IT Solutions with Software Engineering in Practice
Overview of Client Technologies

Course at TU Darmstadt
Alexander Schwartz, 11 May 2015

Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive

TECHNISCHE UNIVERSITÄT DARMSTADT

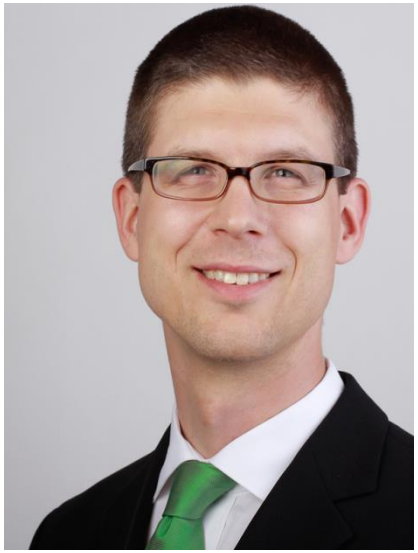
.msg

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Alexander Schwartz

- Studied management science (Betriebswirtschaftslehre) at Philipps-Universität Marburg (DE) and University of Kent (UK)
- 15+ years of experience in web development
- Today: Principal IT Consultant at msg
- Office in Eschborn
- Focus on Java and web technology
- Interests: agile project management, open source, automated tests
- Responsibilities: architecture, technology scouting



@ahus1de

Private

- 39 years old, married, 2 kids
- Hobbies: Family, Geocaching

Sie passen punktgenau bei uns rein!

.consulting .solutions .partnership

.msg

An unserem Karrieretag in der Geschäftsstelle Frankfurt erhalten Sie einen detaillierten Einblick in die Arbeitsmethoden und aktuellen Projekte eines der erfolgreichsten IT-Beratungs- und Systemintegrationsunternehmen in Deutschland.
Darüber hinaus bleibt Raum, sich individuell mit Mitarbeitern aus unseren Branchen zu Einstiegs- und Karrierechancen auszutauschen.

Natürlich sind Sie an diesem Tag unser Gast, Reisekosten werden übernommen.

Wir freuen uns auf Sie!

Anmeldung mit Lebenslauf an: your.future@msg-systems.com

.msg

Karrieretag am 29. Mai 2015, 11 bis 17 Uhr

Exkursion zur msg in Frankfurt

- Projektvorstellungen zu aktuellen IT-Themen
- Berufseinstieg bei der msg – ein persönlicher Erfahrungsbericht
- Berufsbild Informatiker am Beispiel der msg
- Möglichkeit zum fachlichen Austausch

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Client technology interfaces with the user

Characteristics of client technology:

- Presents information to the user on a screen
- Provides controls for interaction
- Provides data validation feedback
- Integrates with backend services for logic
- Provides access to remote storage for information

In scope for today:

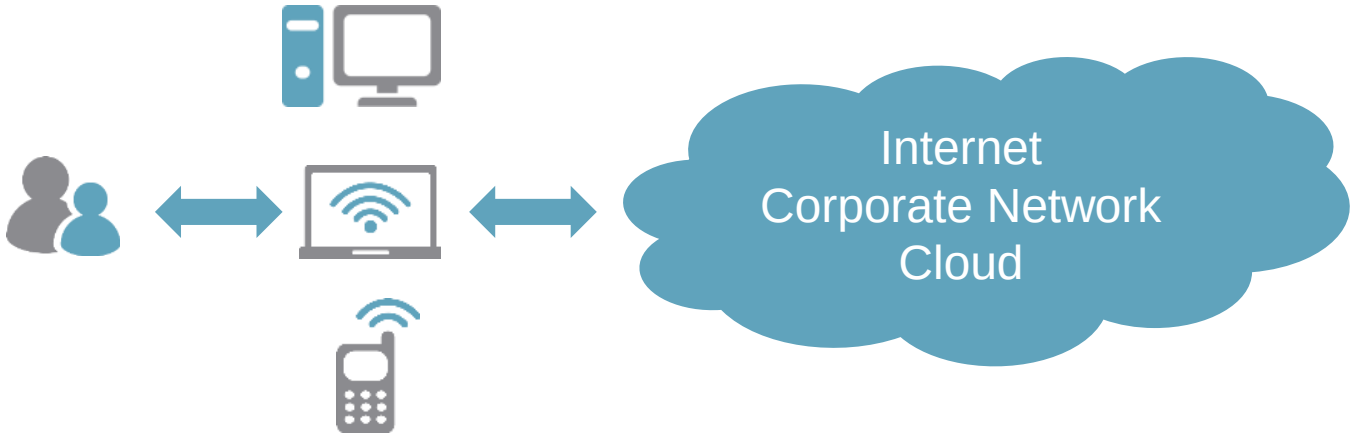
- Desktop
- Mobile (phone/tablet)

Out of scope for today:

- Interactive voice response systems (IVR)
- Embedded systems
- Command lines

Devices have different characteristics

- Specific in operating systems (OS)
- Specific in screen size
- Specific in input methods (mouse, keyboard, touch)
- Specific extras (GPS, printer, scanner, camera, …)



Evaluating different client technologies

- Systematic analysis of aspects
- Use an established standard: ISO 250xx quality model

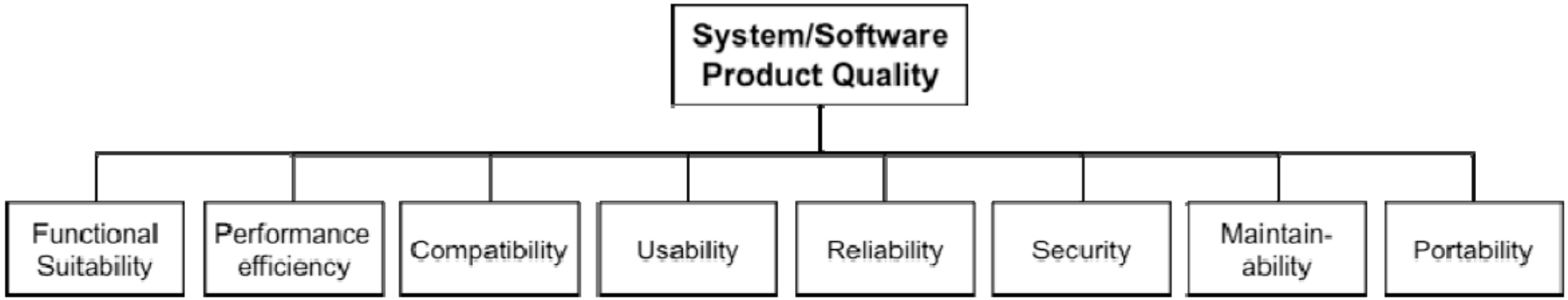
- **User centric**: quality in use model
- **Product centric**: product quality model

Quality in use is applicable to human-computer system

- Centred on the perception of the software by the end user

Product quality is applicable to characterize a software product

- Here: used to characterize a client technology



Sample Application for today: a spread sheet application

- First implementation: VisiCalc (originally released on Apple II, 1979)
- Claim: “turned microcomputers into a serious business tool” *

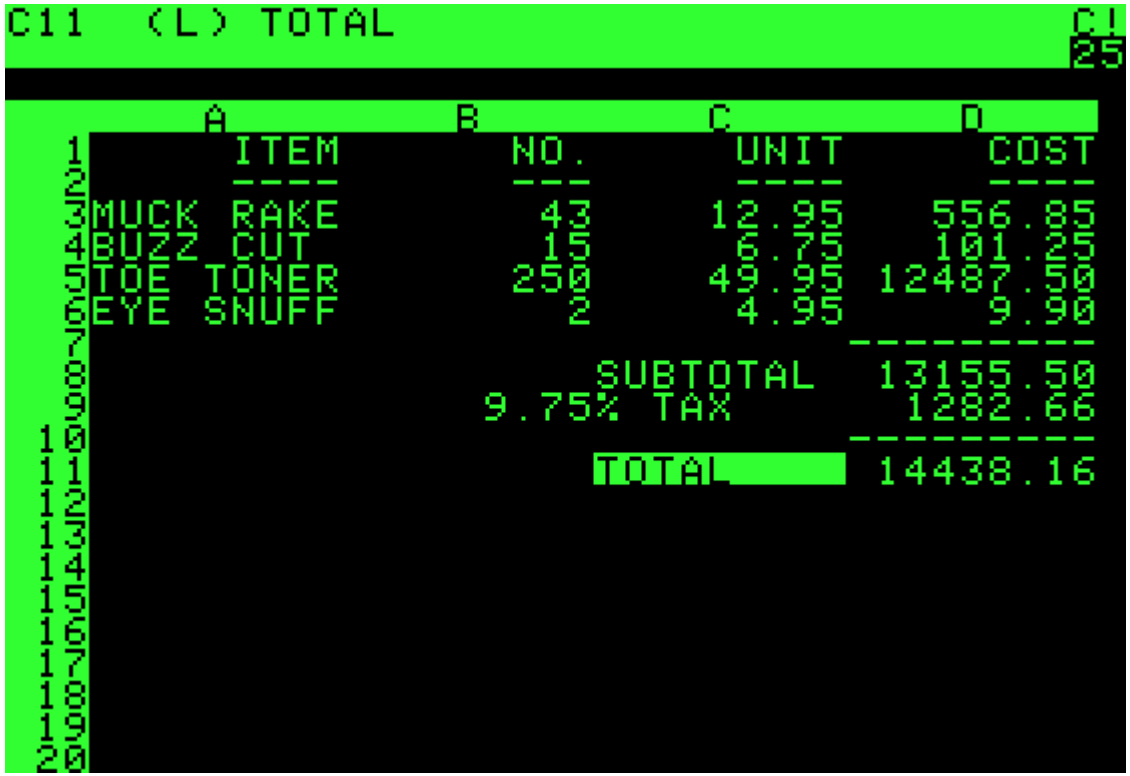
Key functionality

- Simple UI (at first glance)
- Immediate re-calculation when input values change

- High-fidelity graphics when you need them
- Embeddable within text documents and presentations
- Available on all types of devices
- Local storage and cloud storage with real-time collaboration

** http://en.wikipedia.org/wiki/Visicalc*

Screen shot of VisiCalc

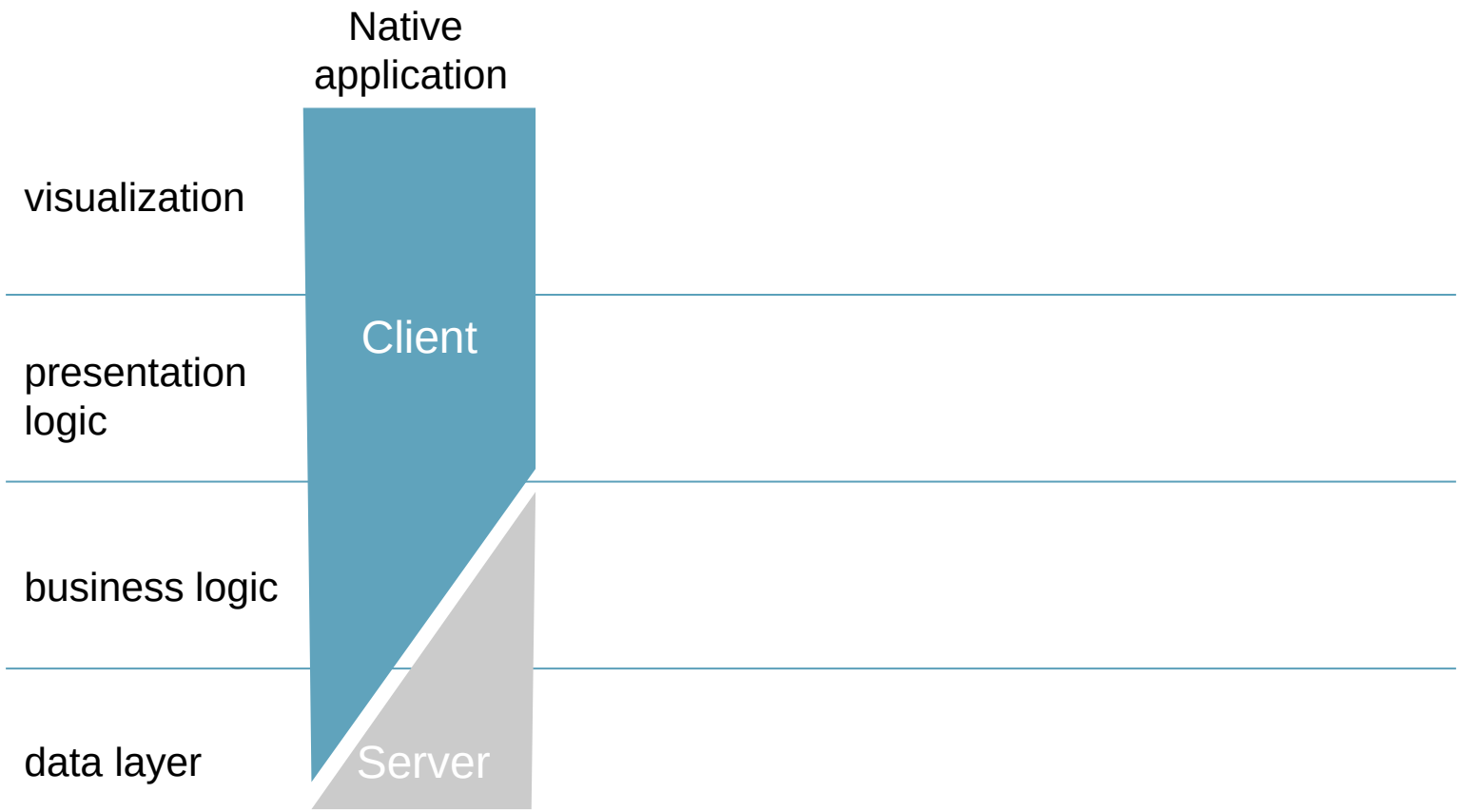


Source: apple2history.org

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

A native application can implement all functionality on the client



Major Platforms for native applications

Desktop

- Microsoft Windows:
  .NET w/ Windows Presentation Foundation (WPF) *
- Apple Mac OS X:
  Objective C w/ Cocoa

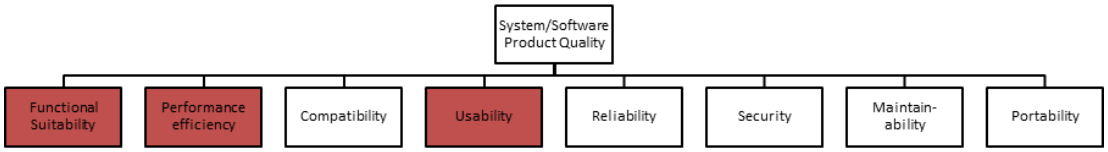
Mobile

- Apple iOS:
  Objective C w/ Cocoa Touch
- Google Android:
  Java w/ Android SDK *

* Also C/C++ are available for native apps (or parts of them)

Native Applications can use all functionality provided by the underlying operating system

- **functional completeness**
  … covers all the specified tasks and user objectives …
- **user interface aesthetics**
  … pleasing and satisfying interaction …
- **time behaviour**
  … response and processing times …
- **learnability**
  … user … learning to use the product …



Microsoft Windows / Desktop:
.NET w/ Windows Presentation Foundation (WPF)

- Can create native applications for Windows platform
- Code is execute on Common Language Runtime (CLR)
- Part of .NET 3.0 (since 2006)
- Latest release 4.5.x (2014)

- All native features of Windows are available to the developer
- Click once installer / auto update functionality
- Windows Store as part of Windows 8
- Plugin Framework (MEF) since 3.5 for runtime extensibility

- Development needs to be done on (latest) Windows

Apple Mac OS X / Desktop:
Objective C w/ Cocoa

- Initial Release 2001
- Latest release 10.10.3 (2015)

- All native features of Mac OS X are available to the developer
- App-Store for applications, including auto-update

- Development needs to be done on (latest) Mac OS X

Apple iOS / Mobile:
Objective C w/ Cocoa touch

- Initial Release 2007
- Latest release 8.3 (2015)

- All native features of iOS are available to the developer
- App-Store for applications, including auto-update

- Development needs to be done on Mac OS X

Google Android / Mobile:
Java w/ Android SDK

- Initial Release 2008
- Latest release 5.1 (2015)

- All native features of Android are available to the developer
- App-Store for applications, including auto-update

- Development can be done on Linux, Mac OS and Windows with Android SDK based around IntelliJ IDE

Downsides of native applications

- Installability depends on local (admin) rights
- Installations needs to be managed by IT operations
- Regular updates need to be rolled out

- Low portability to another operating system

- Technology for front end development might not match technology for back end development (.NET / Objective C / Java)

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Portable Applications run on more than on Operating System

- **portability**
  … can be transferred from one hardware, software … to another
- **reusability**
  … asset can be used in more than one system …
- **maintainability**
  … effectiveness and efficiency … system can be modified …

- Libraries can be re-used on different OS platforms
- Serving multiple platforms with a single release

Desktop:

- Java / Eclipse RCP

Mobile:

- Phonegap / Apache Cordova

Desktop: Java / Eclipse RCP

- Plugin Architecture built upon OSGi for runtime extendibility
- Auto-update mechanisms
- Re-using Java-Know-How that exists i.e. from server side development
- Supported development and target platforms: Windows, Mac OS X, Linux

- Look and Feel is different from a native application
- Access to some native features like tray icons is possible
- Interaction with native application possible, i.e. using OLE (Object Linking and Embedding)

.msg

Mobile: Phone Gap / Apache Cordova

- Use HTML + JavaScript to build you app (w/ jQuery mobile or Dodo Mobile)
- Create an application that is wrapped by native code
- Access to mobile device features via JavaScript abstraction layer

- Build Service in the Cloud to eliminate specific development environment
- Access to 7+ mobile platforms, including Android, iOS, Windows Phone

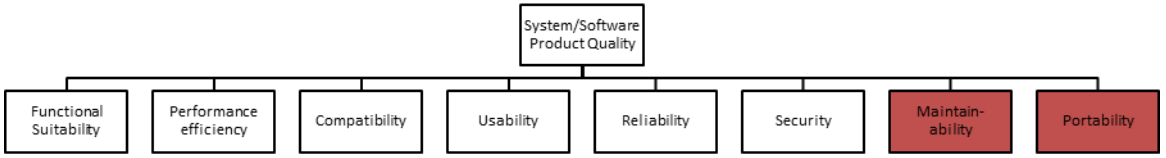
- Look and Feel is (slightly) different from a native application (you can try to adopt you layout via CSS)
- Access to common features of devices
- Slower than a native application

AGENDA

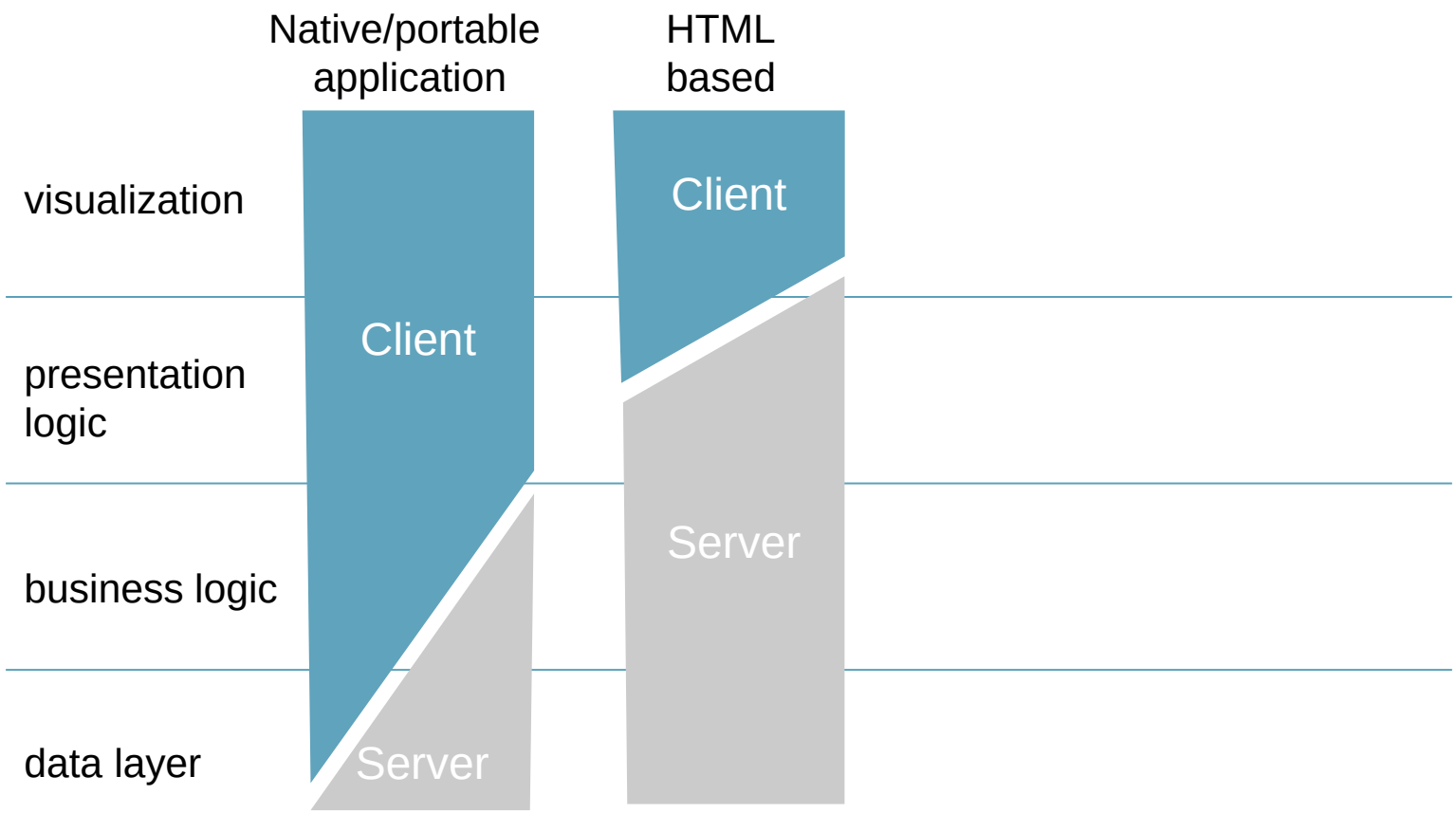
1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Using a browser with server side logic

- **installability**
  … effectiveness and efficiency [to be] installed/uninstalled …
- **replaceability**
  … new version … different product …
- **operability**
  … easy to operate and control …
- **maintainability**
  … effectiveness and efficiency … system can be modified …



Using a browser with server side logic



HTML based server side frameworks

	Java	.NET
MVC style	Spring Web MVC	ASP.NET MVC
component style	Java Server Faces (JSF)	ASP.NET

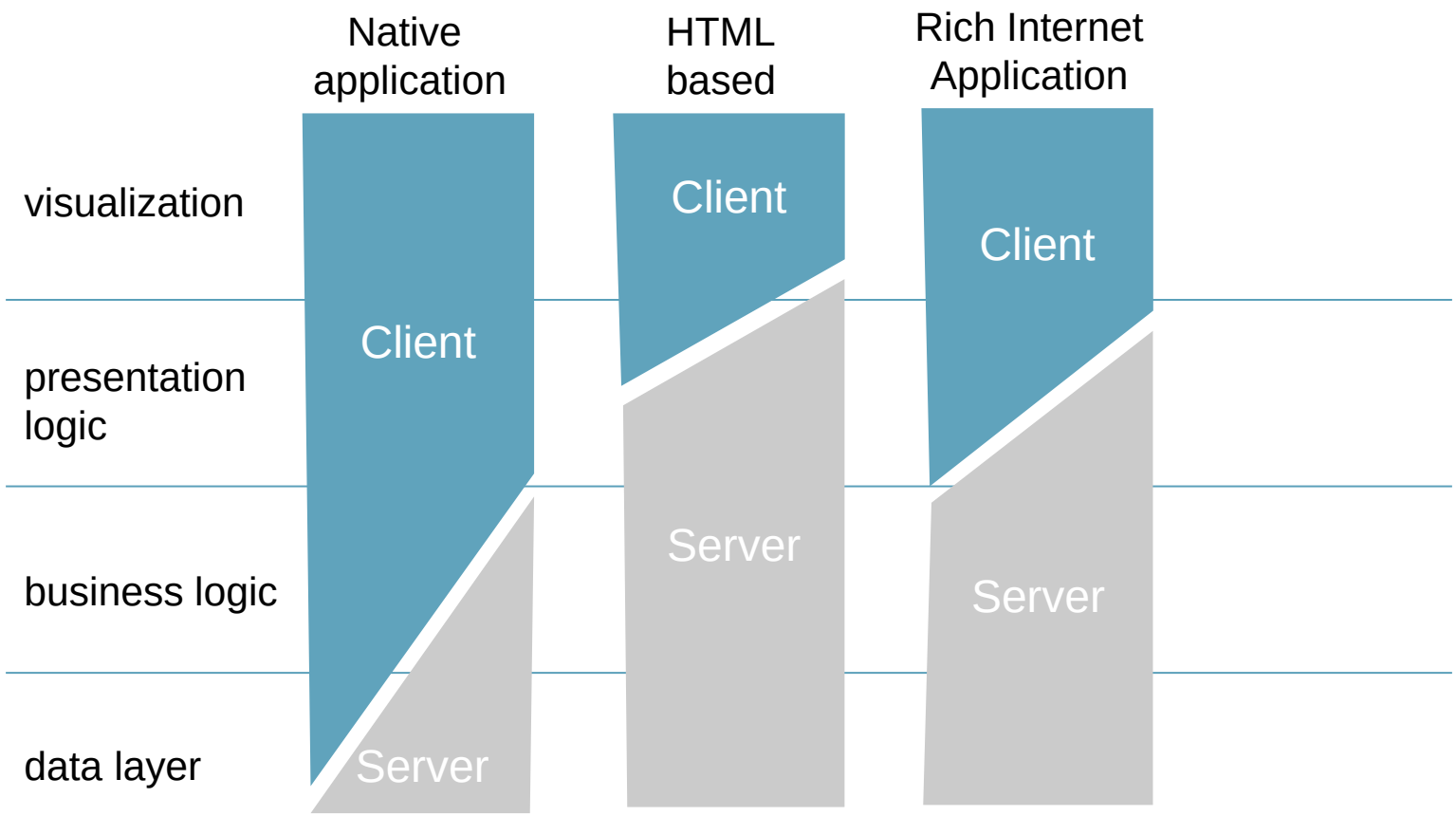
- Minimal use of JavaScript
- Using common browser standards
- Building abstractions on top of HTTP and HTML
- No access to OS specific functionality
- Each user interaction requires feedback from the server

Rich internet applications (RIA)

	Sample Client Technology
AJAX w/ component style	RichFaces, IceFaces, PrimeFaces
Component Based	Google Web Toolkit (GWT)
Browser Plugins	Adobe Flash Microsoft Silverlight

- Heavy use of JavaScript (or browser plugin)
- Building lot's of abstractions on top of HTTP and HTML (GWT cross compiles Java to JavaScript)
- Some user interaction requires feedback from the server

Using a browser with server side logic



Downsides of presented thin client and rich web client solutions

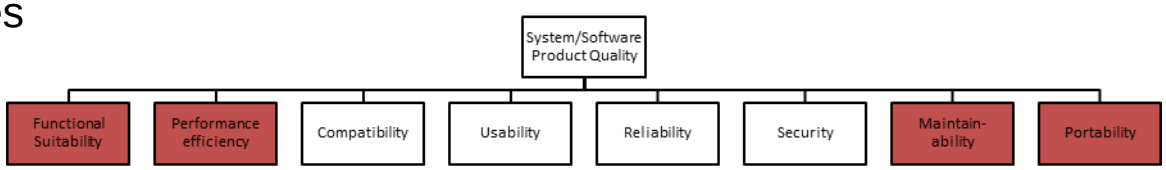
- Reduced user experience due to less interactive technology
- Server side state requires resources per client connected to the system
- Server side state needs to be shared between nodes in a cluster
- Scalability becomes difficult for a high number of users
- Functional completeness problematic when too many levels of abstraction exist
- Browser plugins need to be installed separately
- Browser plugins are famous for security problems

HTML5 + JavaScript as client technology

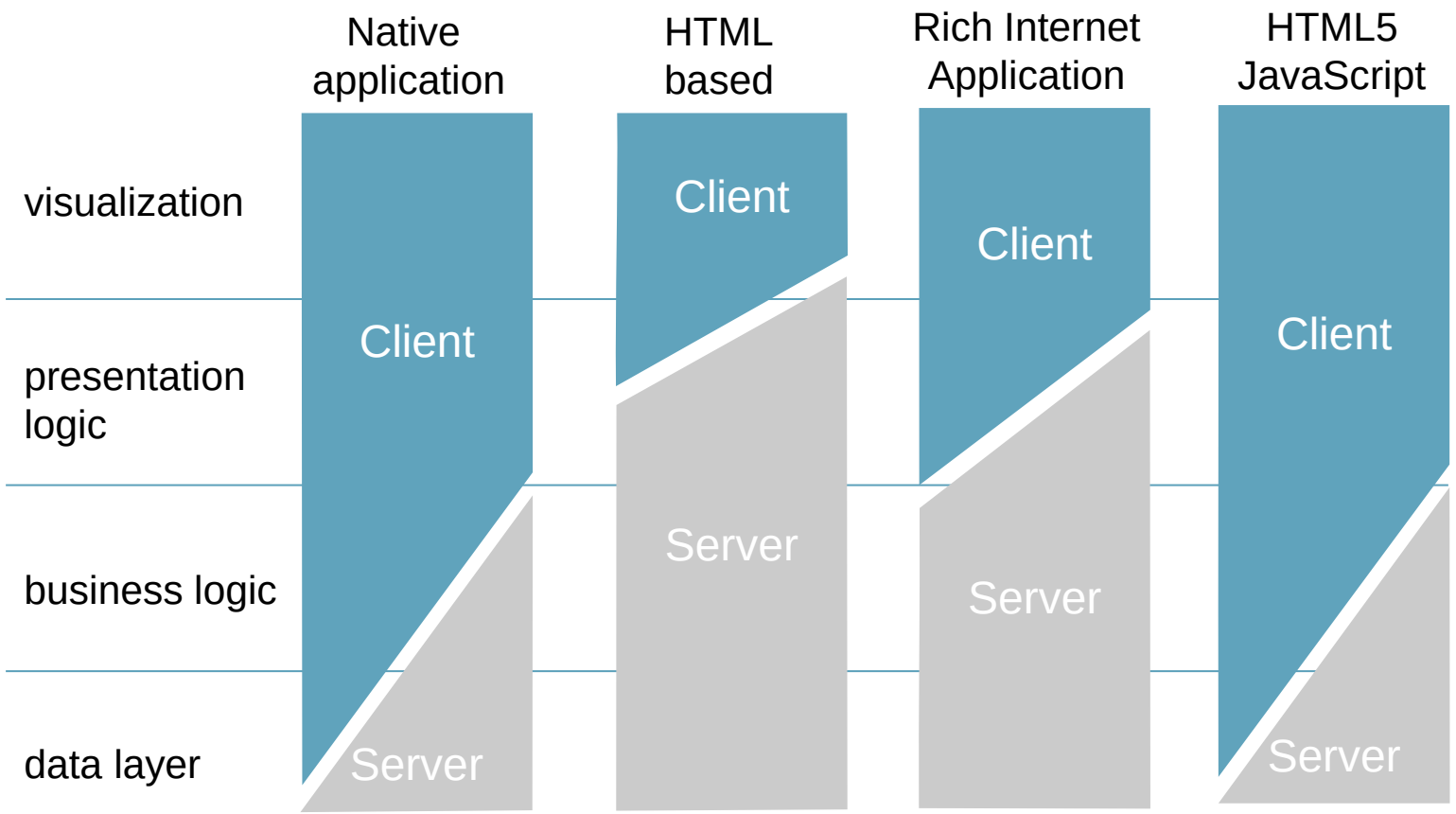
All features of a browser technology plus:

- **capacity**
  … maximum limits of a product …
- **resource utilization**
  … resource used to meet requirements …
- **functional completeness**
  … covers all the specified tasks and user objective …

… when used with REST-ful,
stateless backend services



Using a browser with server side logic



HTML5 + JavaScript timeline

- HTML:
  1990 (Sir Tim Berners-Lee, CERN)
- JavaScript:
  1996 (Brendan Eich, Netscape)
- JavaScript interacting with HTML: Document Object Model (DOM)
  1997 (“intermediate DOM”) … 2004 (“DOM Level 3”)
- JavaScript interacting with the Server: XMLHttpRequest
  2001 (Microsoft / Internet Explorer 6 – but also works with IE 5 as ActiveX)
- Common JavaScript API to manipulate HTML: jQuery
  2006
- Modularization in JavaScript: requireJS
  2010
- Model-View-Binding HTML/JavaScript: Knockout
  2010

(cc) W3C http://www.w3.org/html/logo/

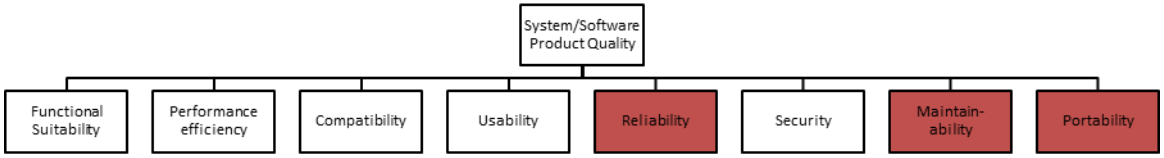
AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Evolution of Client Technology

- **maturity**
  … applied to reliability, but also other quality characteristics …
- **maintainability**
  … corrections, improvements, adoption of the software to changes in the environment …
- **adaptability**
  ... can be adapted for different or evolving hardware, software …

- New frameworks and versions on monthly basis
- Only few get a broader usage and become relevant for enterprise usage
- Some die after lack of vendor or community support



Dead Technology: Struts 1.x

- Java based Web MVC Framework
- Used to be quite popular
- First release: 2000
- Last release: 2008
- Official community statement for “end of life”

- Still existing applications in the enterprise context using Struts 1.x

- Migration path to 2.0 exists – but 2.0 is incompatible

Evolution of a Standard: Java Enterprise Edition / Java Server Faces

- Java Server Faces is part of the Java Enterprise Edition open standard
- First release 1.0: 2004
- Latest release 2.2: 2013
- Old versions still run on new application servers

- Refactoring recommended to use new functionality

- If you used a 3rd party library to achieve functionality that later became standard you'll need to migrate that functionality

Erratic (?) Evolution: HTML and JavaScript

- HTML: Standardized by W3C in different versions
  … now a “living standard” (http://html5.org/)
- JavaScript: Standardized as ISO/IEC 16262
  … but APIs are not part of the standard

After more than 10 years we are able to write modular, responsive, standalone applications

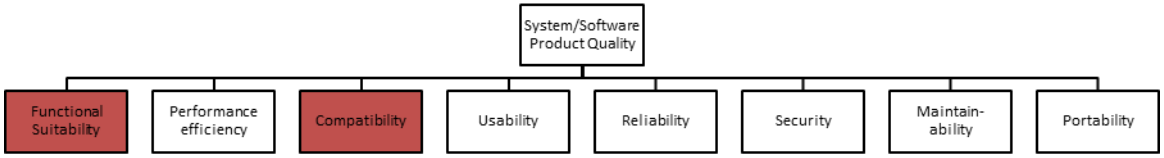
Is this client technology “mature”? Looking at http://caniuse.com/ it is at least fragmented.

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Interoperability of Clients

- **interoperability**
  … can exchange information and use the information …
- **compatibility**
  … can exchange information … sharing the same … environment …
- **functional completeness**
  … covers all specified tasks and user objectives …



Interoperability by Operating System and Client Technology

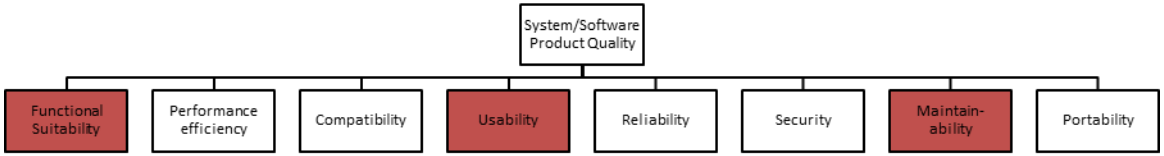
- **Android**
  Intents: publish/subscribe mechanism to receive data and events
- **Windows**
  OLE: specific interactions with applications
  Contracts (since Windows 8): publish/subscribe mechanism to receive data and events
- **HTML / HTML5**
  Links: to forward to another app, optionally passing data
  JavaScript/DOM mash up: “pull” functionality into an app at runtime
- **Eclipse RCP / .NET WPF**
  Plugins that can be installed at runtime

AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Testing Client Technology

- **functional correctness**
  … correct results with the needed degree of precision …
- **usability**
  … achieve specified goals …
- **maintainability**
  … system can be modified by the intended maintainers …
- **testability**
  … test criteria can be established for the system …
  … tests can be performed for [the] criteria …



Elements of automatic testing

- Continuous Integration Build
- Unit Testing
- Automatic Provisioning
- Smoke Tests
- Integration Tests
- Graphical User Interface (GUI) Tests
- Load Tests

- Continuous Integration is available for all technologies
- Automatic Provisioning is the next challenge
- GUI tests are difficult and brittle

Difficulties of GUI testing

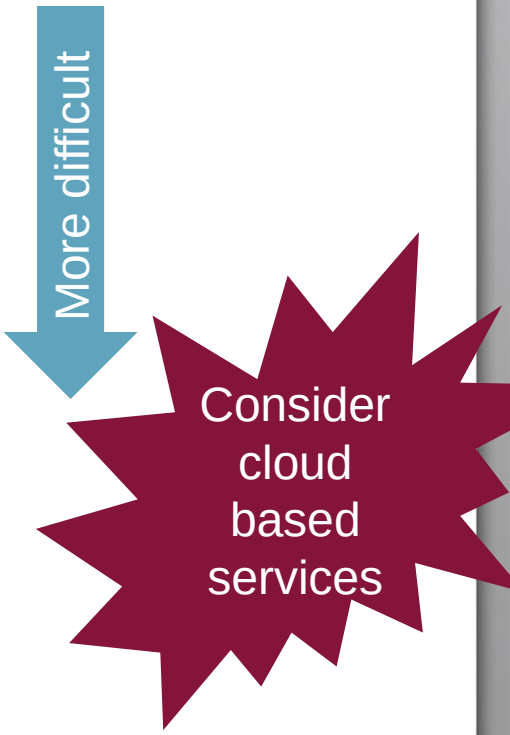
.NET / WPF
- One operating system (few different versions)
- Can be virtualized on very few virtual machines (VMs)

Eclipse RCP
- Several operating systems (few different versions)
- Can be virtualized on few VMs

Web Client / HTML5
- Several operating systems, many browsers
- Can be virtualized utilizing multiple VMs

Android
- Many devices, many screen sizes, many versions
- Can be virtualized – but what about customized Android?

iOS
- Few devices, few screen sizes, few versions
- Difficult to virtualize



AGENDA

1. Introduction
2. Native Applications
3. Portable Applications
4. Thin Clients
5. Evolution
6. Interoperability
7. Testing
8. Choosing a Client Technology

Choosing your Client Technology

- Look at the requirements and prioritize scenarios *
- Estimate the life time of the application
- Double check if the life time of the client is different from the server
- Evaluate using a catalogue like ISO 250xx
- Find out what criteria are relevant in the given situation

- Talk to people who have used the technology
- Build a prototype
- Don’t forget test automation

- Write down why you have chosen the client technology **
- (Maybe) prepare the client technology to be exchangeable

* see [ATAM]

** see [arc42], chapter 9

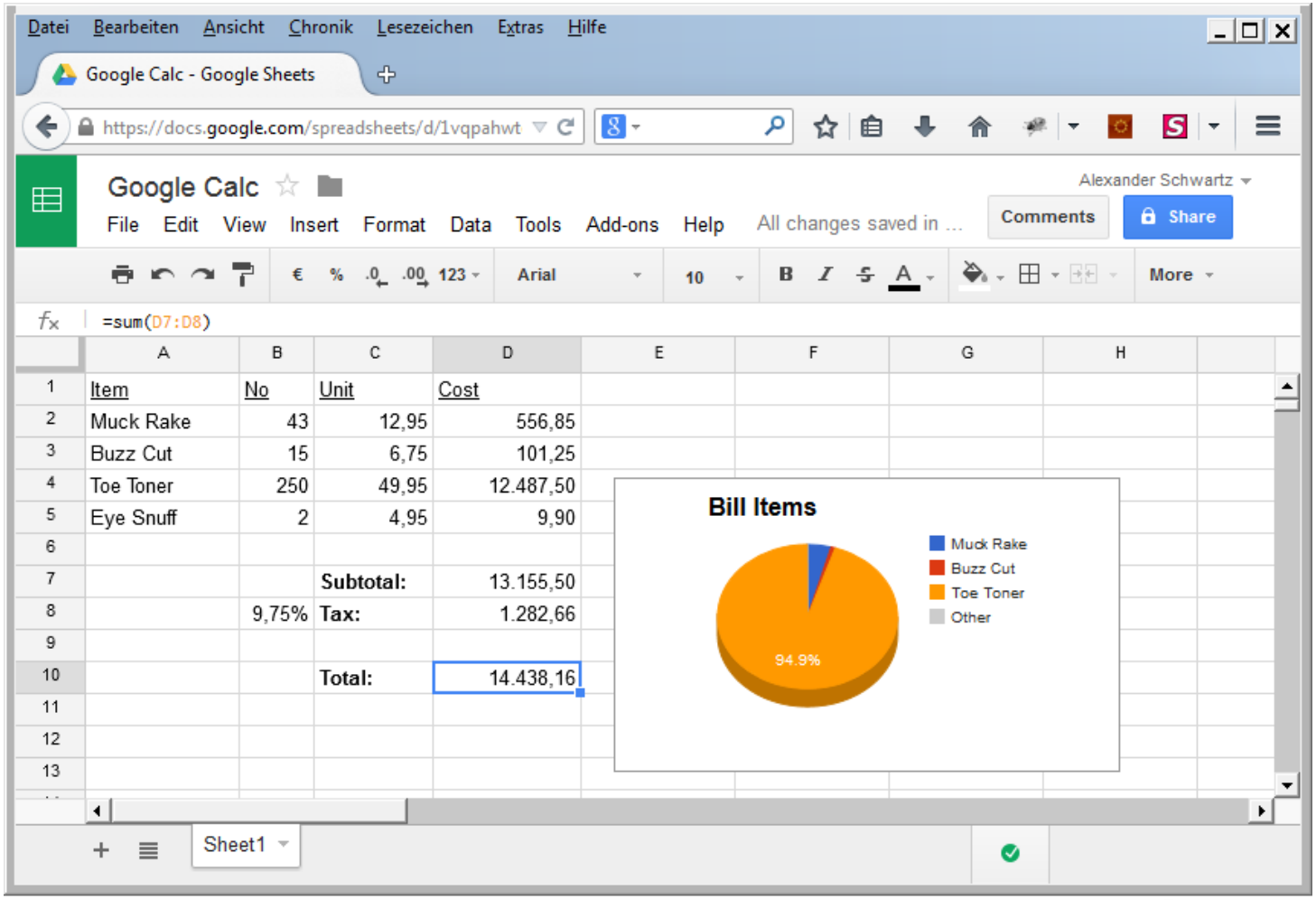
Sweet Spot for a new project

For a new project a technology stack might look like this:

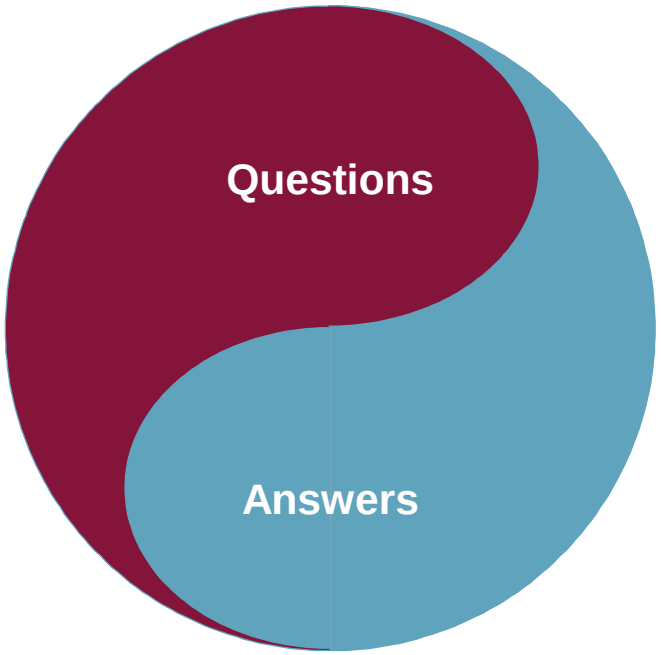
- Choose HTML5/JS as client side technology
  (but don't get lost in too many Java Script frameworks)
- Rendering HTML5 on the server and use HTML links to integrate with other components to keep things simple
- Keep the logic and storage of information on the server
  (probably with a REST service API)
- Consider a mobile first approach
  (would that affect only the layout, or would it require offline capabilities as well?)

(but adopt if this doesn't meet your business requirements)

Screen Shot of Google Calc



Choosing your Client Technology



References

[ISO 25010]
ISO/IEC 25010:2011, Systems and software engineering — Systems and software Product Quality Requirements and Evaluation (SQuaRE) — System and software quality models

[Quamoco]
Quamoco, Software-Qualitätsstandard für Deutschland
http://www.quamoco.de/webmodel/
(free accessible details of Product Quality Model according to ISO 25010)

[ATAM]
ATAM: Method for Architecture Evaluation
http://www.sei.cmu.edu/reports/00tr004.pdf

[arc42]
Template to document system architectures
http://www.arc42.org

Thank you for your attention

msg systems ag

Alexander Schwartz

Mobile: +49 171 5625767
E-Mail: alexander.schwartz@msg-systems.com

Mergenthalerallee 73-75
65760 Eschborn
www.msg-systems.com

.consulting .solutions .partnership