Secure Coordination of RIA Tiers

Vrije Universiteit Brussel

KATHOLIEKE UNIVERSITEIT LEUVEN

AGENTSCHAP INNOVEREN & ONDERNEMEN

AGENTSCHAP INNOVEREN & ONDERNEMEN

Strategic Basic Research

pilot studies

business cases

problems



- Prof. Elisa Gonzalez Boix: distribution
- Prof. Coen De Roover: program analysis
- Prof. Frank Piessens: security

Vrije Universiteit Brussel

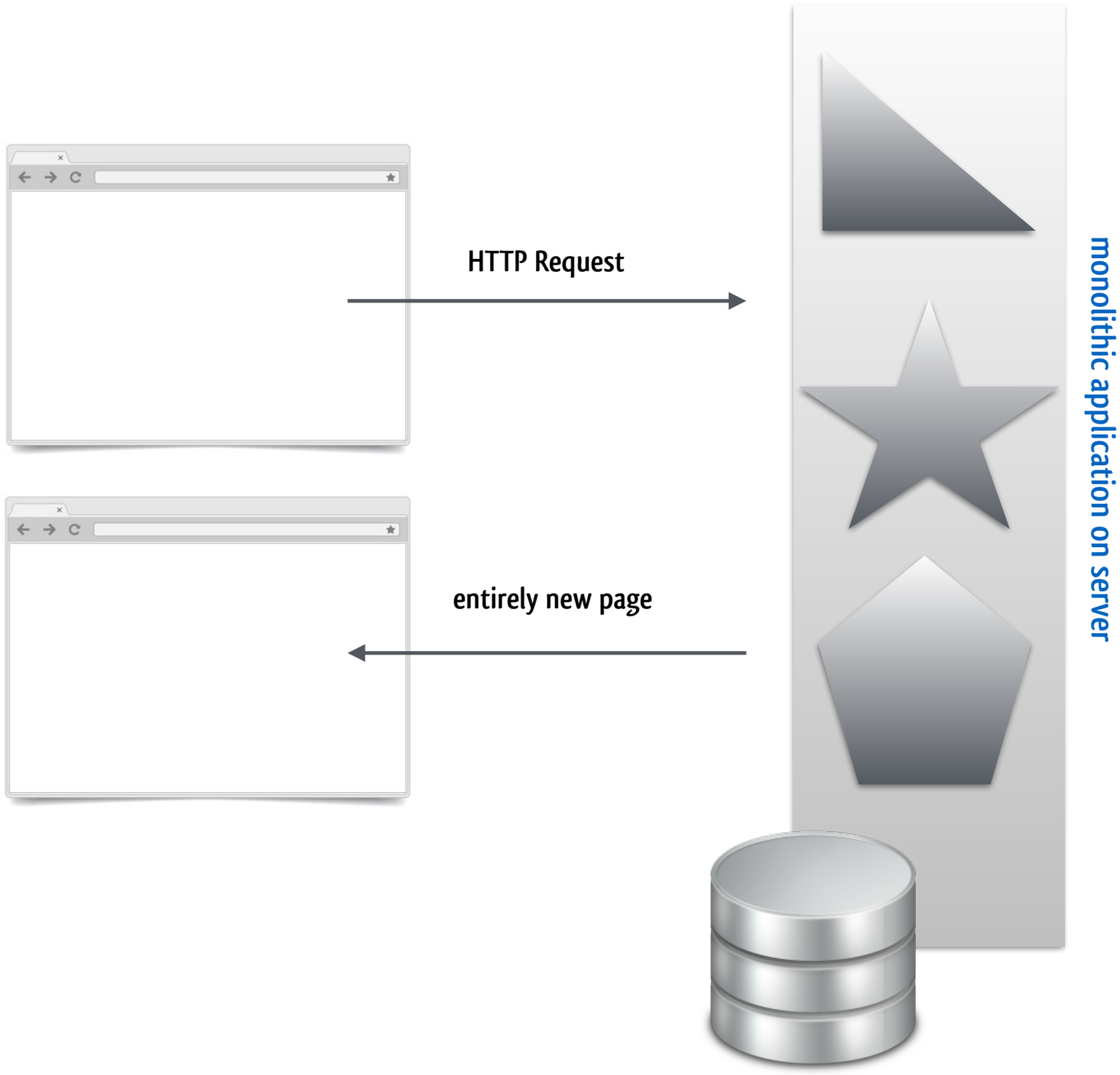
KATHOLIEKE UNIVERSITEIT LEUVEN

technology

demonstrators

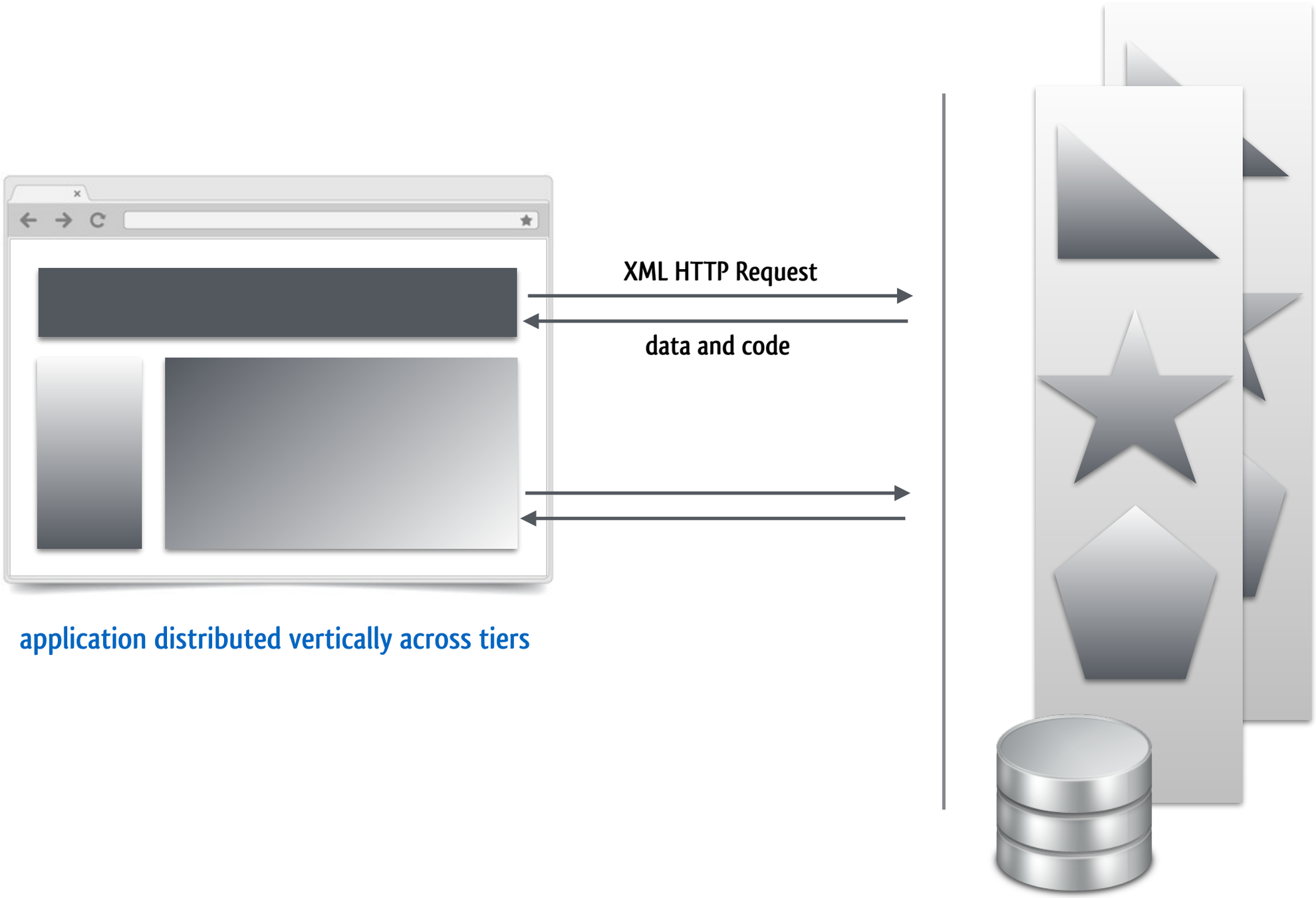
follow-up projects

Evolution of web application architectures



multi-page application

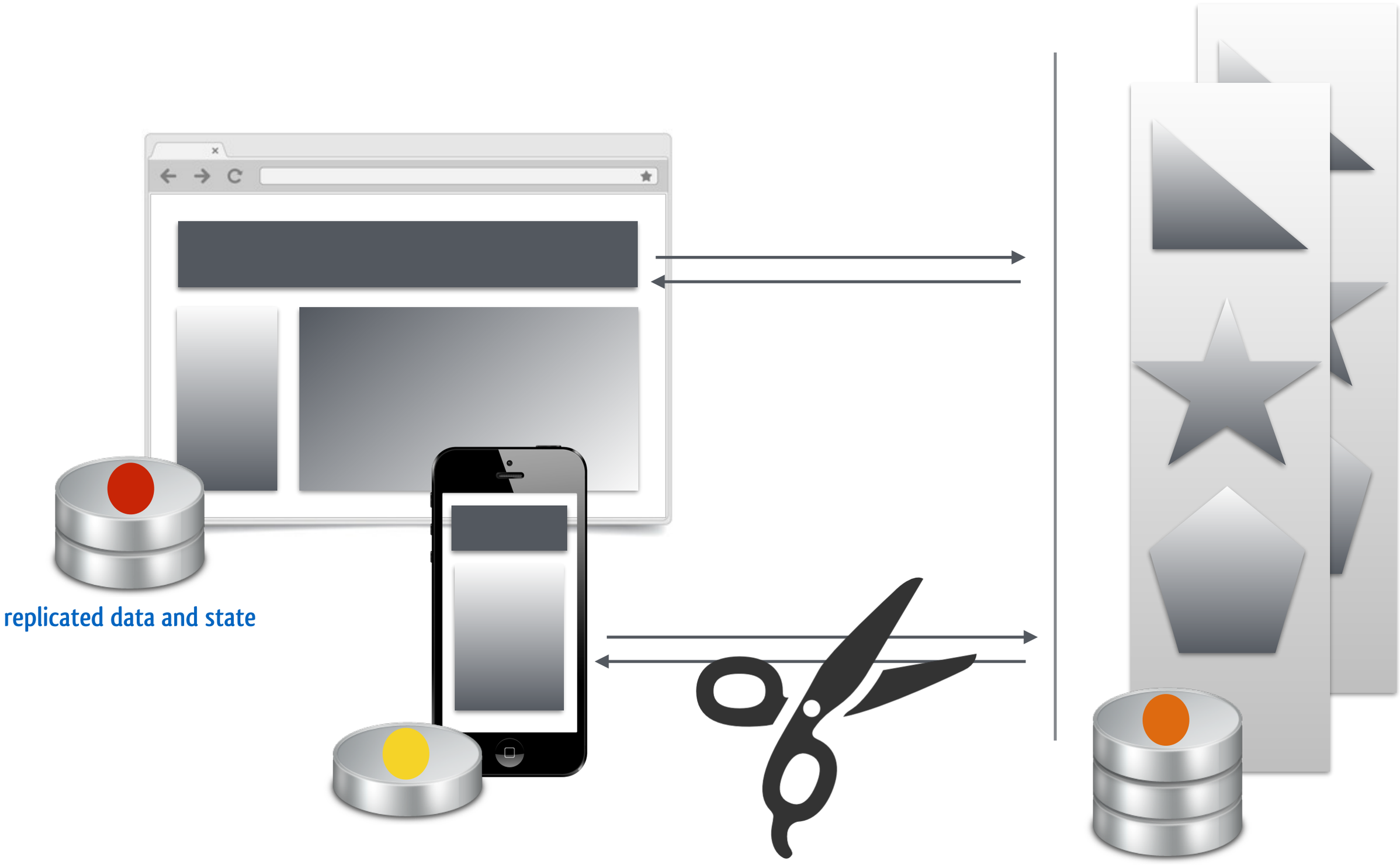
Evolution of web application architectures



application distributed vertically across tiers

single-page application

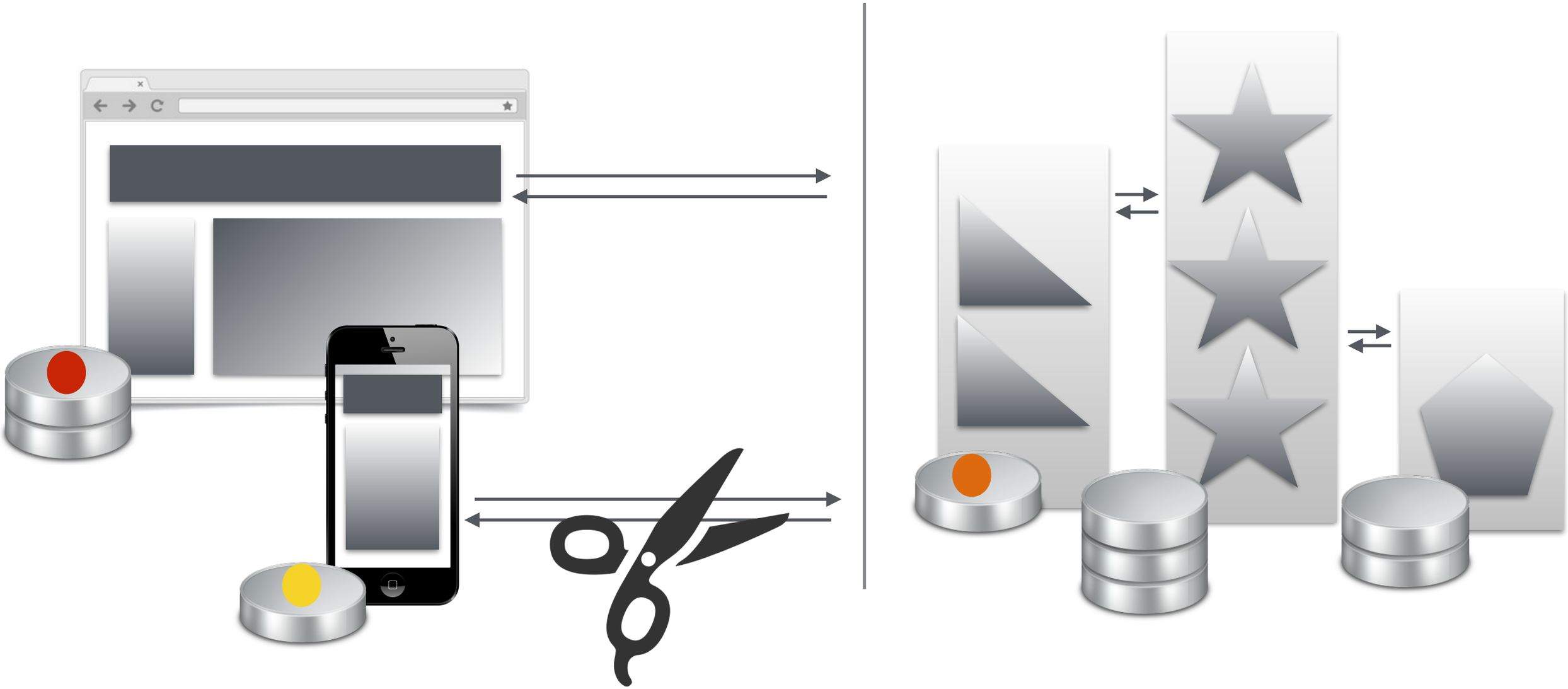
Evolution of web application architectures



rich internet application

Evolution of web application architectures

application distributed horizontally
between instances of the same tier



µ-services on server tier

Evolution of web application architectures

multi-page application

single-page application

application logic and state is increasingly distributed

- vertically across different tiers
  offline and desktop-like functionality
- horizontally between instances of same tier
  collaborative functionality and scalability

rich internet application

µ-services on server tier

Easier said than done

application logic and state is increasingly distributed
- vertically across different tiers
  offline and desktop-like functionality
- horizontally between instances of same tier
  collaborative functionality and scalability

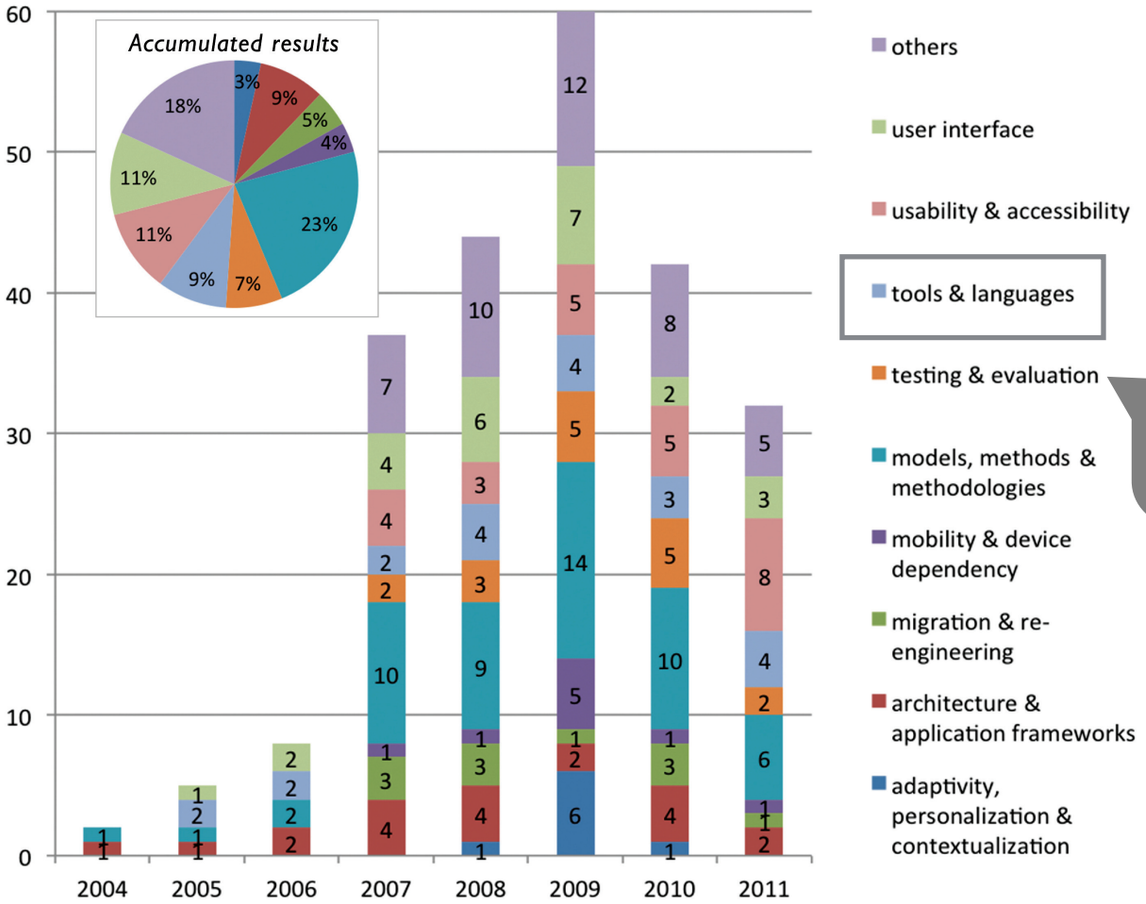
essential complexity

rich internet applications are distributed programs
how to maintain consistency of replicated and shared state?
how to ensure security of replicated and shared state?

accidental complexity

mastering and reconciling a myriad of tier-specific technology

Research is needed



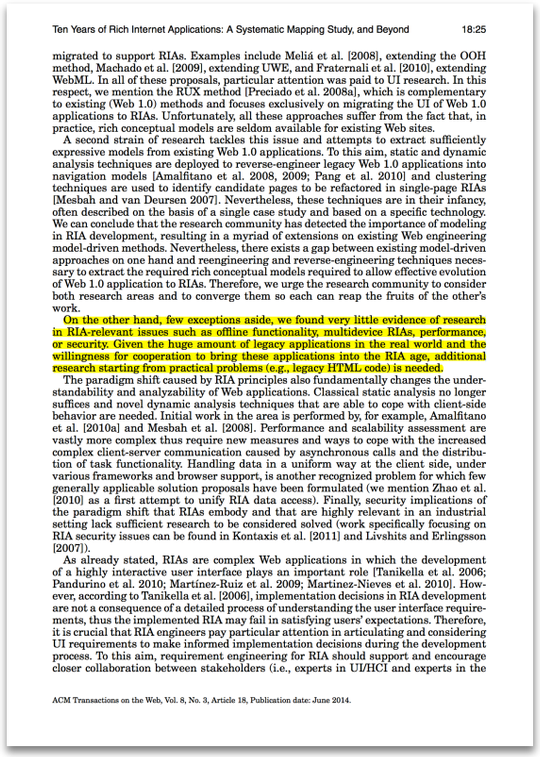
133 publications from 2002-2011

"Ten years of Rich Internet Applications: A systematic mapping study, and beyond."

[Casteleyn, Garrigós and Mazón, 2014]

only 9%, even though these are the concrete means for building a RIA!

"Few exceptions aside, we found very little evidence of research in RIA-relevant issues such as offline functionality, multidevice RIAs, performance, or security. Given the huge amount of legacy applications in the real world and the willingness for cooperation to bring these applications into the RIA age, additional research starting from practical problems (e.g., legacy HTML code) is needed."



Ten Years of Rich Internet Applications: A Systematic Mapping Study, and Beyond 18:25

migrated to support RIAs. Examples include Meliá et al. [2008], extending the OOH method, Machado et al. [2009], extending UWE, and Fraternali et al. [2010], extending WebML. In all of these proposals, particular attention was paid to UI research. In this respect, we mention the RUX method [Preciado et al. 2008a], which is complementary to existing (Web 1.0) methods and focuses exclusively on migrating the UI of Web 1.0 applications to RIAs. Unfortunately, all these approaches suffer from the fact that, in practice, rich conceptual models are seldom available for existing Web sites.

A second strain of research tackles this issue and attempts to extract sufficiently expressive models from existing Web 1.0 applications. To this aim, static and dynamic analysis techniques are deployed to reverse-engineer legacy Web 1.0 applications into navigation models [Amalfitano et al. 2008, 2009; Pang et al. 2010] and clustering techniques are used to identify candidate pages to be refactored in single-page RIAs [Mesbah and van Deursen 2007]. Nevertheless, these techniques are in their infancy, often described on the basis of a single case study and based on a specific technology. We can conclude that the research community has detected the importance of modeling in RIA development, resulting in a myriad of extensions on existing Web engineering model-driven methods. Nevertheless, there exists a gap between existing model-driven approaches on one hand and reengineering and reverse-engineering techniques necessary to extract the required rich conceptual models required to allow effective evolution of Web 1.0 application to RIAs. Therefore, we urge the research community to consider both research areas and to converge them so each can reap the fruits of the other's work.

On the other hand, few exceptions aside, we found very little evidence of research in RIA-relevant issues such as offline functionality, multidevice RIAs, performance, or security. Given the huge amount of legacy applications in the real world and the willingness for cooperation to bring these applications into the RIA age, additional research starting from practical problems (e.g., legacy HTML code) is needed.

The paradigm shift caused by RIA principles also fundamentally changes the understandability and analyzability of Web applications. Classical static analysis no longer suffices and novel dynamic analysis techniques that are able to cope with client-side behavior are needed. Initial work in the area is performed by, for example, Amalfitano et al. [2010a] and Mesbah et al. [2008]. Performance and scalability assessment are vastly more complex thus require new measures and ways to cope with the increased complex client-server communication caused by asynchronous calls and the distribution of task functionality. Handling data in a uniform way at the client side, under various frameworks and browser support, is another recognized problem for which few generally applicable solution proposals have been formulated (we mention Zhao et al. [2010] as a first attempt to unify RIA data access). Finally, security implications of the paradigm shift that RIAs embody and that are highly relevant in an industrial setting lack sufficient research to be considered solved (work specifically focusing on RIA security issues can be found in Kontaxis et al. [2011] and Livshits and Erlingsson [2007]).

As already stated, RIAs are complex Web applications in which the development of a highly interactive user interface plays an important role [Tanikella et al. 2006; Pandurino et al. 2010; Martínez-Ruiz et al. 2009; Martinez-Nieves et al. 2010]. However, according to Tanikella et al. [2006], implementation decisions in RIA development are not a consequence of a detailed process of understanding the user interface requirements, thus the implemented RIA may fail in satisfying users' expectations. Therefore, it is crucial that RIA engineers pay particular attention in articulating and considering UI requirements to make informed implementation decisions during the development process. To this aim, requirement engineering for RIA should support and encourage closer collaboration between stakeholders (i.e., experts in UI/HCI and experts in the

ACM Transactions on the Web, Vol. 8, No. 3, Article 18, Publication date: June 2014.

The Vision of Tearless

tierless programming

enabling technologies

Multi-tier Functional Reactive Programming for the Web

Bob Reynders
iMinds - Distrinet, KU Leuven
bob.reynders@student.kuleuven.be

Dominique Devriese Frank Piessens
iMinds - Distrinet, KU Leuven
{firstname.lastname}@cs.kuleuven.be

Abstract

The development of rob
applications is challeng
deal with multiple progr
events, propagating data
servers, data consistency
for (partly) addressing th
Two relevant ones are (1)
tional reactive programmi
languages support the de
a single language, and hi
to distribution. FRP offe
event-driven programmi
able. However, existing
languages exploit the ben
for example by restrictin
We propose *multi-tier*
to writing web applicatio
multi-tier languages, and
sum of its parts. In multi
server and client togethe
of behaviors (signals) an
where the boundary betw
make our approach more
its potential, this paper p
plementation of a multi-
programming language Si
DSL that makes Scala us
allows us to present initi
multi-tier FRP approach
periment with possible a
Concretely, we show po
exposing client identity
loading clients with the

Towards Tierless Web Development without Tierless Languages

Laure Philips* Coen De Roover‡* Tom Van Cutsem* Wolfgang De Meuter*
* Software Languages Lab, Vrije Universiteit Brussel, Belgium
‡ Software Engineering Laboratory, Osaka University, Japan
lphilips, cderoove, tvcutsem, wdmeuter @vub.ac.be

Abstract

Tierless programming languages enable developing the typical server, client and database tiers of a web application as a single mono-linguistic program. This development style is in stark contrast to the current practice which requires combining multiple technologies and programming languages. A myriad of tierless programming languages has already been proposed, often featuring a JavaScript-like syntax. Instead of introducing yet another, we advocate that it should be possible to develop tierless web applications in existing general-purpose languages. This not only reduces the complexity that developers are exposed to, but also precludes the need for new development tools. We concretize this novel approach to tierless programming by discussing requirements on its future instantiations. We explore the design space of the program analysis for determining and the program transformation for realizing the tier split respectively. The former corresponds to new adaptations of an old familiar, program slicing, for tier splitting. The latter includes several strategies for handling cross-tier function calls and data accesses. Using a prototype instantiation for JavaScript, we demonstrate the feasibility of our approach on an example web application. We conclude with a discussion of open questions and challenges for future research.

Categories and Subject Descriptors D.3.2 [*Programming Languages*]: Concurrent, distributed, and parallel languages; D.2.11 [*Software Architectures*]: Patterns (client/server)

General Terms Languages, Design

Keywords Tier splitting, Program slicing, Tierless Programming, JavaScript

1. Introduction

Contemporary web development has become complex. There is an increasing demand for interactive features, collaboration between clients, support for offline functionality, etc. Realizing such advanced features in a traditional three-tier architecture requires developers to select and master a myriad of technologies. Each tier comes with its own technology stack. Examples include a query language for the database tier, PHP or Java for the server tier, and a combination of JavaScript, HTML and CSS for the client tier —often augmented with cross-tier technology for asynchronous communication such as Ajax and jQuery. It is up to the programmer to combine and align the different technology stacks. This might not only require a lot, but also rather complex glue code for contemporary web applications. For instance, to ensure the different data models of each tier are kept in sync.

Tierless programming languages aim to reduce this complexity. They enable developing a web application as a single mono-linguistic application, which renders its development akin to that of a desktop application. A preprocessor or the runtime of these languages realizes a split into a client, server and sometimes a database tier, where communication between the different tiers is handled transparently. The dynamically-typed functional language Hop [10] is an early example. It discerns code destined for the server and client tier based on developer-provided annotations at the level of individual expressions. The statically-typed functional languages Links [2] and Opa[1] require such annotations at the level of complete functions.

These approaches to tierless web development require an investment in novel and perhaps esoteric programming languages. More importantly, they require developers to annotate code meticulously with tier splitting information —a time-consuming and often error-prone process. This is particularly problematic given the general lack of tool support for these languages. Developers are on their own as far as understanding, testing, validating, debugging and refactoring tierless web applications is concerned. We therefore advocate to develop tierless web applications in a general-

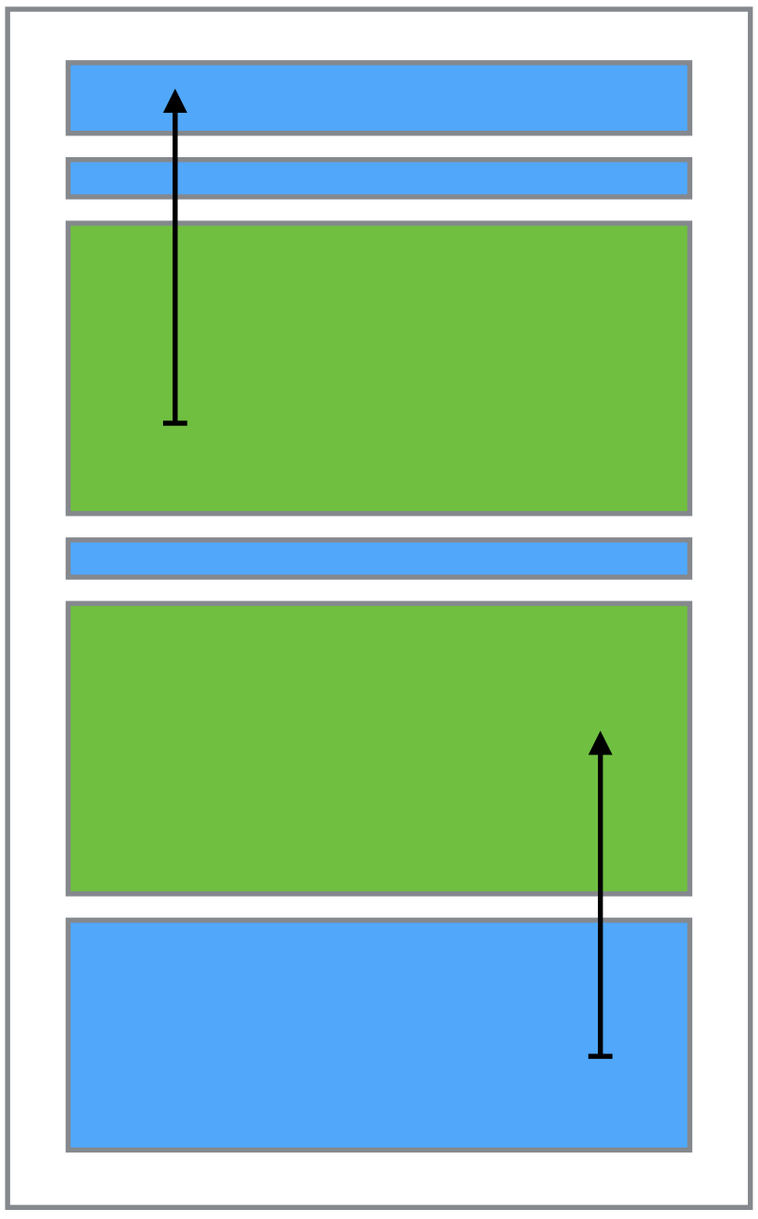
[1] `http://www.opalang.org`

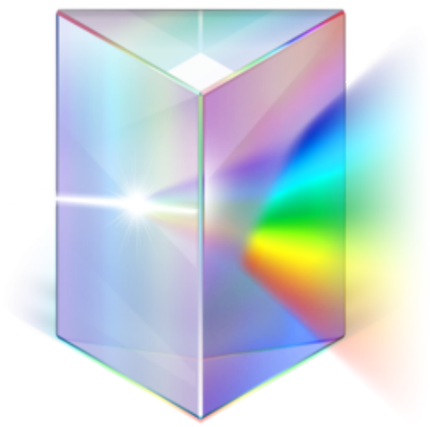
both published at “International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software” (ONWARD14)

“The Tearless project envisions a future in which multi-tier applications are developed, tested and maintained as a single artefact that spans all tiers.

This tierless programming relieves developers of vertical and horizontal distribution concerns, while ensuring the consistency and security of shared logic and state.”

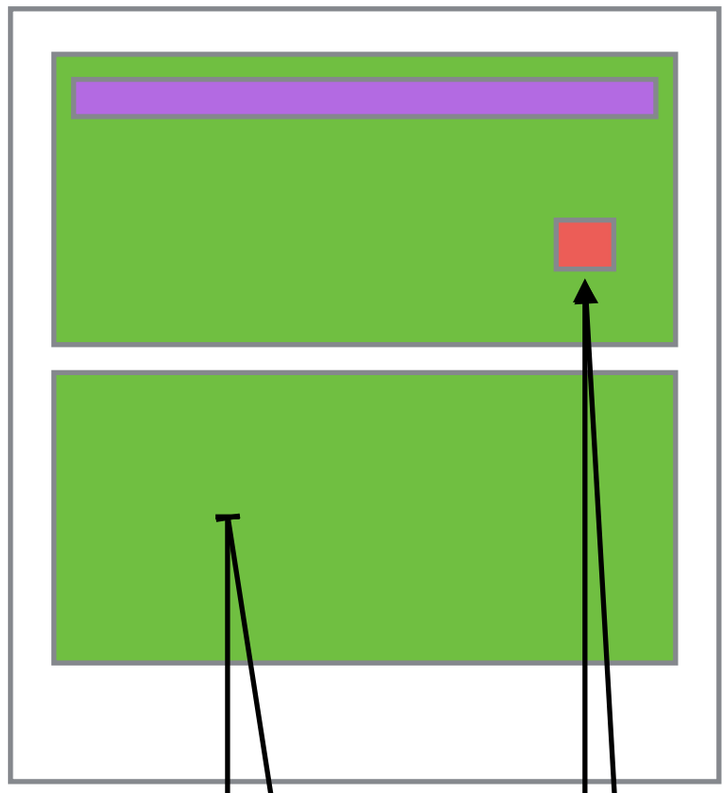
tierless.js



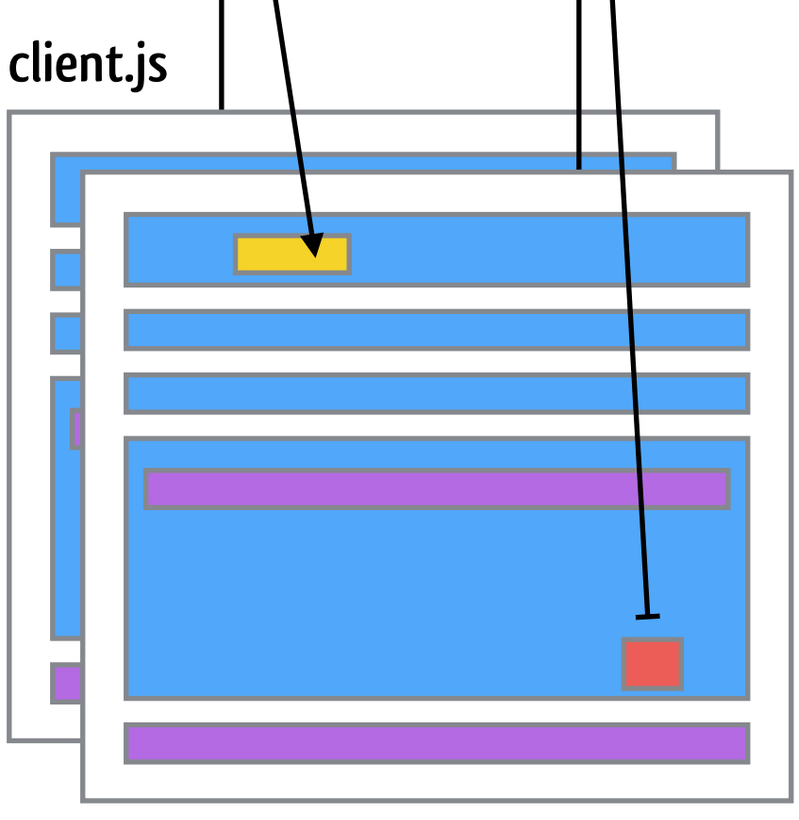


tier splitting

server.js



client.js



server-specific code

client-specific code

generated code implementing distribution concerns

generated code implementing consistency concerns

generated code implementing security concerns

Your JavaScript code: Snippets

```
39
40          ids.push(user_id);
41          names.push(name);
42
43          /*@broadcast*/
44          hear(name + ' joined');
45
46          return user_id;
47      }
48
49      function changeUser(user_id, newName){
50          var idPos = ids.indexOf(user_id);
51
52          if(names.indexOf(newName) >= 0){
53              throw new userExistsError('Userna
```

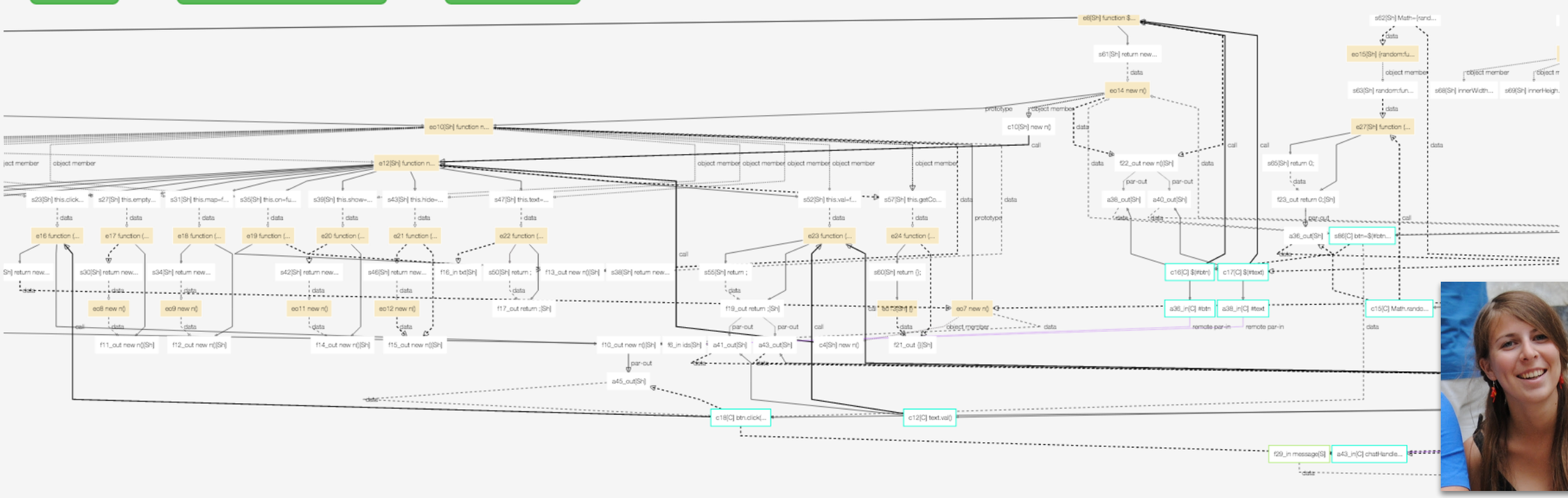
Sliced code: client | server | setup

```
2  var name;
3  var btn;
4  var text;
5  name = 'user' + Math.random();
6  btn = $('#btn');
7  text = $('#text');
8  btn.click(chatHandler);
9  client.expose({
10     'displayMessage': function (name, message
11         text.val(name + ':said ' + message);
12     }
13 });
14 /* SERVER */
15 server.expose({
16     'broadcast': function (name, message, ca
```

STiP | Jipda | Esprima | JSLint

Eval or Transform CPS or Tier split

based on a sophisticated analysis of your code



The Vision of Tearless

tierless programming

enabling technologies

"We will realize our long-term vision gradually through enabling technologies and accompanying tooling."

libraries

for implementing distribution concerns
for implementing consistency concerns
for implementing security concerns

tools

migration assistant for existing code
end-to-end debugger
end-to-end security monitor

Example of library + tooling: consistency

library

prototype JS implementation of Cloud Types [Burckhardt et al, 2012]

CRDTs transposed to a cloud setting

An open implementation of Cloud Types for the Web

Tim Coppieters
Vrije Universiteit Brussel
tcoppiet@vub.ac.be

Laure Philips
Vrije Universiteit Brussel
lphilips@vub.ac.be

Tom Van Cutsem
Vrije Universiteit Brussel
tvcutsem@vub.ac.be

Wolfgang De Meuter
Vrije Universiteit Brussel
wdmeuter@vub.ac.be

Abstract

Cloud Types is an interesting EC model that closely integrates its consistency model with the programming language, rendering EC programming more feasible for the average programmer. We have created an open implementation of the model in a JavaScript end-to-end implementation that can act both as an experimental platform for researchers and as an open EC model for the web. Furthermore we are experimenting with additions to the model such as an RDBM-like authorization system in combination with Views in order to give more control over the state and to allow partitioning.

Categories and Subject Descriptors D.1.3 [*Concurrent Programming*]: Distributed Programming

Keywords Cloud Types, JavaScript, Web Applications, Authorization, Eventual Consistency

1. Introduction

During the last few years first attempts have been made to present easier Eventual Consistency (EC) programming models by integrating it more closely with the programming language. Cloud Types[1] is one of the most recent models that takes such an approach. It allows developers to create an EC application (providing both server and client replication, and thus offline availability of the data) by using automatically replicated data types with commutative operations (as also introduced by CRDT[2]) or last-writer semantics and strong consistency whenever required. While the programming model restricts the developer to certain EC choices (e.g. no conflict detection/resolution), it does provide a more understandable environment for the developers to create EC applications.

2. Open Web Implementation

We believe such an approach is the way to go to make EC programming more feasible. More intelligent language abstractions that cover the complex problems of EC will help and attract more developers to the EC community. In order to further progress our research in that direction, it is key to experiment with and build on previous work. Although the mentioned Cloud Types model is already utilized in TouchDevelop, a closed-source Microsoft product supported by the creators of Cloud Types, it was not yet operationalized in such way that researchers/developers can experiment with it, hands on.

Therefore, we decided to implement the complete model in a JavaScript end-to-end library approach[3]. In short, it allows you to declare the replicated data types using a Node.js server and use the EC data on the client-side (e.g. in your browser) with another JS library, using WebSockets for communication. The library has a clear API, an extensive documentation and an implementation that is kept as close to the original description of the language model as possible. This (1) allows researchers to better understand and experiment with the model and (2) makes the model available in a widely used environment, namely the web platform.

3. More Control

We are currently using this implementation to examine the boundaries of the model for web application programming. Thus far we encountered two main problems in order to use it for our day-to-day web applications. First, there is no way to control what happens with the shared data, i.e. we do not want everyone to see, create, update and delete all data. Second, there is no way to partition the state in such way that the client can decide to only access a part of the data. By allowing to retrieve and synchronize only the necessary data on demand, it becomes feasible to encode larger data sets

PaPEC '14, April 13 - 16, 2014, Amsterdam, Netherlands.

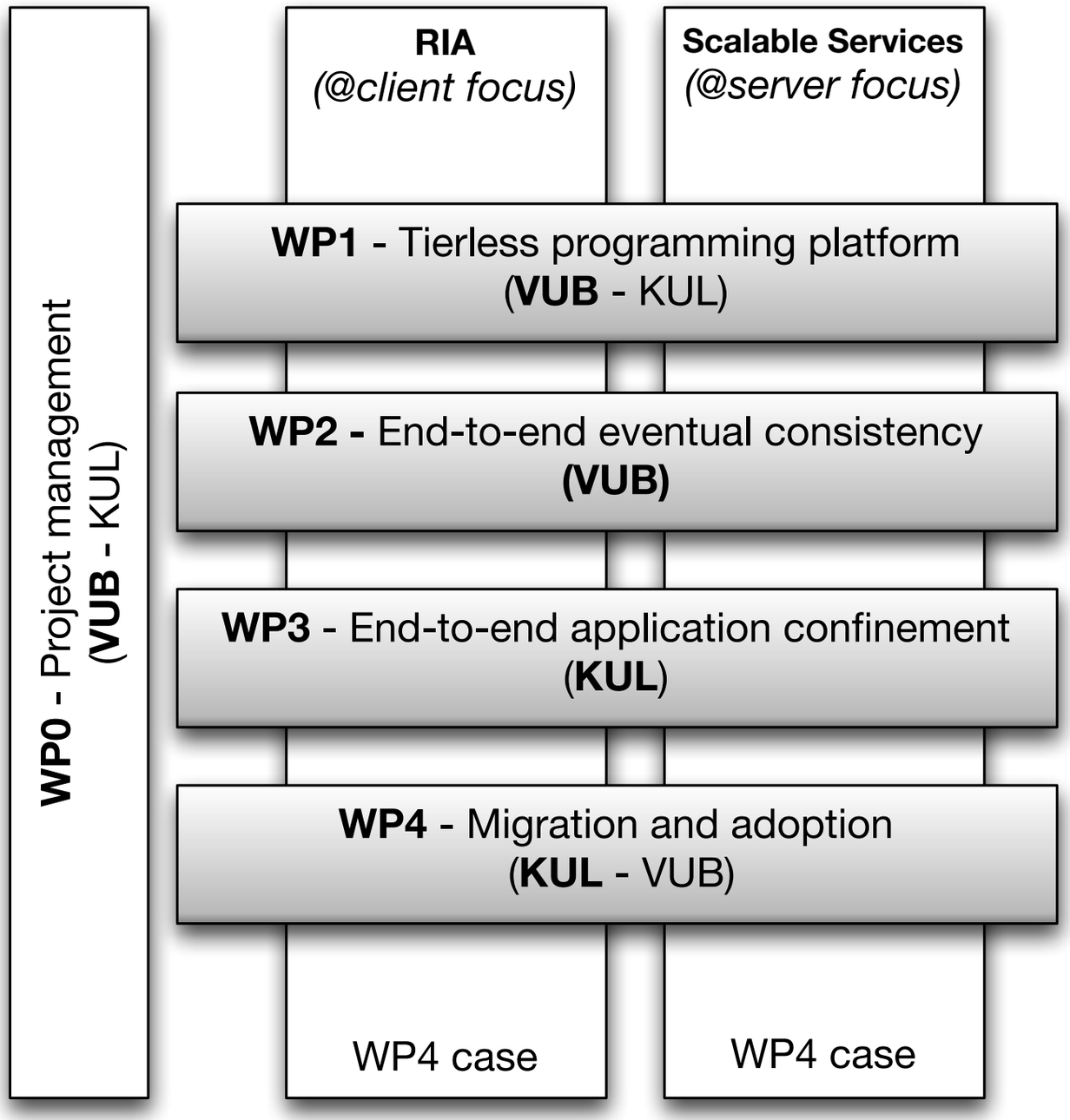
http://dx.doi.org/10.1145/2596631.2596640

tool

how to determine which data needs to be replicated for a desired offline service level?

Major research questions per work package

"Does our technology help stakeholders make the leap towards rich client tiers with offline and collaborative functionality, and server tiers capable of handling global scales?"



to what extent can tier splitting be automated?
how to debug application logic across tiers, and third-party code?

how to support user-defined cloud types?
how to determine which data needs to be replicated for a desired offline service level?

how to secure assets distributed across tiers, and isolate them from untrusted code on the same tier?
how to track policy violations across tiers?

how to help developers renovate existing code?
how to help developers identify assets to secure?

demonstrations of technology on 4 synthetic cases

Valorization opportunities

still up-to-date?

	PaaS Providers	**SaaS Providers**	**Software-Intensive**	**Consultants**
WP1: Tierless Programming Platform	A.-Lucent Bell	HealthConnect YesPlan Xenit Up-nxt	MobileVikings IOS Int.	AE
WP2: End-to-end Eventual Consistency	A.-Lucent Bell	HealthConnect YesPlan Up-nxt XAOP	MobileVikings IOS Int.	AE
WP3: End-to-end Application Confinement	IS4U	Xenit Up-nxt XAOP		NVISO
WP4: Migration and Adoption	IS4U			NVISO AE

based on your letters of intent

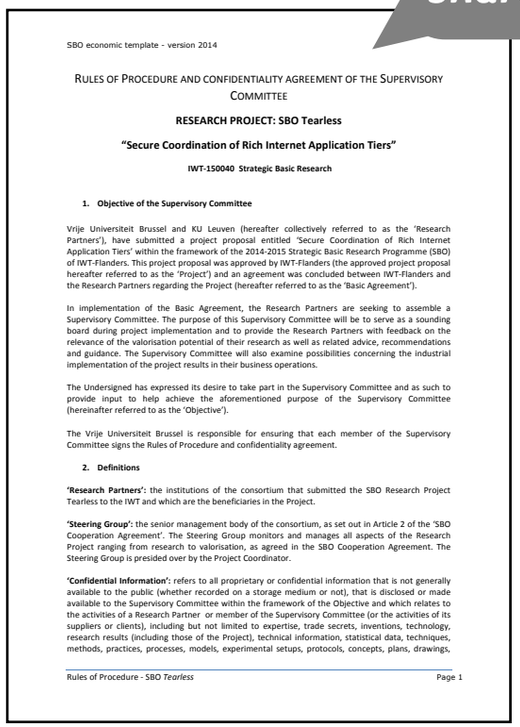
Administrativa

pilot studies
business cases
problems

- Prof. Elisa Gonzalez Boix: distribution
- Prof. Coen De Roover: program analysis
- Prof. Frank Piessens: security

technology
demonstrators
follow-up projects

confidentiality of shared information



SBO economic template - version 2014

RULES OF PROCEDURE AND CONFIDENTIALITY AGREEMENT OF THE SUPERVISORY COMMITTEE

RESEARCH PROJECT: SBO Tearless

"Secure Coordination of Rich Internet Application Tiers"

IWT-150040 Strategic Basic Research

1. Objective of the Supervisory Committee

Vrije Universiteit Brussel and KU Leuven (hereafter collectively referred to as the 'Research Partners'), have submitted a project proposal entitled 'Secure Coordination of Rich Internet Application Tiers' within the framework of the 2014-2015 Strategic Basic Research Programme (SBO) of IWT-Flanders. This project proposal was approved by IWT-Flanders (the approved project proposal hereafter referred to as the 'Project') and an agreement was concluded between IWT-Flanders and the Research Partners regarding the Project (hereafter referred to as the 'Basic Agreement').

In implementation of the Basic Agreement, the Research Partners are seeking to assemble a Supervisory Committee. The purpose of this Supervisory Committee will be to serve as a sounding board during project implementation and to provide the Research Partners with feedback on the relevance of the valorisation potential of their research as well as related advice, recommendations and guidance. The Supervisory Committee will also examine possibilities concerning the industrial implementation of the project results in their business operations.

The Undersigned has expressed its desire to take part in the Supervisory Committee and as such to provide input to help achieve the aforementioned purpose of the Supervisory Committee (hereinafter referred to as the 'Objective').

The Vrije Universiteit Brussel is responsible for ensuring that each member of the Supervisory Committee signs the Rules of Procedure and confidentiality agreement.

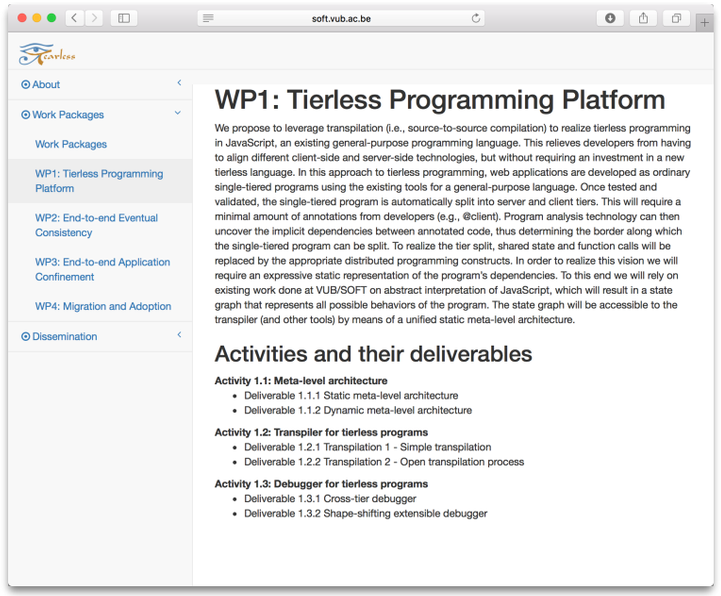
2. Definitions

'Research Partners': the institutions of the consortium that submitted the SBO Research Project Tearless to the IWT and which are the beneficiaries in the Project.

'Steering Group': the senior management body of the consortium, as set out in Article 2 of the 'SBO Cooperation Agreement'. The Steering Group monitors and manages all aspects of the Research Project ranging from research to valorisation, as agreed in the SBO Cooperation Agreement. The Steering Group is presided over by the Project Coordinator.

'Confidential Information': refers to all proprietary or confidential information that is not generally available to the public (whether recorded on a storage medium or not), that is disclosed or made available to the Supervisory Committee within the framework of the Objective and which relates to the activities of a Research Partner or member of the Supervisory Committee (or the activities of its suppliers or clients), including but not limited to expertise, trade secrets, inventions, technology, research results (including those of the Project), technical information, statistical data, techniques, methods, practices, processes, models, experimental setups, protocols, concepts, plans, drawings,

Rules of Procedure - SBO *Tearless* Page 1



we will meet twice a year

rules of procedure

soft.vub.ac.be/tearless/

	2016				2017				2018				2019			
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
WP0				D0.1.1				D0.1.2				D0.1.3			D0.2.1	D0.1.4
WP1		D1.1.2 D1.1.3		D1.2.1		D1.3.1					D1.2.2			D1.3.2		
WP2			D2.1.1			D2.1.2		D2.1.3		D2.2.1		D2.2.2			D2.3	
WP3			D3.2		D3.2.1	D3.1.1		D3.1.3		D3.1.2		D3.3.1		D3.3.2		
WP4				D4.4.1				D4.1.1 D4.4.2		D4.1.2		D4.2 D4.4.3				D4.3 D4.4.4

deliverables will be made available on website

Agenda for the remainder of the meeting

- 15:00 Technical presentation about the state of the art
  - in securing RIAs (Philippe De Ryck, KUL)
  - in implementing offline functionality for RIAs (Joeri De Koster, VUB)
- 16:30 Reception with networking opportunities