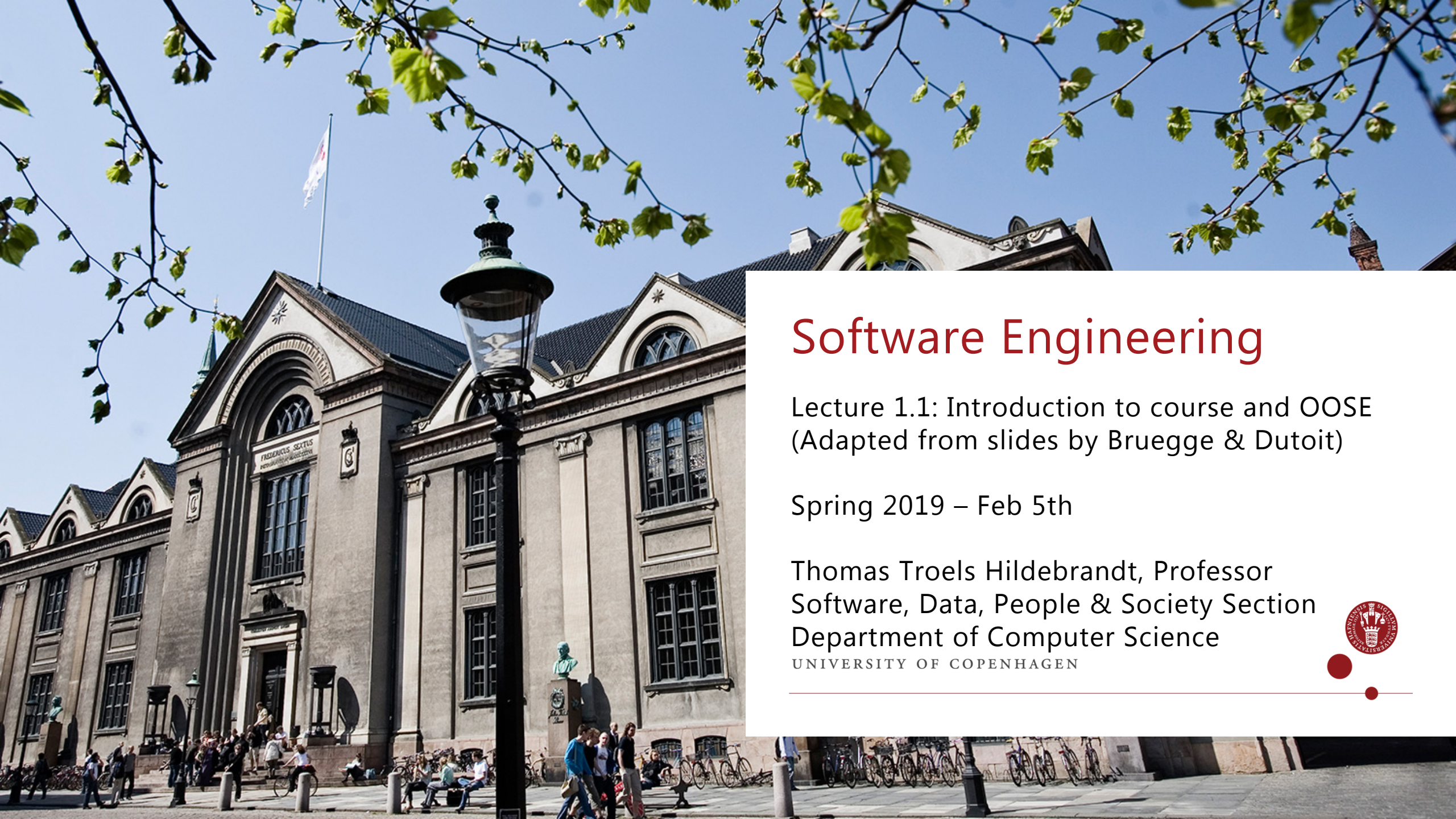
EcoKnow

Effective, co-created & compliant adaptive case management for Knowledge workers

KK – September 18th 2018

Thomas Troels Hildebrandt, Professor
Software, Data, People & Society Section
Department of Computer Science
UNIVERSITY OF COPENHAGEN

Innovationsfonden
Grand Solutions Projekt



Software Engineering

Lecture 1.1: Introduction to course and OOSE
(Adapted from slides by Bruegge & Dutoit)

Spring 2019 – Feb 5th

Thomas Troels Hildebrandt, Professor
Software, Data, People & Society Section
Department of Computer Science
UNIVERSITY OF COPENHAGEN

Roadmap for today

- 9:15-9:40 Lecture 1.1: Introduction to course and OOSE
- 9:40-10 Exercises
- 10:15-10:40 Lecture 1.2: Introduction to UML
- 10:40-11 Exercises
- 11:15-11.40 Lecture 1.3: Dynamic Condition Response Graphs
- 11:40-12.00 Exercises

Welcome to the Software Engineering Course!

- Who am I (professionally)?
  - Started to program in the 80'ties…
  - 1999 PhD in Computer Science with focus on formal process modelling
  - +20 years of teaching & research experience (Aarhus, ITU, Copenhagen University)
  - Inventor of new declarative process modelling language (DCRGraphs) used in industry
  - Board member of the start-up DCRSolutions.net based on DCRGraphs
- Who is the TA ?
  - Philip Falck
- Who are you?
  - Experience with OO programming, but not based on models

Course goals

- Learn fundamentals of model-based software engineering & maintanance
  - The Software Engineering Project
  - Object-oriented Modelling, Analysis & Design
  - Requirements Elicitation
  - Relating models and code
  - Testing, Maintanance, Change & Evolution
  - Apply the knowledge on a real case – a Case Management System

Practicalities

- Schedule (Weeks 6-12 – Feb 6th to March 21th)
  - Tuesdays 9:15-12 we have 3 Lectures including Excercises (20 minutes)
  - Thursdays 9:15-12 we have Q&A and project work
- You need to form projects groups of 3-4 persons.
- Requirements for exam: Hand in the two assignments and final report
- **Exam**:
  - Joint group presentation of project (15-20 minutes)
  - Individual oral examination (20 minutes, including grading) based on report and curriculum – we evaluate the documentation and reflection, not the amount of code
  - **Aids:** Project report, group presentation slides, source code & models
  - **Grade & censorship:** 7-point scale, internal censor

Schedule Overview

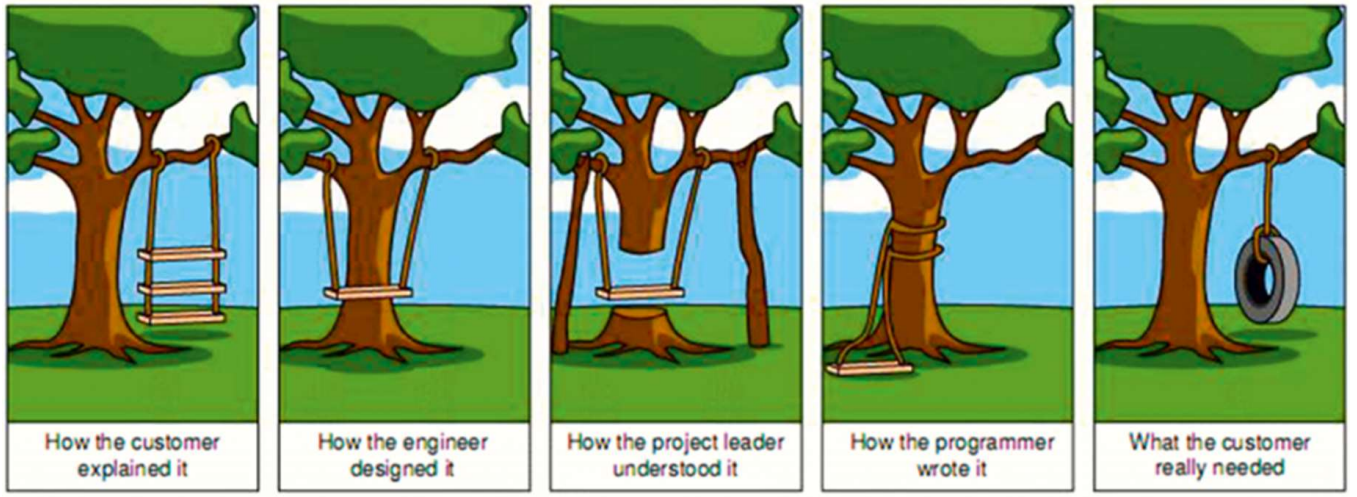
- Week 1: Introduction to course, UML & DCR (Ch 1+2 + DCR highlighter)
- Week 2: Rationale and Project Management (ch 3+12+14)
- Week 3: Requirements Elicitation (ch 4 + DCR) – 1st Hand in: Requirements
- Week 4: Analysis (ch 5)
- Week 5: Design (Ch 6+7) – 2nd Hand in: Analysis & Design Document
- Week 6: Mapping Models to Code and Testing (Ch 10+11)
- Week 7: Software Life Cycle & Conclusion (ch 15) Final hand in
- Week 8(?) Exam
- Deadlines: Feb 21st, March 7th, March 21st?
- Changes may happen!

Software Engineering: A working definition

- Software engineering is a collection of techniques, methodologies and tools that help with the production of
  - A high quality software system developed with a given budget and before a given deadline while changes occurs
- High quality: Addresses the problem satisfactory

Why is Software Engineering so difficult?

- The problem is usually ambiguous
- The requirements are usually unclear and changing when they become clearer
- The problem domain (part of application domain) is complex, and so is the solution domain
- The development process is difficult to manage
- Software offers extreme flexibility and technology keeps changing!



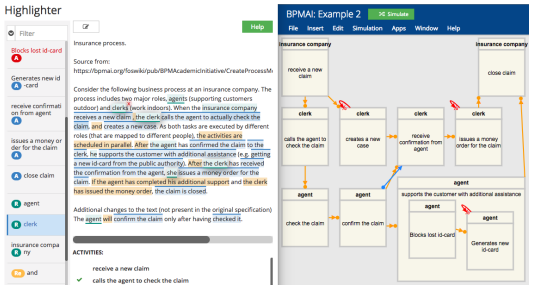
Application/Problem Domain vs. Solution Domain

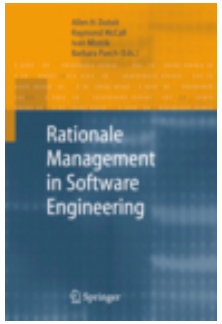
- The **problem and application domain** describe respectively what problem the system addresses and in which context
  - Described e.g. with rich pictures, use cases, sequence diagrams

- The **Solution** (design or implementation) **domain** describes how the system is to be build
  - Described e.g. with system diagrams, class diagrams, code

Much more than just writing code

- Problem solving
  - Understanding a problem
  - Proposing a solution and plan
  - Engineering a system based on the proposed solution
- Dealing with complexity
  - Creating abstractions and models
  - Notations for abstractions
- Knowledge management
  - Elicitation, analysis, design, validation of the system and the solution process
- Rationale management
  - Making the design and development decisions explicit to all stakeholders involved.





Systems

- A *system* is an organized set of communicating parts
  - Natural system: A system whose ultimate purpose is not known
  - Engineered system: A system which is designed and built by engineers for a specific purpose
- The parts of the system can be considered as systems again
  - In this case we call them *subsystems*

Examples of natural systems:

- Universe, earth, ocean

Examples of engineered systems:

- Airplane, watch, GPS

Examples of subsystems:

- Jet engine, battery, satellite.

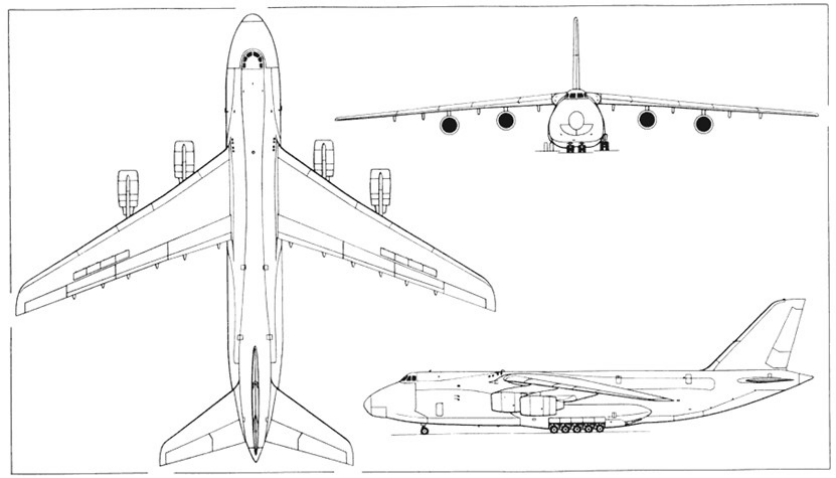
Models, Notations, Techniques, Methodologies and Tools

- Model:
  - Abstract representation of the system of interest
- Modelling notation:
  - Graphical or textual formal language for expressing models
- Techniques:
  - Formal procedures for producing results using well-defined notation
- Methodologies:
  - Collection of techniques applied across software development
- Tools:
  - Instruments or automated systems to accomplish techniques and support methodologies, e.g. integrated development environments (IDEs) or modelling and simulation tools such as DCRGraphs.net from DCRSolutions.net

Abstraction and Models

We create **models** of a system to abstract away from its complexity to something manageable and understandable.

By using formal and standardized **notations** for our models we avoid ambiguity.

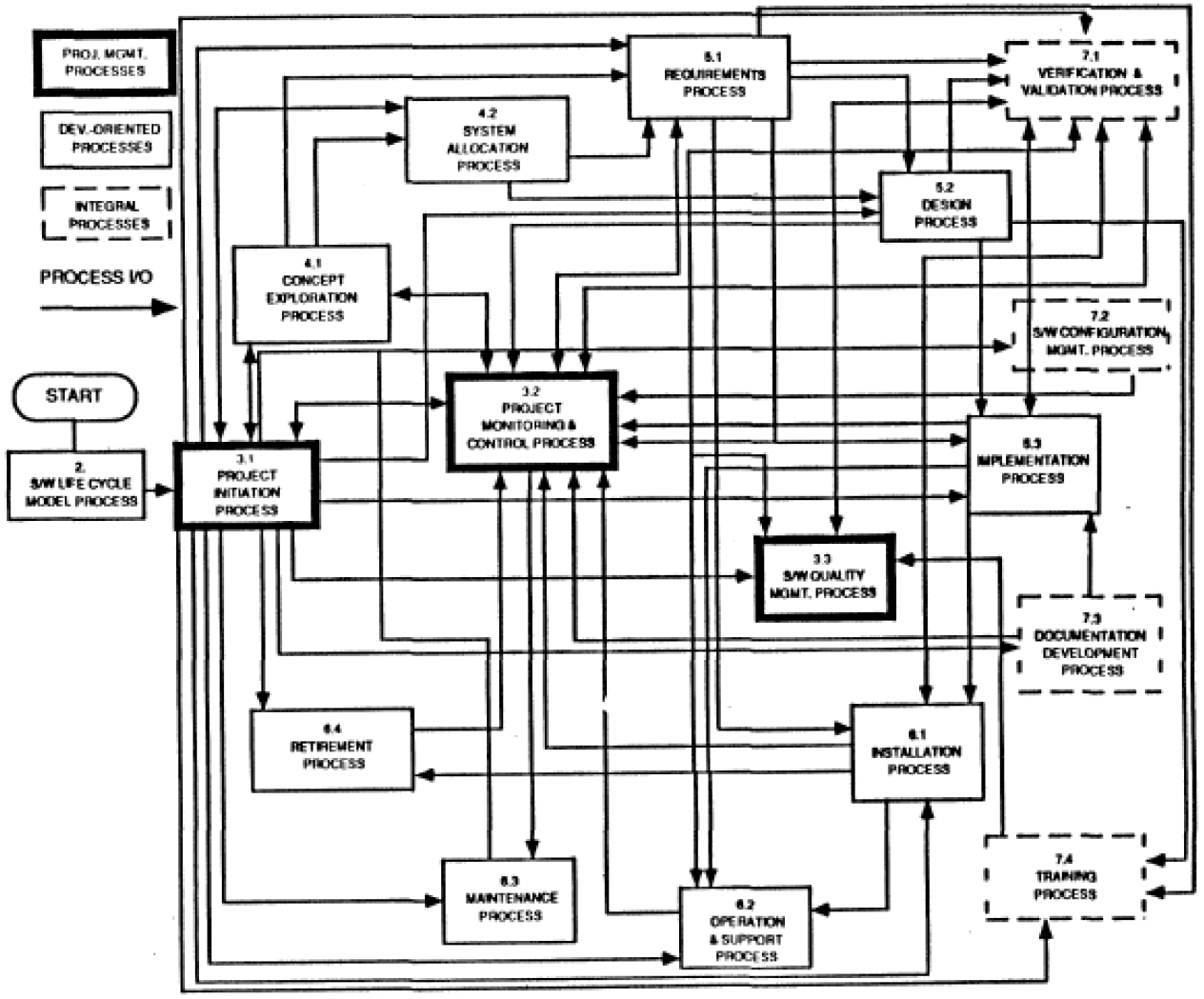


Models must be falsifiable

- Karl Popper ("Objective Knowledge):
  - There is no absolute truth when trying to understand reality
  - One can only build theories, that are "true" until somebody finds a counter example
    - Falsification: The act of disproving a theory or hypothesis
- The truth of a theory is never certain. We must use phrases like:
  - "by our best judgement", "using state-of-the-art knowledge"
- In software engineering any model is a theory:
  - We build models and try to find counter examples by:
    - Requirements validation, user interface testing, review of the design, source code testing, system testing, etc.
- Testing: The act of disproving a model.

Dealing with Complexity

- Abstraction & Models
- Decomposition
- Hierarchy



Abstraction

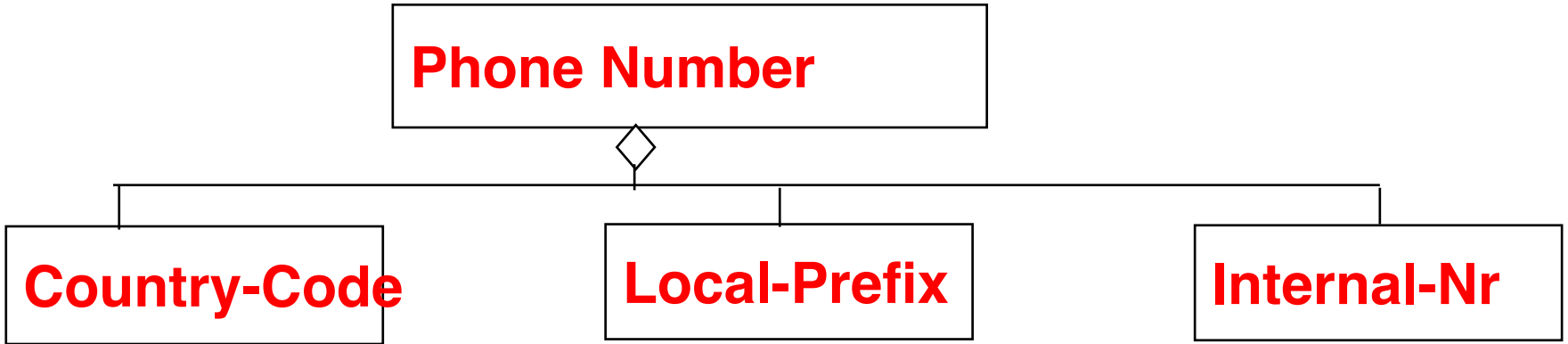
- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Old ITU Phone Number: +4572185279

Abstraction

- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Old ITU Phone Number: +4572185279
- Chunking:
  - Group collection of objects to reduce complexity
  - 3 chunks:
  - Country-code, Local-Prefix, Internal-Nr

Abstraction

- Complex systems are hard to understand
  - The 7 +- 2 phenomena
    - Our short term memory cannot store more than 7+-2 pieces at the same time -> limitation of the brain
    - My Old ITU Phone Number: +4572185279
- Chunking:
  - Group collection of objects to reduce complexity
  - 3 chunks:
  - Country-code, Local-Prefix, Internal-Nr



Models to describe Software Systems

- Object model: What is the structure of the system?
- Functional model: What are the functions of the system?
- Dynamic model: How does the system react to external events?

- System Model: Object model + functional model + dynamic model

Decomposition

- A technique used to master complexity (“divide and conquer”)
- Two major types of decomposition
  - Functional decomposition
  - Object-oriented decomposition
- Functional decomposition
  - The system is decomposed into modules
  - Each module is a major function in the application domain
  - Modules can be decomposed into smaller modules.
- Object-oriented decomposition
  - The system is decomposed into classes (concepts describing objects)
  - Each class is a major entity in the application domain
  - Classes can be decomposed into smaller classes

Concepts and Phenomena

- Phenomenon
  - An object in the world of a domain as you perceive it
    - Examples: This lecture, my black watch
- Concept
  - Describes the common properties of phenomena
    - Example: All lectures on software engineering
    - Example: All black watches
- A Concept is a 3-tuple:
  - **Name:** The name distinguishes the concept from other concepts
  - **Purpose:** Properties that determine if a phenomenon is a member of a concept
  - **Members:** The set of phenomena which are part of the concept.

Types and values

- Type:
  - A concept in the context of programming languages
  - Name: int
  - Purpose: integral number
  - Members: `0, -1, 1, 2, -2,…`
- Instance:
  - Value/Member of a specific type

- The type of a variable represents all possible instances, i.e. values, the variable can hold

Example: A watch as a concept

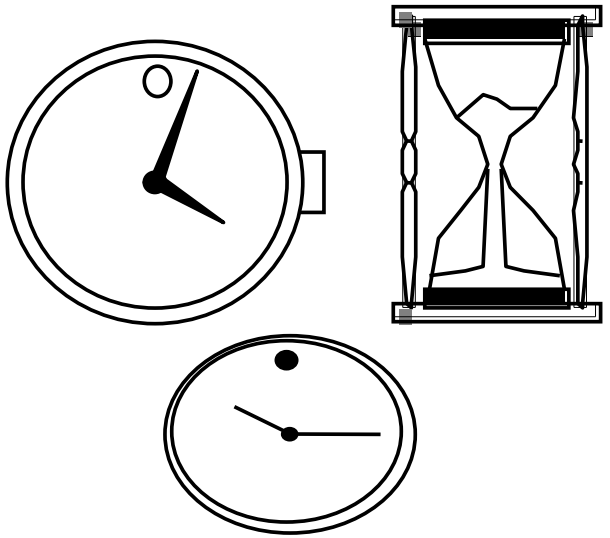
Name

Watch

Purpose

A device that measures time.

Members



Definition Abstraction:

- Classification of phenomena into concepts

Definition Modeling:

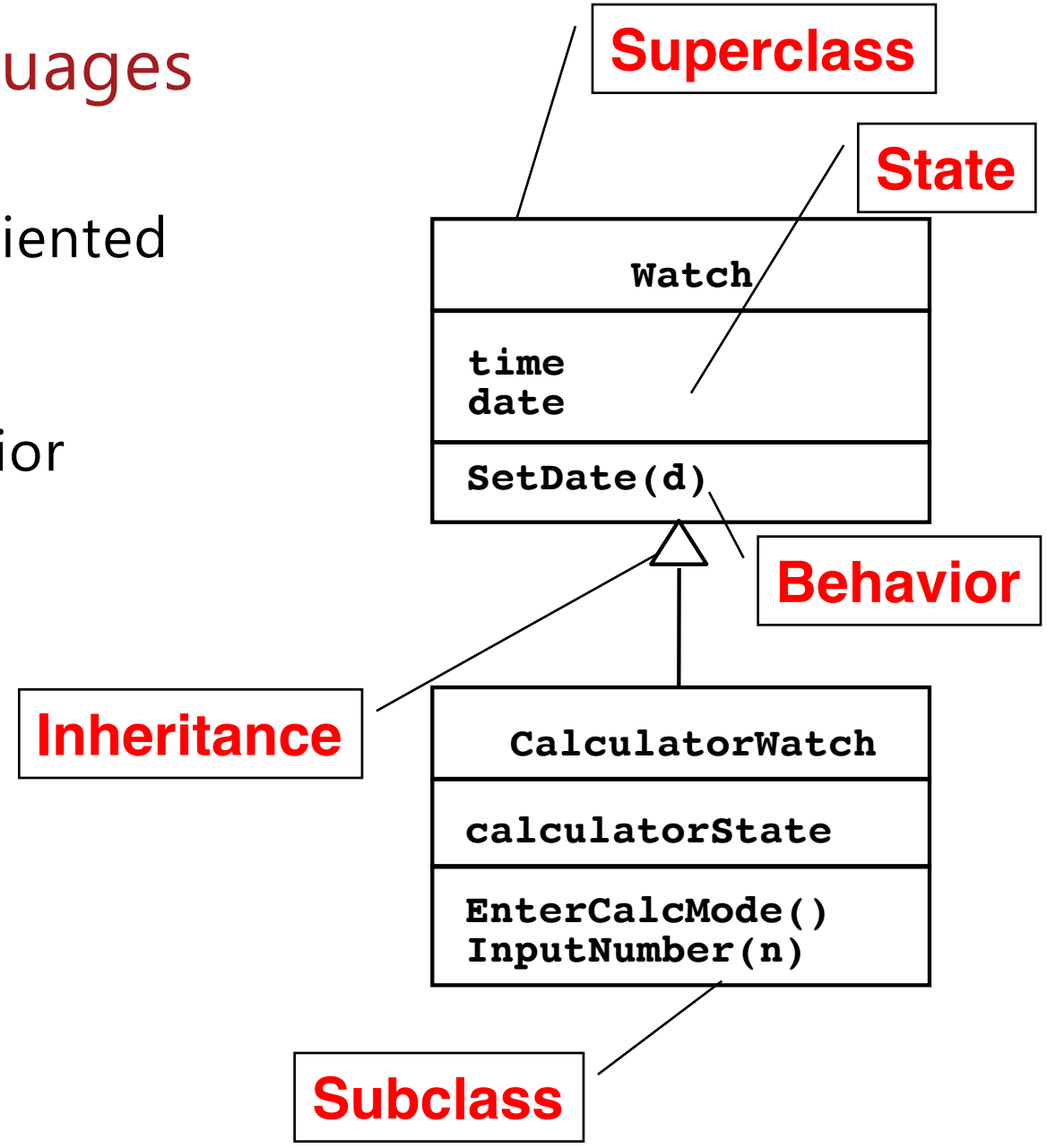
- Development of abstractions to answer specific questions about a set of phenomena while ignoring irrelevant details.

Classes in Object-Oriented Languages

- A concept in the context of object-oriented languages
- A class encapsulates state and behavior

Subclasses can be defined in terms of other classes using inheritance.

Example of using hierarchy to handle complexity



Class identification

- **Basic assumptions:**
  - We can find the *classes for a new software system:* Greenfield Engineering
  - We can identify the *classes in an existing system*: Reengineering
  - We can create a *class-based interface to an existing system:* Interface Engineering

Model-driven Software Development

- *Application and problem domain:* A stock exchange lists many companies. Each company is identified by a ticker symbol

- *Analysis* results in analysis object model (UML Class Diagram):

- *Implementation* results in source code (Java):

```
public class StockExchange {
    public m_Company = new Vector();
 };
public class Company  {
 public int m_tickerSymbol;
 public Vector m_StockExchange = new Vector();
};
```

Model-driven Development and Architecture

1. Build a platform-independent model of an applications functionality and behavior
   a) Describe model in modeling notation (UML)
   b) Convert model into platform-specific model
2. Generate executable from platform-specific model

Advantages:

- Code is generated from model (“mostly”)
- Portability and interoperability

- Model Driven Architecture effort:
  - http://www.omg.org/mda/
- OMG: Object Management Group

Project theme for this course: Municipal Case Management

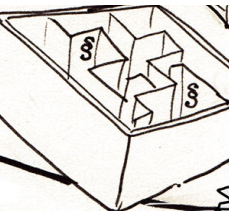
- We have 98 municipalities in Denmark
- 1.500 – 550.000 citizens and 8 - 1.500 km^2
- Just as many other organisations and companies, municipalities use many it systems in their daily work processes.
- Need to comply with many legal constraints

EcoKnow: Effective, co-created & compliant adaptive case management for knowledge workers (2017-2021)

Innovationsfonden
Grand Solutions Projekt

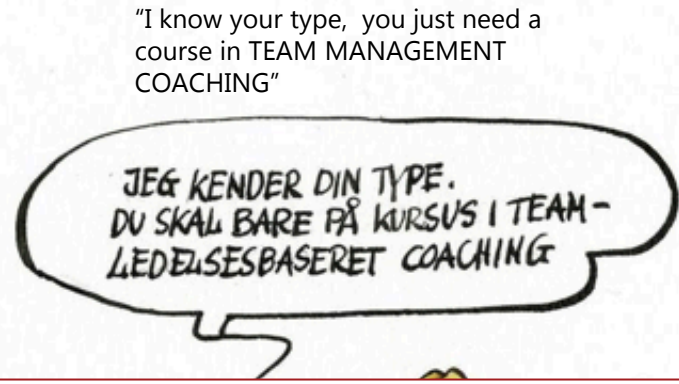
WP1: Computer Supported Cooperative Work: Understanding case management practices, discretion & automation

WP2: AI & Prescriptive Process Management: Application of historical data for finding effective paths to the goal



WP3: Adaptable digitalisation of the law and agile case management support

WP4: Understandability and end-user adaptability of proces descriptions



Goals:

1. Reduce cost of digitalisation
2. Increase quality of case management
3. Increase effectiveness
4. Local anchoring – national sharing
5. Support for co-creation

Participants

KØBENHAVNS UNIVERSITET

IT-UNIVERSITETET I KØBENHAVN

DTU Danmarks Tekniske Universitet

KMD WORKZONE

KMD MOMENTUM

FUJITSU

DCR

INFO VENTURE

MAPS SHARING KNOWLEDGE

Syddjurs KOMMUNE

KØBENHAVNS KOMMUNE

Dansk Socialrådgiverforening

GLOBETEAM

Kammeradvokaten Advokatfirmaet Poul Schmith

GLADSAXE

ETH zürich

TARTU ÜLIKOOL 1632 UNIVERSITAS TARTUENSIS

VU VRIJE UNIVERSITEIT AMSTERDAM

Advisory board

AACHEN UNIVERSITY

TU WIEN TECHNISCHE UNIVERSITÄT WIEN

KL

Background: Our work & life processes are being digitalised

Travel bookings

Transport infrastructure

Bank loans

Research grants

Tax payments

Hospital treatments

Hospital construction

Emergency management

Social services & unemployment

Pushed strongly by industry & government

A FUTURE THAT WORKS:

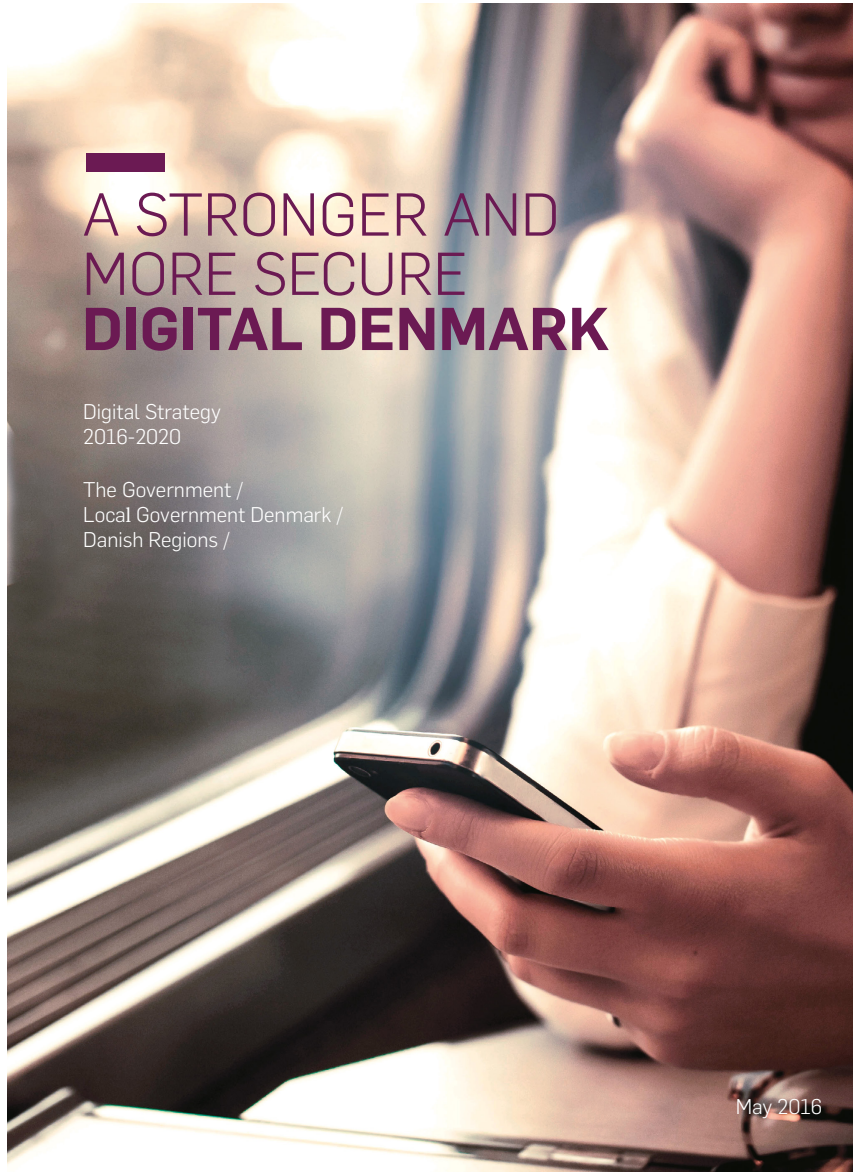
the impact of automation in Denmark

McKinsey&Company

AARHUS BSS THE TUBORG RESEARCH CENTRE FOR GLOBALISATION AND FIRMS
INSTITUT FOR ØKONOMI
AARHUS UNIVERSITET

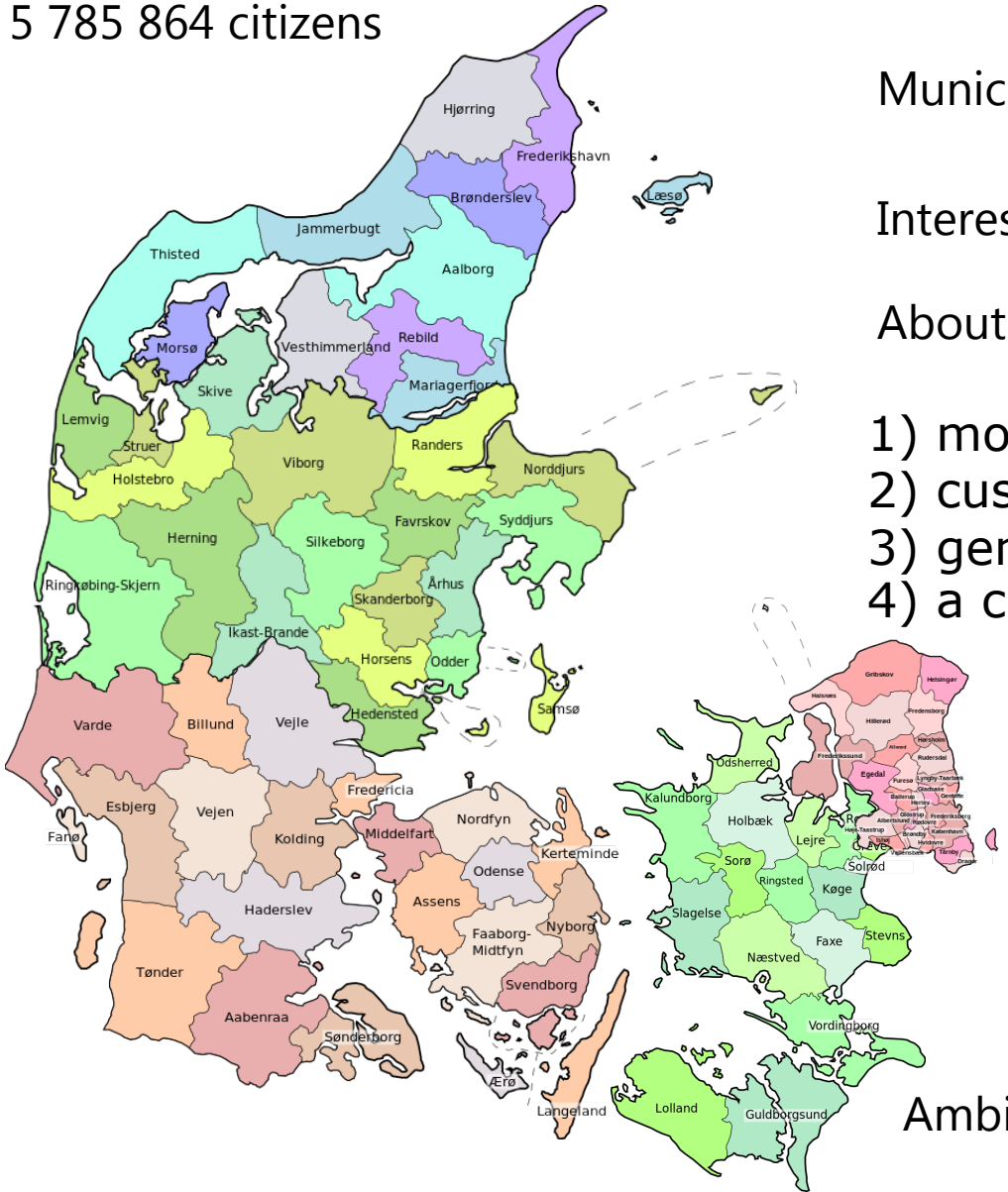
"~27% of all tasks in public services in DK potentially automatic"

October 2017: Digitalisation ready regulations:
Save 7.5 million work hours & one billion DKK yearly by 2020



98 Municipalities, 5 Regions and X governmental services

5 785 864 citizens



Municipalities: From 1.500 – 550.000 citizens and from 8 - 1.500 km^2

Interest organisation for municipalities: Local Government Denmark

About 2600 different case types in municipalities

1) monolithic domain-specific systems
2) customised general case management systems,
3) general document management systems
4) a combination of many of these systems.

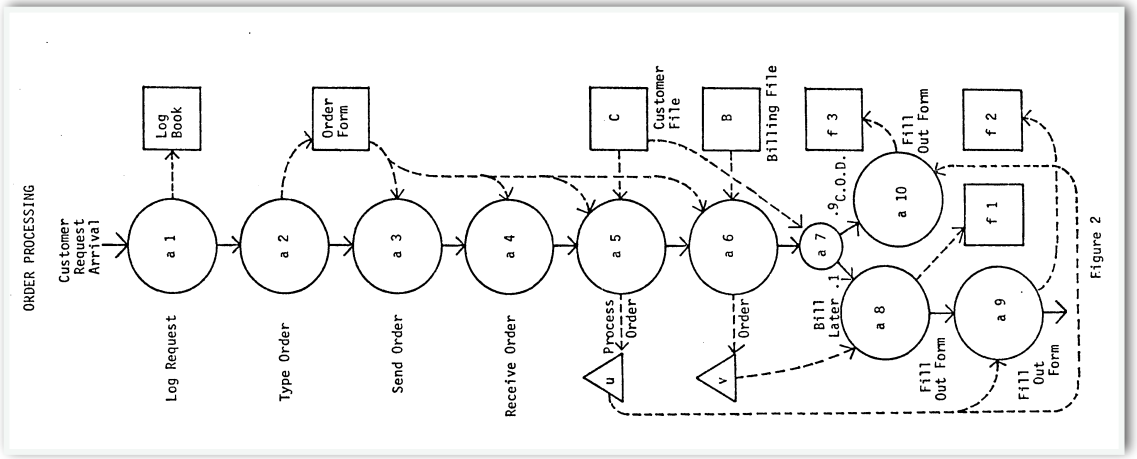
CPR (Central Personal Register) numbers since 1968

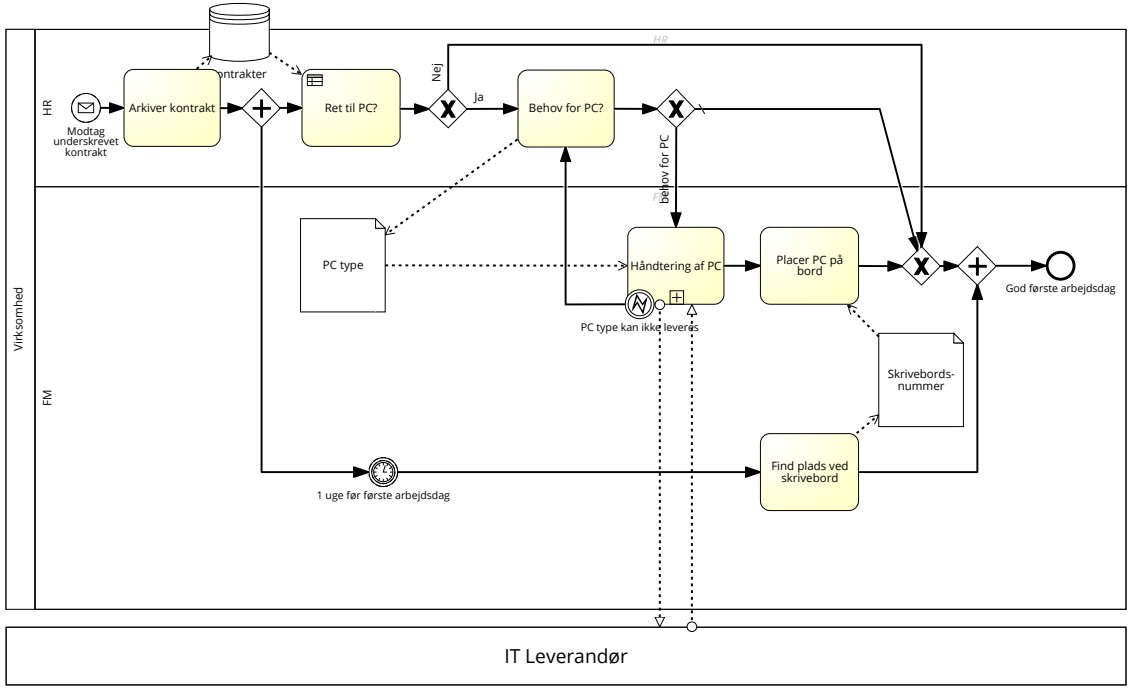
Ambitious BPMN workflow repository attempted but given up in 2013

Digitalizing processes as flows and procedures is challenging

- 40 year old dream, but…
- Using standard procedural flow-diagram notations we usually describe only a few ideal paths
- Do not capture *why* activities are ordered as they are
- Difficult to maintain
- Difficult to validate against regulations and law

Ambitious Municipal BPMN workflow repository given up in 2013

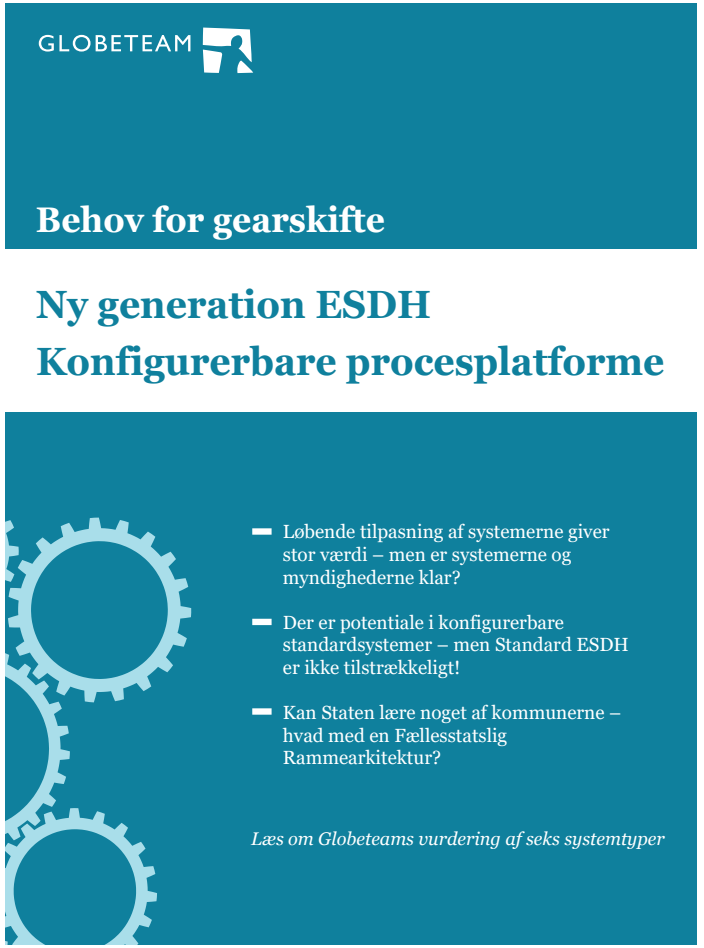




Most sought for qualities towards successful public digitisation

- Support for easy continuous adaptation
- User-friendliness
- Automatisation
- Local anchoring of the digitalisation

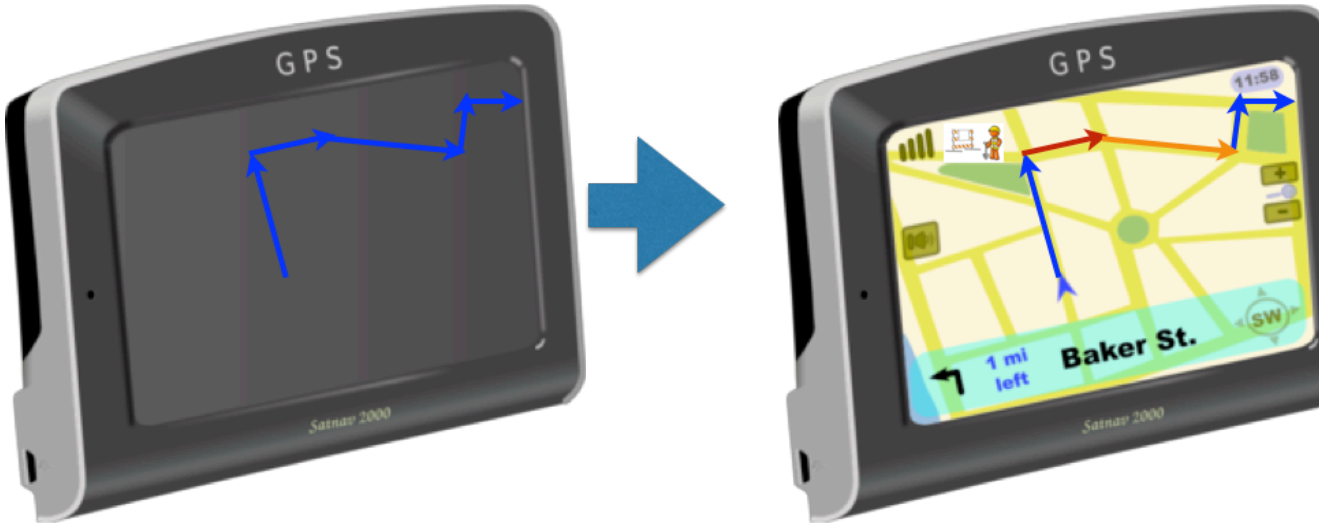
New generation of document management (ESDH) and configurable process platforms ?



Inge Bograd and Per Andreasen. "Ny generation ESDH / Konfigurerbare procesplatforme". Globeteam, December 2016.

Key Enabling Technology: Dynamic Condition Response Graphs

- Declarative notation: Digitalise regulations and allow all compliant paths
- Capture *why* activities are ordered as they are
- Fast & incremental, agile digitalisation
- Easier to maintain when regulations change
- Scenario-driven modelling
- Supports validation via simulation and formal analysis



DCRGraphs.net

DCRSolutions.net

Inter-disciplinary research and development

- Work package 1: Field studies of work practices and research in computer supported cooperative work (ITU & KU)
- Work package 2:
  Data & process mining for predictive and prescriptive process management (ITU, KU, KMD, VU Amsterdam, Tartu University)
- Work package 3:
  Formal models of law and processes (ITU, KU, ETH Zurich)
- Work package 4: Understandability studies of modelling tools and methods (DTU & KU)

Best Practices & Robotic Process Automation

KØBENHAVNS KOMMUNE

GLOBETEAM

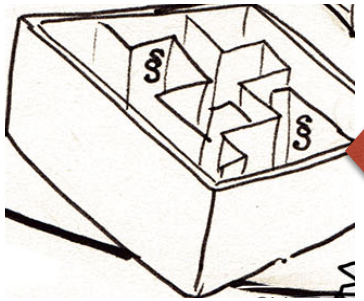
Syddjurs KOMMUNE

Dansk Socialrådgiverforening

End-user path-finding

Declarative Process Mapping & Security (ETH Zurich, ITU)

Kammeradvokaten



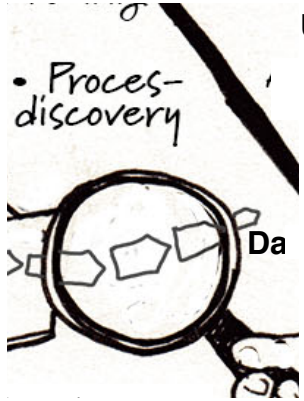
Human Centred Computing (KU
CSCW & Agile (ITU)
Understandability (DTU)

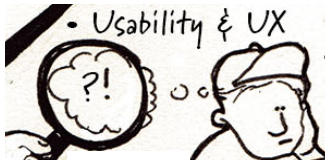
Formalisation of law & regulations

INFO VENTURE

MAPS SHARING KNOWLEDGE

Data analytics (process mining)





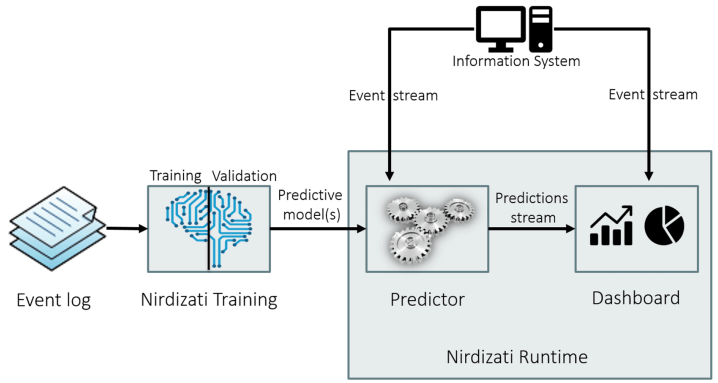
Tartu University
VU Amsterdam

NIRDIZATI

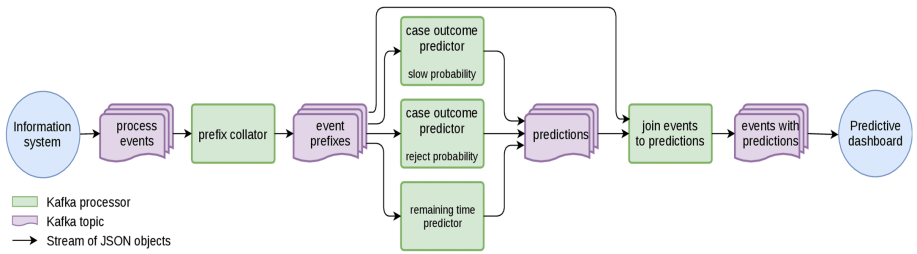
Open source predictive process monitoring dashboard

- Predict **process outcome** (“*Is this loan offer going to be rejected?*”)
- Predict **process performance** (“*Will this claim take longer than 5 days to be handled?*”)
- Predict **future events** (“*What activity is likely to be executed next? And after that?*”)

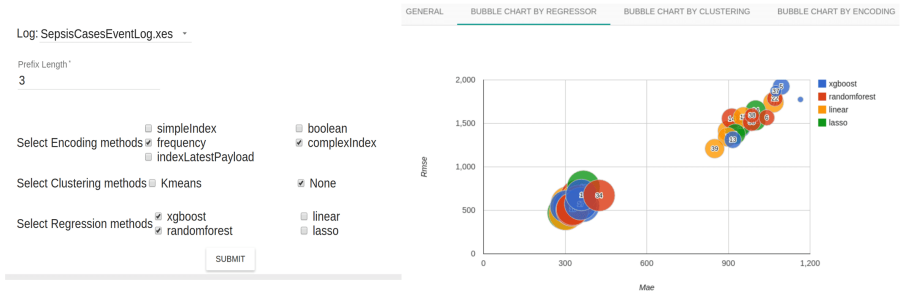
High-level overview



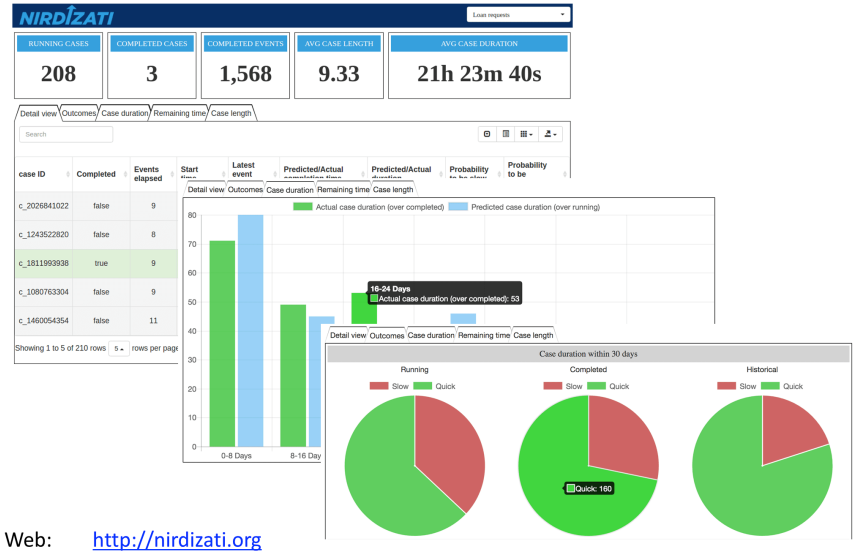
Dataflow diagram of the Runtime component



Training component



Runtime component



Web: http://nirdizati.org
Code: https://github.com/nirdizati
Runtime: http://dashboard.nirdizati.org
Training: http://training.nirdizati.org

Events?

- Assign target group 2.x (x = 1..14)
- Close target group + reason
- Assign goal (Education goal, job goal)
  - Education goal: type is taken from AMU, end date is expected start of education, "borgerens frist" is not allowed to be in the past – the citizen do something before. TRIN 1-4.
  - Job goal: type is taken from Jobstillingsdatabase/jobnet. Start date always today. End date.
- Offerings (Tilbud): Pratik, Øvrige kurser, psykisk mestring, TILBUD iD.
- Anden samtale

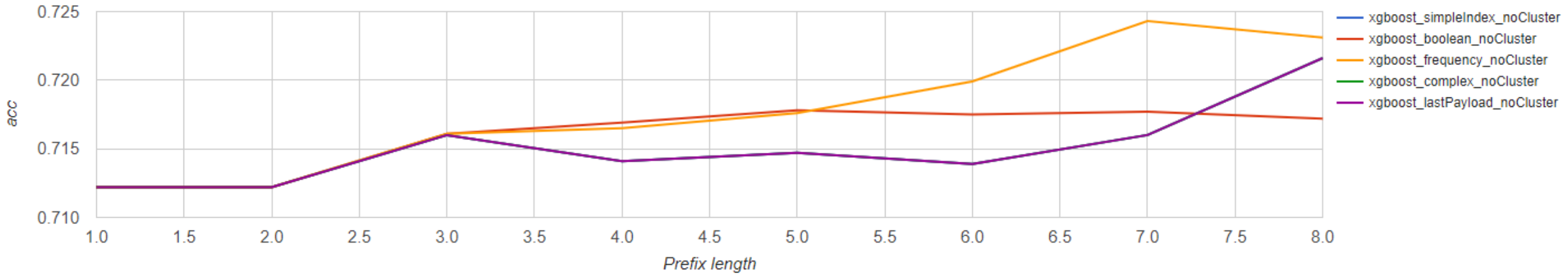
Target group	Explanation
2.1	Receiving "dagpenge" (unemployment insurance)
2.2	Receiving "kontanthjælp" or "integrationsydelse" and ready for a job[1]
2.3	Receiving "kontanthjælp" or "integrationsydelse" [2]
2.4	Limited ability to work [3]
2.5	Receiving "sygedagpenge" (illness)[4]
2.6	Receiving pension as early retired[5]
2.7	Unable to get normal job but not eligable for early retirement
2.8	Handicap and eligable for unemployment insurance[6]
2.9	Under 18 and need for help getting education or employment
2.10	Not eligable to any above
2.11	"Resourceforløb"[7]
2.12	Under 30 and ready for education
2.13	Under 30 and not ready for education
2.14	"Resourceforløb" and "jobafklaring"

Table 1. Target groups

```
<?xml version='1.0' encoding='UTF-8'?>
<log>
  <extension name="Time" prefix="time" uri="http://www.xes-standard.org/time.xesext"/>
  <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
  <trace>
    <string key="concept:name" value="7cb9e7b4-47cd-63cb-67fd-a2af5703e4ed"/>
    <event>
      <int key="weekly_hours" value="2"/>
      <date key="time:timestamp" value="2014-01-28T00:00:00.000Z"/>
      <string key="concept:name" value="Education offer: Særligt tilrettelagt projekt (LAB §32.1.2/INL §23a, §24f)"/>
    </event>
    <event>
      <int key="weekly_hours" value="37"/>
      <date key="time:timestamp" value="2017-01-01T00:00:00.000Z"/>
      <string key="concept:name" value="Education offer: AMU (LAB §32/INL §23a)"/>
    </event>
```

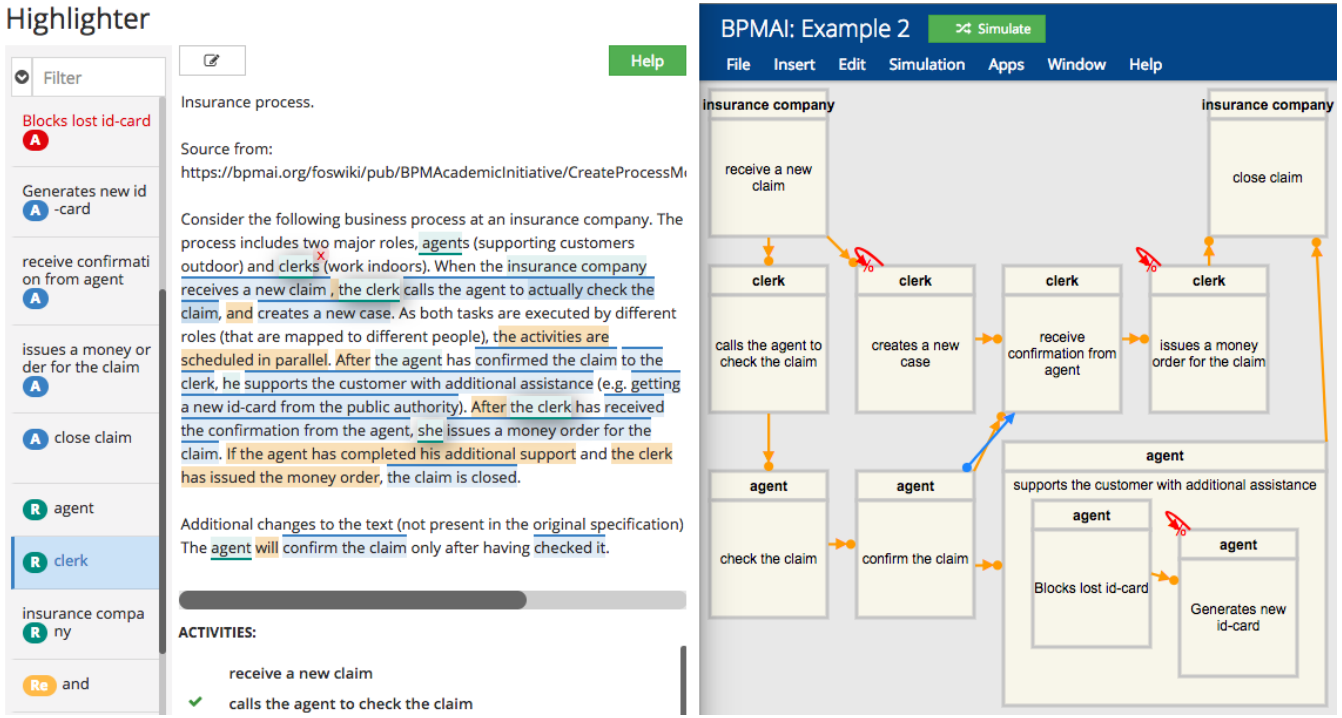
Example results

Prefix length by acc



ID	Task identity	F1 score	AUC	Accuracy	Prefix length	Precision	Recall	True positive	True negative	False positive	False negative
419	xgboost_frequency_noCluster	0.836	0.674	0.724	7	0.725	0.986	6,637	208	2,512	94
420	xgboost_frequency_noCluster	0.835	0.676	0.723	8	0.725	0.985	6,628	206	2,514	103
404	xgboost_simpleIndex_noCluster	0.833	0.678	0.722	8	0.727	0.976	6,569	251	2,469	162
428	xgboost_complex_noCluster	0.833	0.678	0.722	8	0.727	0.976	6,569	251	2,469	162
436	xgboost_lastPayload_noCluster	0.833	0.678	0.722	8	0.727	0.976	6,569	251	2,469	162
418	xgboost_frequency_noCluster	0.833	0.684	0.72	6	0.724	0.98	6,599	205	2,515	132
251	decisionTree_boolean_noCluster	0.833	0.611	0.719	7	0.722	0.985	6,627	169	2,551	104
252	decisionTree_boolean_noCluster	0.833	0.611	0.719	8	0.723	0.982	6,607	187	2,533	124
332	randomForest_boolean_noCluster	0.833	0.612	0.719	8	0.723	0.982	6,612	181	2,539	119
331	randomForest_boolean_noCluster	0.833	0.612	0.719	7	0.722	0.984	6,626	166	2,554	105

The Process Highlighter: From Texts to Declarative Processes and Back (BPM 2018)



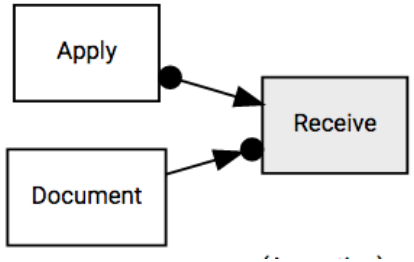
The Process Highlighter: From Texts to Declarative Processes and Back

Hugo A. López[1,2], Søren Debois[1], Thomas T. Hildebrandt[3], and Morten Marquard[2]

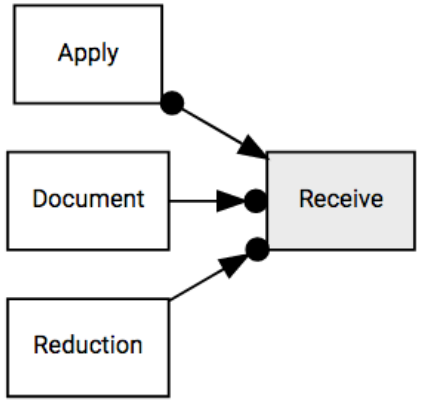
[1] IT University of Copenhagen, Denmark
{hual,debois}@itu.dk
[2] DCR Solutions A/S, Copenhagen, Denmark
{hual,mm}@dcrgraphs.net
[3] Copenhagen University, Denmark
hilde@di.ku.dk

Abstract. The adoption of formal models by process specialists has faced two challenges: First, it requires process specialists to get training in formal modeling. Second, the resulting specifications bear little resemblance wrt. the original descriptions. We introduce a tool that supports translations between natural language descriptions and declarative process models. The resulting models are given in a graphical formalism, DCR Graphs. Traceability is at the core of the tool: Later changes in the process model due to, e.g., ambiguity resolution are traced *back* into the text. This allows users to either correct and complete their descriptions, or to derive models more refined than the text. In this paper, we describe the mechanics of the tool and provide examples of its use. Finally, we report on experiences using the tool in a Danish Municipal government.

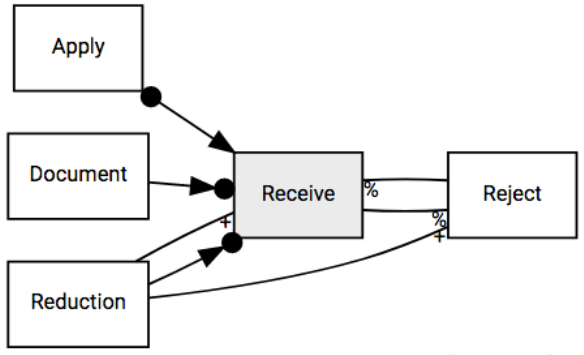
Open to Change: A Theory for Iterative Test-Driven Modelling (BPM 2018)



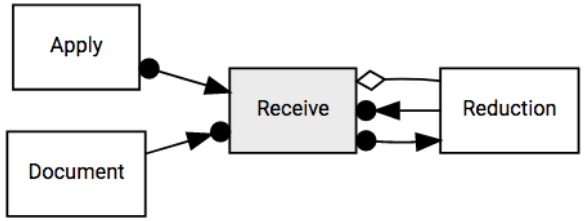
(a) I_1: Iteration 1



(b) I_2: Iteration 2



(c) I_3: Iteration 3



(d) I_3': Variant iteration 3

Open to Change:
A Theory for Iterative Test-Driven Modelling*

Tijs Slaats[1], Søren Debois[2], and Thomas Hildebrandt[1]

[1] University of Copenhagen, Denmark
slaats@di.ku.dk, hilde@di.ku.dk
[2] IT University of Copenhagen, Denmark
debois@itu.dk

Abstract. We introduce *open tests* to support iterative test-driven process modelling. Open tests generalise the trace-based tests of Zugal et. al. to achieve *modularity*: whereas a trace-based test passes if a model exhibits a particular trace, an open test passes if a model exhibits a particular trace *up to* abstraction from additional activities not relevant for the test. This generalisation aligns open tests better with iterative test-driven development: open tests may survive the addition of activities and rules to the model in cases where trace-based tests do not. To reduce overhead in re-running tests, we establishing sufficient conditions for a model update to preserve test outcomes. We introduce open tests in an abstract setting that applies to any process notation with trace semantics, and give our main preservation result in this setting. Finally, we instantiate the general theory for the DCR Graph process notation, obtaining a method for iterative test-driven DCR process modelling.

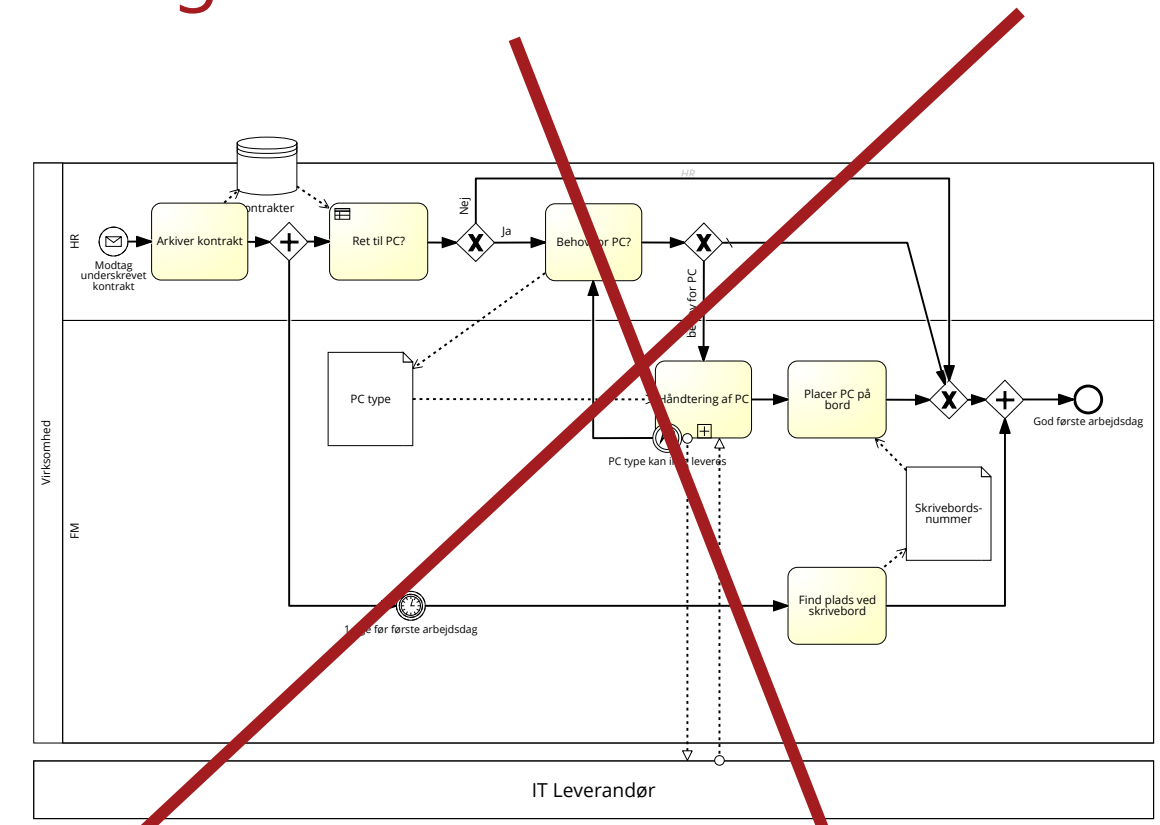
Keywords: Test-driven Modelling, Abstraction, Declarative, DCR Graphs

Taking a step back from the flow graphs: Scenario-driven Agile Declarative Digitalisation

Identify roles, activities, data

Identify constraints

Identify scenarios



Interviews, post-its, role-play, simulations

Case: Managing Social Services in Danish Municipalities

- Case work in municipalities is highly regulated by law
- The law changes often
- Current systems either do not provide support for compliance or have built in constraints that are difficult to adapt locally

Case: Digitalising Social Services Law Declaratively

Executive Order no. 1053 of 8 September 2015 (Current) | **Print-out date: 28 October 2015**

Ministry: The Danish Ministry of Social Affairs and the Interior File number: Ministry of Social Affairs and the Interior, file no. 2015-4958

Later amendments to the regulation ACT No. 495 of 21 May 2013 ACT No. 722 of 25 June 2014

Consolidation Act on Social Services

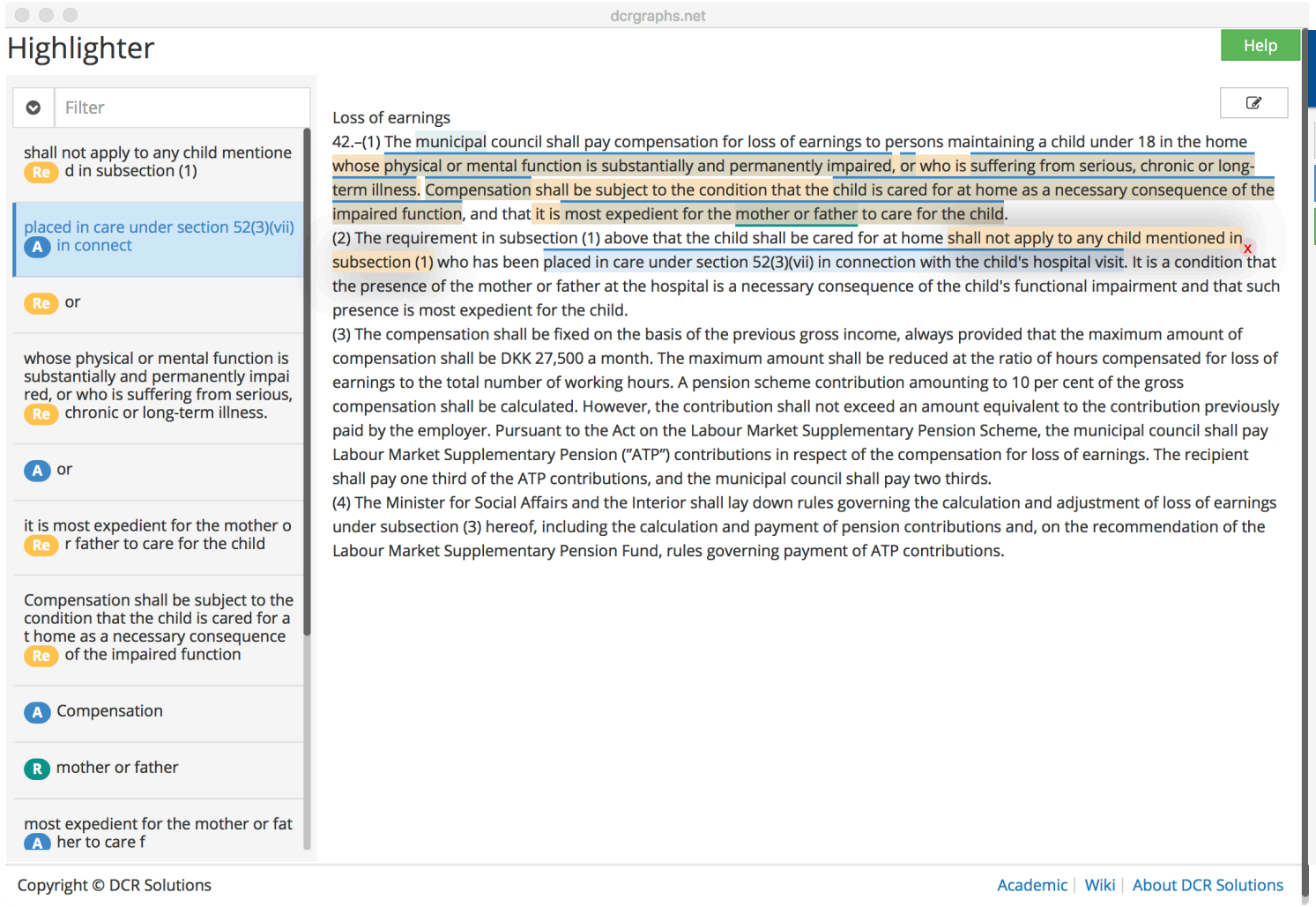
Loss of earnings

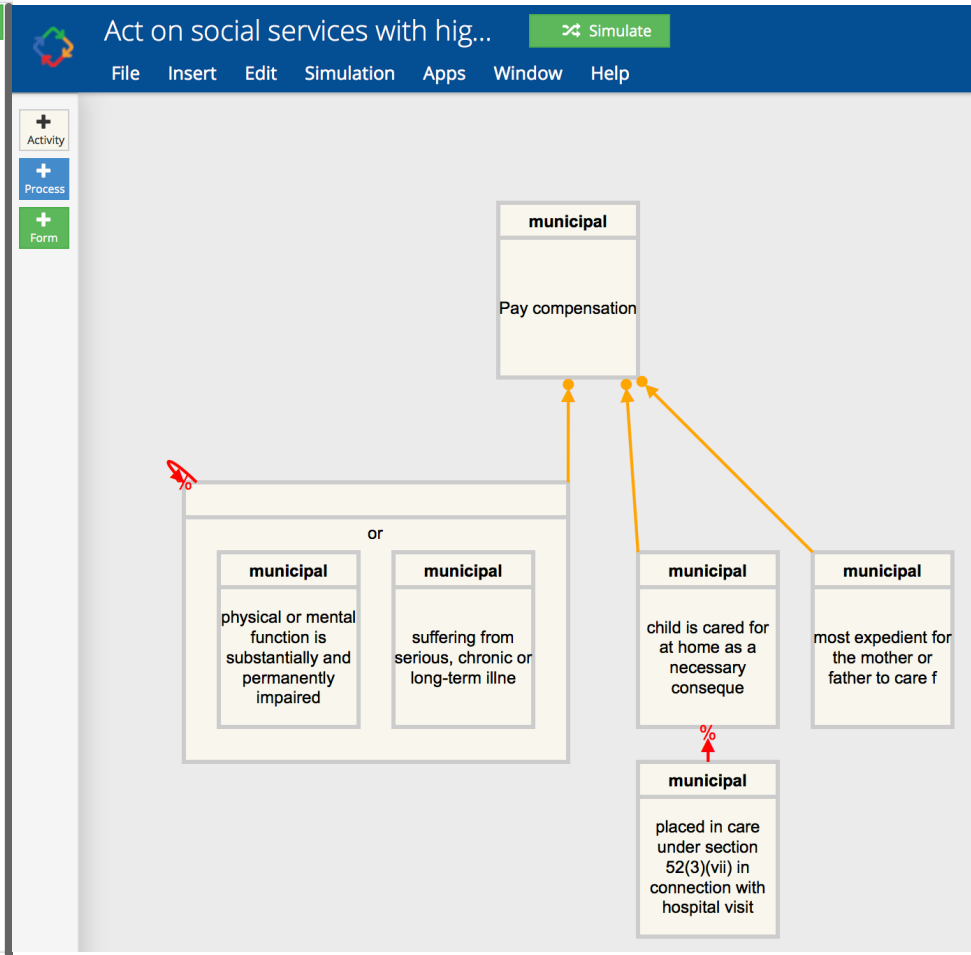
42.–(1) The municipal council shall pay compensation for loss of earnings to persons maintaining a child under 18 in the home whose physical or mental function is substantially and permanently impaired, or who is suffering from serious, chronic or long-term illness. Compensation shall be subject to the condition that the child is cared for at home as a necessary consequence of the impaired function, and that it is most expedient for the mother or father to care for the child.

(2) The requirement in subsection (1) above that the child shall be cared for at home shall not apply to any child mentioned in subsection (1) who has been placed in care under section 52(3)(vii) in connection with the child's hospital visit. It is a condition that the presence of the mother or father at the hospital is a necessary consequence of the child's functional impairment and that such presence is most expedient for the child.

(3) The compensation shall be fixed on the basis of the previous gross income, always provided that the maximum amount of compensation shall be DKK 27,500 a month. The maximum amount shall be reduced at the ratio of hours compensated for loss of earnings to the total number of working hours. A pension scheme contribution amounting to 10 per cent of the gross compensation shall be calculated. However, the contribution shall not exceed an amount equivalent to the contribution previously paid by the employer. Pursuant to the Act on the Labour Market Supplementary Pension Scheme, the municipal council shall pay Labour Market Supplementary Pension (”ATP”)

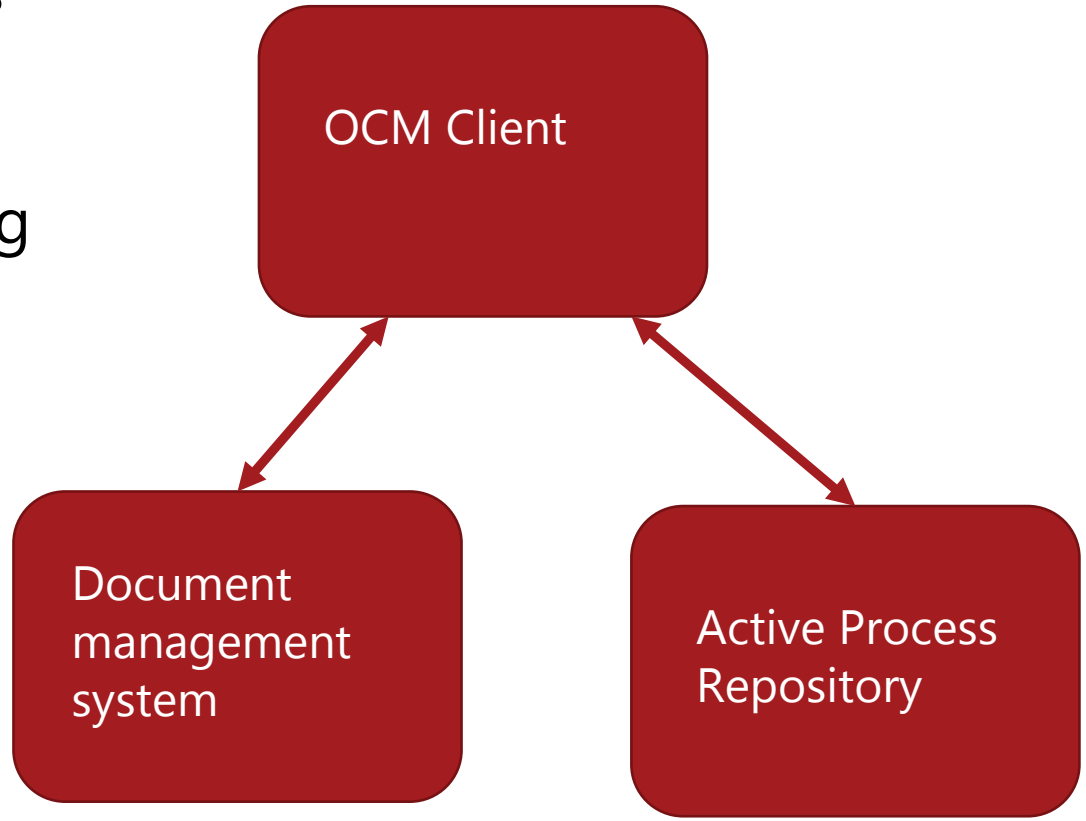
DCRGraphs.net highligher tool





Open Case Manager (OCM)

- Lightweight open source process manager client interacting with existing systems, e.g. document management (Elektronisk Sags og Dokument Håndtering - ESDH) systems and an active process repository, e.g. DCR engine and repository



https://github.com/DCRGraphsNet/DCROpenCaseManager/wiki/

http://opencasemanager.azurewebsites.net/

http://opencasemanager.azurewebsites.net/Admin/Processes/

Exercises

- Form groups of 3-4
- In groups (with help from TA and lecturer)
  - Discuss and draw informal pictures of the application and problem domain for
    - Keeping track of time
    - Municipal case management
  - What are the relevant concepts the application and problem domain of municipal case management ?

Cross organisational

- Share processes but limiting access to activities (e.g. using roles) and separating personal data from process data.