

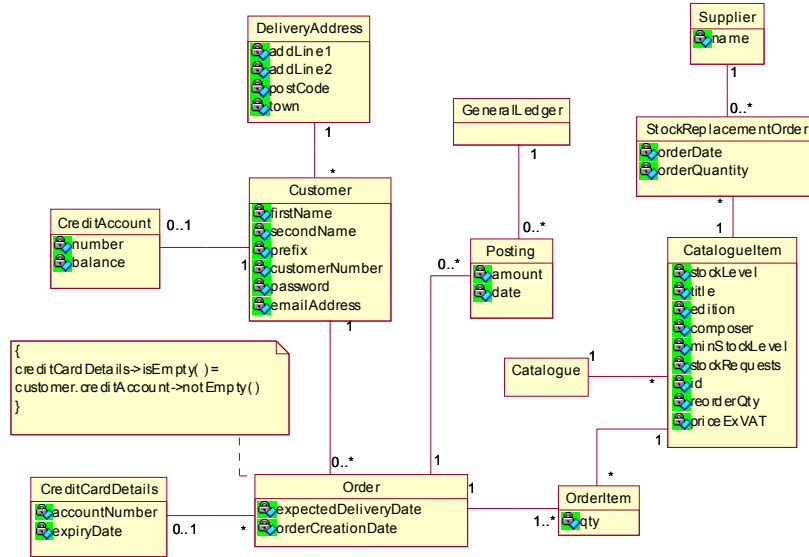
# A Survey of Modelling

John Daniels  
Syntropy Limited, UK  
[www.syntropy.co.uk](http://www.syntropy.co.uk)

© Syntropy Limited 2005

You all know what modelling is...

## Spend a lot of time drawing loads of these...



www.syntropy.co.uk

Syntropy Limited

## ..throw them away and get on with the code

- But wait!
- Isn't the code also a model?
- So what exactly do we mean by "model"?

```

package com.fastnloose.utilities;
/**
 * @author danielsj
 */
public class Stopwatch {

    protected long display, snap;

    /**
     * Constructor for Stopwatch.
     */
    public Stopwatch() {
        super();
        display = 0;
        snap = 0;
    }

    public void start() {
        snap = System.currentTimeMillis();
    }

    public long stop() {
        display = display +
            (System.currentTimeMillis() - snap);
        snap = 0;
        return display;
    }

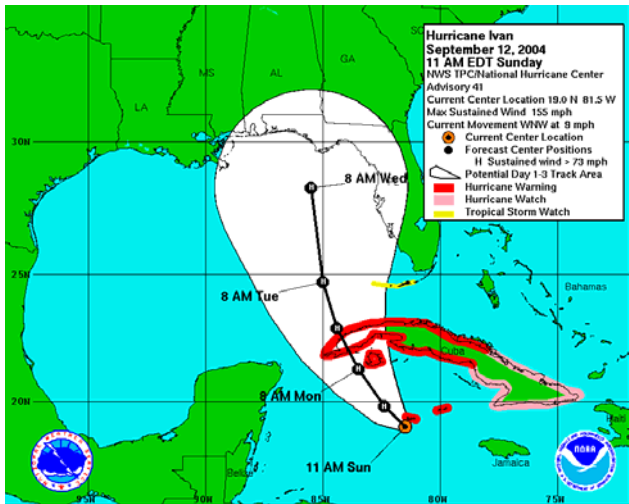
    public void reset() {
        display = 0;
    }
}

```

www.syntropy.co.uk

Syntropy Limited

Are our models "mathematical models"?



www.syntropy.co.uk

Syntropy Limited

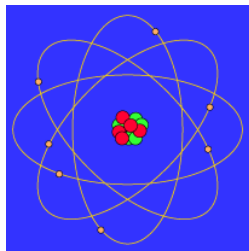
Are they "scale models"?



www.syntropy.co.uk

Syntropy Limited

Are they "theoretical models"?



[www.syntropy.co.uk](http://www.syntropy.co.uk)

**Syntropy Limited**

Definitely not this kind of model...



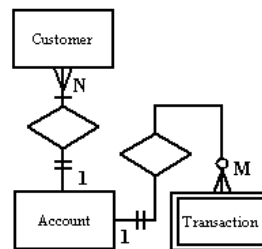
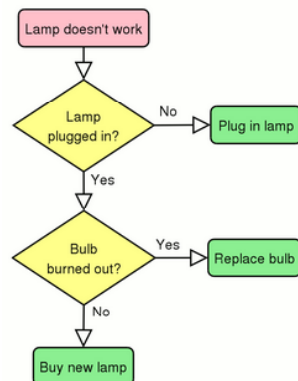
[www.syntropy.co.uk](http://www.syntropy.co.uk)

**Syntropy Limited**

## A software "model" is

- partly about visualisation
  - abstraction for clarity
  - pictures worth thousands of words
- sometimes about algorithms
  - expression in a convenient form
- often not really a model at all, more a set of instructions
  - a recipe

## Modelling has a long history in software

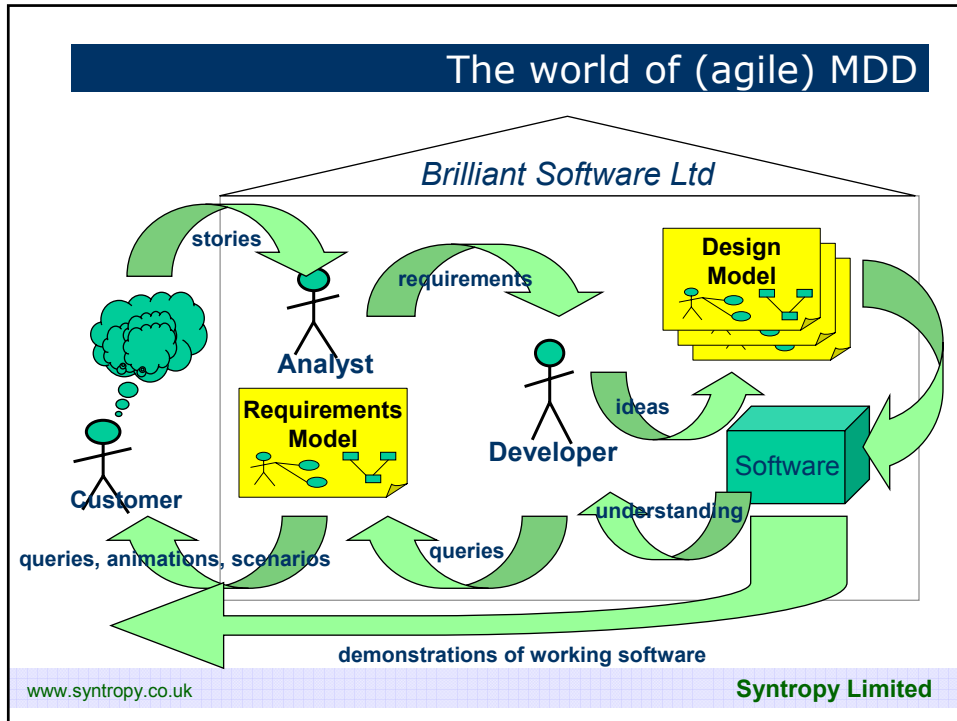


## So why the interest now?

- Cost, Complexity and Communication
  - Can't understand large systems at the code level
  - Systems are increasingly about multiple interacting processes - far beyond the "design of the program"
  - Much coding is repetitive and inefficient
  - Teams need a shared understanding
  - Better communication with the customer

## "Model-Driven Development"

- MDD puts (graphic, abstract) models at the centre of the development process
  - Use of models to allow customers, analysts and developers to visualise, share and discuss requirements, design ideas, solutions
  - Use of models to allow newcomers to see the big picture quickly
  - Need tools that can move quickly and easily between code and diagrams



- ## Reasons why modelling is working now
- A better understanding of the purpose of models
  - Standardisation of modelling languages
  - Advances in automated model transformations
- www.syntropy.co.uk Syntropy Limited

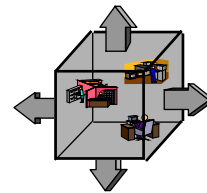
## Perspectives and intentions

## Model perspectives

	Conceptual Model	Software Specification Model	Software Implementation Model
Concerned with:	How the problem domain operates	Required behaviour of software	Internal design of software

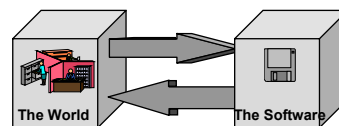
## What is the nature of the problem?

- Often we simply want ways to understand a complex situation
- We can build a model for this!
- This is nothing to do with software design
- Examples:
  - Corporate information models
  - Business processes
- **Conceptual models**



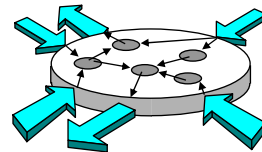
## What should this system do?

- Part of requirements definition is specifying:
  - what information the system must manage
  - how the system should react to inputs
- We can build a model for this!
- Says what, not how
- Examples:
  - Data models
  - Stimulus/response mechanisms
- **Specification models**



## How is this system constructed?

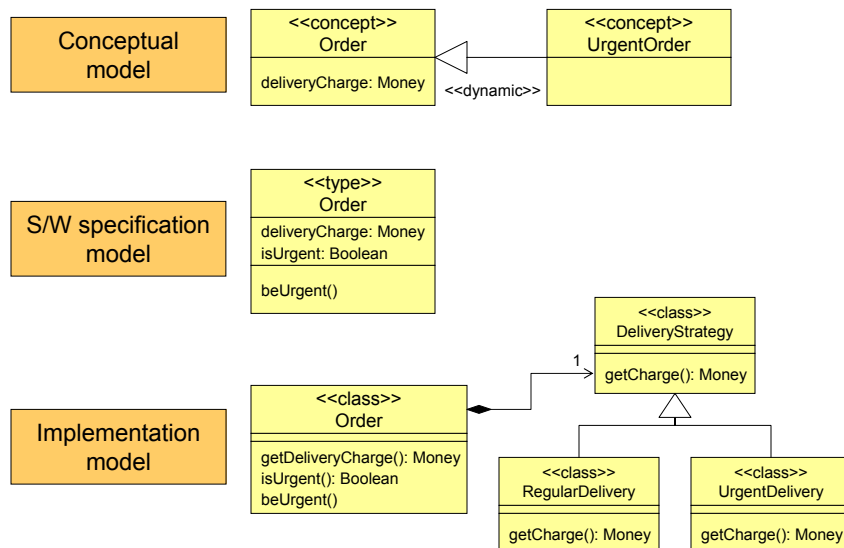
- Technicians often need ways of understanding how a system has been built or is to be built
- We can build a model for this!
- The content of these models is influenced by the implementation technology
- Examples:
  - Software structure models
  - Control flow models
- **Implementation models**



www.syntropy.co.uk

Syntropy Limited

## Same name, different perspective



www.syntropy.co.uk

Syntropy Limited

## Model intentions

	Intention:
Sketch	Informal, partial, not designed for automation. Intended as human-to-human communication.
Blueprint	Formal, fairly complete, precise. Intended to be translated, manually or automatically, into another blueprint or a program.
Program	Intended to be executed by a "machine", not translated into another model. Takes inputs, generates outputs.

## Put the two dimensions together...

	Conceptual Model	Software Specification Model	Software Implementation Model
Sketch	Description of a problem situation	(partial) Description of some s/w that is to be built	Ideas about how some s/w was/should be built
Blueprint	Spec. for a problem situation	Spec. for some s/w that is to be built	Spec for how some s/w was/should be built (e.g. rev. eng.)
Program	Simulation of a problem situation	Simulation of some s/w that is to be built <b>OR</b> Software that can be used directly (with a <b>very</b> smart machine)	Software that can be used directly

## Modelling Languages

## UML

### Unified Modeling Language

- The UML is a standardised language for describing the structure and behaviour of things
- UML emerged from the world of object-oriented programming
- UML has a set of notations, mostly graphical
- There are tools that support some parts of the UML

## UML is

- big
  - notations for classes, components, composites, deployment, activities, interactions, collaborations, use cases, state machines, etc., etc; it's own textual logic language (OCL); profiles for everything you can imagine; etc., etc.
- general purpose
  - can be used for anything, but...
  - has to be tailored for specific usages
- increasingly positioned for *blueprints* and *programs*
- a standard!

## Common usage of UML notations

	Conceptual	S/W spec	Implementation
Use case		boundary interactions	
Class diagram	information models	class/component structures	class/component structures
Seq./comm. diagram		required object interactions	designed object interactions
Activity diagram	business processes		algorithms
Statechart		object lifecycles	object lifecycles

## Architectural Description Languages

- Special-purpose languages for describing systems
  - components
  - connectors
  - configurations
- E.g.: ACME, AESOP, UniCon, Wright, Darwin
  - tend to combine text with diagrams
- Industry isn't interested, and
- UML is now positioned in this space

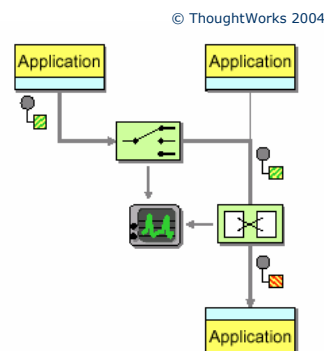
see <http://sunset.usc.edu/~nenopapers/TSE-ADL.pdf> for a survey

[www.syntropy.co.uk](http://www.syntropy.co.uk)

**Syntropy Limited**

## "Gregor-grams"

- Rare example of a useful ADL
- A language for Enterprise Application Integration
- Devised by Gregor Hohpe, ThoughtWorks
- Highly specialised and very visual

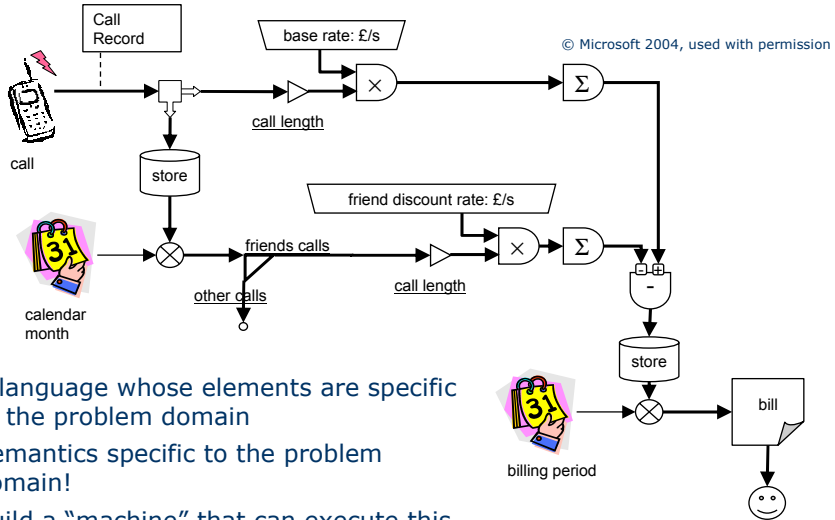


see [www.eaipatterns.com](http://www.eaipatterns.com)

[www.syntropy.co.uk](http://www.syntropy.co.uk)

**Syntropy Limited**

## Domain-Specific Languages



[www.syntropy.co.uk](http://www.syntropy.co.uk)

Syntropy Limited

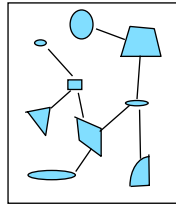
## Model Transformations

[www.syntropy.co.uk](http://www.syntropy.co.uk)

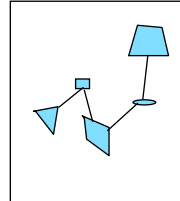
Syntropy Limited

## We want seamlessness...

a situation in the world



software that supports the situation



The basic principle of object-oriented design:

Objects identified in a situation in the world can be directly represented as software objects in a computer system that interacts with the situation

[www.syntropy.co.uk](http://www.syntropy.co.uk)

Syntropy Limited

## but how do we obtain it?

The great 1980's debate

**Elaboration**

**versus**

**Transformation**



The model of the problem is refined into a design for implementation

The problem model and the implementation design are based on quite different sets of concepts

[www.syntropy.co.uk](http://www.syntropy.co.uk)

Syntropy Limited

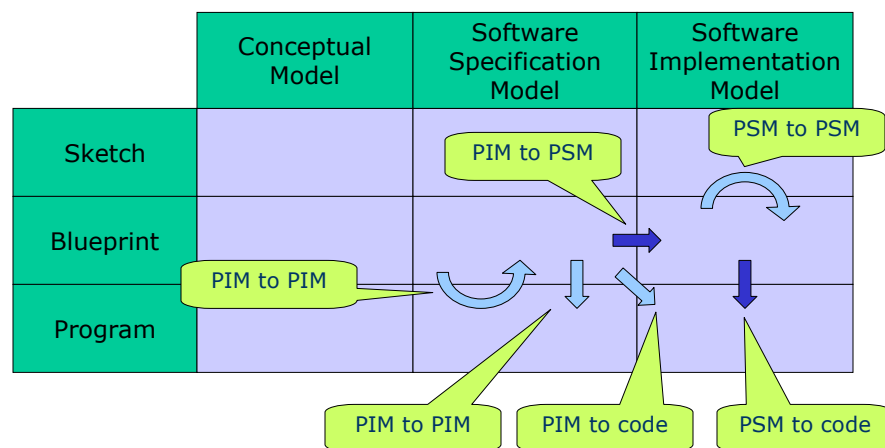
## The result?

### Transformation won

because:

- The world isn't software
- We want portability (or so we say...)
  - the OMG MDA story

## Transformations in MDA



that was a  
**Survey of Modelling**

John Daniels  
Syntropy Limited, UK  
[www.syntropy.co.uk](http://www.syntropy.co.uk)