2015-01-30

Reijo Soréus

Bringing a 10 year old admissions system into the future

From waterfall to agile

Universitets- och högskolerådet

The Basics

- Me
  - Reijo Soréus reijo.soreus@uhr.se
  - Technical application manager for the admissions system since 2013
- My organization
  - The Swedish Council for Higher Education (government agency)
  - http://www.uhr.se/sv/Information-in-English/
  - Established in 2013 as a result of an re-organization three former agencies
- My system
  - NyA – the national Swedish admissions system for higher education
    - https://www.antagning.se/se/start (in Swedish for domestic applicant)
    - https://www.universityadmissions.se/intl/start (in English for international applicants)

This presentation

- When you are done fixing the rest list and the quality issues it is time to renovate your system
- Things happens in the world that sends plans into the bin
- Yes, agile methods works but you have to find an adaptation that suits you
- External, non domestic, authentication is complicated

The system

NyA – The Swedish system for admissions to higher education

Universitets- och högskolerådet

Higher education in Sweden

UNIVERSITY ADMISSIONS.SE

ANTAGNING.SE

The admissions process

Upper sec. grades

Academic records

Course offering

Application

+

Credentials

Eligibility check

Assessment and selection

Admission

Rules

Rules

Publishes

Defines

Some 2014 figures

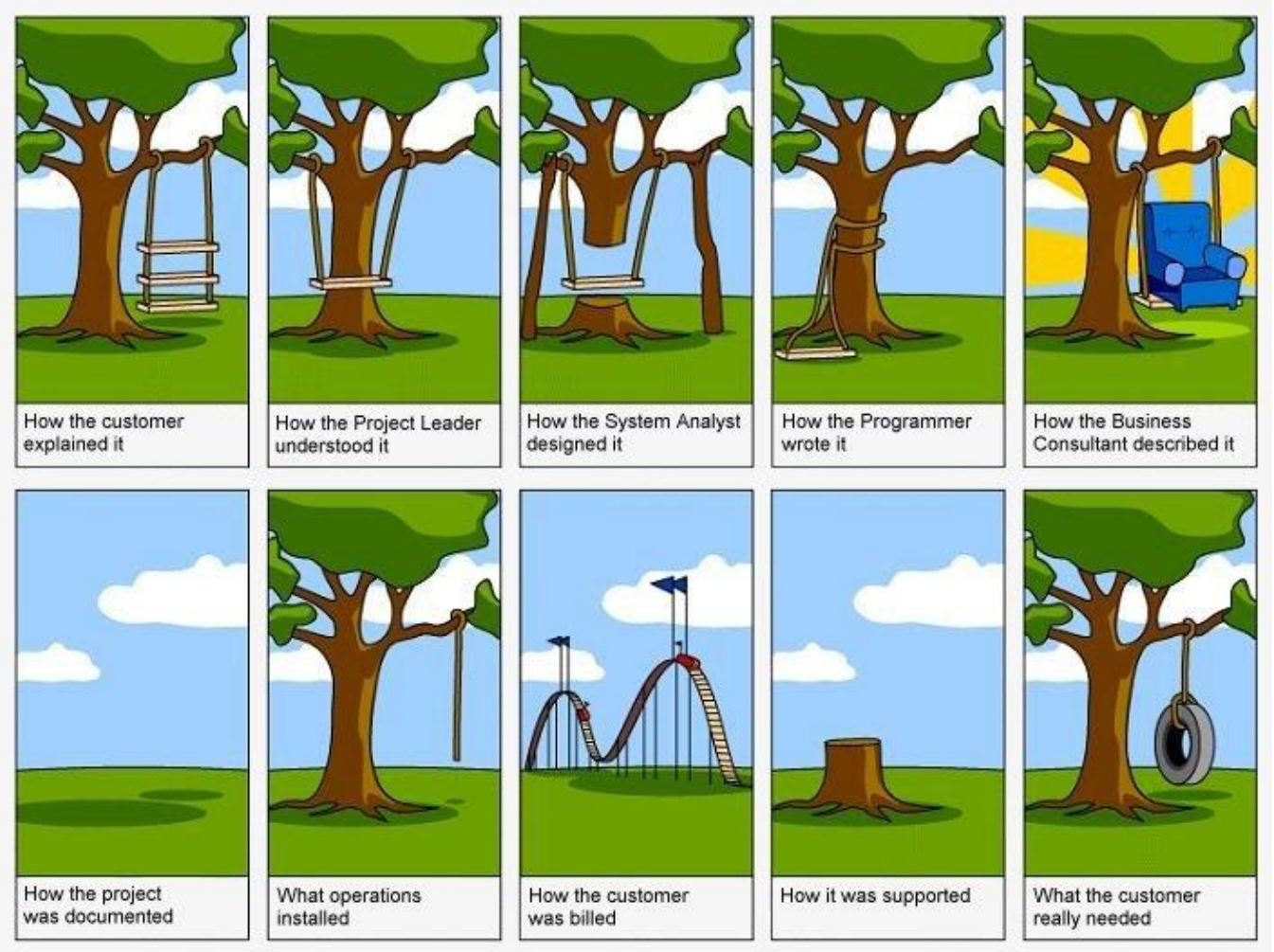
- 37 participating Universities and University colleges (HEI's)
- 831 113 Applications in total
- Total annual budget, € 20 M
  - Includes financial, system and labour costs
  - Financed by a license fee for participating universities
- Average handling cost per application: SEK 233

Background

Original project and lessons from the first 10 years

Universitets- och högskolerådet

The development project



From proposal to production

June 2000, 80 MSEK

10 years of operations

The world keeps changing…

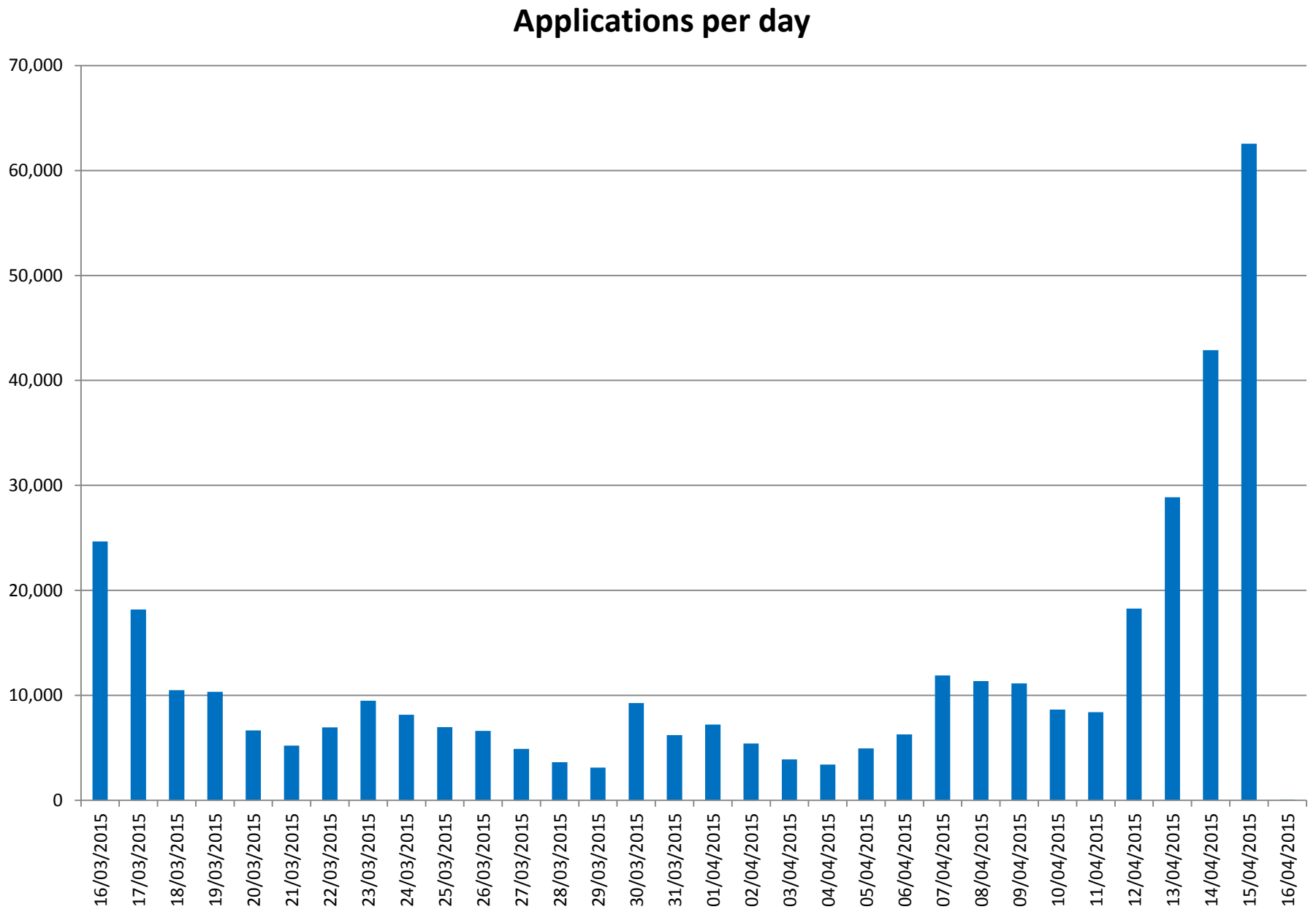
Universitets- och högskolerådet

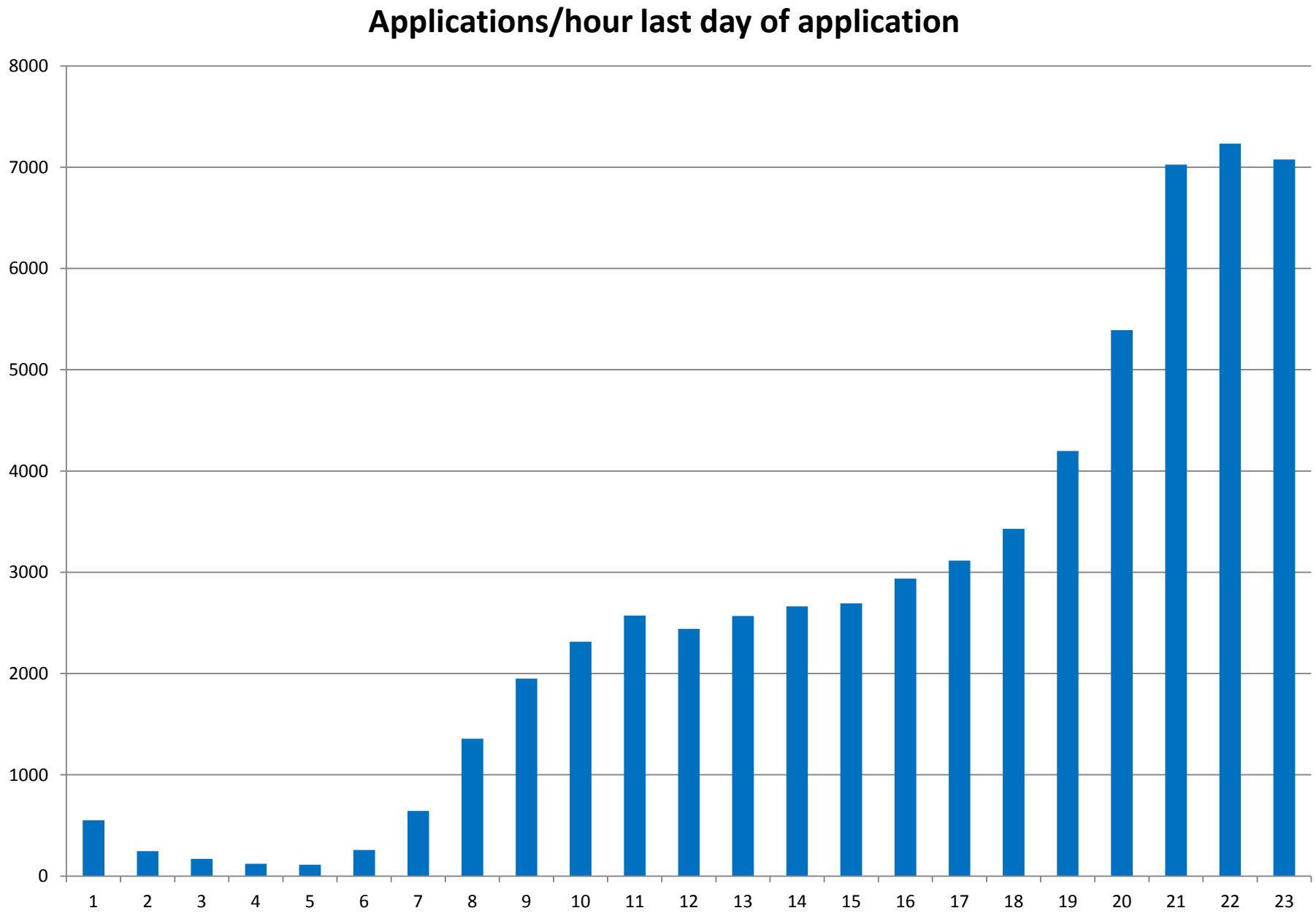
The first years

- Fixing the rest list
  - Actually making the system usable
- Cleaning out bugs – and building technical debt
- Adapting to new regulations
  - Admission and study fees
  - New grading system for upper secondary school leaving certificates
  - …
- Local demands on special admissions
  - Shared programmes
  - Special requirements
- Master programmes
- Resulting in a constant development level of 40’-70’ hours/year
- …and some quality issues

Challenges

- Constant high rate of change
  - The only thing that does not change
- Organization
  - Academic customer, lot of experts…
  - Consensus based decision process
  - Dispersed development organization
    - Application management in Stockholm
    - Development in Umeå
    - Requirement analysts distributed all over Sweden
- Technical
  - Complicated processes and rules
  - Monolithic architecture, technical debt, dependencies
  - Client trends – Java Swing, browsers
  - Security
  - Performance (peak problem)


Applications per day
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
16/03/2015
17/03/2015
18/03/2015
19/03/2015
20/03/2015
21/03/2015
22/03/2015
23/03/2015
24/03/2015
25/03/2015
26/03/2015
27/03/2015
28/03/2015
29/03/2015
30/03/2015
31/03/2015
01/04/2015
02/04/2015
03/04/2015
04/04/2015
05/04/2015
06/04/2015
07/04/2015
08/04/2015
09/04/2015
10/04/2015
11/04/2015
12/04/2015
13/04/2015
14/04/2015
15/04/2015
16/04/2015


Applications/hour last day of application
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

Change management in a nutshell

Where we are

How to get there

Where we want to be

So, where were we?

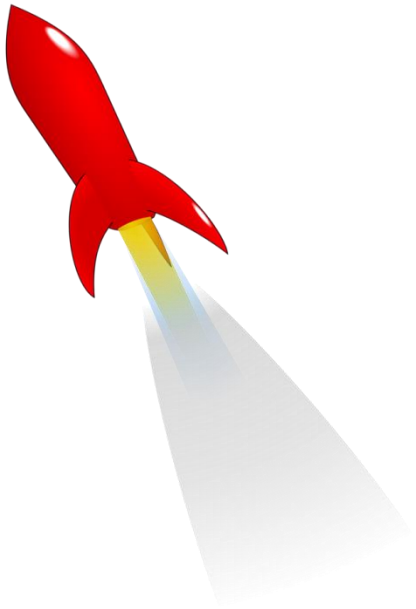
- Very long time from proposal to production
  - Complicated, overly detailed, budget process
  - Detailed requirements analysis before centralized decisions
  - Long (two months) acceptance testing period
  - Two major releases per year
- Quality issues
  - Technical debt
  - Internal dependencies
  - Challenging data model
  - Monolithic application
  - Multiple user interface technologies and generations
  - Steep learning curve for developers
  - Complicated branching
- High unpredictable rate of change
  - Political changes of rules (grading, assessment, fees…)
  - Ambitious academy (new business models)

Where do we want to be?

- Quick response to change
  - Less rigid budget process
  - Flexible project portfolio management
  - Deliver new functionality when needed
- Reduced organizational dependencies
  - Clear responsibilities and mandates
  - Move decision making closer to the users
  - Independent teams
- Improved quality
  - No known errors in delivered code
  - Fix severe bugs fast
  - Build the right thing in the right way
  - Improved (automatic) testability

How to get there…

- Flexible budgeting and planning
  - Minimize the “have to” projects, elastic planning
  - Finish the most important things first
  - Plans are made to be changed
- Agile methods
  - Reduced planning, trust the product owner
  - Engage the users early
  - Testing as early as possible
  - Trust the teams, ensure the improvement process
- New tools
  - Automatic testing
  - Continuous integration (and deploy)
  - Communication solutions
- Architecture
  - Reduce dependencies, modularization, services
  - Refactoring to reduce technical debt
  - New communication technologies (Atom feeds, REST)



Agile challenges

- Trust
  - The product owner is the business expert
  - The development team knows what they are doing
  - They all learn
- Quality is a process
  - The retrospective is the tool
- How to measure improvements?
- Expectations – management and customer
  - Budgeting and planning
  - External processes and integration
- Coherence that allows for experimentation

Agile at UHR

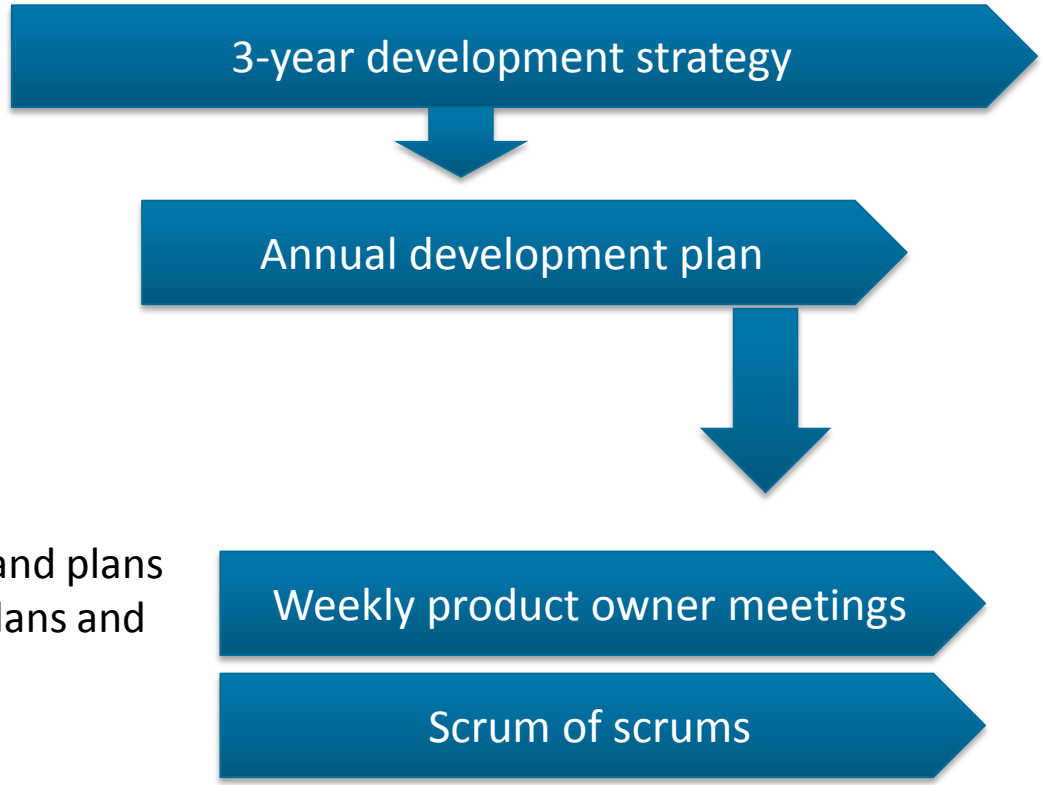
- 1 head product owner/application manager
  - Assisted by a controller and a requirements coordinator
- Five user focused development tracks
  - One product owner
  - Supporting business experts when needed
  - One Scrum master
  - Development team (5-8 developers) per track
  - Reference groups, customer teams etc. according to need
- One technology track
  - One technical product owner
  - Two system architects
  - One Scrum master
  - Developer team
- Supporting functions
  - CM, DBA etc.

The difficult parts

- Difficult to throw away development proposals
  - Kept for a rainy day…
  - Learn to just say no if it can’t be prioritized
- Grooming
  - Need to define ”Definition of ready”
- What to estimate? How? And how far in advance?
  - We tend to fall back to Kanban
- Prioritizing function over code quality

Planning

- 3-year development strategy
  - Long term goals
  - Budget forecasting
  - High level prioritizing
- Annual development plan
  - Prioritized projects and goals
  - Track budgeting adjustments
- Weekly project owner meetings
  - Coordinate functional requirements and plans
  - Update the release manager about plans and progress
- Scrum of scrums
  - Coordinate development
  - Schedule shared resources



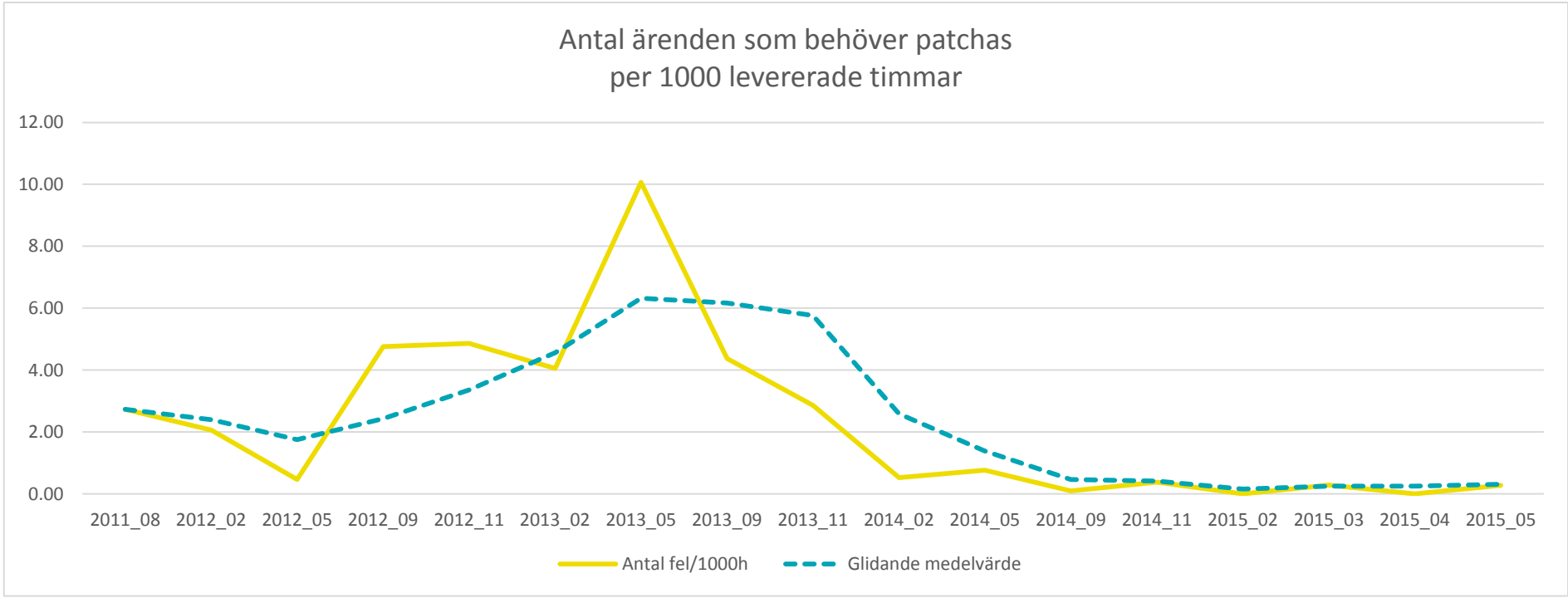
And the result?

- Quality has improved
  - Teams takes responsibility for their code
  - Team learn business rules from the product owner
- More frequent deliveries
  - New functionality comes out quicker
  - Smaller changes means smaller risks
  - Bugs are corrected faster
- Bad code is identified and corrected
  - Refactoring is the norm

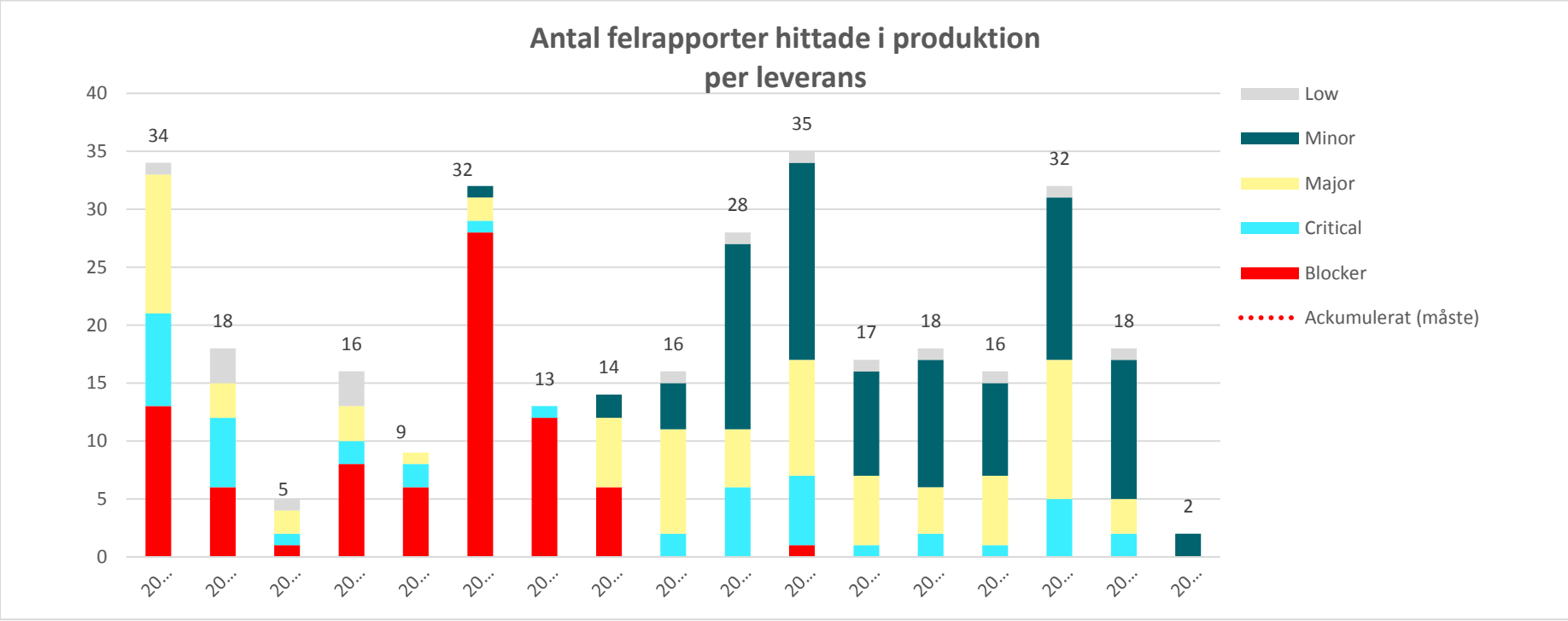
Self improvement

- Retrospectives
  - Sprint retrospectives
  - Application management retrospectives
- Encourage experimentation
  - Try out new methods and new tools
  - Keep what works and let other teams test as well
- Measure quality factors
  - Error rates
  - Released patches
  - Time to correct bugs
  - System stability

Patched bugs per 1000 hours of development



Errors in production



Technical challenges

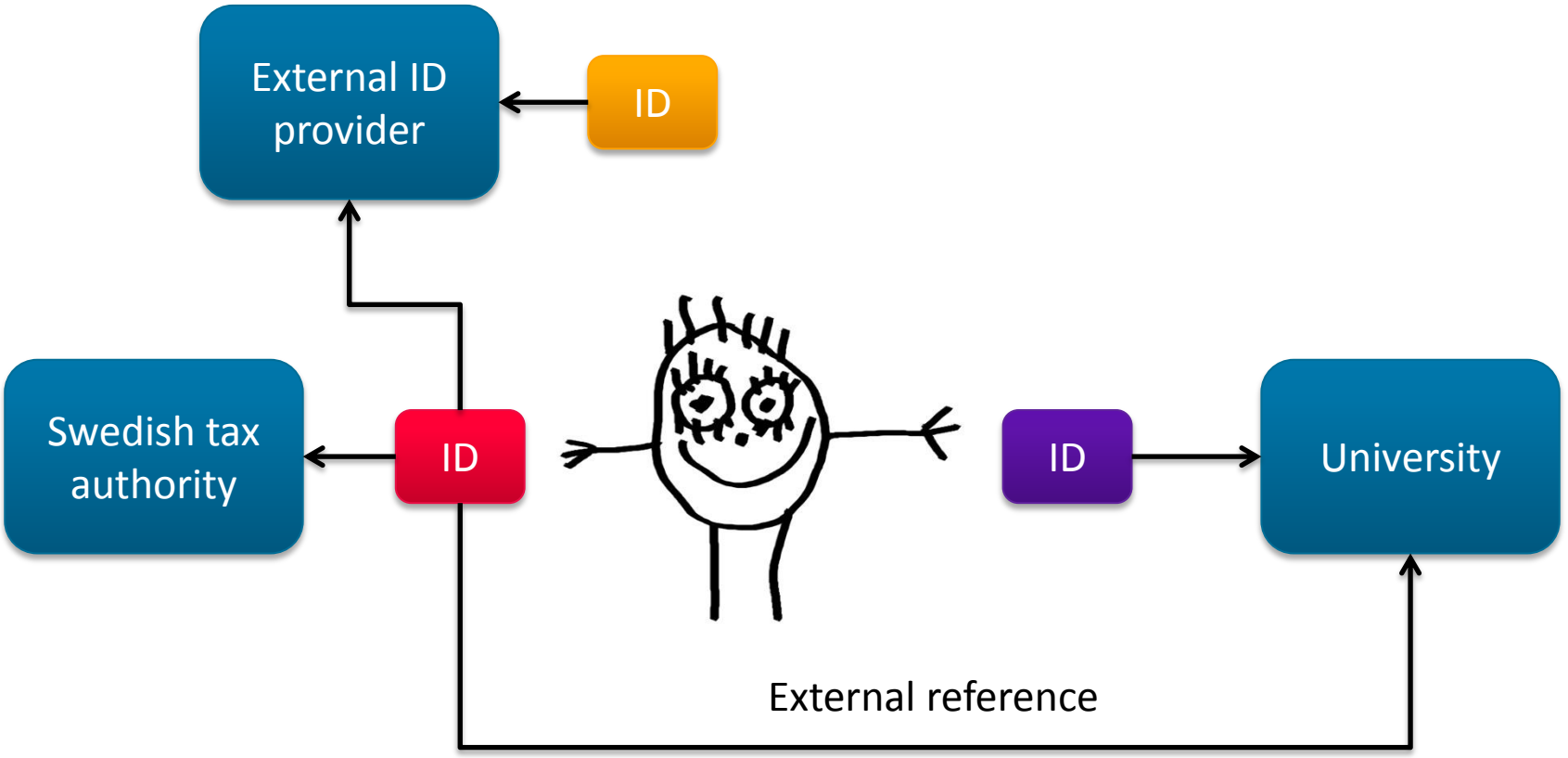
- Java Swing based expert client
  - Difficult to test
- Monolithic architecture with dependencies
  - Service based architecture to make updates easier
- Mobile terminals taking over from PC’s
  - Introduction of responsive design rather than apps
- Continuous integration and deploy
  - New versioning support and build solutions
- System monitoring
  - DevOps for connecting developers and operation
  - Logs, health checks, tools
- Statistical needs, big data
  - The need for analytics increases

What goes on in the world?

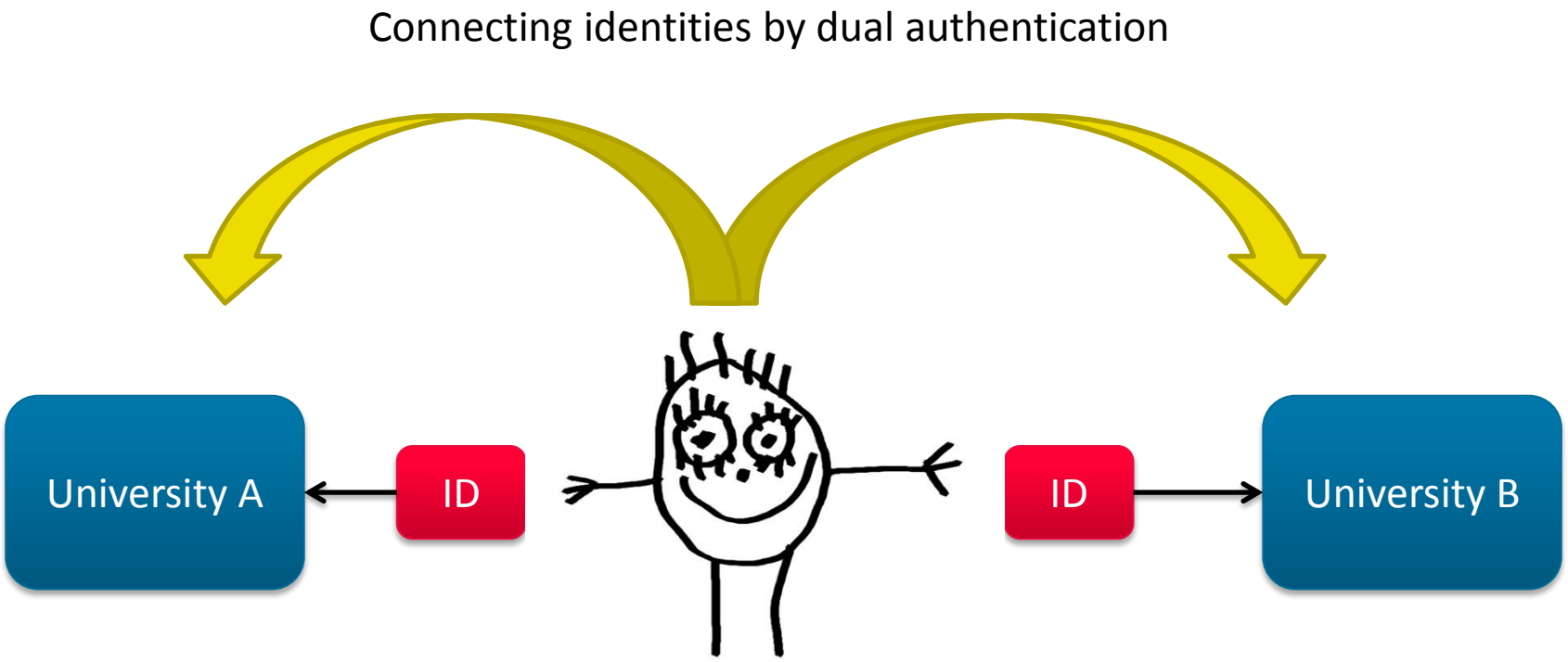
- Development of new Swedish SIS (Ladok3)
  - Introduces new technologies
  - Redefines application integration
- Technological trends
  - Containers
  - Microservices
  - Whatever as a service (XaaS)
  - Outsourcing
- Integration and authentication
  - Federations
  - E-identities crosses borders
  - Transfer of credits and academic qualifications
- Get prepared!

The problem with external authentication

- How to identify individuals?
  - Swedish national solution assigns civic registration numbers for everybody



The EMREX way



Get prepared – the future is coming!