Model Validation
Gain without Pain

MDA Implementer's Workshop
Orlando, May 2004

Ashley McNeile
Metamaxim Ltd.
www.metamaxim.com

Agenda

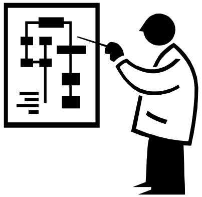
- The Three Ages of System Development
- A Different Perspective on MDA
- Model Validation in Practice
- Modelling Techniques
- Closing Remarks

The Three Ages

- 1970s
  - First big systems
  - “We have a problem”
  - First ideas of method

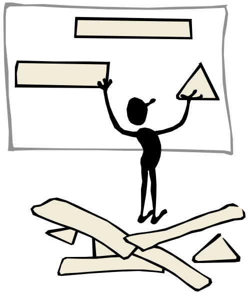
Honeymoon

- 1980s and 1990s
  - High Ceremony
  - “Squash the Chaos”
  - Analysis Paralysis



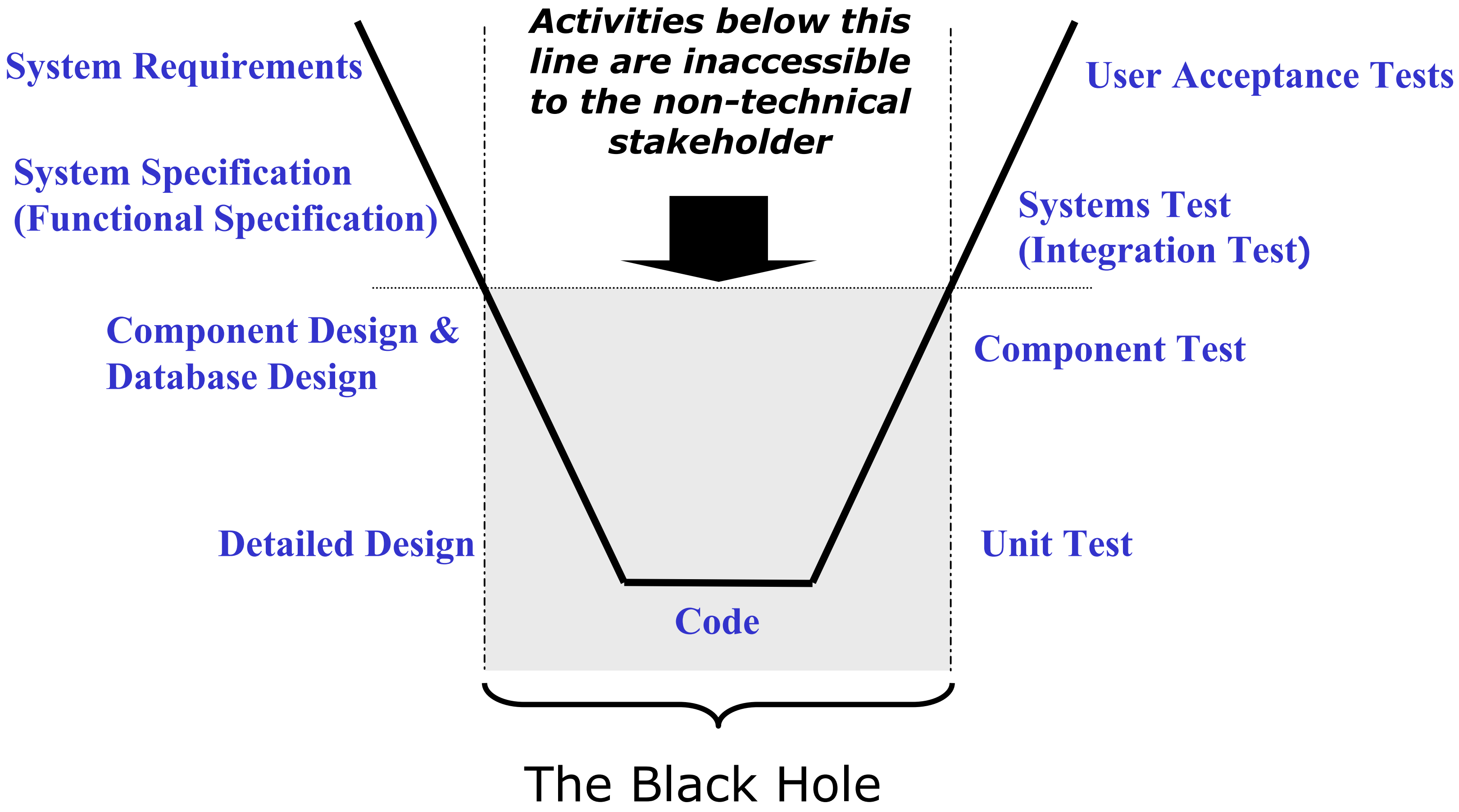
Methodical

- Now
  - Living with Change
  - Incremental/Iterative
  - Early Feedback



Agile

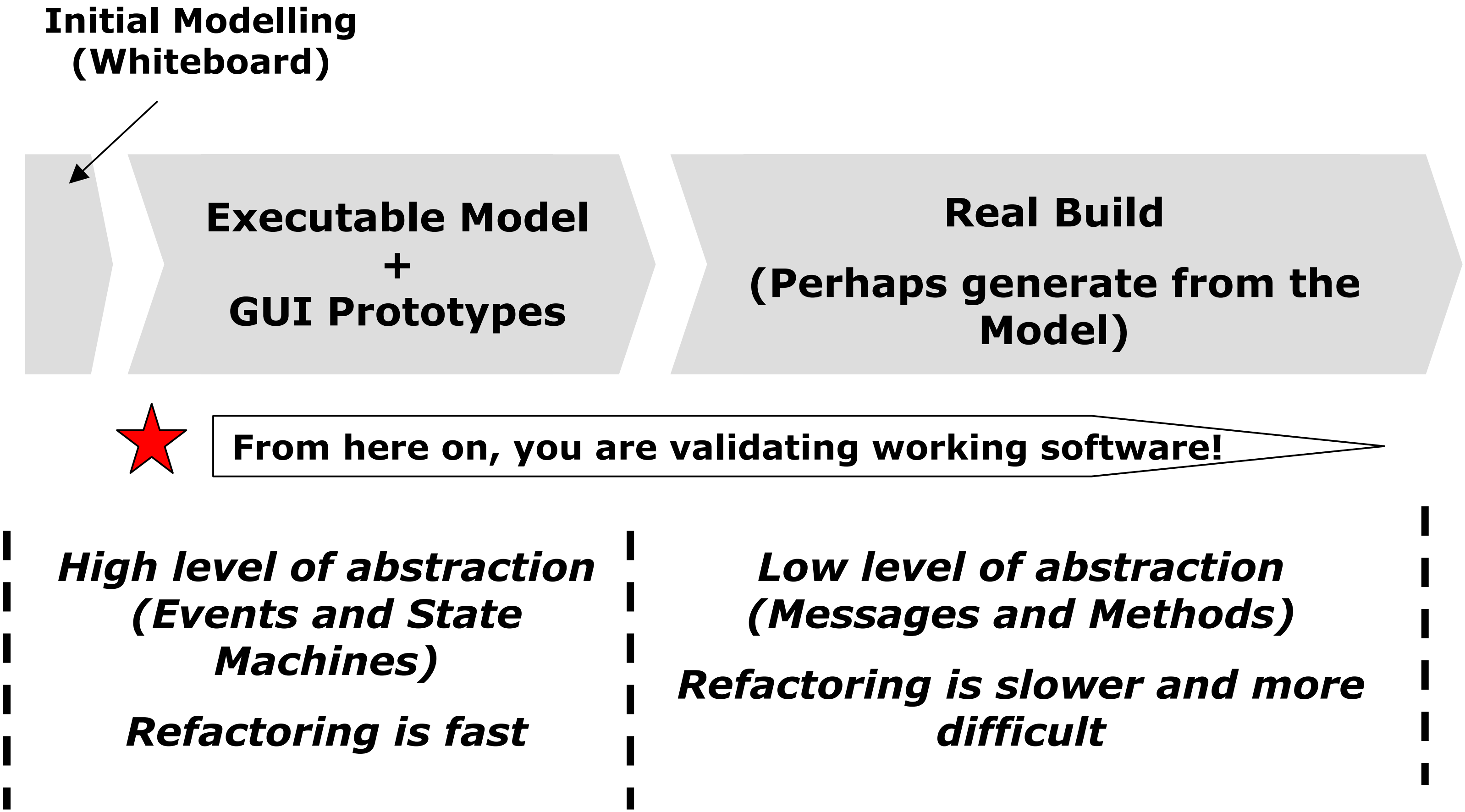
The "Waterfall Black Hole"


System Requirements
System Specification (Functional Specification)
Component Design & Database Design
Detailed Design
Code
Unit Test
Component Test
Systems Test (Integration Test)
User Acceptance Tests
Activities below this line are inaccessible to the non-technical stakeholder
The Black Hole

Agenda

- The Three Ages of System Development
- A Different Perspective on MDA
- Model Validation in Practice
- Modelling Techniques
- Closing Remarks

Model Driven Agility



Initial Modelling (Whiteboard)

Executable Model + GUI Prototypes

Real Build

(Perhaps generate from the Model)

From here on, you are validating working software!

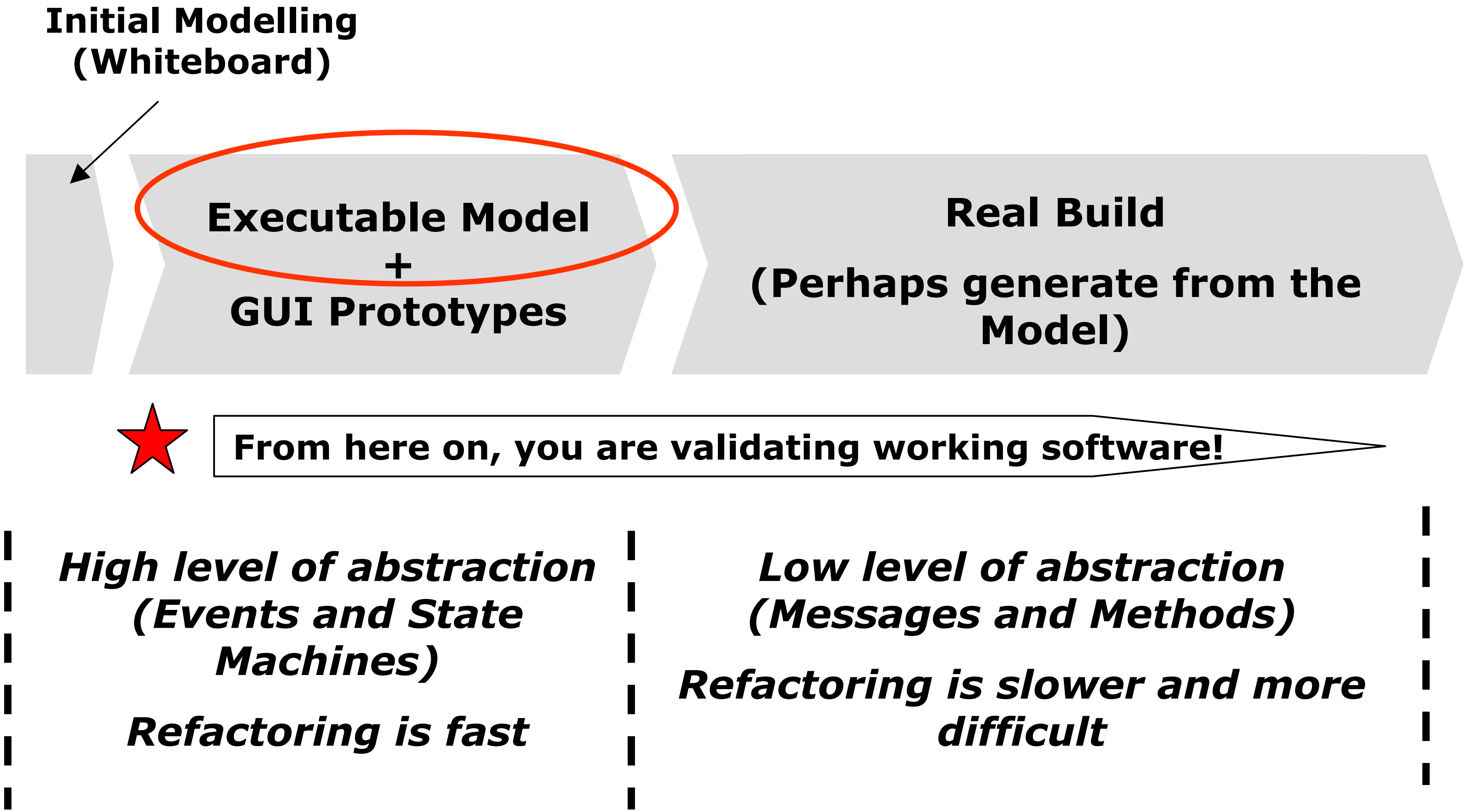
High level of abstraction (Events and State Machines)

Refactoring is fast

Low level of abstraction (Messages and Methods)

Refactoring is slower and more difficult

Our Focus



Initial Modelling (Whiteboard)

Executable Model + GUI Prototypes

Real Build

(Perhaps generate from the Model)

From here on, you are validating working software!

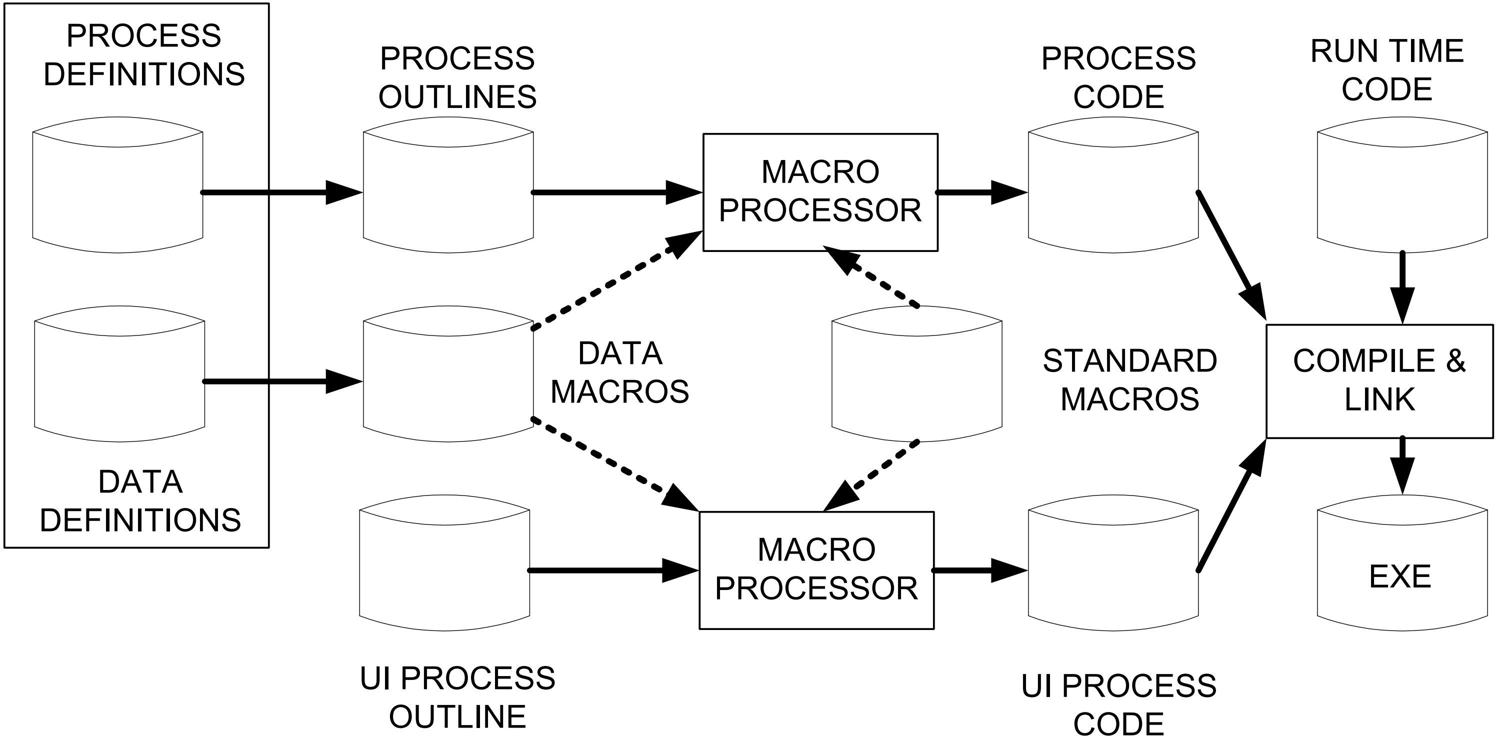
High level of abstraction (Events and State Machines)

Refactoring is fast

Low level of abstraction (Messages and Methods)

Refactoring is slower and more difficult

JSD “Model Compiler” (1989)



From a presentation to 11th ICSE Conference, May 1989

Evolution Since JSD

- 1993: Smalltalk model interpreter (OME)
  - Interprets instead of compiles
  - State Diagrams instead of Jackson diagrams
- 1996: First Java Version (DOME)
  - Metadata separated from code
  - Browser front-end
- 2001: Enhanced DOME
- 2003: ModelScope
  - Complete Java rewrite
  - First commercial release

Agenda

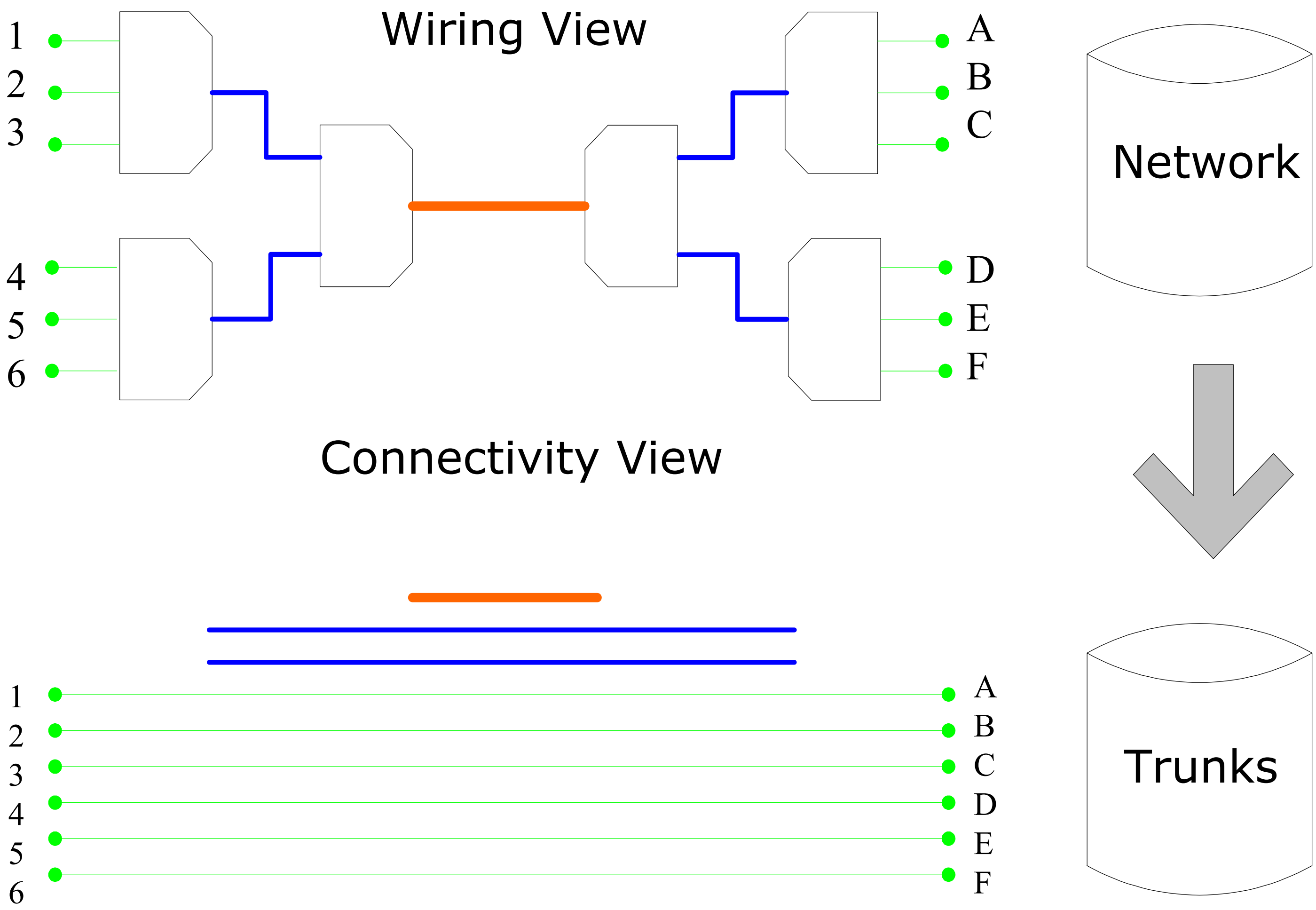
- The Three Ages of System Development
- A Different Perspective on MDA
- Model Validation in Practice
- Modelling Techniques
- Closing Remarks

Case Study 1: Telecoms

- Problem
  - Large COBOL system (~2m LOC)
  - Design “lost” during development
  - Code incomprehensible
  - No documentation
  - Enhancement impossible

- Solution
  - Rewrite of core sub-system (~ 0.8m LOC)
  - Executable Modelling to validate model
  - “Conventional” final build

Multiplex Engineering



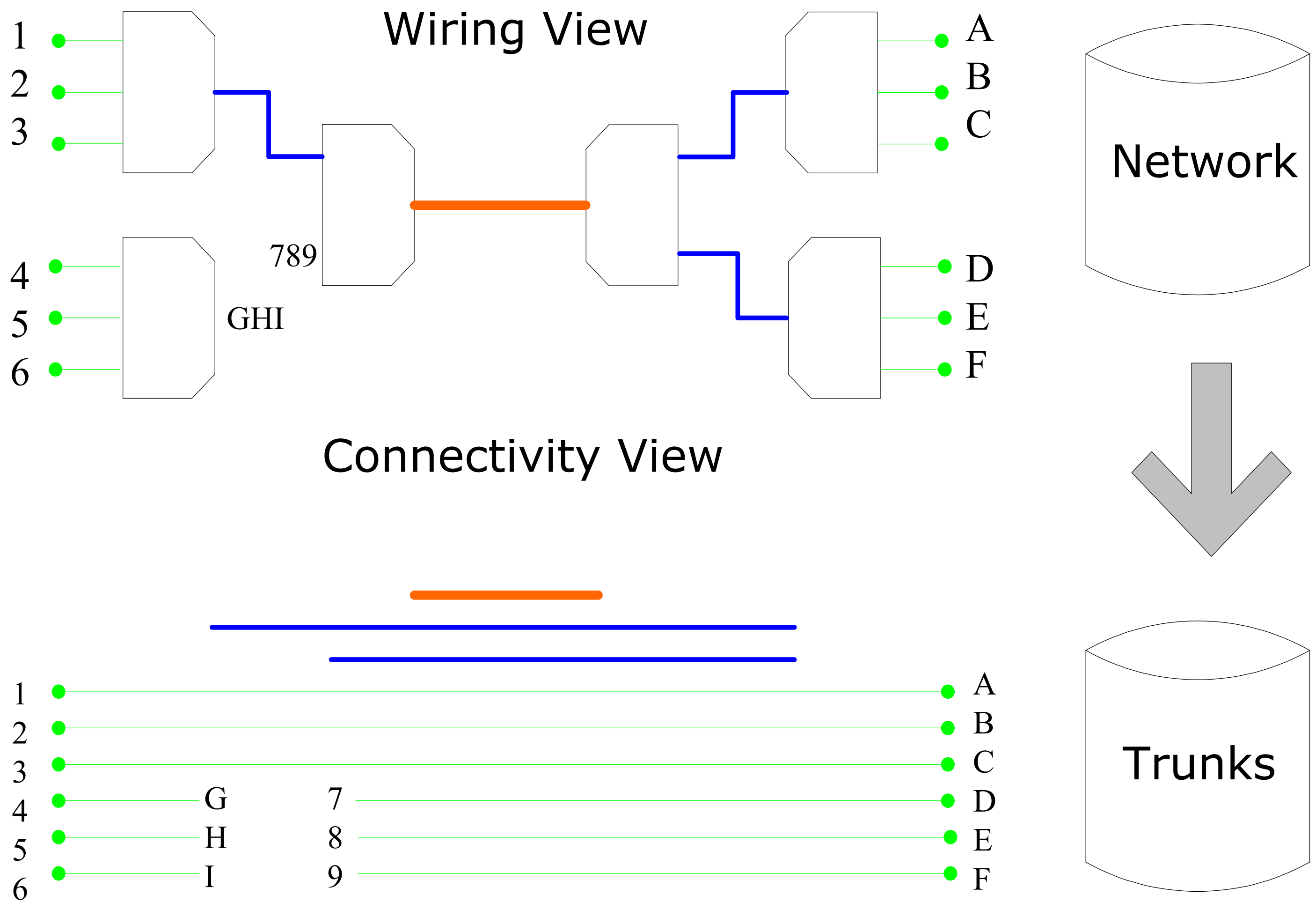
Wiring View

Connectivity View

Network

Trunks

Multiplex Engineering



Wiring View

Connectivity View

Network

Trunks

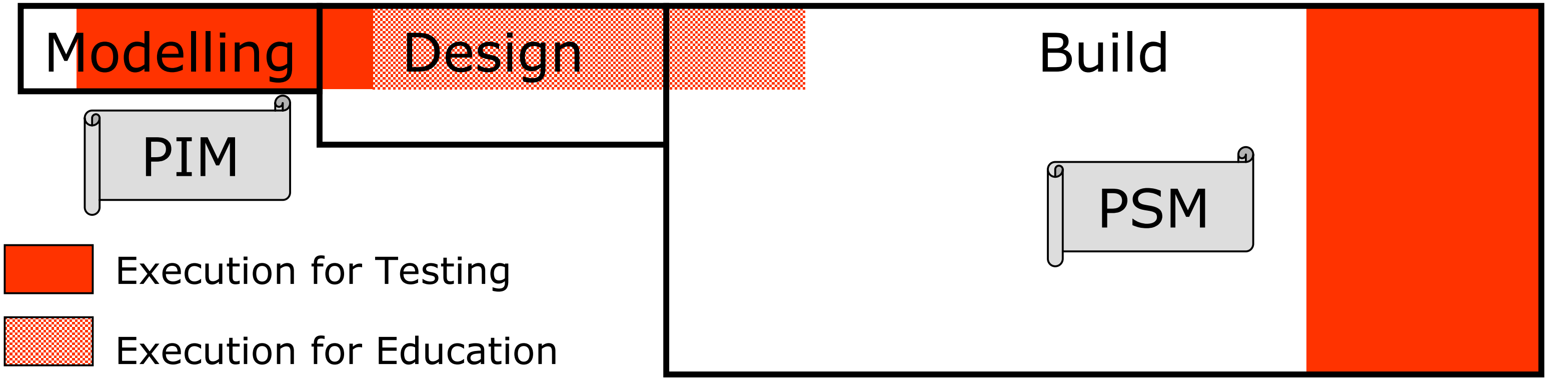
Design Issues

- At what multiplex levels does connectivity information need to be stored?
- How should the system determine fieldwork constraints?
- How should the system retain and manage multiple images of the network?
- How should complex (DCXs and Bridges) be represented?

The Shape of the Project

- Modelling Effort
  - 4 people for 3 months
- Whole Project
  - Peak headcount ~60
  - 20 months

Compare



🟥 Execution for Testing

▨ Execution for Education

Results

- Proof of Design
  - We didn’t get it right at first …
  - … but it was right before we started Design
- Synchronisation of Team
  - Concrete shared understanding
  - Enabled a successful “cavalry charge”
- Quality of the delivered system

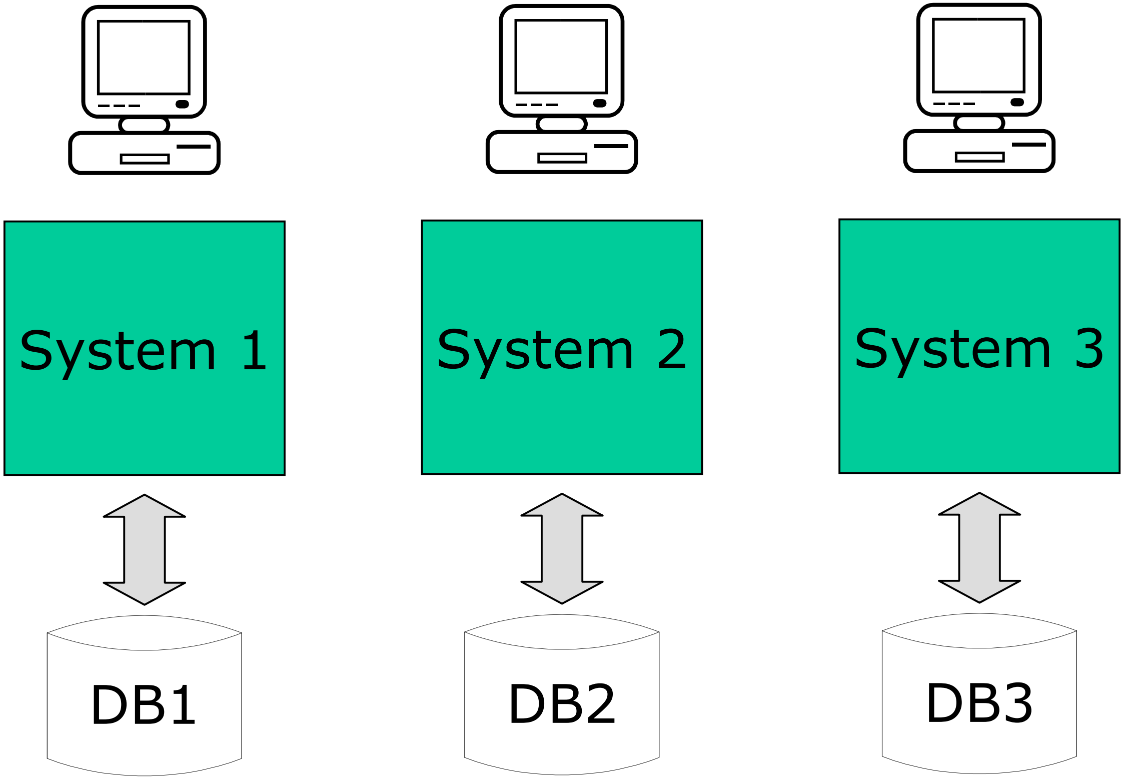
  “If you guys intended to develop a system which was stable and maintainable in production, you succeeded.” (System Manager, 4 years later)

Case Study 2: Workflow

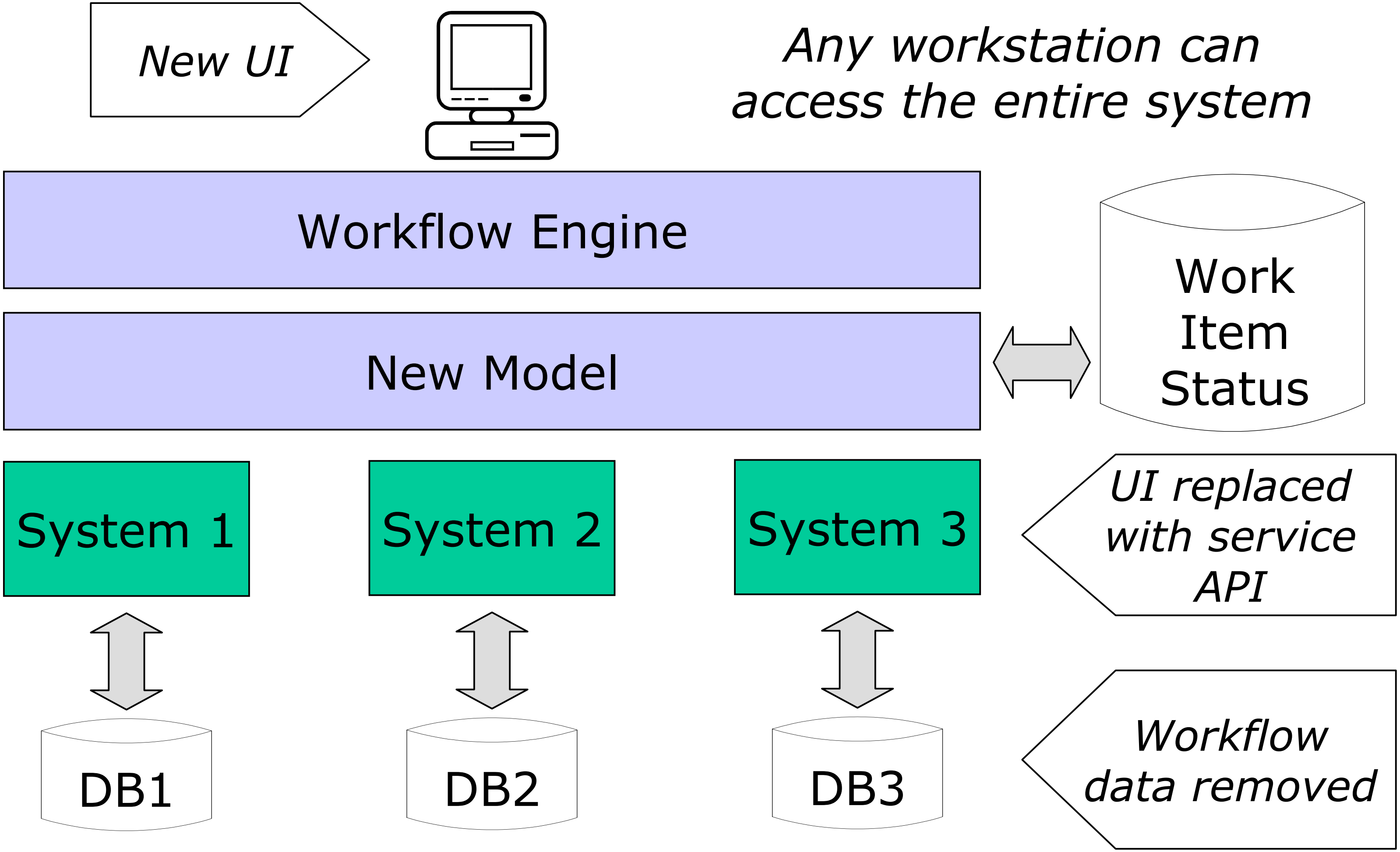
- Problem
  - Paper-based workflow using multiple legacy systems
  - Processes constrained by legacy system structure
  - No basis for progressive automation
- Solution
  - Replace paper with document imaging
  - Layer a new model over the legacy systems
  - Use executable modelling to design the new model ..
  - .. And verify that it can drive automated workflow

Old Architecture

Each stage of the process uses a different departmental system



New Architecture

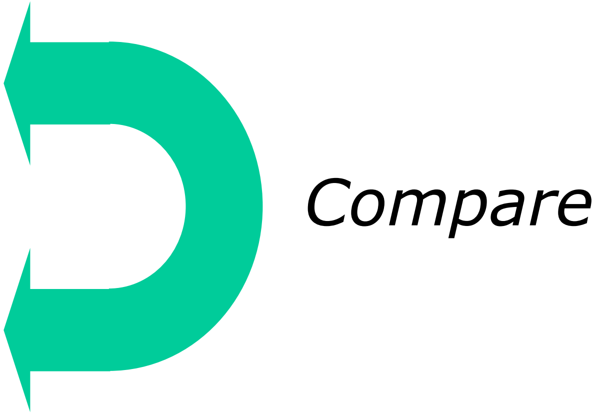


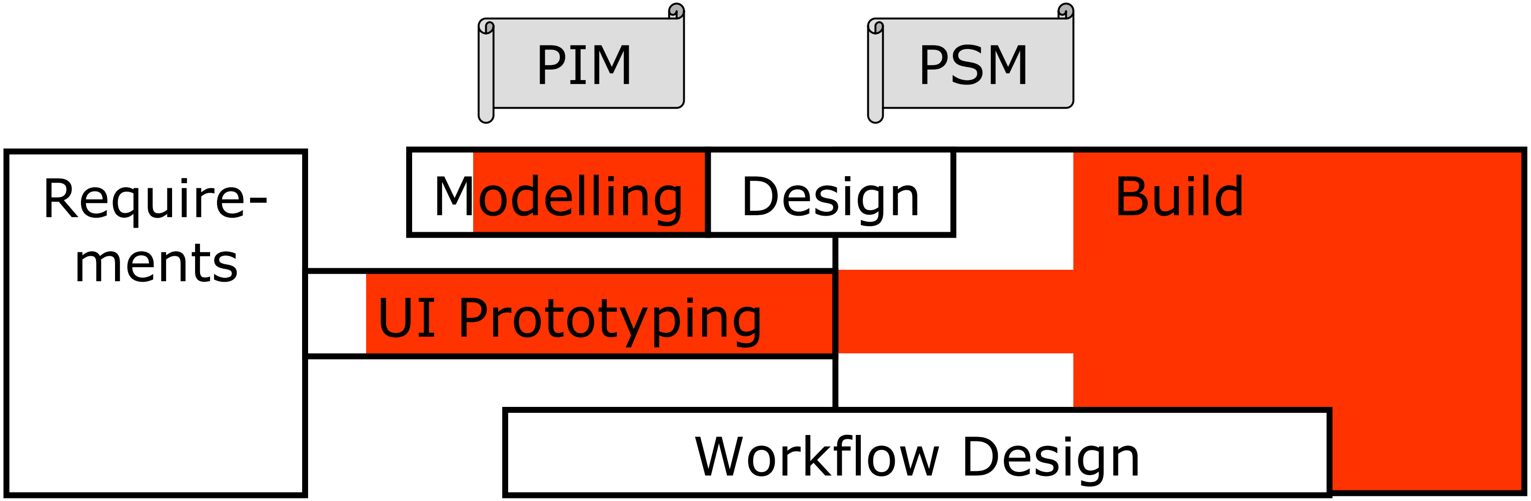
Design Issues

- How are multiple levels of workflow synchronised?
- How is workflow de-coupled from the legacy systems as much as possible?
- How is the new model synchronised with the legacy systems?

The Shape of the Project

- Modelling Effort
  - 2 people for 1 month
- Whole Project
  - Peak headcount ~20
  - 8 months





Results

- Validation of Requirements
  - Identification of gaps (e.g., re-linking documents)
- Communication
  - Demonstration of the new model to wider business community
- Quality of Process and Result
  - *Overall lower cost and reduced time to market*
  - *Efficiency and quality advantages from parallel construction*
  - *Test time reduced, regression testing not always required*
  - *High modularity (e.g., business transactions plug and play) and easy extensibility*

  (BIS Business Process and Workflow Conference, Orlando Florida, February 1995)

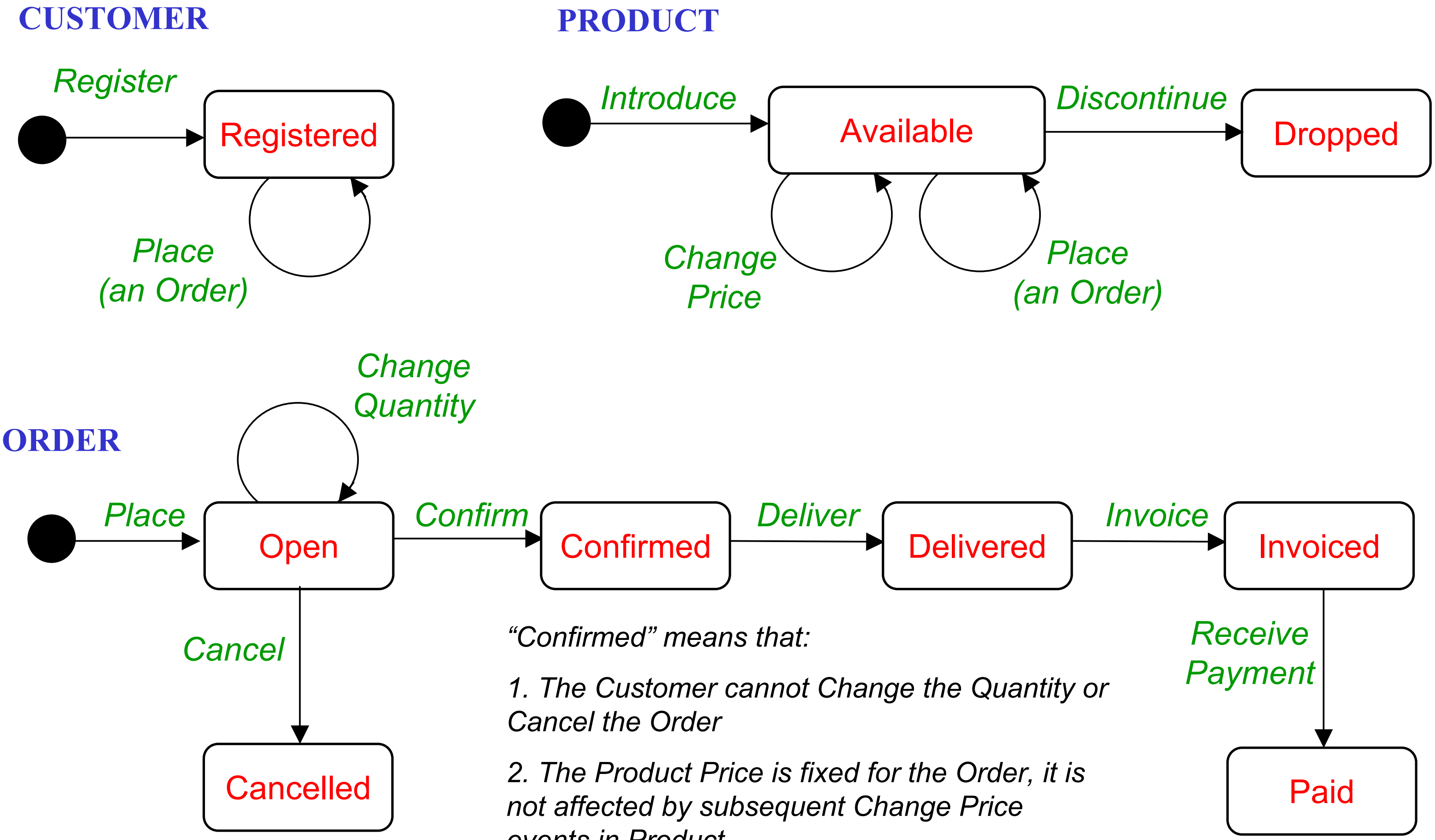
Agenda

- The Three Ages of System Development
- A Different Perspective on MDA
- Model Validation in Practice
- Modelling Techniques
- Closing Remarks

Modelling Technique

- Behaviour Modelling is Crucial
  - Static Models are good for generating *Infrastructure* code
  - Infrastructure is not important when validating behaviour
  - The user is interested in the behaviour of the application, not the structure of the software
- We use State Transition Diagrams
  - The only Show in Town for executable behaviour modelling!

Order Processing Model



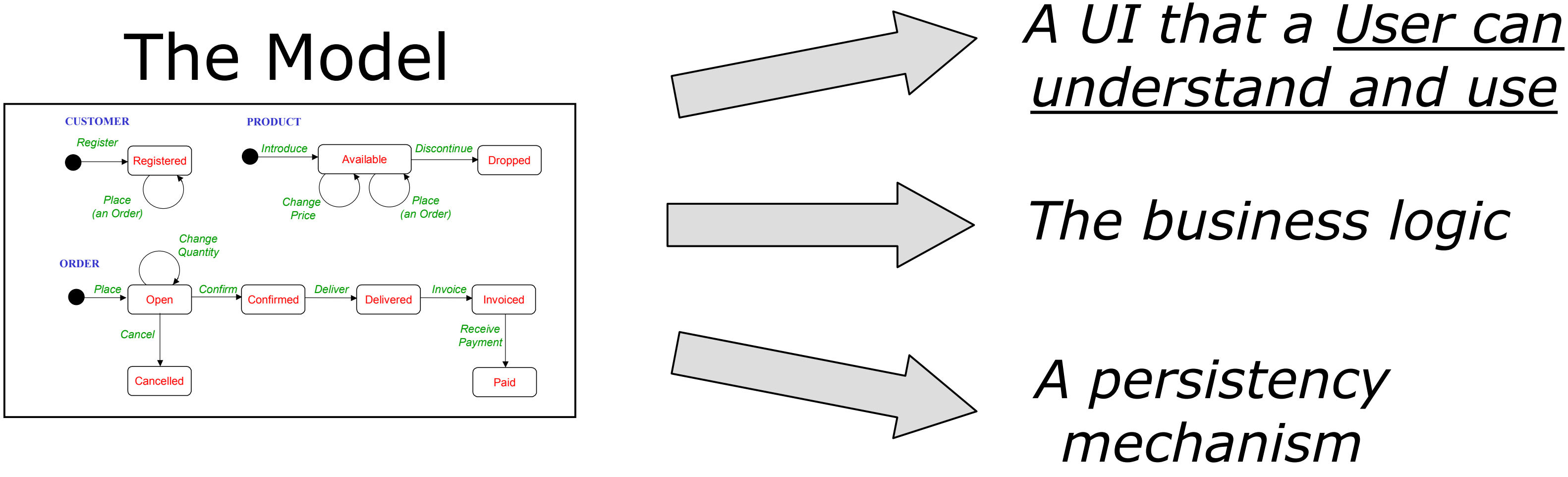
"Confirmed" means that:

1. The Customer cannot Change the Quantity or Cancel the Order

2. The Product Price is fixed for the Order, it is not affected by subsequent Change Price events in Product.

Extreme Modelling

- Extreme Modelling
  - Tight-loop model, test, demonstrate, change
  - Refactoring to improve the quality of the abstractions
- Requires full leverage of the Model



The Model

A UI that a User can understand and use

The business logic

A persistency mechanism

Isn’t this just Functional Prototyping?

- Functional Prototypes
  - Constructed at a lower level of abstraction
  - The required behaviour is coerced at the code level, because that is quickest
  - The model and code diverge, and the model is lost
  - The protoype is thrown away (we hope!)
- Executable Models
  - The behaviour is described using model level primitives (state machines)
  - Actual behaviour and model cannot diverge
  - The model is used for the final build

Agenda

- The Three Ages of System Development
- A Different Perspective on MDA
- Model Validation in Practice
- Modelling Techniques
- Closing Remarks

Closing Remarks

- Our view is that Executable Modelling used early in the project can deliver real value
  - Reviewers need no knowledge of modelling
  - Retains a high level of abstraction (contrast XP)
  - Enables acceptance testing of business behaviour while the model is still easy to change
- The value is not dependent on the use of model driven code generation
  - Generating the final code improves productivity
  - But hand crafting code from a validated model is not hard

Finally

- The tools required are simple and cheap
  - At least an order of magnitude simpler and cheaper than a full MDA final code generation toolset
- “Lock in” can be completely avoided
  - If the final code is written by hand, the deployed system has no dependency on the model execution tool
- This is a good way to start using MDA
  - Low cost and risk
  - Low exit cost if unsuccessful
  - Experience gained builds a foundation for progressing to full MDA
  - “Gain without Pain”

Thank You

Any Questions?