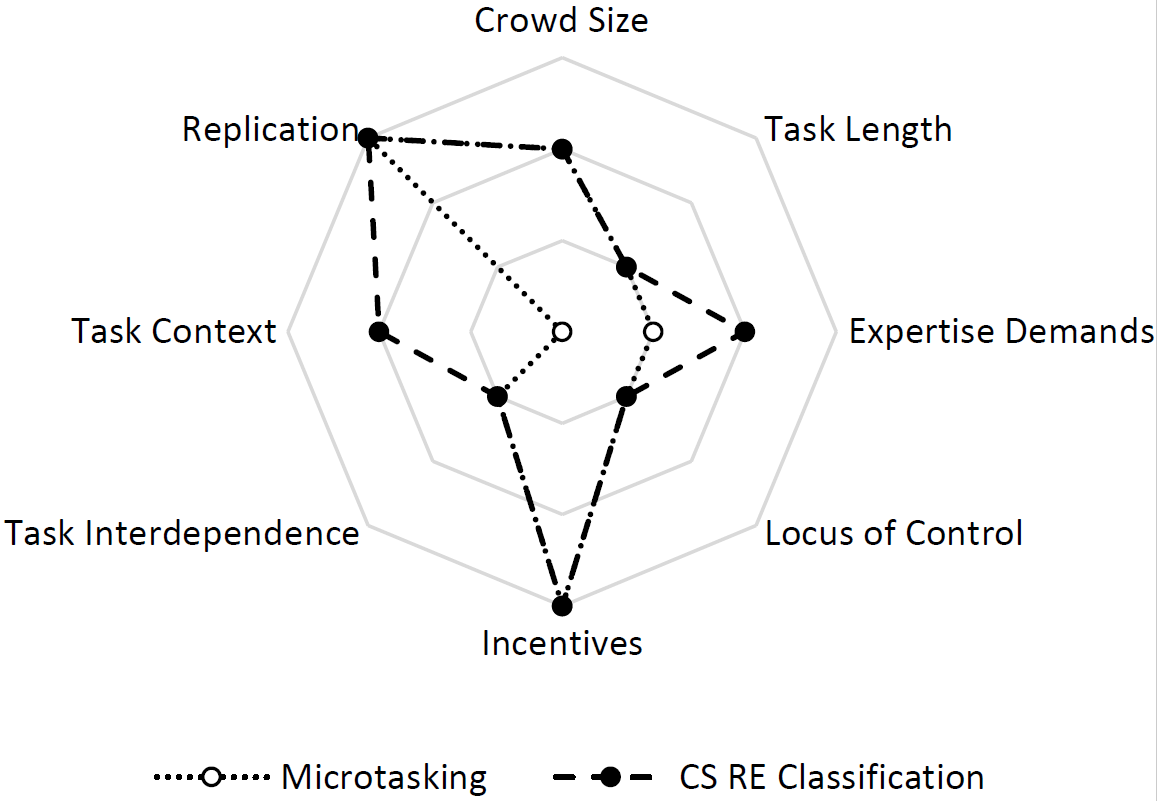
Solution

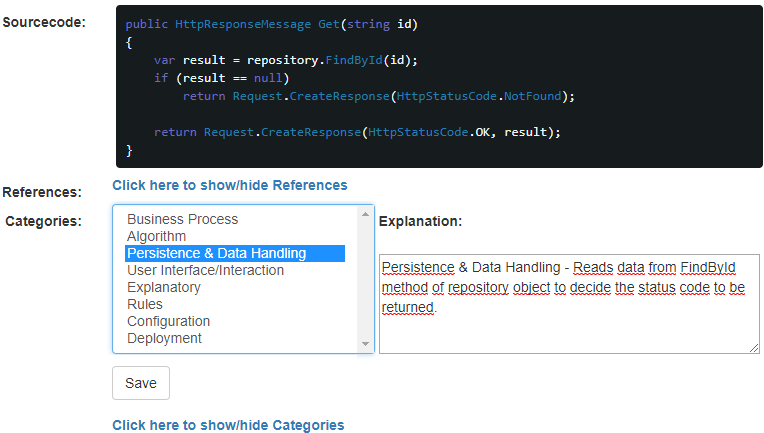
* Principles
  + \*\*Low Resource Reverse Engineering\*\* Improving the understanding of the legacy system and securing valuable knowledge against loss is an important principle of web migration which is achieved through reverse engineering. It involves identification of the components of the legacy system and their relationships and their representation at higher levels of abstraction [@Chikofsky1990ReverseEngineering]. Reverse engineering forms the basis of reengineering and transformation approaches and comprises \*redocumentation\* and \*design recovery\* activities. According to RO2 reverse engineering needs to be achieved with only limited resources.
* Concept Assignment with AWSM Platform [17]
  + AS PRINCIPLE
  + [@Biggerstaff1994ConceptAssignmentJournal;@Biggerstaff1993ConceptAssignmentICSE]
  + Concept assignment as reverse engineering technique to identify valuable knowledge/models [@Gold2005ConceptSlicing] TODO:REF\_TO\_OCCURENCE\_IN\_STATE\_OF\_THE\_ART- slicing is only useful for wrapper approaches or automatic service extraction
  + Process follows idea of longterm migration Borchers et al. 1997 (only migrate those

parts when they require maintenance anyway) [@Borchers1998Longterm] but for reverse engineering as integration with ongoing agile development



* + ﻿“need to identify relevant and non-relevant legacy components” [@﻿Lewis2008SMART]
  + Supports static code analysis
  + Can be integrate results from dynamic code analysis like in [@﻿Fuhr2013SOAMIG]:
    - Use aspect-oriented tracing of method calls per scenario / business process (e.g. using AspectC++)
    - but some requirements artifacts such as user stories, fine-grain requirement descriptions etc. are significantly smaller and have different perspective than scenarios, require manual annotation nonetheless
    - even [@﻿Fuhr2013SOAMIG] argues that dynamic analysis results need to be checked manually
  + Can be used to distinguish business-process-specific from general helper classes following the method described in [@﻿Fuhr2013SOAMIG]
  + ﻿SOA-MF conceptual reference framework for SOA Migration [@﻿Razavian2010SOA-MF] distinguishes knowledge on four levels: code-related, design-element-related, composition, business domain knowledge
  + Architectural knowledge elicitation in SAPIENSA [@﻿Razavian2010SAPIENSA]
    - Architectural Knowledge [@﻿DeBoear2007ArchitecturalKnowledge ;@﻿Farenhorst2009PhD]
    - ﻿Problem-related knowledge
      * business processes, business rules
    - ﻿non-functional requirements
      * quality attributes such as performance
    - ﻿Solution-related knowledge
      * structural design
      * design decisions, discarded alternatives
    - Requires a generic and extensible and queryable knowledge representation model
    - ﻿“to enable an effective knowledge capturing and sharing, the knowledge extracted in this phase should be explicitly represented”
    - As-is business process portfolio ﻿allows “the various stakeholders to understand the portfolio of available (internal and external) applications and business processes”
  + Generic conceptual model would allow integration with various different existing methods, e.g. with metamodels from UWAT+-based reengineering [@﻿﻿Distante2006a] contributing the missing support tool for the reverse engineering phase
  + AWSM models are at PSM or CIM level of abstraction
* OWL Ontology and SPARQL Endpoint
  + Queryable source code knowledge as PRINCIPLE
  + Ontology as flexible conceptual model allowing to associate arbitrary knowledge models with legacy codebases
  + Show as KDM^[KDM has been adopted as ISO/IEC 19506:2012 standard] extension using UML DecoratorProfile, define mapping onto OWL
    - Cf. [@﻿Perez-Castillo2011KDM] KDM as ontology of legacy systems
    - KDM defines XMI schema (several XML schemas), but OWL advantages and queryable endpoint
    - OMG standards ADM, KDM, ASTM in use for model-driven modernization, commonly extended for specific scopes like RSL in REMICS
      * The from-scratch creation of Extended KDM (EKDM for short). EKDM is a set of metamodels backed up by a UML-like profile named “DecoratorProfile” (see D3.3). EKDM realizes a consistent link between the ASTM and KDM OMG standards that cannot, because of poor maturity, be used directly in industrial software modernization projects. [@Remics2013RecoverToolkit]
  + [@﻿Tilley1996ProgramUnderstanding]
    - Representation according to data model that supports abstraction
    - Information exploration through navigation, hyperspace, hypertext-base navigation
    - Powerful query languages
      * Cf. code query technique, program database in [@﻿Paul1996CodeQuery], but on a higher level of abstraction
      * Code query technologies “﻿support the so-called extract-abstract-present paradigm” [@﻿Khadka2011ServiciFi]
  + Model-driven approaches provide ﻿1. formal definition of valid models (metamodels), **2. querying of models (query languages)** and 3. transformation of models (transformation languages) [@﻿Fuhr2013SOAMIG]
    - Many existing metamodels, e.g. MOF, KM3
    - Many existing transformation languages, e.g. QVT, ATL
  + ﻿Cf. utility services: non-business-centric portions of code like logging [@Marchetto2008]
  + Cf. Taxonomy of legacy artifacts in ARTIST [@﻿ARTIST2013Taxonomy]
    - Configuration: ﻿artefacts that are part of the legacy system sources but more dedicated to the systems’ parameterization and customization
    - Documentation: ﻿artefacts written during or after the implementation of the legacy systems and intended to be provided to future users and/or developers
    - Data
    - Communication
    - Logic
    - Presentation
  + Supports feature-driven incremental migration through marking of features to be migrated [@﻿Menychtas2014ARTISTJournal]: “﻿For instance, persistence could be a feature to iterate on: during this iteration all data ﻿sources marked for migration are transformed.“
* Crowdsourced Reverse Engineering [18]
  + Crowdsourcing as principle
  + value-added knowledge discovery of concepts is difficult and requires involvement of domain experts and application experts [@﻿Perez-Castillo2011KDM]
* 

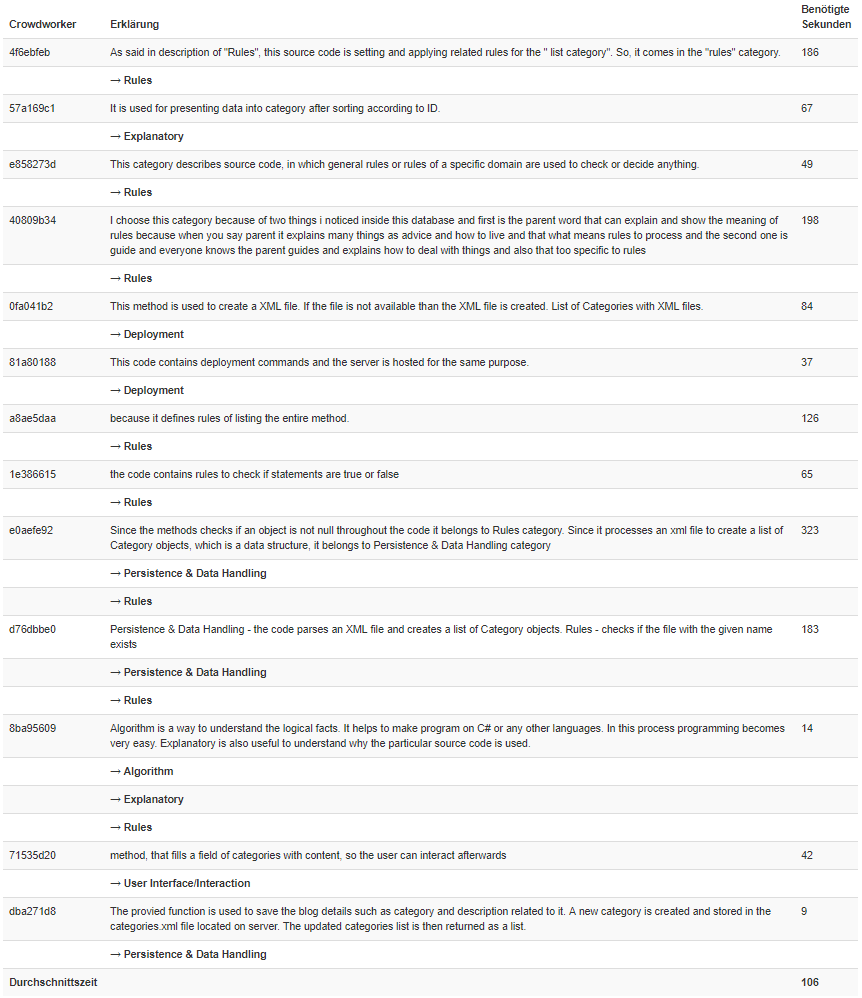
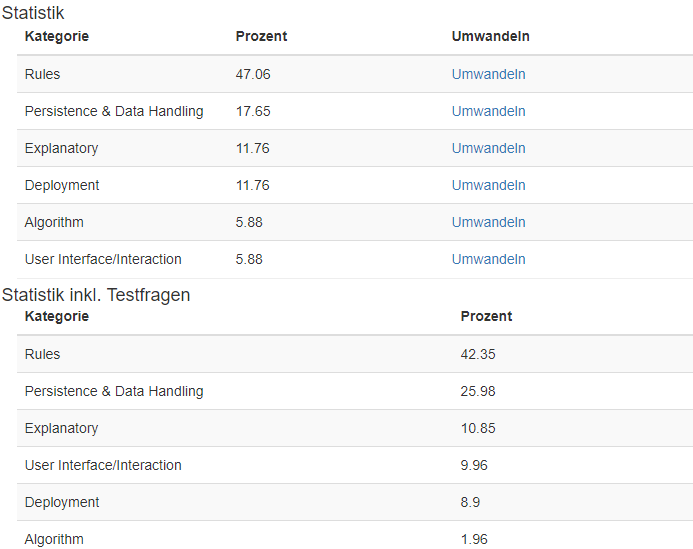




Cf. also: Panos Ipeirotis: Adventures in Crowdsourcing

Evaluation

* Experimental Results of CSRE



# Rapid Web Migration Prototyping

Situation

* Migration pilots in Software/Cloud Migration [@AmazonWebServices2018Migration] in a wider sense, not specific to SaaS/Application Migration
* Rapid Prototyping is used in early stages of Web Engineering to address desirability, feasibility
* Cf. RIA features in [@﻿Rodriguez-Echeverria2011]
* Even comprehensive methodologies like ARTIST fail to address benefits and early customer feedback in pre-migration phases [@﻿ARTIST2014Methodology]:
  + In BUSFEAS.A3, analyst estimates cost and benefits, but this requires well-informed staff to derive profit estimates without clear understanding of benefits in terms of functional and non-functional advantages for customers without feedback mechanism. Only after migration decision, activity VALPRO.A1 addresses customer needs through interviews, surveys etc., but it lacks a method for providing a concrete, tangible prototype to serve as means of communication.

Existing Technologies

* Rapid Prototyping in Web Engineering
* <https://www.atlassian.com/blog/agile/agile-design-prototype>
  + Demo Trust
* Cloud migration prototyping [@﻿Fowley2017CloudSME]
  + Focus on technical feasibility, comparing reengineering options, performance comparisons
  + Top-level perspective (component allocation, e.g. relational database virtualized vs PaaS SQL service vs NoSQL etc.)

Problem

* Web Migration differs from forward Web Engineering
* Rapid Prototyping has to be adapted to Web Migration Context

Question

* How to apply Rapid Prototyping in Web Migration

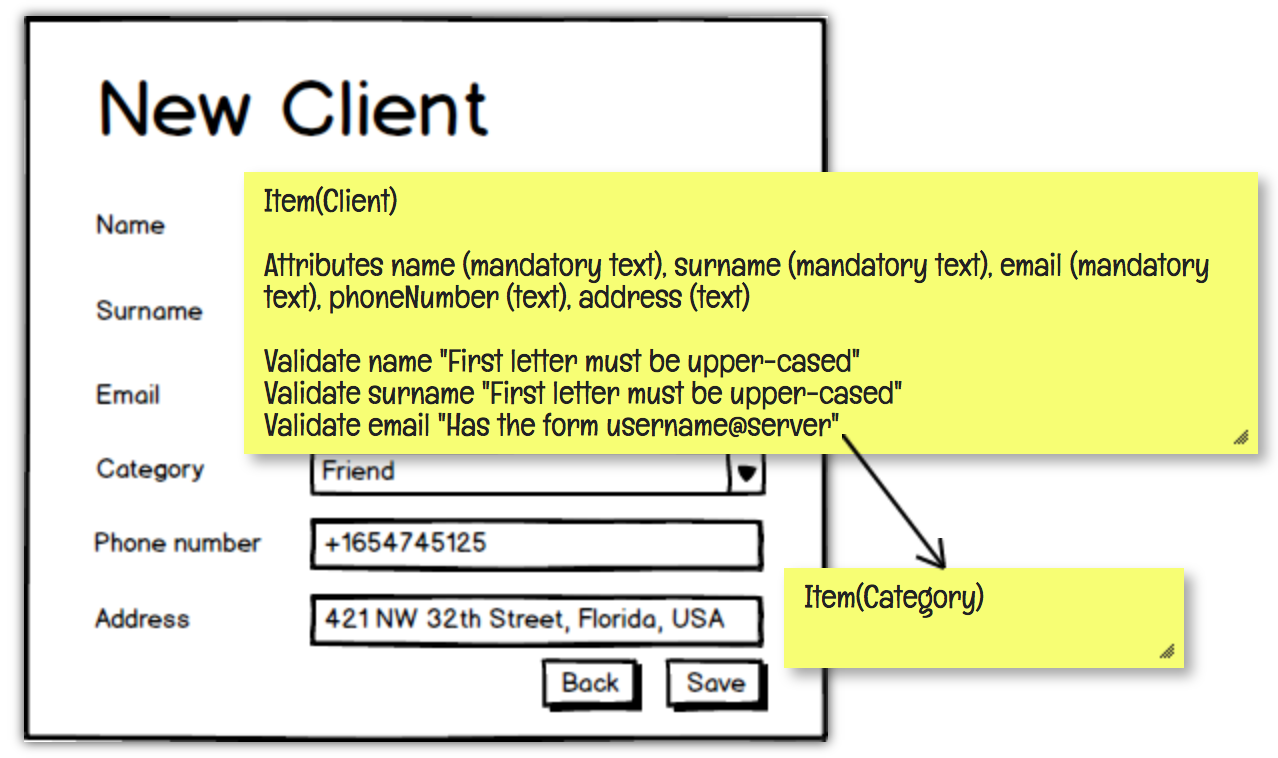
Solution

* ﻿[@ForresterResearch2011Modernization] accelerate plans to modernize: lessen risk, communicate, [@﻿Khadka2014ProfessionalsModernization] resistance from organization, ﻿[@﻿Khadka2014ProfessionalsModernization] communicate reasons/consequences of Modernization and ROI
* ReWaMP Process [19] + Tool Support (MA Tobias Lang)



* UI Generation (BA Thomas Weber)
* API Generation (MockAPI) [12] [13] (1 Journal submitted)
  + Public API benefits: enable integration of third parties, cf. to scenario
  + Vast majority of public APIs are REST APIs [@﻿Neumann2018PublicApis]





* WASM-based Prototyping [19]



Evaluation

# Similarity of User Interfaces in Web and non-Web contexts

Situation

* Legacy system user base is familiar with and well-adapted to legacy user interface
  + Often a requirement to limit differences and maintain the look and feel of the legacy user interface to avoid forcing end-users to change their working habits [@Lucia2008;@﻿Distante2002]
* It is worth noting that the business processes in many doctors' offices are based on the processes represented in the PMS, making it hard to introduce larger changes in user interaction patterns or even replace the PMS. (cf. Trello meeting notes)
* Web Migration visibly impacts the user interface
  + ﻿Changing the interaction paradigm towards navigational one is one of the most important challenges of web migration [@Distante2002]
* Measurability is a recognized necessity when dealing with legacy systems to quantify their state [@﻿Masak2006], however existing metrics are based on the legacy system only, do not consider target and source system together
* Existing migration strategies do not consider UI similarity, or do that only for TUIs which have a restricted design space (e.g. [@﻿Bodhuin2002DesktopWebMVC])

Existing Technologies

* General Image Similarity Metrics
* DOM-based UI Analysis
* Clustering-based similarity for legacy mainframe UI snapshots to identify compound states in state-transition interaction model [@﻿Stroulia2002] based on custom feature set [@﻿Stroulia1999]
  + UIs are simpler, less diverse and very similar because all from the same system, just in different UI states
* UI Generation step in CelLEST
  + Generated UIs are very simple representations of TUIs, not real web UIs, similarity is easy to achieve [@﻿Stroulia2002]
* ﻿[@Grechanik2018]
* [@﻿Lucia2007SimilarPages]
* <https://www.heise.de/developer/meldung/Machine-Learning-Microsofts-Sketch2Code-erzeugt-HTML-Code-aus-Zeichnungen-4146813.html>
* <https://www.computer.org/csdl/mags/so/1995/01/s1064.pdf>

Problem

* Changes due to Web Migration risk losing users
  + [@﻿Khadka2014ProfessionalsModernization] resistance from organization
* Learning the new Web User Interface requires effort
  + ﻿Keeping look and feel of Web UI very similar to original often is a requirement imposed by companies to avoid forcing the end-users to change their working habits [@Lucia2008]
* “Modernization customers require ﻿new applications to look like the old ones through similar screens and interaction ways even if new applications are Web-based” [@﻿Remics2013RecoverToolkit]
* “﻿In this phase it is important to comply with the cognitive characteristics of the legacy application from which the Web application derives (User Interface Constraints). To avoid presenting the user with a ﻿drastically different application, it is important to preserve the old (established) mode of operation as far as possible.” [@﻿Distante2006a], but no concrete approach for measuring/ensuring this similarity is provided in UWAT+-based reengineering
  + UWAT+ addresses user interaction to some extent by mapping business process and web transactions to create navigation model, but UI layout is not considered
* Existing UI similarity approaches cannot be applied to legacy and web UIs because they are based on code analysis

Question

* How to measure the similarity of non-web and web versions of a user interface

Solution

* Similarity Analysis based on visual aspects [20] [21] [22]
  + Visual complexity [@﻿Tuch2009VisualComplexity]

Evaluation

* Experimental Evaluation with UI pairs and test subjects [20] [21] [22]
* TODO: Check Table 11 in “MINING DATA, IMAGES AND WEBSITE CODE FOR HCI ENGINEERING” https://www.researchgate.net/profile/Maxim\_Bakaev/publication/329443196\_Mining\_data\_images\_and\_website\_code\_for\_HCI\_Engineering/links/5c08e42692851c39ebd63dba/Mining-data-images-and-website-code-for-HCI-Engineering.pdf

# Evaluation

Of overall Solution: Situation: Evidence of S1-S3 working, show that S is working based on S1-S3 (better than SotA if existing)

* Objective-based Evaluation?

# Conclusion & Outlook

[1] R. Khadka, B. V. Batlajery, A. M. Saeidi, S. Jansen, and J. Hage, “How do professionals perceive legacy systems and software modernization?,” in *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, 2014, pp. 36–47.

[2] V. Khusidman and W. Ulrich, “Architecture-driven modernization: Transforming the enterprise,” 2007.

[3] D. Distante, G. Canfora, S. Tilley, and S. Huang, “Redesigning legacy applications for the web with UWAT+: A Case Study,” in *Proceeding of the 28th international conference on Software engineering - ICSE ’06*, 2006, pp. 482–491.

[4] M. Bernhart, A. Mauczka, M. Fiedler, S. Strobl, and T. Grechenig, “Incremental reengineering and migration of a 40 year old airport operations system,” in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, 2012, pp. 503–510.

[5] L. Aversano, G. Canfora, A. Cimitile, and A. De Lucia, “Migrating legacy systems to the Web: an experience report,” in *Proceedings Fifth European Conference on Software Maintenance and Reengineering*, 2001, pp. 148–157.

[6] P.-J. Maenhaut, H. Moens, V. Ongenae, and F. De Turck, “Migrating legacy software to the cloud: approach and verification by means of two medical software use cases,” *Softw. Pract. Exp.*, vol. 46, no. 1, pp. 31–54, Jan. 2016.

[7] S. Singh, “Estimating Effort for Corrective Software Maintenance Using Neuralnetworks and Regression Technique,” *Int. J. Comput. Sci. Commun. Eng.*, vol. 4, no. 1, 2015.

[8] B. V Batlajery, R. Khadka, a M. Saeidi, S. Jansen, and J. Hage, “Industrial Perception of Legacy Software System and their Modernization,” 2014.

[9] S. Heil and M. Gaedke, “Web Migration - A Survey Considering the SME Perspective,” in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering*, 2017, pp. 255–262.

[10] N. Koch, A. Knapp, G. Zhang, and H. Baumeister, “Uml-Based Web Engineering,” in *Web Engineering: Modelling and Implementing Web Applications*, R. G., P. O., S. D., and O. L., Eds. London: Springer London, 2008, pp. 157–191.

[11] A. McDonald and R. Welland, “Agile Web Engineering (AWE) Process: Multidisciplinary Stakeholders and Team Communication,” in *Web Engineering*, 2003, pp. 515–518.

[12] J. M. Rivero, S. Heil, J. Grigera, M. Gaedke, and G. Rossi, “MockAPI: An Agile Approach Supporting API-first Web Application Development,” in *ICWE 2013, LNCS 7977*, 2013, pp. 7–21.

[13] J. M. J. M. Rivero *et al.*, “An Extensible, Model-Driven and End-User Centric Approach for API Building,” in *Web Engineering: Proceedings of the 14th International Conference, ICWE 2014*, 2014, vol. 8541, pp. 494–497.

[14] J. M. Rivero and G. Rossi, “MockupDD: Facilitating Agile Support for Model-Driven Web Engineering,” in *Current Trends in Web Engineering: ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers*, Q. Z. Sheng and J. Kjeldskov, Eds. Cham: Springer International Publishing, 2013, pp. 325–329.

[15] T. J. Biggerstaff, B. G. Mitbander, and D. E. Webster, “Program understanding and the concept assignment problem,” *Commun. ACM*, vol. 37, no. 5, pp. 72–82, May 1994.

[16] T. J. Biggerstaff, B. G. Mitbander, and D. Webster, “The concept assignment problem in program understanding,” in *Proceedings of 1993 15th International Conference on Software Engineering*, 1993, vol. 37, no. 5, pp. 482–498.

[17] S. Heil and M. Gaedke, “AWSM - Agile Web Migration for SMEs,” in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, 2016, no. Enase, pp. 189–194.

[18] S. Heil, F. Förster, and M. Gaedke, “Exploring Crowdsourced Reverse Engineering,” in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE*, 2018, pp. 147–158.

[19] S. Heil, V. Siegert, and M. Gaedke, “ReWaMP : Rapid Web Migration Prototyping leveraging WebAssembly,” in *Proceedings of 18th International Conference on Web Engineering (ICWE2018)*, 2018.

[20] S. Heil, M. Bakaev, and M. Gaedke, *Measuring and ensuring similarity of user interfaces: The impact of web layout*, vol. 10041 LNCS. 2016.

[21] M. Bakaev, V. Khvorostov, S. Heil, and M. Gaedke, *Web intelligence linked open data for website design reuse*, vol. 10360 LNCS. 2017.

[22] M. Bakaev, V. Khvorostov, S. Heil, and M. Gaedke, “Evaluation of user-subjective web interface similarity with Kansei engineering-based ANN,” in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*, 2017.