Mission Invitée supported by IUT Nantes

The Art of Decision Making in Software Engineering

Dr. Joost Noppen, School of Computing Sciences
University of East Anglia

Decision Making in Software Engineering

Software Engineering/Development

Hundreds of decisions

Which language?

Which algorithm?

Which architectural style?

How to support extension?

...

```
OUTPUT_BYTE(h>>BYTE_SIZE);
OUTPUT_BYTE(h);
OUTPUT_BYTE(w>>BYTE_SIZE);
OUTPUT_BYTE(w);

 order1 = order>>4;
 order2 = order & 15;

#ifdef TRACE
    if (!(fpm = fopen("ppmenc.doc", "wb"
    {
        fprintf(stderr, " \n Error: Ca
        exit(2);
    }
#endif

        /* allocate 'order+1' elements
                       is used to sto
```

Decision Making in Software Engineering

Software Engineering/Development

Decisions vary in magnitude of impact

Some decisions are fairly minor

- SVN vs. GIT
- Eclipse vs. NetBeans vs. Visual Studio
- Which coffee brand

Others can be with you for a long time

- Which platform to use
- Which future extensions to support
- Which budgets (to ask for)

It is the latter ones you want to get right

Decision Making in Software Engineering

"OK, all those in favour of delegating decision-making, shrug your shoulders"

In short, decisions can be very important

Some decisions may stay with you for the complete lifespan of your system

Modern developments (Software Product Lines, Complex Systems) therefore put pressure on decision making

The longer the life span, the higher the impact when you get it wrong

It may bread indecisiveness...

… or deflecting and postponing decisions

Decision Making in Software Engineering

Good vs. Bad Decisions

So what makes a decision good and what makes it bad?

Is bad all about getting it wrong?

- Predicting the wrong evolutions
- Underestimating costs
- …

And is good all about getting it right?

Perhaps, it is certainly a viewpoint that directly relates to success or failure of a software system.

But it means the quality of your decision making depends on your ability to predict the future.

I prefer to say that decisions are good if they are taken from an informed point of view. Bad are decisions taken from a confused point of view.

Decision Making in Software Engineering

Good vs. Bad Decisions Illustration

Suppose we are playing a simple gambling game

We get to bet four 1 pound coins on the outcome of the roll of a die

If we get it right, we get three times our bet

If we get it wrong, all money is lost

We can play as many times as we want

Bad Decision

Putting all your money on a single face every time.

You might get lucky, but you loose in the long run. You would win once every six times, losing 20 pounds and winning 12 pounds.

Good Decision

Putting one coin on four different faces

You might get unlucky, but you win in the long run.

You would win four times out of six, losing 8 pounds and winning 12 pounds.

The Anatomy of a Decision

What does a decision entail?

A decision is typically seen as selecting a course of action when faced with a particular choice...

… from a set of **alternative actions** …

… given a set of **objectives** ...

… aided by a certain **body of knowledge** …

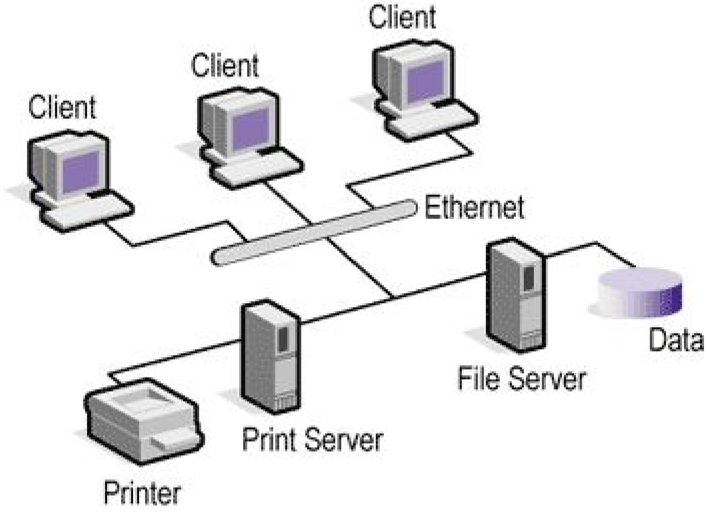
… a level of insight and **experience** ...

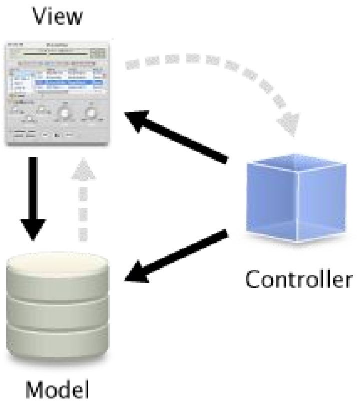
… and where needed a set of **assumptions**

The goal of decision making then?

The goal is to select the course of action that best satisfies the objectives. To determine this, you can use the knowledge, experience and assumptions.

The Anatomy of a Decision





A software architecture decision

Alternative actions: MVC or Client-Server

Objectives: Performance and Evolvability

Body of knowledge: books, ...

Experience: Similar systems we worked on

Assumptions: performance is this, evolvability that

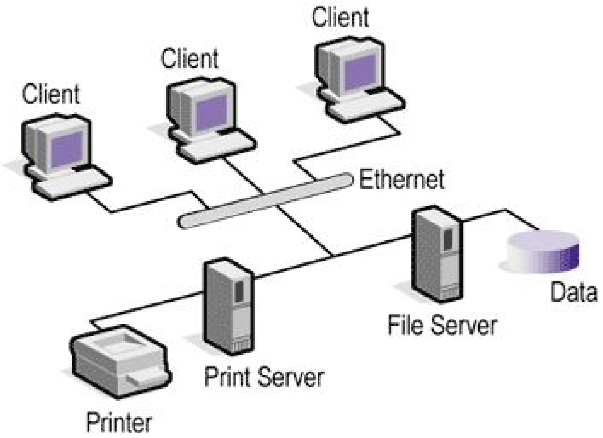
So where does it go wrong?

All too frequently we lack substantial knowledge and we have to fill the gap with experience and assumptions.

To make matters worse, alternatives, objectives and what we know is not always accurate and complete.

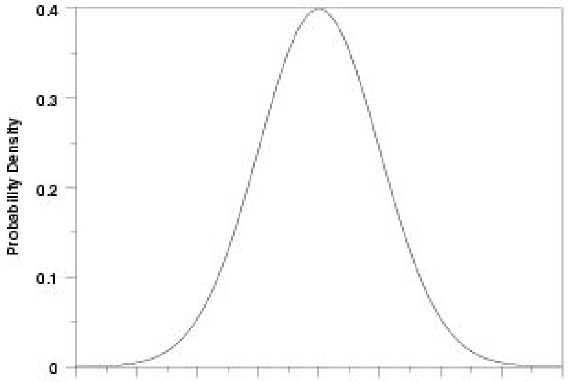
In short, we end up in a state of confusion (even if we do not realise it). And this leads to bad decisions.

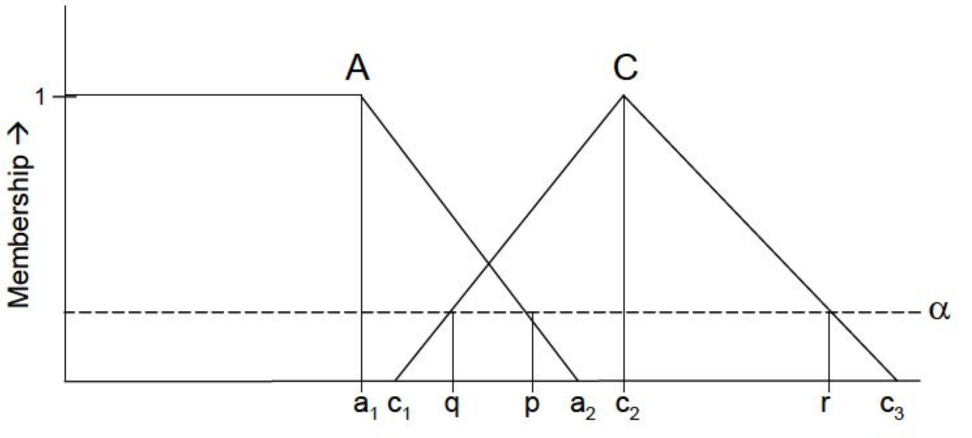
Modelling and Analysis of Information



Modelling our information accurately

- Not knowing everything is not a bad thing
- As long as we are certain about our uncertainty
- For example, consider the response time in a client-server architecture system
- A single number might not be possible, but a probability distribution would do
- Similarly, a vague budget objective and estimated cost can accurately be described using a fuzzy set





Modelling and Analysis of Information

Benefits:

- Sound insight into risk
- Mostly automatable
- Takes away the sting of intuitive assumptions

Example 1: Tossing a Coin

Suppose you have tossed a coin 99 time and every single time it came out heads. What is the chance of it coming up heads again?

Example 2: Illness and Test

Suppose in a population there is an illness that will be fatal in a week and affects 10% of the people. Also suppose there is a test that is accurate 90% of the time.

If you would receive a positive test result, what is the probability of survival for longer than a week?

Answer: 50%!

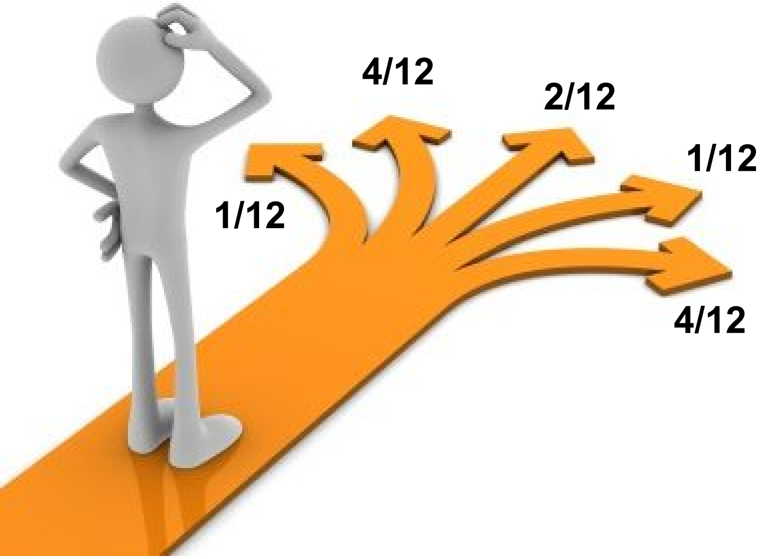
Initially, you will be in one of four groups:

- Ill and test correct (1/10 * 9/10 = 9/100)
- Ill and test wrong (1/10 * 1/10 = 1/100)
- Well and test correct (9/10 * 9/10 = 81/100)
- Well and test wrong (9/10 * 1/10 = 9/100)

Once you know the test is positive:

- Ill and test correct (1/10 * 9/10 = 9/100)
- Well and test wrong (9/10 * 1/10 = 9/100)

Modelling and Analysis of Information



A better insight on the credibility of our information

We now have insight into the risks that come with a specific path of action

Risks that arise from objectives, assumptions, information can be treated uniformly

And we can include that accordingly in our assessment

But what if there is no clear winner?

Systematic support for deferring risky decisions

Commit to more than one path of action

Higher workload but lower risk

Requires modelling extensions

Deferring Decision that are Too Risky

From and Object-Oriented point of view, nothing new

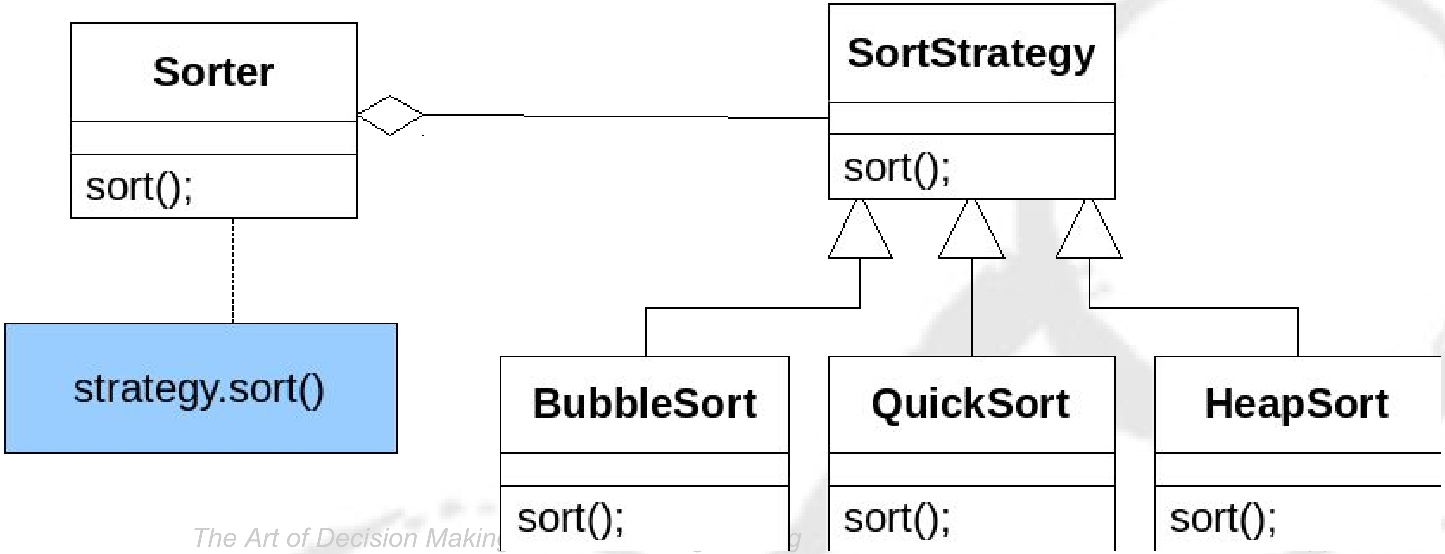
The bread and butter, design patterns, plugins, etc.

Requires planning and insight, some knowledge of the future

And it is expensive if you predict it wrong

More importantly, it needs support on more levels

Requirements, feature modelling, architecture, etc.



Leveraging Experience

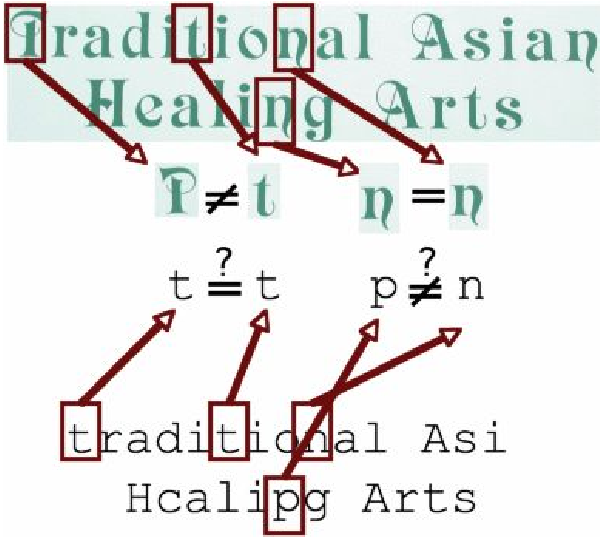
And what about the knowledge already out there?

- How do we use that to our advantage?
- Experts in your team are very useful
- Domain specific approaches are gaining momentum
- The internet with fora contain a wealth of knowledge

But there is more...

- Similar projects by you and others
- Best practices
- The question of how to capture the knowledge of experts

Leveraging Experience



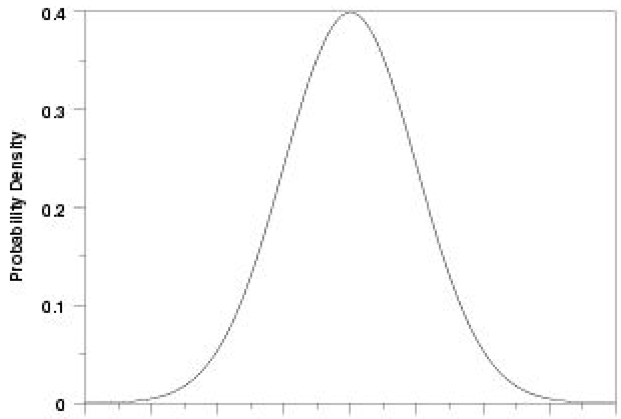
Learning from the past, not that easy

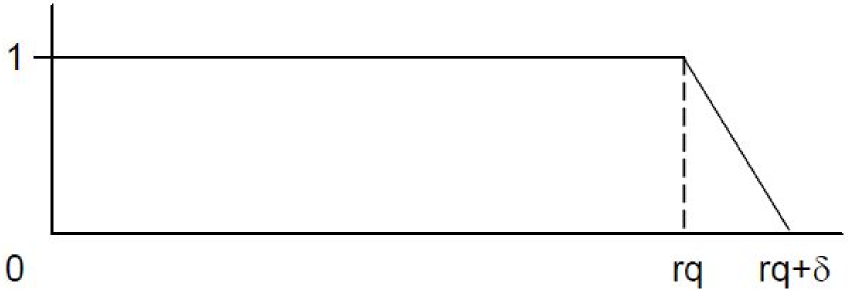
- If you want to leverage from existing projects, there are some issues to resolve
- Different project have different focus and terms
- Bridge the differences, establish similarities using
  - Natural Language Processing
  - Graph matching
  - …
- Identify which parts can be of use to you
  - Manually
  - Automatically: pattern recognition, data mining, etc.

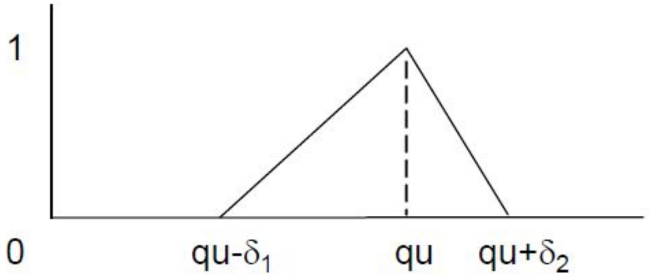
Example Approaches

Modelling Vague Requirements and Estimations

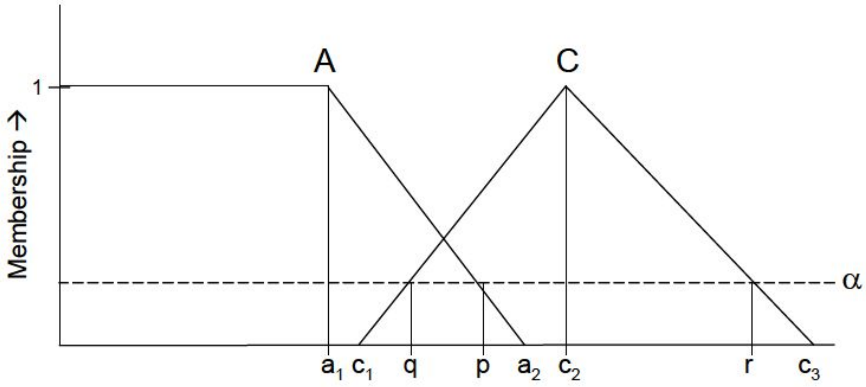
- Sometimes it is not possible to provide strict requirements on budget, performance, etc.
- Equally, you might not be certain about the exact performance and cost that can be expected.
- How would you assess alternatives without making strong assumptions and falling into the confused decision trap?







Example Approaches

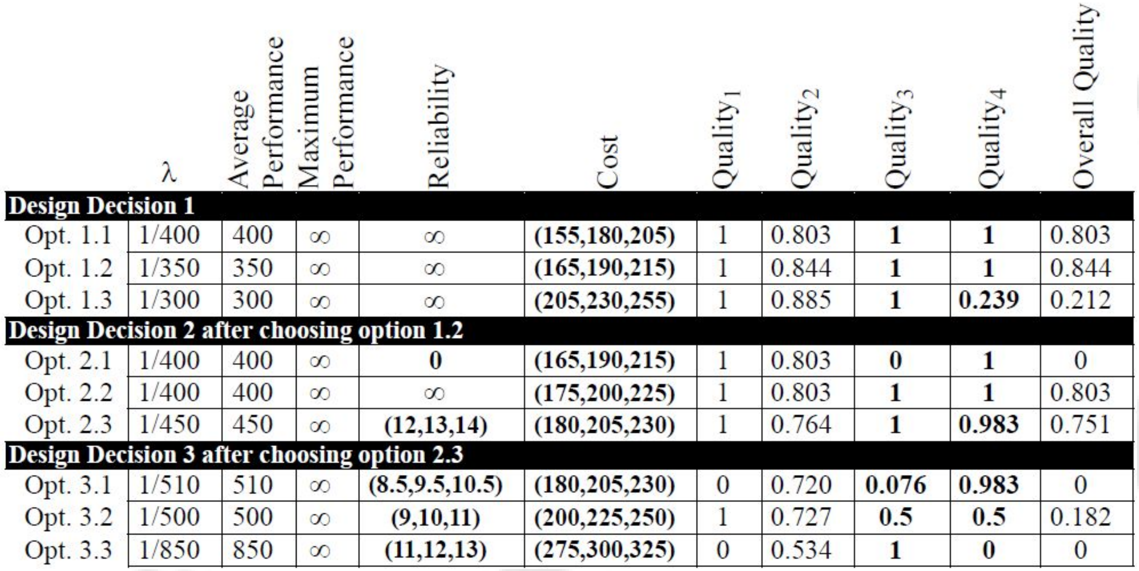


How to analyse?

- How do you compare fuzzy estimations and fuzzy requirements?
- Or even fuzzy estimations and probabilistic requirements?
- A new analysis method was needed, a specialised comparison operator

And the results?

- Vagueness in requirements and estimations captured
- Evaluation results in a risk indicator for more insight
- Automated support for calculations

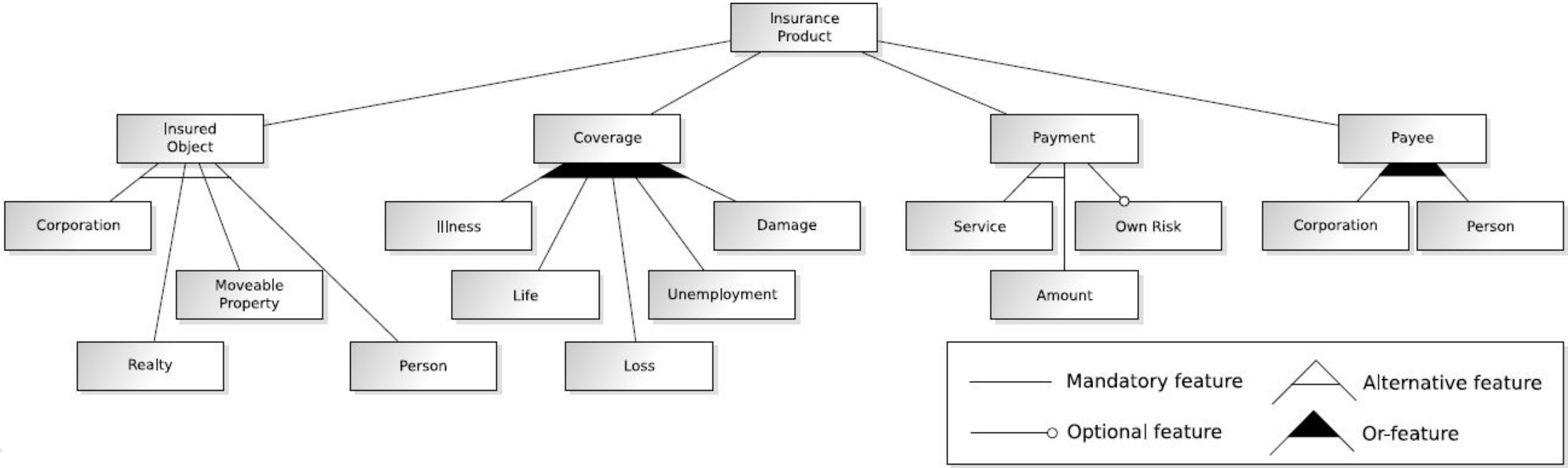


	λ	Average Performance	Maximum Performance	Reliability	Cost	Quality_1	Quality_2	Quality_3	Quality_4	Overall Quality
Design Decision 1										
Opt. 1.1	1/400	400	∞	∞	**(155,180,205)**	1	0.803	**1**	**1**	0.803
Opt. 1.2	1/350	350	∞	∞	**(165,190,215)**	1	0.844	**1**	**1**	0.844
Opt. 1.3	1/300	300	∞	∞	**(205,230,255)**	1	0.885	**1**	**0.239**	0.212
Design Decision 2 after choosing option 1.2										
Opt. 2.1	1/400	400	∞	**0**	**(165,190,215)**	1	0.803	**0**	**1**	0
Opt. 2.2	1/400	400	∞	∞	**(175,200,225)**	1	0.803	**1**	**1**	0.803
Opt. 2.3	1/450	450	∞	**(12,13,14)**	**(180,205,230)**	1	0.764	**1**	**0.983**	0.751
Design Decision 3 after choosing option 2.3										
Opt. 3.1	1/510	510	∞	**(8.5,9.5,10.5)**	**(180,205,230)**	0	0.720	**0.076**	**0.983**	0
Opt. 3.2	1/500	500	∞	**(9,10,11)**	**(200,225,250)**	1	0.727	**0.5**	**0.5**	0.182
Opt. 3.3	1/850	850	∞	**(11,12,13)**	**(275,300,325)**	0	0.534	**1**	**0**	0

Example Approaches

Examining Feature Models for Advice

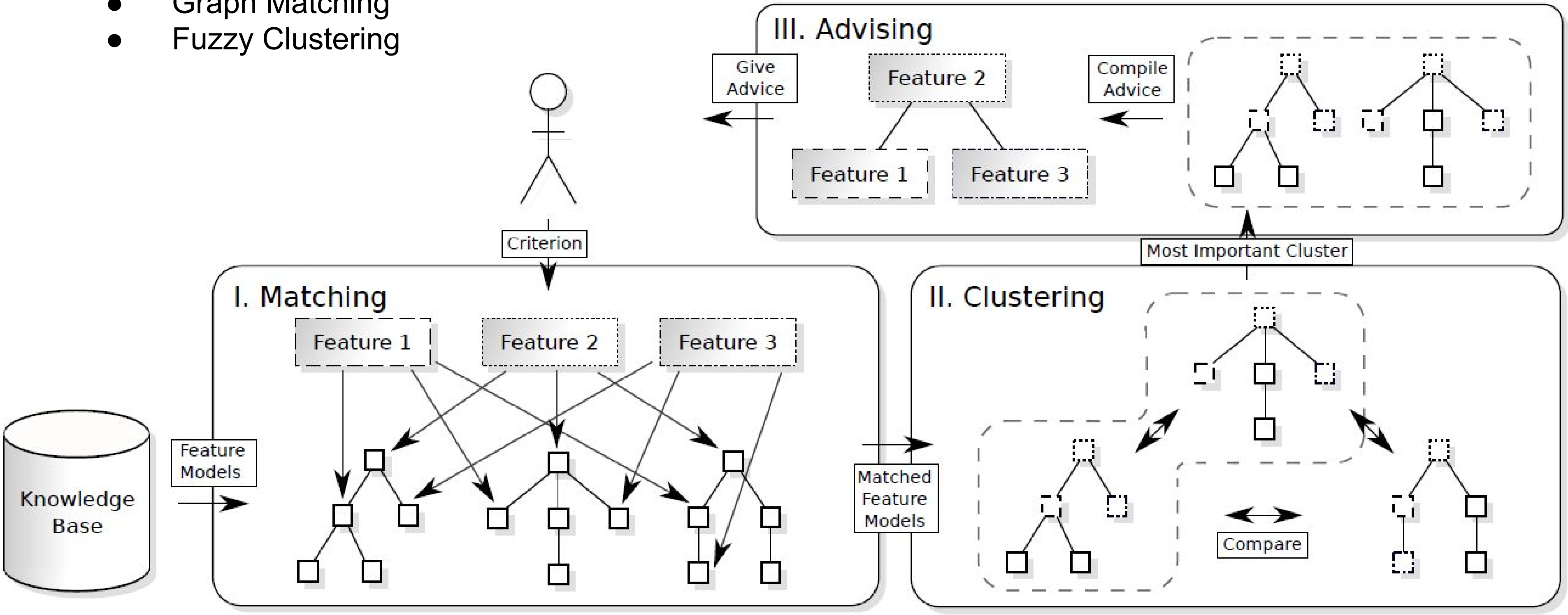
- When building a new Software Product Line, one might take inspiration from others
- By looking at models from existing, similar systems, you can identify recurring patterns
- So how to do that?



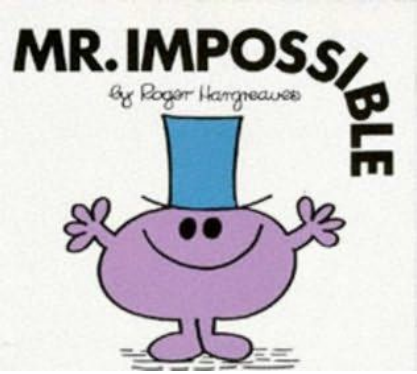
Example Approaches

A combination of techniques

- Natural Language Processing
- Graph Matching
- Fuzzy Clustering



Not Just Ideas, We Need Tools!



The typical research my group tends to...

- … focuses on trade-offs in software development
- … involves interpreting historical data
- … builds on connections between development artefacts (ie traces)
- … deals with natural language as well as formalised models
- … requires defining partial knowledge representations
- … as well as logical reasoners that can interpret these

The tools that result have to provide…

- … sophisticated optimisation implementations for the trade-offs
- … visualisations of the results to aid the developer
- … integration with industry standard tools and formats
- ... and it needs to return the results fast, no waiting around for days

And finally it needs to get in front of developers fast!!

The Vision of a Software Engineering GPS

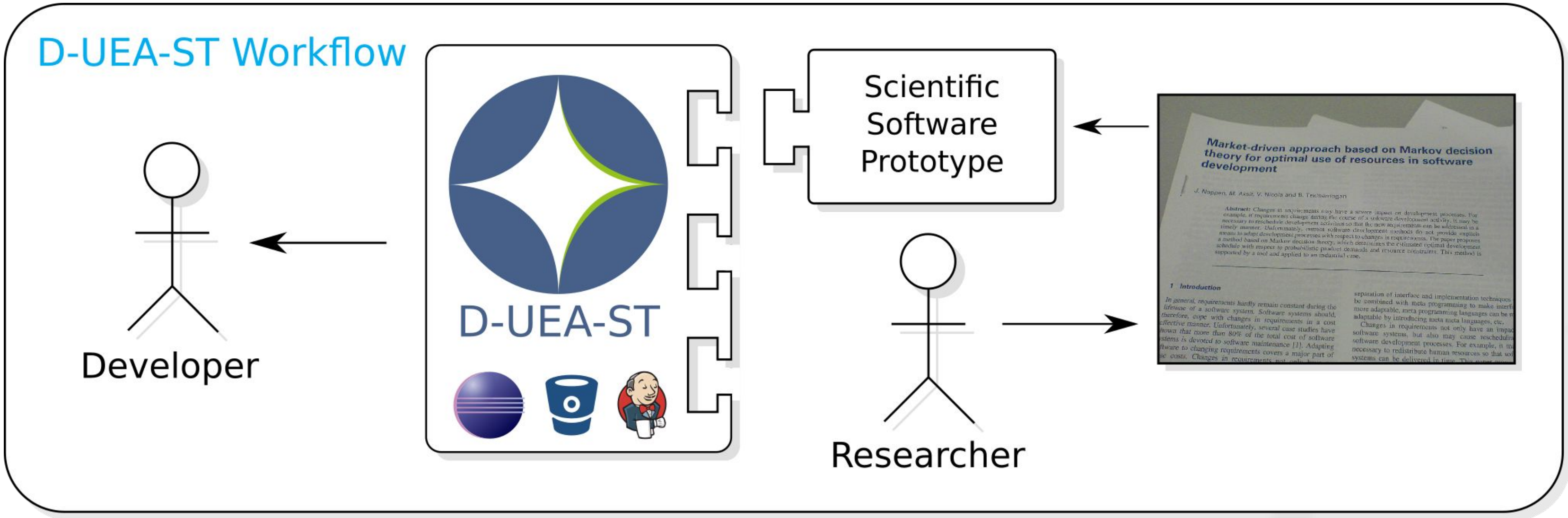
What would be the ideal?

- The integrated tools must be a decision support approaches (SatNav) for developers
- It should take in data, run a bespoke analysis algorithm and visualise the results
- It can be triggered and driven by the user or do this autonomously

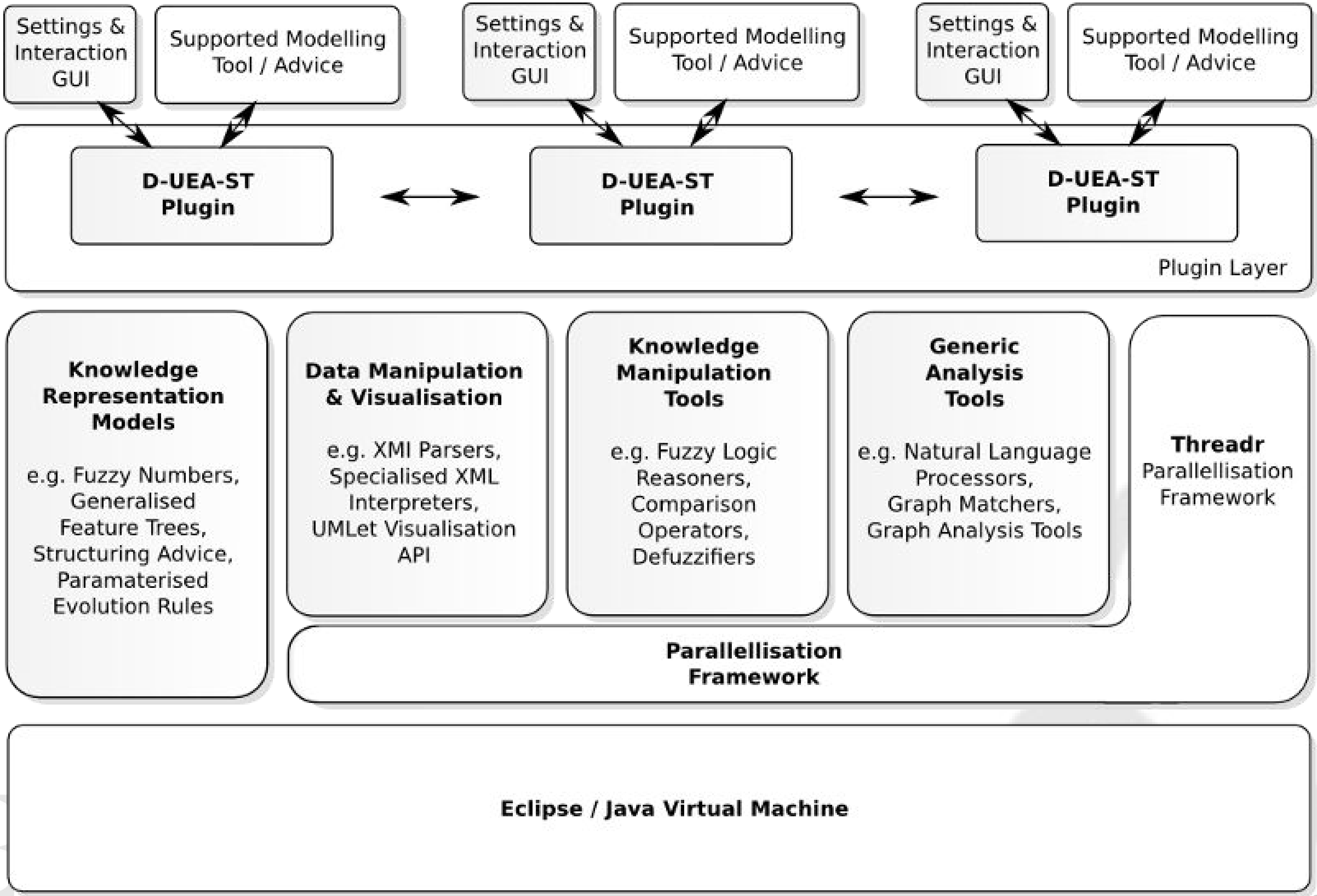
An architecture that allows for easy creation of such tools can support:

- Sharing of models and analysis algorithms
- Sharing of visualisation mechanisms
- Sharing of standards integratio
- Uniform parallellisation of tasks

The Vision of a Software Engineering GPS

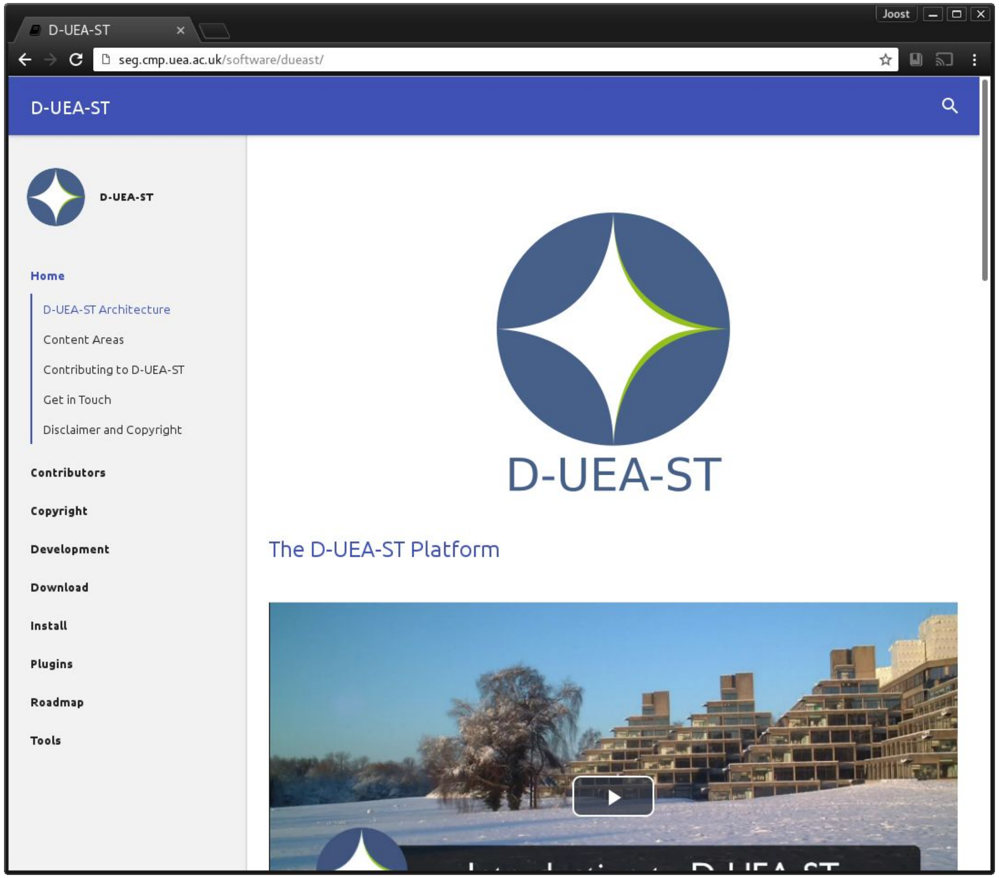

D-UEA-ST Workflow
Developer
D-UEA-ST
Scientific Software Prototype
Researcher
Market-driven approach based on Markov decision theory for optimal use of resources in software development

D-UEA-ST Plugin Architecture



Settings & Interaction GUI
Supported Modelling Tool / Advice
Settings & Interaction GUI
Supported Modelling Tool / Advice
Settings & Interaction GUI
Supported Modelling Tool / Advice
D-UEA-ST Plugin
D-UEA-ST Plugin
D-UEA-ST Plugin
Plugin Layer
Knowledge Representation Models
e.g. Fuzzy Numbers, Generalised Feature Trees, Structuring Advice, Paramaterised Evolution Rules
Data Manipulation & Visualisation
e.g. XMI Parsers, Specialised XML Interpreters, UMLet Visualisation API
Knowledge Manipulation Tools
e.g. Fuzzy Logic Reasoners, Comparison Operators, Defuzzifiers
Generic Analysis Tools
e.g. Natural Language Processors, Graph Matchers, Graph Analysis Tools
Threadr Parallellisation Framework
Parallellisation Framework
Eclipse / Java Virtual Machine

The Vision of a Software Engineering GPS



D-UEA-ST: A Flexible Reuse Platform

- Architecture aimed at the creation and support of decision support for software engineering
- Currently runs as an Eclipse Plugin
- Multiple plugins created as a result of research.

- Continuous integration and deployment for collaboration and easy feedback
- Usable as a delivery mechanism and student projects

http://seg.cmp.uea.ac.uk/software/dueast