

# Lecture 13

## Software business, Software startup

14.4.2014

Week	Lecture	Exercise
10.3	Quality in general; Quality management systems	Patterns
17.3	Dependable and safety-critical systems	ISO9001
24.3	Work planning; effort estimation	Code inspections
31.3	Version and configuration management	<i>Effort estimation</i>
7.4	Role of software architecture software evolution	
14.4	Software business, software start-ups, IPR	Break
21.4	Easter	Break
28.4	Specifics of some domains, e.g. web system and/or embedded and real time systems	?
5.5	Last lecture; summary; recap for exam	?
15.5	EXAM 9-12	

# Content of the lecture

- Regap of last weeks lecture
- Business aspects of software
  - Those who attended basic course fall 2013 might find some slides familiar
- Software Startups
  - Output from recent seminar

# Definition

(<http://csse.usc.edu/csse/TECHRPTS/1995/usccse95-500/usccse95-500.pdf>)

A software system architecture comprises

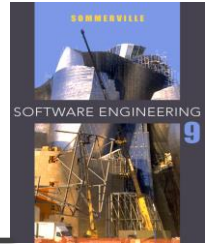
- A collection of software and system components, connections, and constraints.
- A collection of system stakeholders-need statements.  
(a collection of system requirements)
- a rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system requirements

# Two (three) levels

Sommerville

- **Architecture in the small** is concerned with the architecture of individual programs. At this level, we are concerned with the way that an individual program is decomposed into components.
- **Architecture in the large** is concerned with the architecture of complex enterprise systems that include other systems, programs, and program components. These enterprise systems are distributed over different computers, which may be owned and managed by different companies.
- **Enterprise architecture** Enterprise architecture is the organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the company's operating model. The operating model is the desired state of business process integration and business process standardization for delivering goods and services to customers.[]

# 4 + 1 view model of software architecture



- ✧ A logical view, which shows the key abstractions in the system as objects or object classes.
- ✧ A process view, which shows how, at run-time, the system is composed of interacting processes.
- ✧ A development view, which shows how the software is decomposed for development.
  - Haikala&Mikkonen: toteutusnäköymä; implementation view
- ✧ A physical view, which shows the system hardware and how software components are distributed across the processors in the system.
  - Haikala&Mikkonen: sijoittelunäköymä (deployment view)
- ✧ Related using use cases or scenarios (+1)

# Stakeholder concerns

<http://www.codingthearchitecture.com/pages/book/role.html>

Stakeholder	Concern
Customer	<ul style="list-style-type: none"><li>• Schedule and budget estimation</li><li>• Feasibility and risk assessment</li><li>• Requirements traceability</li><li>• Progress tracking</li></ul>
User	<ul style="list-style-type: none"><li>• Consistency with requirements and usage scenarios</li><li>• Future requirement growth accommodation</li><li>• Performance, reliability, interoperability, etc.</li></ul>
Architect	<ul style="list-style-type: none"><li>• Requirements traceability</li><li>• Support of tradeoff analyses</li><li>• Completeness, consistency of architecture</li></ul>
Developer	<ul style="list-style-type: none"><li>• Sufficient detail for design</li><li>• Reference for selecting / assembling components</li><li>• Maintain interoperability with existing systems</li></ul>
Maintainer	<ul style="list-style-type: none"><li>• Guidance on software modification</li><li>• Guidance on architecture evolution</li><li>• Maintain interoperability with existing systems</li></ul>

# Conway's law

- organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations
- Conway, Melvin E. (April 1968), "How do Committees Invent?", Datamation **14** (5): 28–31,
  - Reprint available:  
<http://www.melconway.com/research/committees.html>



# **SOFTWARE BUSINESS**

# Background

- The local job-market has changed during last years
- In the past most students were hired by big companies and "business" was a concern of "somebody else"
- Today small companies do most of the hiring
- In small companies everybody need understand something about the business
- Many of you should create a start-up company
- Latest trend in software engineering research have approached business

# TLO-35246 Software Business, 4 cr

## Learning outcomes

- Student understands the basic principles of software business and the special characteristics of software industry. He/she can critically analyze and develop software business models. Student can apply theoretical knowledge and understanding of the software business characteristics to create a solid business plan for a software start-up.

# TLO-35246 Software Business, 4 cr

<b>Core content</b>	<b>Complementary knowledge</b>	<b>Specialist knowledge</b>
1. Software industry	Historical development and status of the industry. Value networks in the industry.	
2. Software business models	SaaS models.	
3. Management and leadership in software business	Leading professionals in software business. Productization and marketing.	
4. Business plan for a software intensive company	Software start ups.	Financing, IPR's.

# A rough categorization

- Software as part of the product
  - Value of software is increasing
- Simple sub-contracting of software resources
- Development of custom software as a project
- "Shrink-wrapped" software
  - Although the delivery channel is changing
- Software as a service (SaaS)
- How about open source?

# Example SaaS: Adobe Creative Cloud

(<http://www.paulpehrson.com/2011/04/11/adobes-new-software-as-a-service-model/>)

Product	Full	Upgrade*	SAAS**	Months to justify initial investment***
Design Premium	\$1899	\$399	\$95	20
Web Premium	\$1799	\$399	\$89	20
Production Premium	\$1699	\$399	\$85	20
Master Collection	\$2599	\$549	\$129	20
Photoshop	\$699	\$199	\$35	20
Illustrator	\$599	\$199	\$29	20

# **CHALLENGE NUMBER 1: BUDGETING**

# Hypothetical example

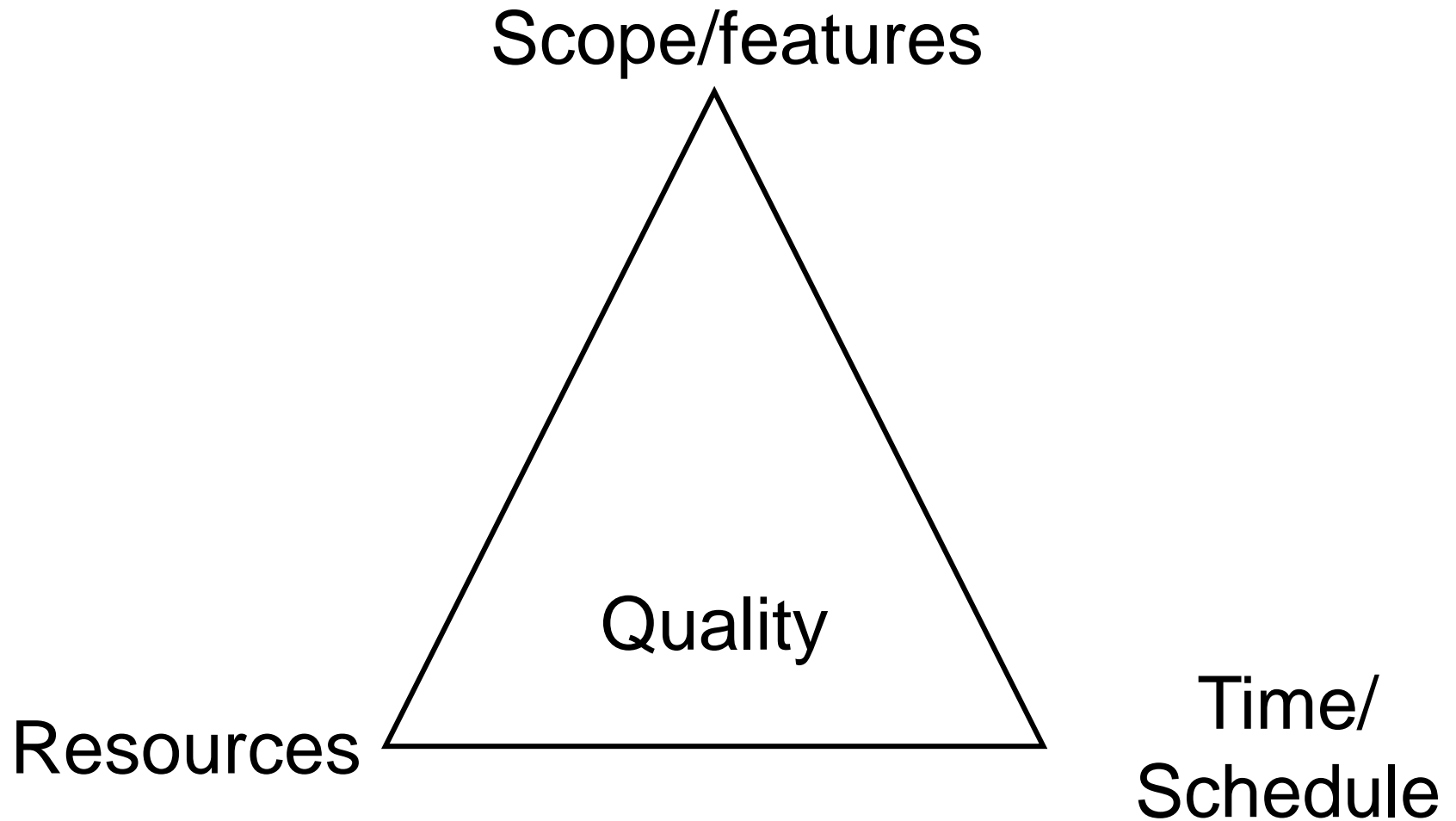
- Software developer company with 10 employees
- Sells programming work by charging developer hours
- Full-time manager, no assisting work force
- Other employees do invoiced work 75% of their worktime
- Rotation-rate about 1 person / year
- Salaries with compulsory indirect costs  
 $1.6 * 3 \text{ k€} * 12 \text{ month} = \sim 58 \text{ k€}$
- Add office, equipment, etc, by multiplying with 1.5  
 $\Rightarrow \sim 86 \text{ k€}$
- 10 persons  $\Rightarrow$  costs of the company are  $\sim 860 \text{ k€/year}$



## ... continues

- 9 people create the income.
- Due to rotation, sick leaves etc we use factor 8:
- $8 \cdot 1700 \cdot 0.75 = \text{about } 10000 \text{ hours to be charged.}$
- Because we need to cover 860k€/year a hour will cost about ~86€ and price of a day is 650€.
- One person year costs ~150k€
  
- Note: does not include any profit  
(Nobody gets a Ferrari)

# Remember the iron triangle



# The "Competition"

## Vendor

- As little as possible
- As expensive as possible

## Maintenance

- As a new project

## New features

- Buy us

## Customer

- As much as possible
- As cheap as possible

## Maintenance

- Belongs to warranty

## New features

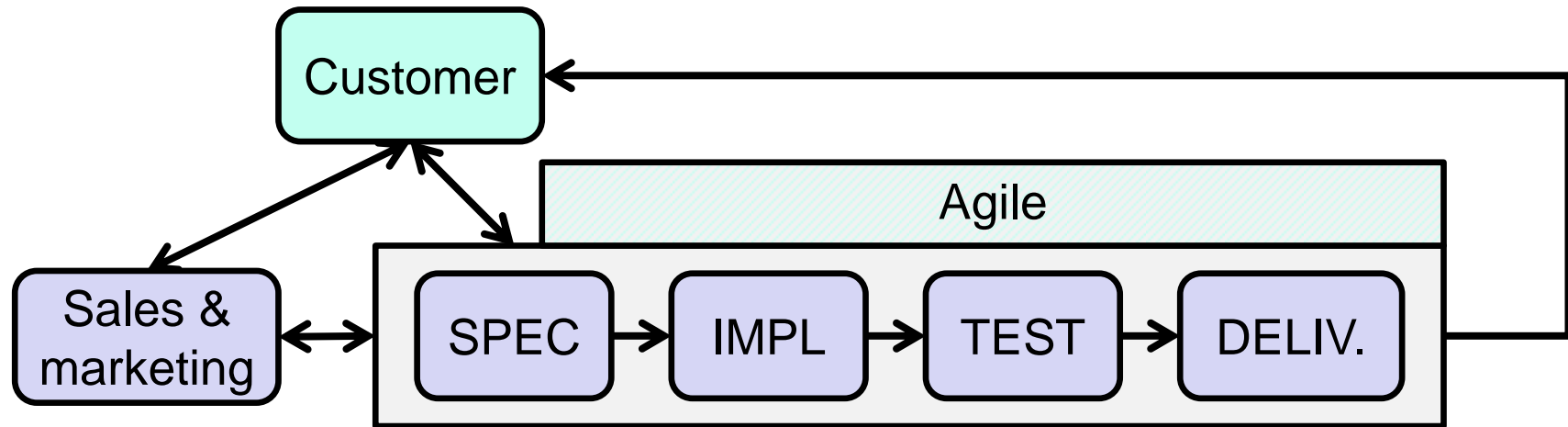
- By anybody

# **CHALLENGE 2**

## **FROM CUSTOM TO PRODUCT**

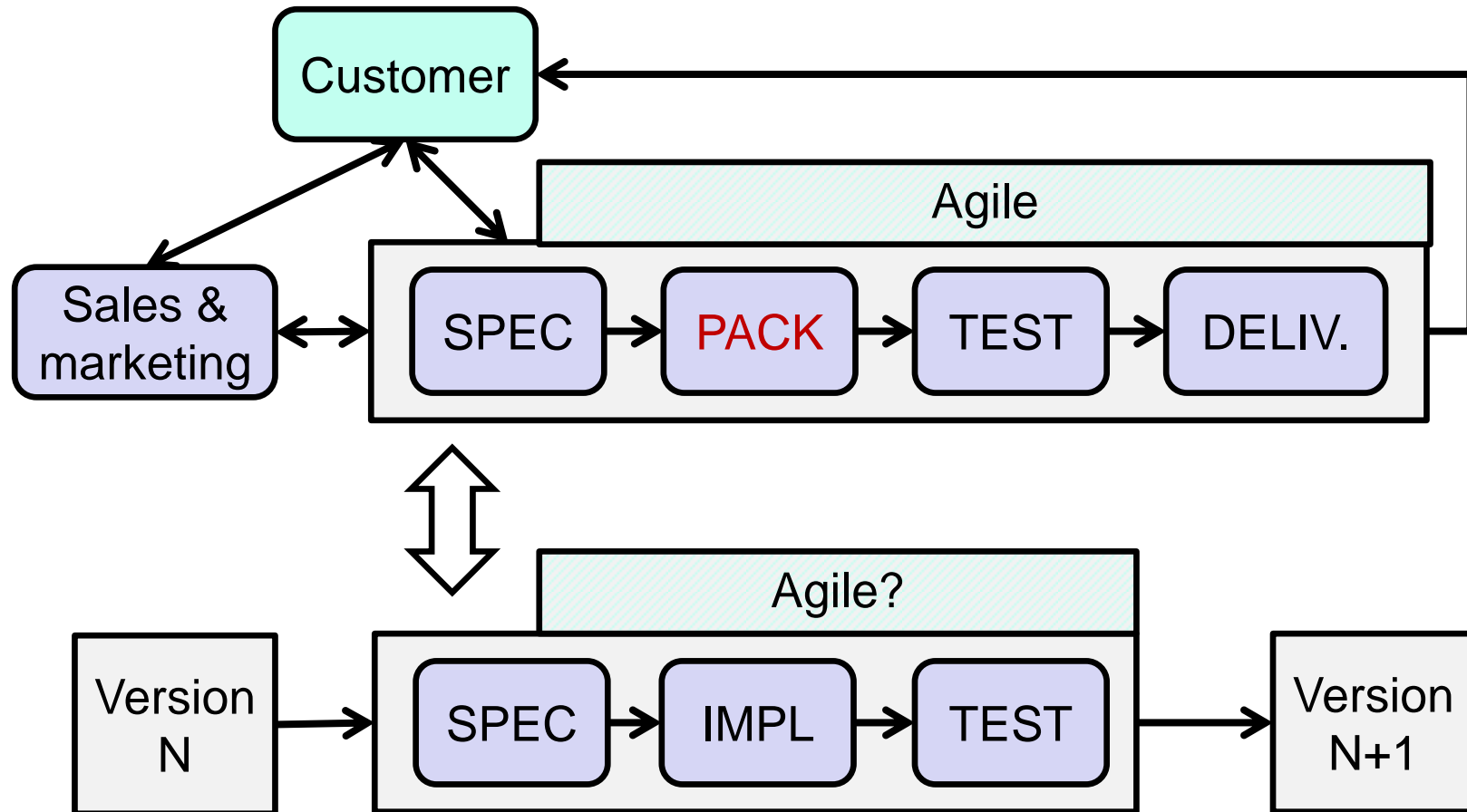
# Step 0: Customer-specific project

Adapted from Haikala&Mikkonen Fig 11.3



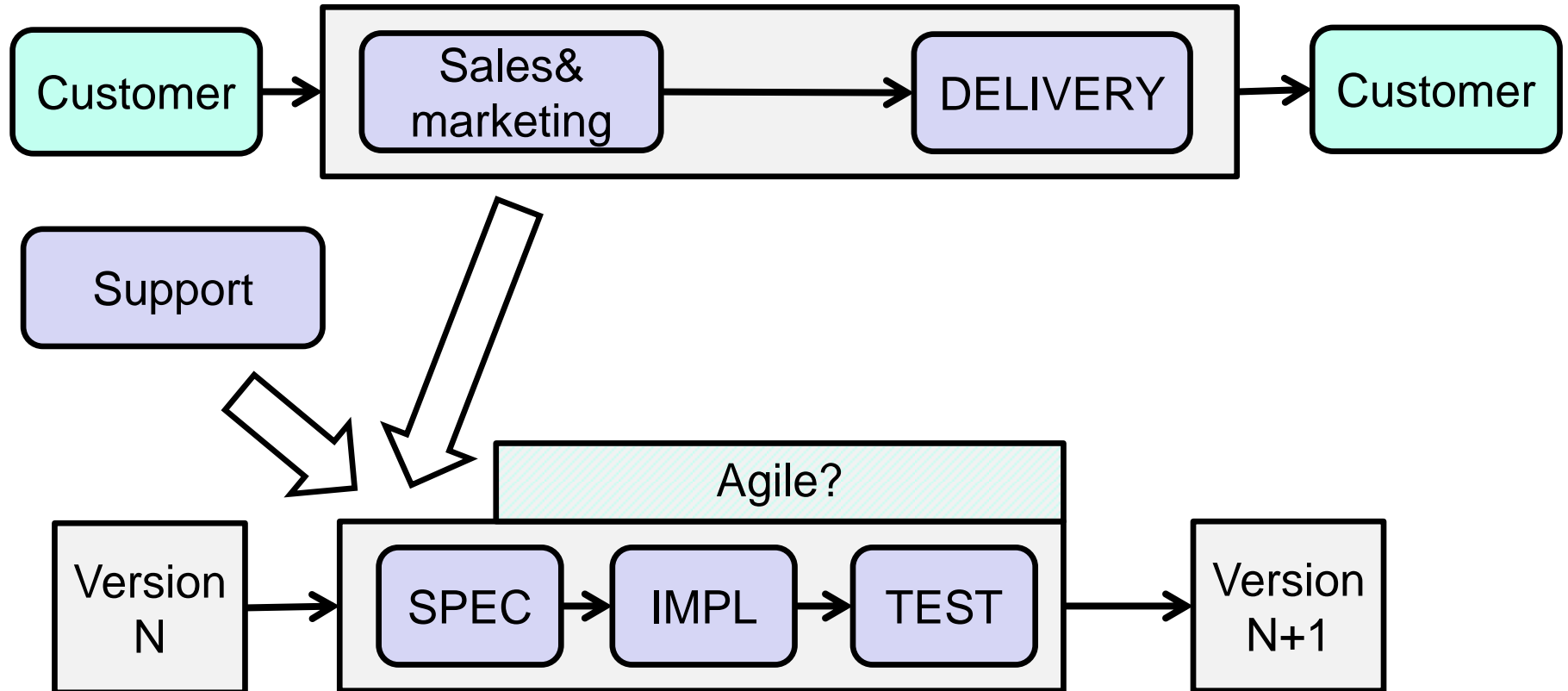
# Step 1: Packetized project

Adapted from Haikala&Mikkonen Fig 11.4



## Step 2: Product process

Adapted from Haikala&Mikkonen Fig 11.5



# **CHALLENGE 3**

## **IPR PROTECTION, OPEN SOURCE OR SOMETHING ELSE**



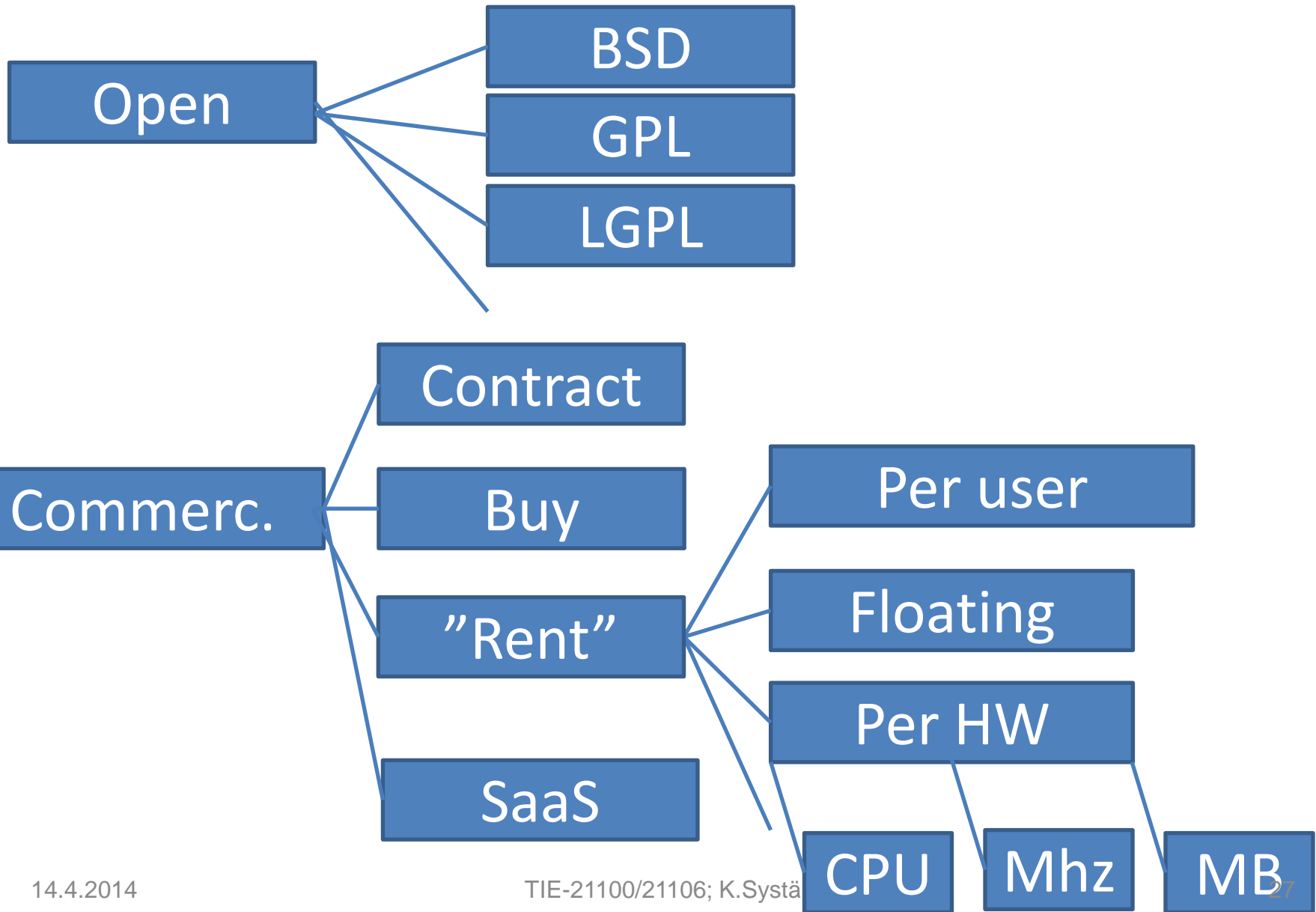
# The software vendors need to protect their business

- IPR protection
  - Based on legislation
  - Different countries have different laws
- License agreements
  - Pricing
  - The included software
  - Conditions of use; constraints
  - Duties, responsibilities, liability

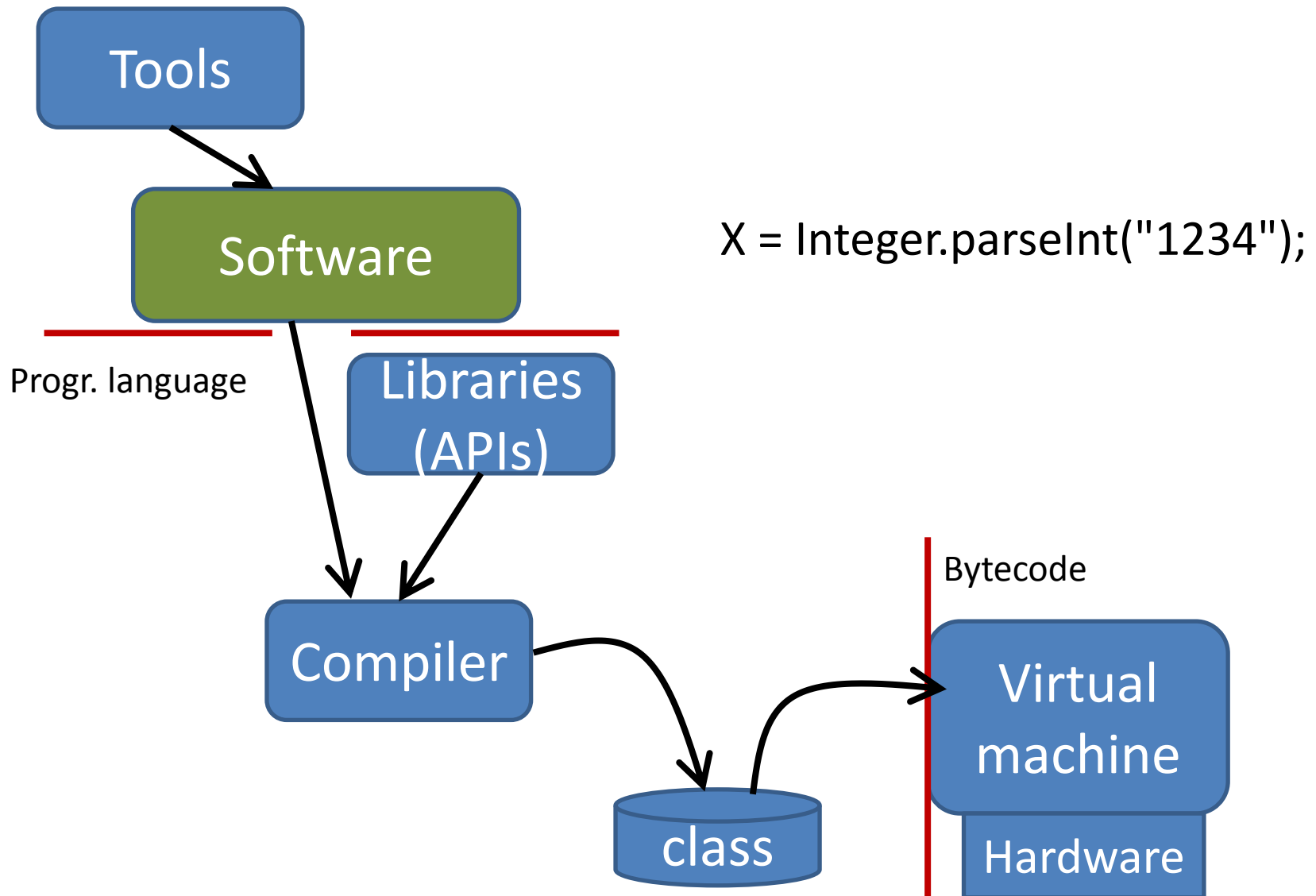
# Three main types of IPR

- Patent
  - Against common believe, SW can be protected with patents
- Copyright
  - Source code, user interface, API
- Trade secret

# World of software licenses



# Java as an example



# *Beware – Open vs. Free Software*

- Free software (1983) is:
  - A philosophy
  - A social movement
  - FSF, free software foundation
  - Stallmanism



Richard Stallman

- Open source (1998) is:
  - A business model
  - A development methodology
  - OSI, open source initiative
  - Raymondism



Eric Raymond

- Both approaches share a common vision on access to source code
- Free as in free speech, not as in free beer

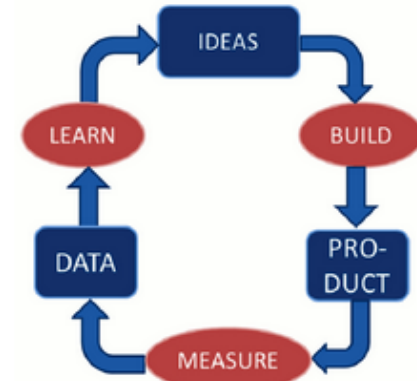
# *Elements of Open Source Software*

- Open development methodology
  - Constant and thorough peer reviews
  - Transparency of development process
  - Global distribution
- Open Source Software license
  - Set of well-defined licenses whose terms define what and what not can be done with the software
  - Lot of incompatibilities; do not always mix with proprietary code
- Community
  - Individuals, companies, and organizations are free to participate
  - (Somewhat) shared mission often needed for driving the community towards a common goal

# **STARTUP SW DEVELOPMENT**



Tampere University of Technology  
Department of Pervasive Computing



## Software Startup Day

11th of December 2013

TUT/Information Technology (Tietotalo), room TB109

*A growing trend in industrial software engineering is that new software products and information services are developed under conditions of notable uncertainty. This is especially visible in startup enterprises which aim at new kinds of products and services especially in web and mobile platforms. Startups have also other typical characteristics like small, committed teams, high degree of agility, and fast development cycles. However, also established companies are more and more seeking proactively new business opportunities with their customers, without having clear understanding of the requirements of the products and services that are actually desired by the customers. In this event, we explore the challenges of startups especially from the viewpoint of developing software: what is important to understand in working in a software startup, which practices are useful and which are harmful, and how to manage software development in a startup. The speakers are leading experts and professionals presenting their viewpoints on software startups.*



## Research Seminar on Startup Software Engineering

TIE-12206

### Seminar introduction

A growing trend in industrial software engineering is that new software products and information services are developed under conditions of notable uncertainty. This is especially visible in startup enterprises which aim at new kinds of products and services in rapidly changing social web, where potential customers can quickly adopt new behavior. However, also established, large companies are more and more seeking proactively new business opportunities with their customers, without having clear understanding of the requirements of the products and services that are actually desired by the customers. Another trend in information industry is the emphasis of fast development: in many areas, and especially in startups, the development speed is much more important than the cost level, as far as the economic value of a product is concerned. These trends have resulted in a software engineering paradigm (called here Startup Software Engineering, SSE), where the development and the usage of a product or service are not separated, but merged into a fast build-use-measure-learn cycle. This is in sharp contrast with existing software engineering paradigms which assume that requirements have been elicited before the actual development starts, and which emphasize controlled process rather than fast delivery. The seminar is carried out in coordination with the [University of Applied Sciences and Arts Northwestern Switzerland](#) in Zurich. The interviews are performed in startup companies both in Finland and in Switzerland, and the results of the interviews are merged in the common set of pattern descriptions.

# What is start-up

- A startup is a human institution designed to create a new product or service under conditions of extreme uncertainty. (Eric Ries 2011)
- Software startup: temporary organizations focused on the creation of high-tech and innovative products, with little or no operating history, aiming to grow by aggressively scaling their business in highly scalable markets (Giardino & Paternoster 2012)
- Software startups are becoming more and more important because
  - information infrastructure enables new kinds of behavior
  - new products & services based on this infrastructure can be developed with little resources

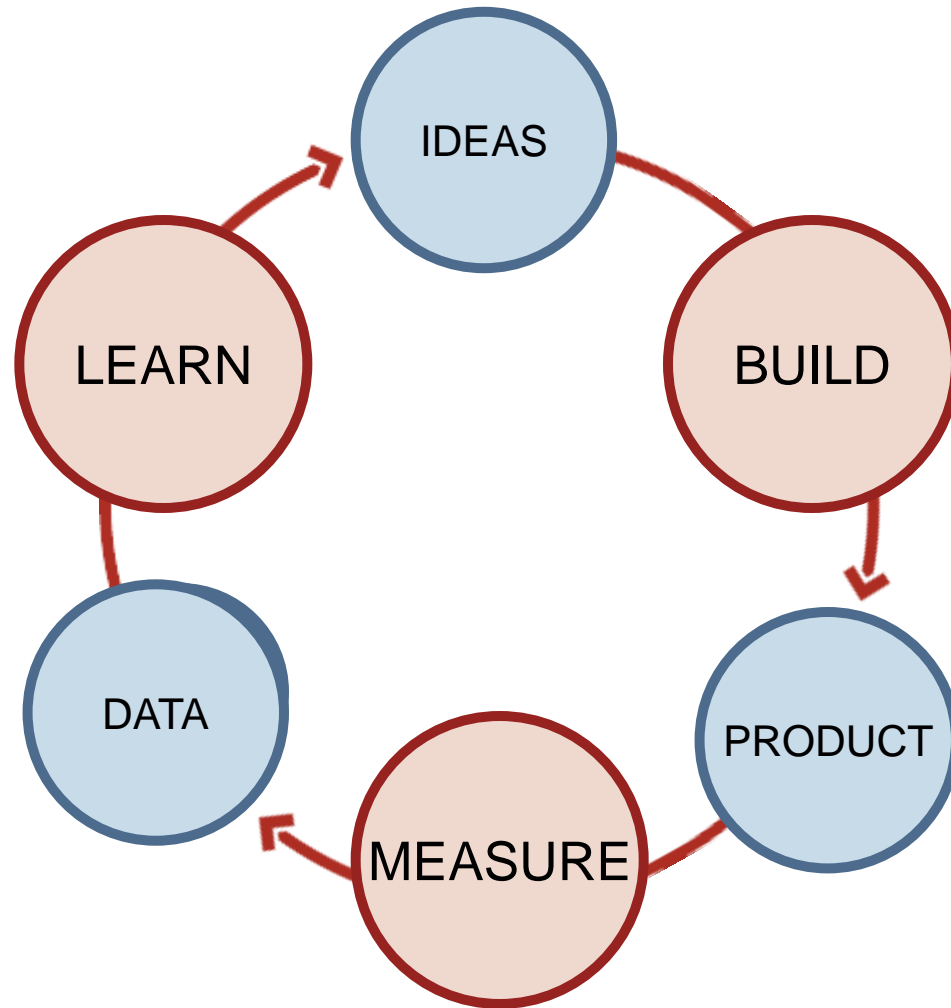
# Crucial questions for start-ups

- Do consumers recognize they have a problem you are trying to solve?
  - If there was a solution, would they buy it?
  - Would they buy it from us?
  - Can we build a solution to that problem?
- Typically, companies start with the last question...
- Manager: I just want this!
- Engineer: I am going to build this!

# More important questions

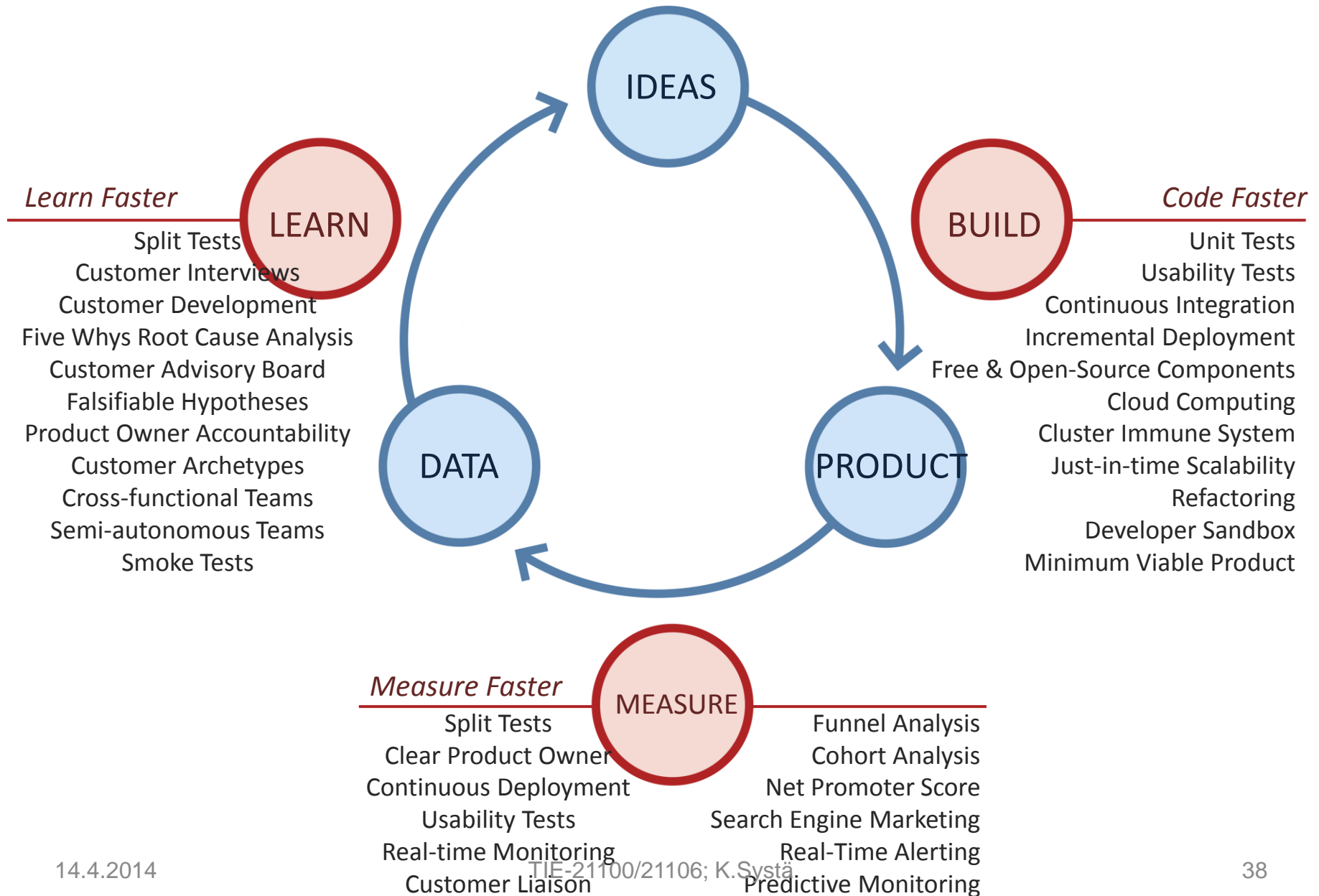
- Which customer opinions should we listen to, if any?
- How should we prioritize across the many features we could build?
- Which features are essential to the product's success and which are secondary?
- What can be changed safely, and what might anger customers?
- What should we work on next?

# The Startup OODA Loop



Minimize *TOTAL* time through the loop

# There's much more...



# IMVU story

- Instant messaging application with customizable avatars (2004)
- A lot of free messaging services available
- First idea: provide the virtual world using existing messaging services
  - Customers would be able to chat online using their IMVU avatars without having to switch IM providers or learn a new user interface. They wouldn't have to persuade their friends to switch, either.
- A first low quality product with the capability to integrate existing IM networks was created
  - Built in 6 months
- The product was launched, but nothing happened: no customers at all
- Quality etc improvements, but still few customers
- Something wrong, but what?

<http://www.inc.com/magazine/201110/eric-ries-usability-testing-product-development.html>

- Eventually, out of desperation, we began bringing people into our office for **in-person interviews and usability tests**. Imagine a 17-year-old girl sitting down with us at a computer. We say, "Try this new product; it's IMVU." She chooses her avatar and says, "**Oh, this is really fun.**" She's customizing the avatar, deciding how it's going to look. Then we say, "All right, it's time to download the instant messaging add-on," and she responds, "**What's that?**"
- "Well, it's this thing that interoperates with the instant messaging client," we say. She has no idea what we're talking about. But because she's in the room with us, we're able to talk her into doing it.
- Then we say, "OK, invite one of your friends to chat." And she says, "No way!" We say, "Why not?" And she says, "Well, I don't know if this thing is cool yet. You want me to risk inviting one of my friends? If it sucks, they're going to think I suck, right?" And we say, "No, no, it's going to be so much fun once you get the person in there; it's a social product." She looks at us, her face filled with doubt; you can see that this is a deal breaker.



- Experiments with customers revealed that they liked to make avatars, but not socialize with messaging and invite friends
- Team created a single-player mode: no better success
- New feature introduced: ChatNow. Allows to be randomly matched with someone else pushing the button at the same time
- Customers liked this. Adding such friends to existing buddy lists?
- Assumption: customers can add such a friend to an existing buddy list
- Reality: they don't, and they don't want to download a whole new IM just for this

# <http://www.inc.com/magazine/201110/eric-ries-usability-testing-product-development.html>

- Then, maybe they would meet somebody they thought was cool. They'd say, "Hey, that guy was neat; I want to add him to my buddy list. Where's my buddy list?" And we'd say, "Oh, no, you don't want a new buddy list; you want to use your regular AOL buddy list." You could see their eyes go wide, and they'd say, "Are you kidding me? A stranger on my buddy list?" To which we'd respond, "Yes; otherwise you'd have to download a whole new IM program with a new buddy list." And they'd say, "Do you have any idea how many IM programs I already run?"
- "No," we'd say. "One or two, maybe?" That's how many each of us used. To which the teenager would say, "Duh! I run eight." It started to dawn on us that our concept was flawed.
- Our early adopters didn't think that having to learn a new IM program was a barrier. Even more surprising, our assumption that customers would want to use IMVU primarily with their existing friends was also wrong. They wanted to make new friends, an activity that 3-D avatars are particularly well suited to facilitating. Bit by bit, customers tore apart our seemingly brilliant initial strategy.

# IMVU story

## Lessons learnt

- Customers don't want an IM add-on, but a stand-alone IM network service
- Having to learn a new IM program is not a barrier
- Customers want to use avatar-based IM also for making new friends

## Afterthoughts

- Especially in a startup, it is unknown who is the customer and what the customer considers valuable
- Strategic assumptions (integration approach) were wrong
- It would have been possible to learn the same things with less effort: work on “possibly valuable” features mostly waste

## New way of working

- Emphasis on experiments. E.g., new customers were split automatically to two different websites, and it was observed which produces more buying customers.
- Working hypothesis: Customers use IMVU for making new friends
- Experiments supported this hypothesis
- Customers start to increase

# IMVU today

## (source wikipedia)

- IMVU, Inc. is an online social entertainment website founded in 2004, in which members use 3D avatars to meet new people, chat, create, and play games.
- IMVU has over 3 million active users and currently has the largest virtual goods catalog of more than 6 million items.
- The business is located in Mountain View, California and currently has 120 full-time employees.
- It is also known as one of the leading practitioners of the Lean Startup approach.

# **SOME FINDINGS FROM THE SEMINAR**

# Patterns were discovered

- The study resulted in 69 pattern candidates. After a screening process, 14 pattern candidates were further elaborated using the pattern workshop method into a more refined form.
- Examples
  - Create the development culture before processes [#54]
  - Keep customer communication simple and natural [#45]
  - Don't grow in personnel [#57]
  - Flat Organization [#3]
  - Unique value proposition [#38]
  - Start with small and experienced team and expand as needed [#64]
  - Time process improvements right [#32]
  - Develop only what is needed now [#27]

# Develop only what is needed now [#27]

## Context

- Startups often think of the extensibility of their products. It is clear from the beginning that the first versions will be extended later for certain customers or to the general market. This is an essential issue in all startup lifecycle phases.

## Problem

- How to tackle the extensibility issue? How to find the best basic approach to development that makes the extension and alterations as easy as possible?

## Solution

- For efficient extensibility, develop only for what you need soon: what this customer requires, what the next release should include, what the current user stories require.
- Don't generalize designs. Don't implement for the next project. Plan only for what is known.
- This is very important for startups due to the lack of resources. Once the directions for products stabilize, other strategies may gain value and that applies to later stages of companies.

# Create the development culture before processes [#54]

## Context

- Often, when a new company is formed, all the elements of organizational activity are missing – there is only a core team of people and its competences, a goal and some vague visions of how to proceed. Everything else must be built.

## Problem

- Quite soon the team must be capable of producing software systems that provide good value to the first customers. That ability requires many things and decisions must be made upon what to develop first? Should processes be the first priority or something else?

## Solution

- Consciously develop from the beginning a company culture that supports what you want to be. (Note: Culture here means the values of the company, company's identity, assumptions, general ways of activity: what we are, what is special about us, how we approach things, are we for example more creative than systematic, or do we put preference to human issues over technology etc.)
- For example, if your key success factor is flexibility, develop competence, general professional skill, collaboration, quality culture and not rigid processes



Week	Lecture	Exercise
10.3	Quality in general; Quality management systems	Patterns
17.3	Dependable and safety-critical systems	ISO9001
24.3	Work planning; effort estimation	Code inspections
31.3	Version and configuration management	<i>Effort estimation</i>
7.4	Role of software architecture software evolution	?
14.4	Software business, software start-ups, IPR	Break?
21.4	Easter	Break?
28.4	Specifics of some domains, e.g. web system and/or embedded and real time systems	?
5.5	Last lecture; summary; recap for exam	?
15.5	EXAM 9-12	