DevOps Meets Formal Modelling in High-Criticality Complex Systems

Marta Olszewska, Marina Waldén

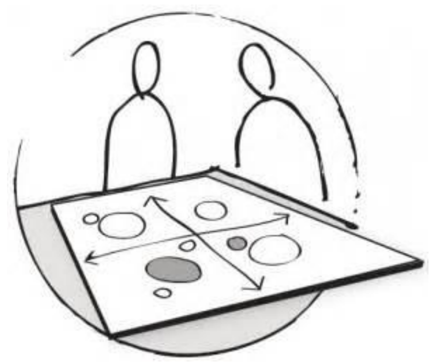
Åbo Akademi University
Science and Engineering

TUCS

1st International Workshop on Quality-Aware DevOps (QUDOS 2015)
1st September 2015, Bergamo, Italy

Roadmap

- Why?
  - Motivation and goals
- How?
  - Existing methods, tools and processes
  - Strategy
- What?
  - DevOps umbrella



Why?

The world is not enough

- Priority: quality
  - Human lives or major financial losses
- Need for speed and a bit more
- System development nowadays
  - Requires to be responsive to change and actionable
  - Provide faster delivery
  - Enable communication and collaboration

Currently we…

- Develop high-criticality complex systems
  - Assure *correctness*
- Focus on modelling
  - Early stage *development*
- Ensure *quality*
  - E.g. to enable standardisation

Formal Methods

But we also need to…

- Timely identify bottlenecks
- Increase the speed of development
  - Reduce friction in the development time
  - Faster delivery of artefacts
- Improve communication
  - Within development team
  - With stakeholders
- Support functioning of interdependencies in a project

Agile methods

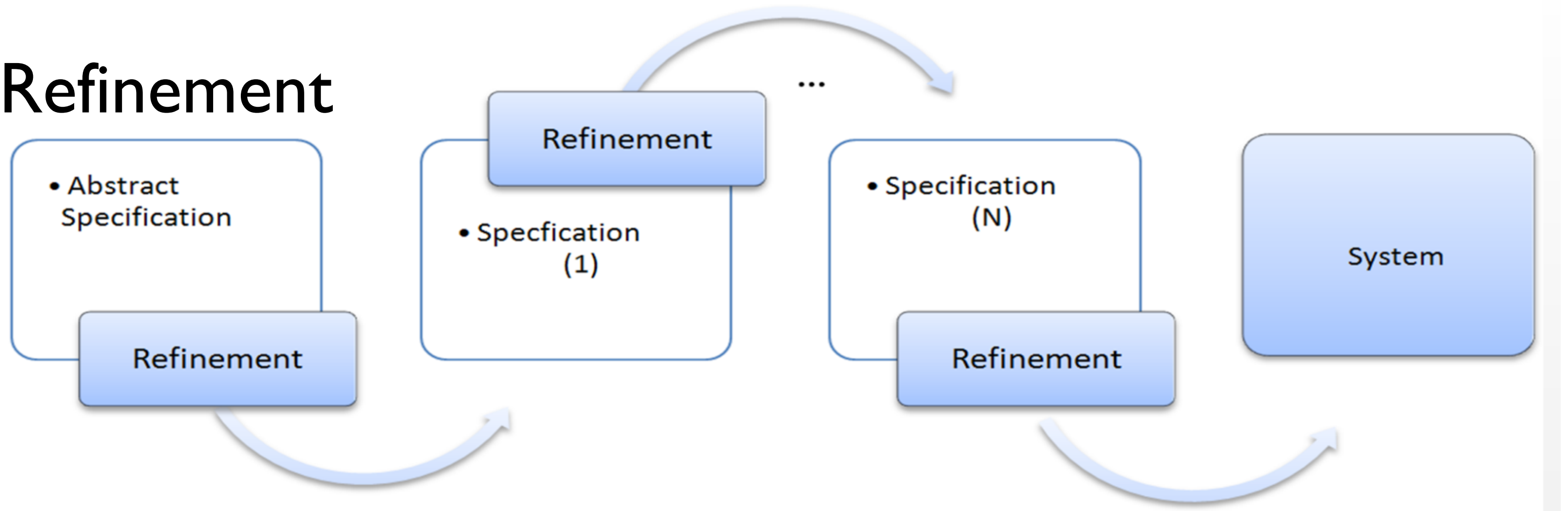
How?

Åbo Akademi University
Science and Engineering

TUCS

Focus on correctness and quality

- Refinement

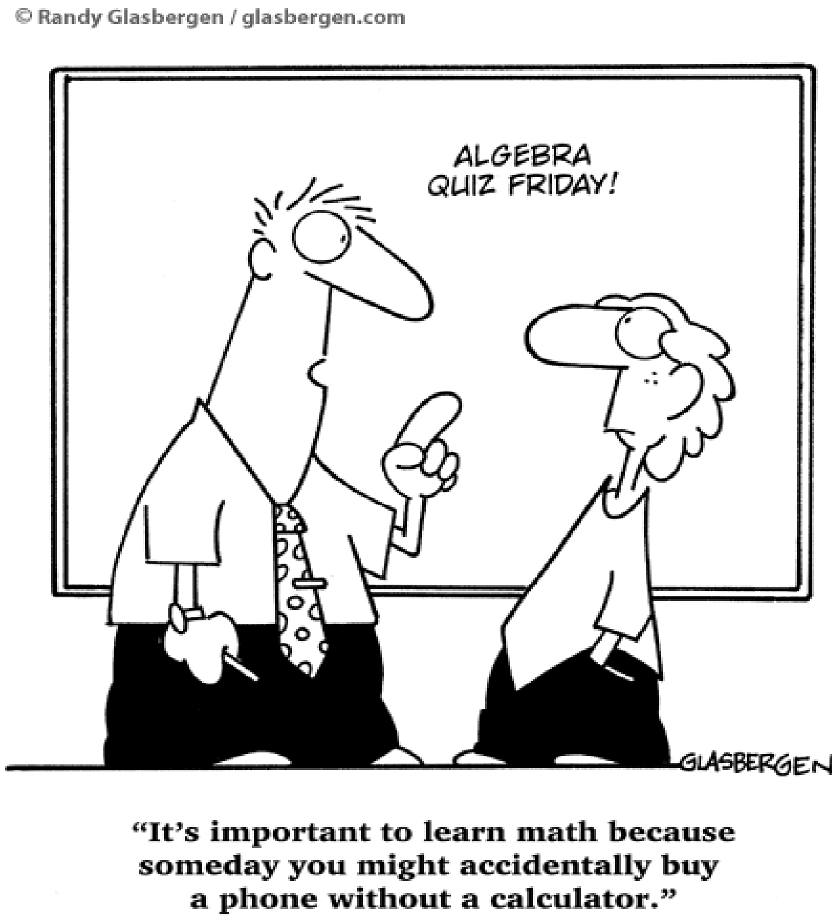


- Mathematically proving that the abstract model is consistent and feasible
  - Model preserves invariant
  - Tool supported
- Complexity control

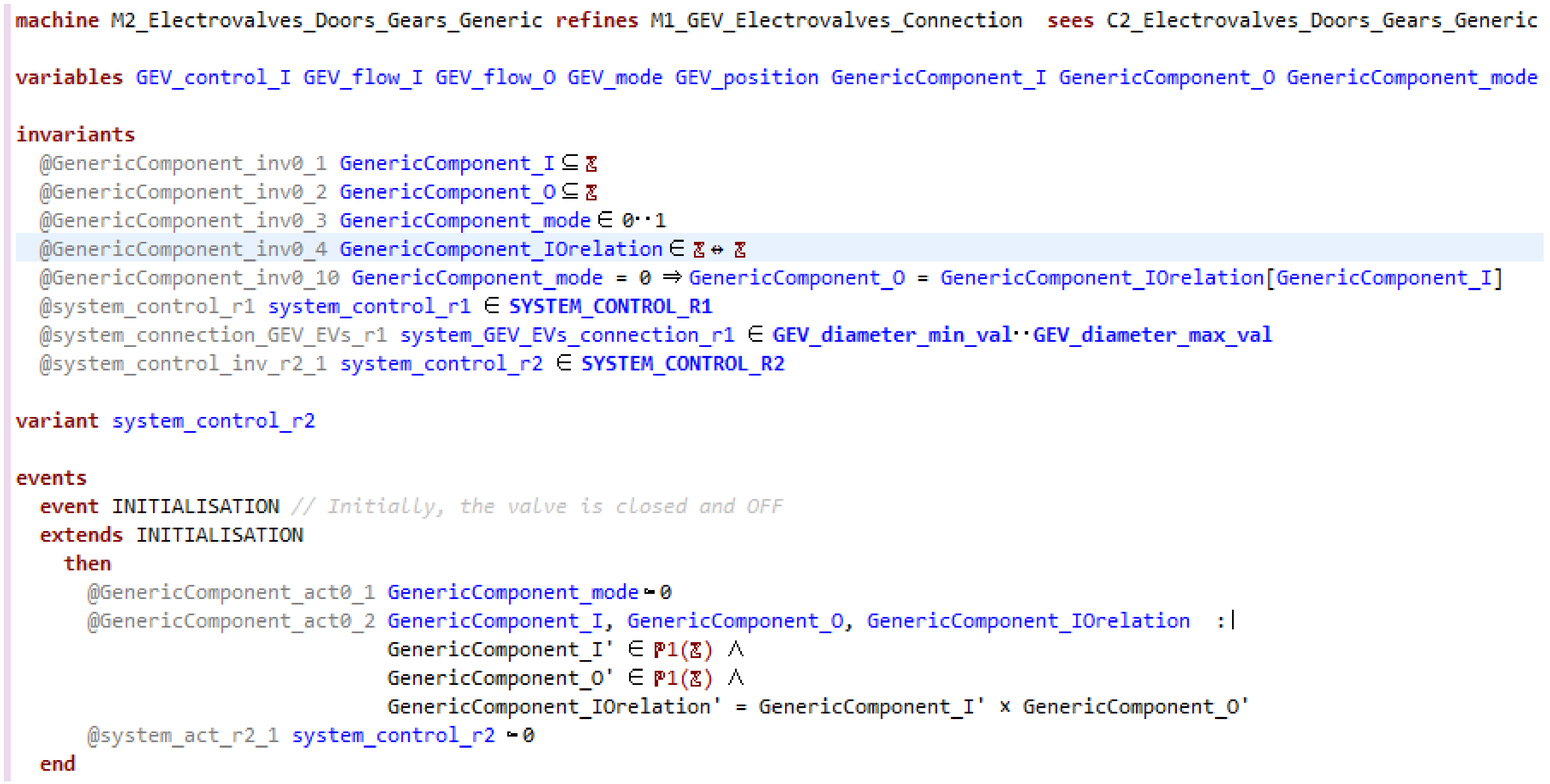
Event-B

- Formal method
  - Uses Abstract Machine Notation
  - Utilises refinement
  - Models complete systems
- Tool supported
  - Rodin platform
  - Multiple plugins

- Development *method*


© Randy Glasbergen / glasbergen.com
ALGEBRA QUIZ FRIDAY!
GLASBERGEN
"It's important to learn math because someday you might accidentally buy a phone without a calculator."

Event-B code



```
machine M2_Electrovalves_Doors_Gears_Generic refines M1_GEV_Electrovalves_Connection  sees C2_Electrovalves_Doors_Gears_Generic

variables GEV_control_I GEV_flow_I GEV_flow_O GEV_mode GEV_position GenericComponent_I GenericComponent_O GenericComponent_mode

invariants
  @GenericComponent_inv0_1 GenericComponent_I ⊆ ℤ
  @GenericComponent_inv0_2 GenericComponent_O ⊆ ℤ
  @GenericComponent_inv0_3 GenericComponent_mode ∈ 0‥1
  @GenericComponent_inv0_4 GenericComponent_IOrelation ∈ ℤ ↔ ℤ
  @GenericComponent_inv0_10 GenericComponent_mode = 0 ⇒ GenericComponent_O = GenericComponent_IOrelation[GenericComponent_I]
  @system_control_r1 system_control_r1 ∈ SYSTEM_CONTROL_R1
  @system_connection_GEV_EVs_r1 system_GEV_EVs_connection_r1 ∈ GEV_diameter_min_val‥GEV_diameter_max_val
  @system_control_inv_r2_1 system_control_r2 ∈ SYSTEM_CONTROL_R2

variant system_control_r2

events
  event INITIALISATION // Initially, the valve is closed and OFF
  extends INITIALISATION
    then
      @GenericComponent_act0_1 GenericComponent_mode ≔ 0
      @GenericComponent_act0_2 GenericComponent_I, GenericComponent_O, GenericComponent_IOrelation  :∣
                               GenericComponent_I' ∈ ℙ1(ℤ) ∧
                               GenericComponent_O' ∈ ℙ1(ℤ) ∧
                               GenericComponent_IOrelation' = GenericComponent_I' × GenericComponent_O'
      @system_act_r2_1 system_control_r2 ≔ 0
  end
```

Åbo Akademi University
Science and Engineering
TUCS

Agile methods

- Flexible development
- Responsiveness to change
- Ability to meet stakeholders’ needs within the given time
- Facilitating collaboration

- Development *process*

Iterative

Incremental

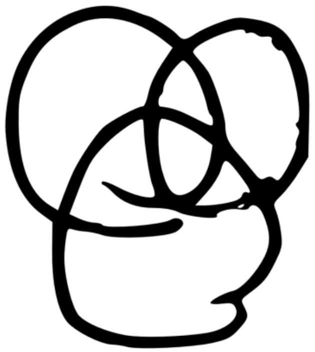
Software development method

Adaptive and flexible

Evolutionary development and delivery

Synergy

- Emphasis on collaboration, integration, communication and automation
- Increasing comprehension
  - Effectively mapping real world to code
- Development philosophy*
  - Quality assurance mechanisms
  - IT operations
  - SwEng practices



a set of ideas about how to do something

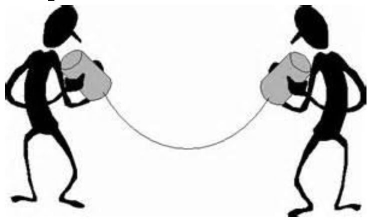
FormAgi framework

- Relates agile principles, practices and values to formal setting
  - To create a synergy between these two

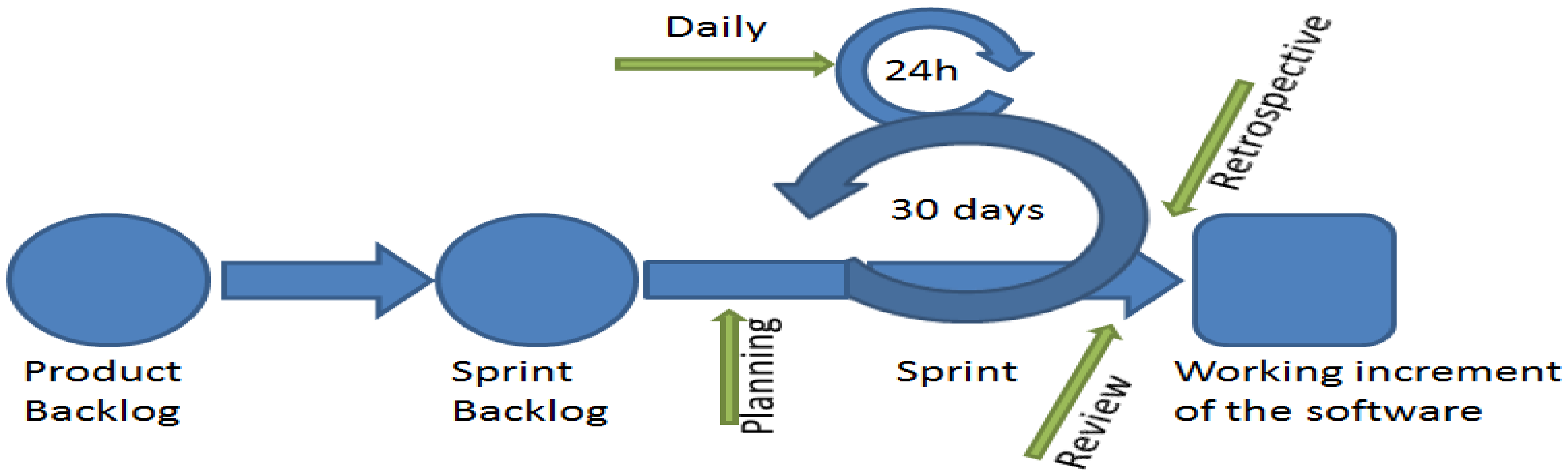
- Agile concepts set in the context of safety-critical development providing:
  - Guidelines on what concerns should be tackled before committing to a certain agile method
  - Pointers in which aspects an agile method can be a facilitator in the formal development
- Idea of tailoring: *merge and adapt*

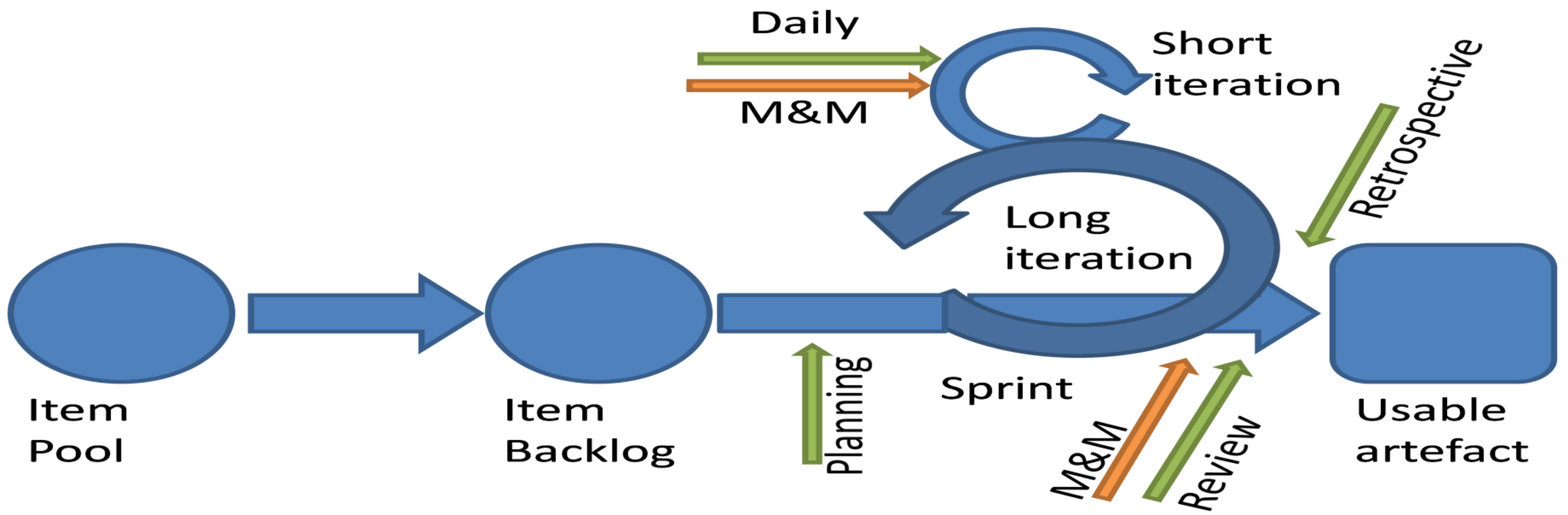
Why Scrum?

- Clear definition of time frames for iterations
  - Organisation of sprints
- Set of meetings to be held during the development process
  - Supports communication
- Similarity in iterations and refinement steps
- Short development cycles
  - Smoothening development process
- Supports process improvement



Scrum and formal modelling





Åbo Akademi University Science and Engineering

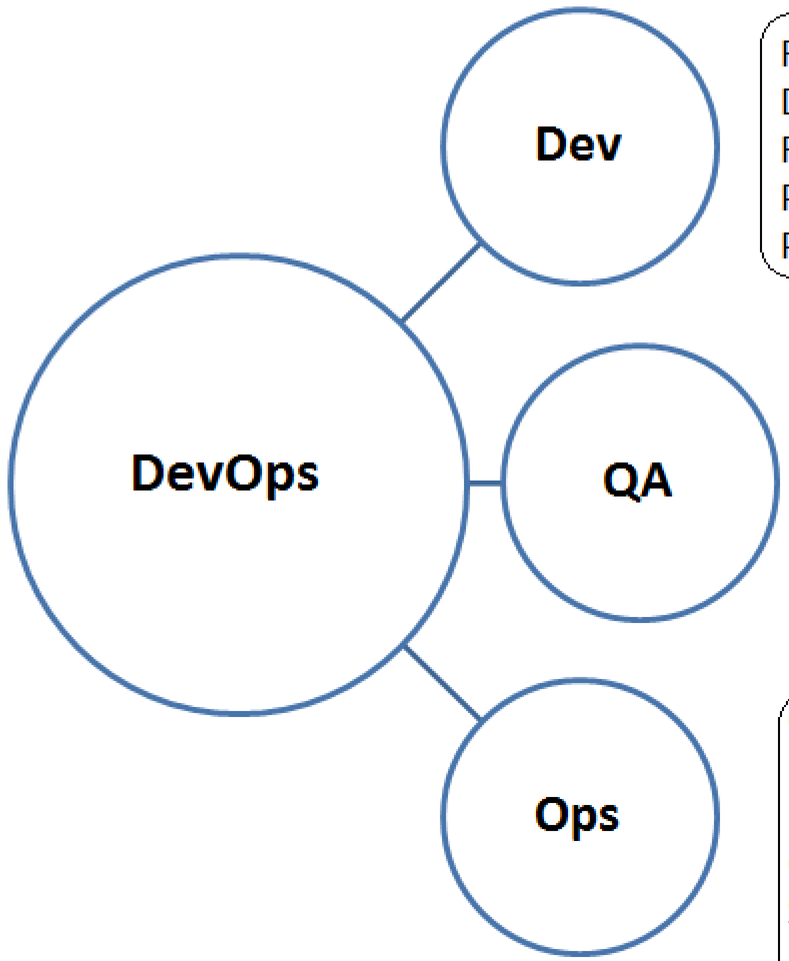
TUCS

What?

Åbo Akademi University
Science and Engineering

TUCS

Formal modelling in DevOps

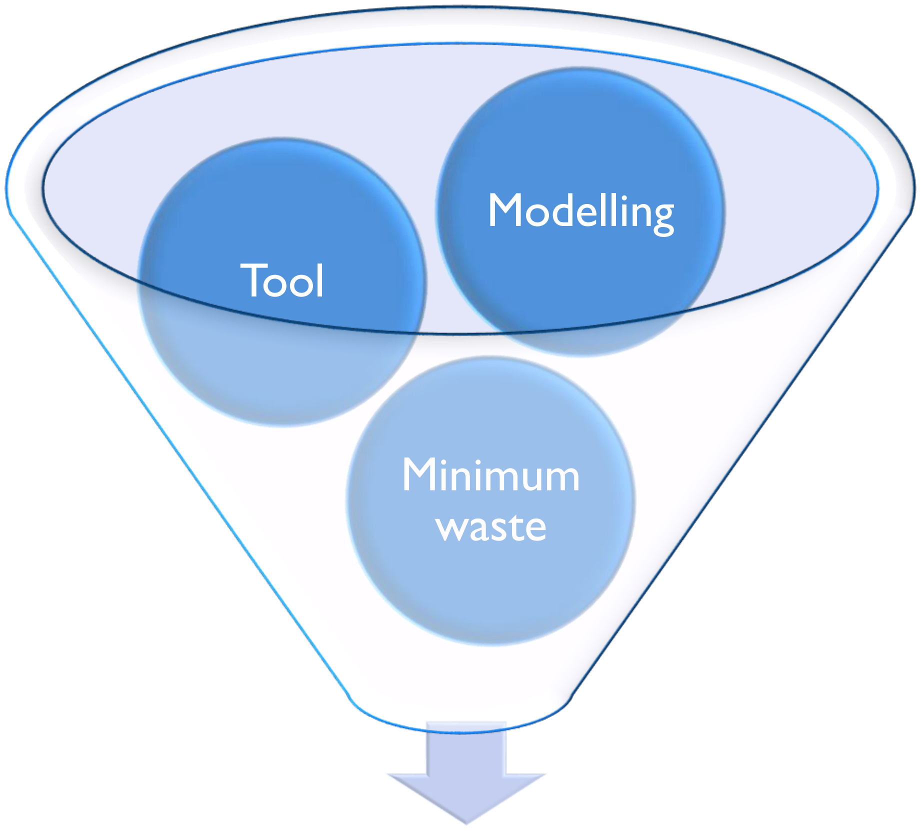


Formal method (Event-B)
Development process (Scrum)
FormAgi
Practices
Patterns, dev. guidelines

Correctness (by construction)
Proofs
Metrics
Monitoring
Integration tests

Communication
(Inner-team/organisation;
Outer (stakeholder))
Collaboration
Standards
Documentation

Facilitating *Dev*



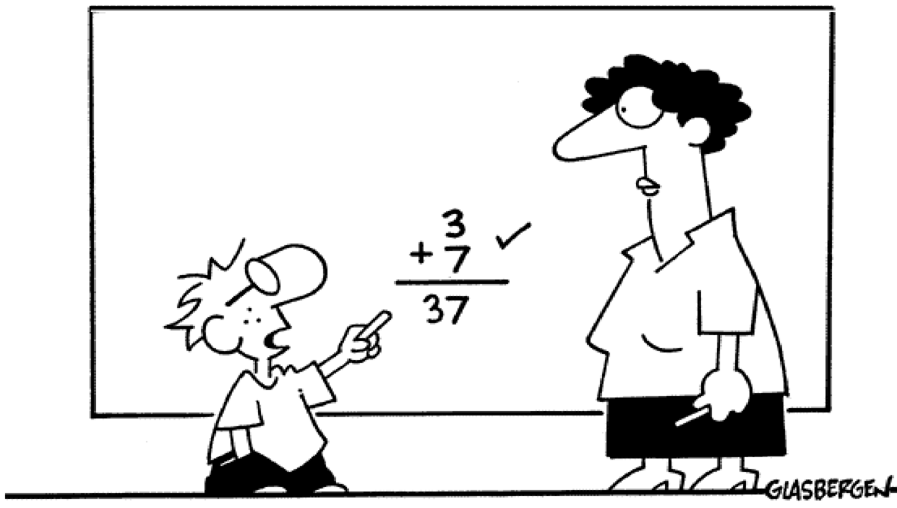
Speed of delivery
&
Continuous delivery

Supporting development – Rodin tool

- Visualisations and animations
  - To show the results of the modelling to team members and stakeholders
    - E.g. after a short / long iteration
    - No need to provide executable code
- Code generation
  - To various programming languages
  - Different level of technical detail
  - Once the model is at a lower level of abstraction

Guiding development - Modelling strategy

- Patterns
  - Generic
  - Related to modelling strategy
- Components (library)
  - Generic components, visualised
  - Support reuse and modularity
- Decomposition
- Abstraction



"In the corporate world they pay you big bucks for thinking outside of the box!"

Waste

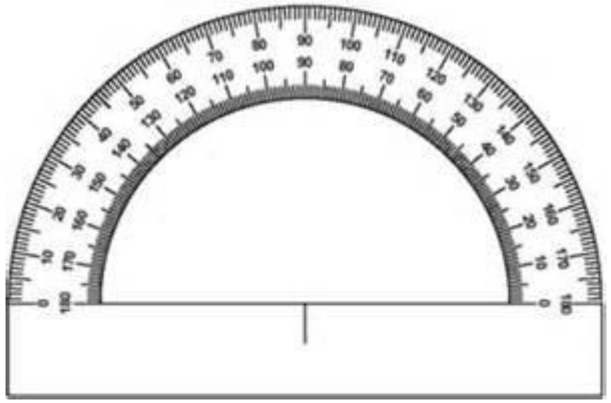
- Waste can be generated when
  - Insufficient time is spent on requirements modelling
    - Can lead to spending excessive time on modelling and then cause cumbersome proving
  - Detailing the model too early
    - It increases the complexity of the model and its related proofs.
- Avoiding waste by
  - Requirement prioritisation
  - Providing strategy in modelling
  - Via decomposition and abstraction mechanisms

Assuring *quality*

- Refinement
- Complexity control
  - Concentrating on what matters the most
    - At a particular point in the development
  - Matching the level of abstraction with the current development stage
- Feedback mechanisms
  - Monitor & Measure
  - Analyse
- Standardisation
  - Documenting modelling decisions

Metrics and monitoring

- Feedback mechanism
  - Identifying bottlenecks
  - Prioritising the improvement areas
- Short and long iteration
  - Model metrics
    - Size, complexity, proof obligations
  - Project oriented metrics
    - Delivered functionality, velocity
  - Process metrics
    - Time invested, activity time, change cycle time



Post-mortems

- Team
- Stakeholders
- Additional "check" mechanism
  - Could be incorporated in the development process
    - Once a bigger milestone is achieved
  - Integrating current development with other part of a system

Operations from DevOps

- Emphasis on communication
  - The team members and stakeholders
- Standups
  - Pinpointing difficulties with the modelling or proving
- Knowledge sharing
- Raising understanding and awareness
- „Reusable team”
  - Expertise of every group member is known
  - How-to can be utilised whenever needed

In the next episode…

…meaning: after the paper submission*

Åbo Akademi University
Science and Engineering

TUCS

* Involvement of Sergey Ostroumov, PhD

Experimentation

- Need to check technical details
  - To validate our claims
  - And our „advocacy” in the publication
- Two-fold experimentation
  - Case study of a landing gear
    - Industrial case study
    - Execution in academic / research setting
  - Project course
    - Case study where core functionality is in Event-B
    - Execution in academic setting - students as developers

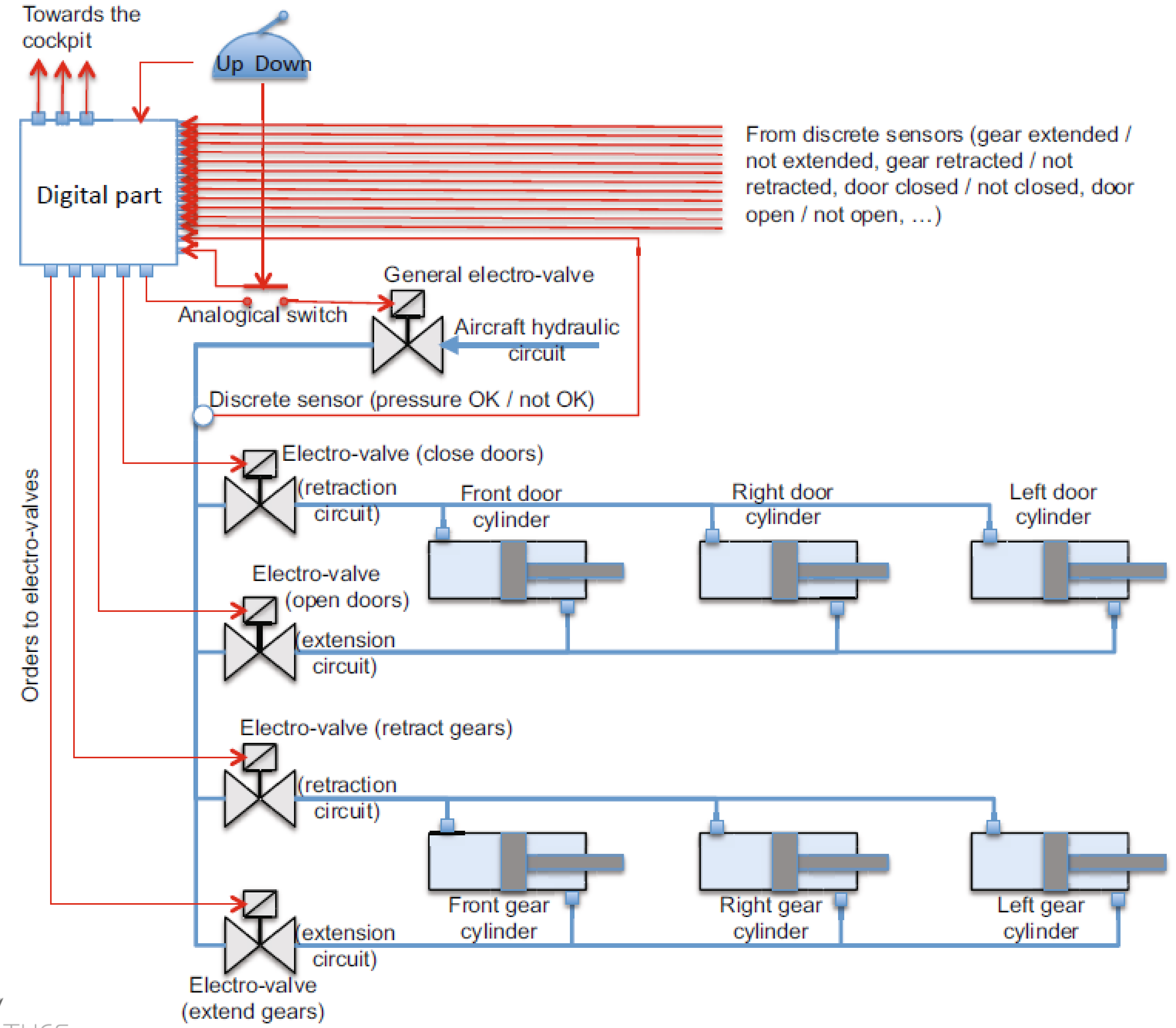
Landing gear

Scrum

- People
  - Formal modelling expert
  - Developer and stakeholder
  - Product owner and quality assurance specialist
  - External consultant
- Two one-week sprints
  - Plus „0” sprint
- Daily standups
- Retrospectives
- Trac document + formal requirements documentation

Event-B

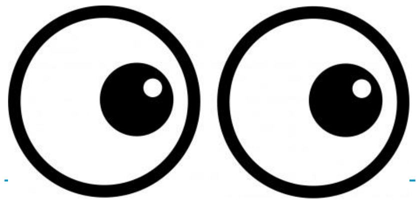
- Component-based modelling
- Some characteristics of OO programming
- Challenge: connecting components
- Restrictions: sequential nature of refining models
  - Opposes flexibility


Towards the cockpit
Up Down
Digital part
From discrete sensors (gear extended / not extended, gear retracted / not retracted, door closed / not closed, door open / not open, …)
General electro-valve
Analogical switch
Aircraft hydraulic circuit
Discrete sensor (pressure OK / not OK)
Orders to electro-valves
Electro-valve (close doors)
(retraction circuit)
Front door cylinder
Right door cylinder
Left door cylinder
Electro-valve (open doors)
(extension circuit)
Electro-valve (retract gears)
(retraction circuit)
Front gear cylinder
Right gear cylinder
Left gear cylinder
(extension circuit)
Electro-valve (extend gears)

Åbo Akademi University Science and Engineering

TUCS

Observations



- The need for good governance doesn't vanish with agile
  - Monitoring and documentation still needed
- An agile transformation / DevOps adoption is a journey, not a destination
  - Continuous tweaks and tuning of process
- Boost in communication
- Expert's consultation needed
- Iterative nature of refinement vs agile approach
  - Not hand-in-hand
- Model review needed

Discussion

1) How to effectively experiment with FM-DevOps concept?
   - What are the potential bottlenecks?
   - What should be in (more) focus?
2) Formal Methods are ready for Dev (agility), but are they ready for Ops?