TGA2015

Disciplined Agile Delivery

for

Critical System Development

Andy Edmunds

Åbo Akademi, Turku

aedmunds@abo.fi

The Research Space (i)

- High Integrity Systems,
  - can be business-, mission-, or safety-critical.
  - often require certification.
- Formal methods,
  - provide a precise specification of the system.
  - with mathematical underpinning, often hidden.
  - use abstraction and stepwise development (refinement)
- Agile: a new area for us.
  - Disciplined Agile Delivery (Ambler and Lines).
  - A process goal-driven, Pick 'n' Mix approach.

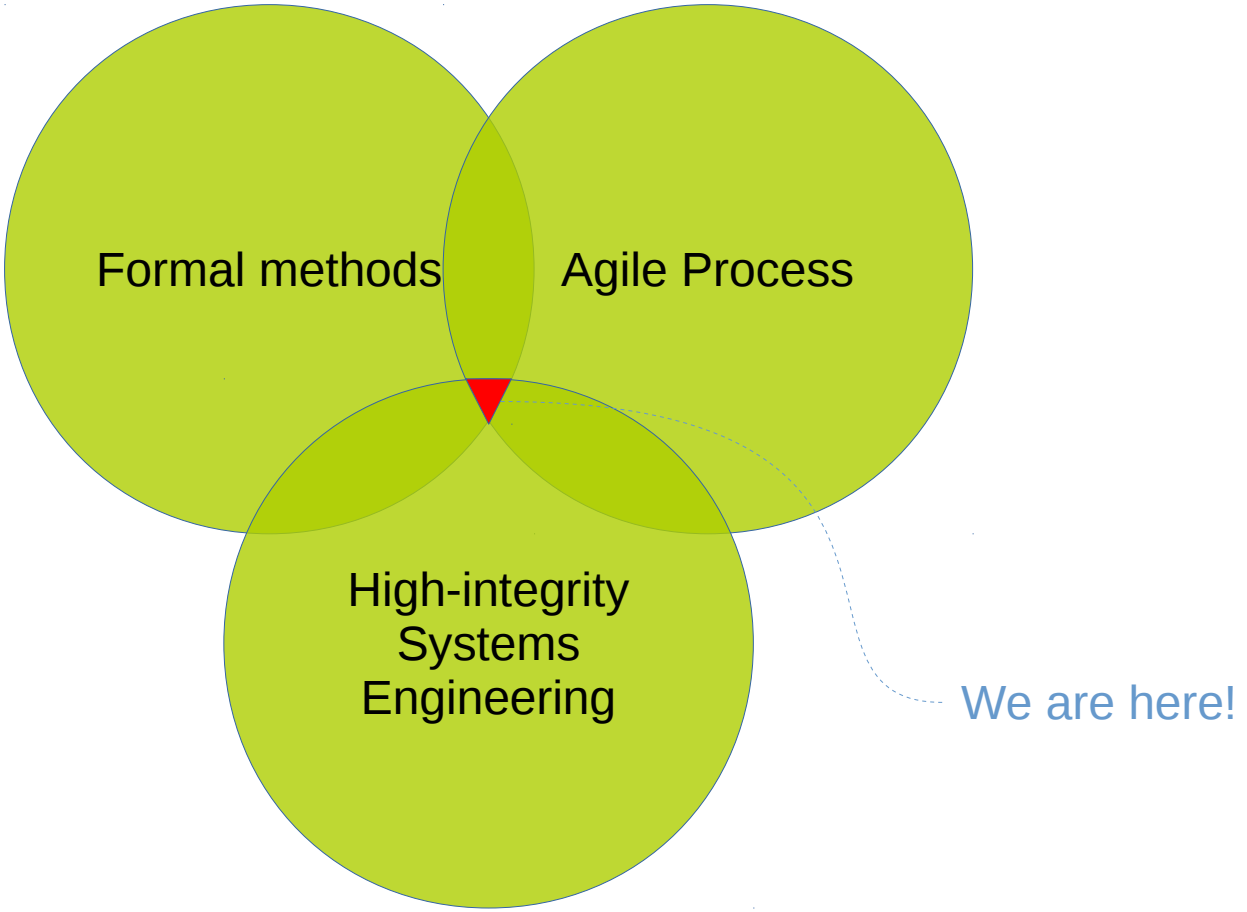
The Research Space (i)

- High Integrity Systems,
  - can be business-, mission-, or safety-critical.
  - often require certification.
- Formal methods,
  - provide a precise specification of the system.
  - with mathematical underpinning, often hidden.
  - use abstraction and stepwise development (refinement)
- Agile: a new area for us.
  - Disciplined Agile Delivery (Ambler and Lines).
  - A process goal-driven, Pick 'n' Mix approach.

The Research Space (i)

- High Integrity Systems,
  - can be business-, mission-, or safety-critical.
  - often require certification.
- Formal methods,
  - provide a precise specification of the system.
  - with mathematical underpinning, often hidden.
  - use abstraction and stepwise development (refinement)
- Agile: a new area for us.
  - Disciplined Agile Delivery (Ambler and Lines).
  - A process goal-driven, Pick 'n' Mix approach.

The Research Space (ii)



High Integrity Systems

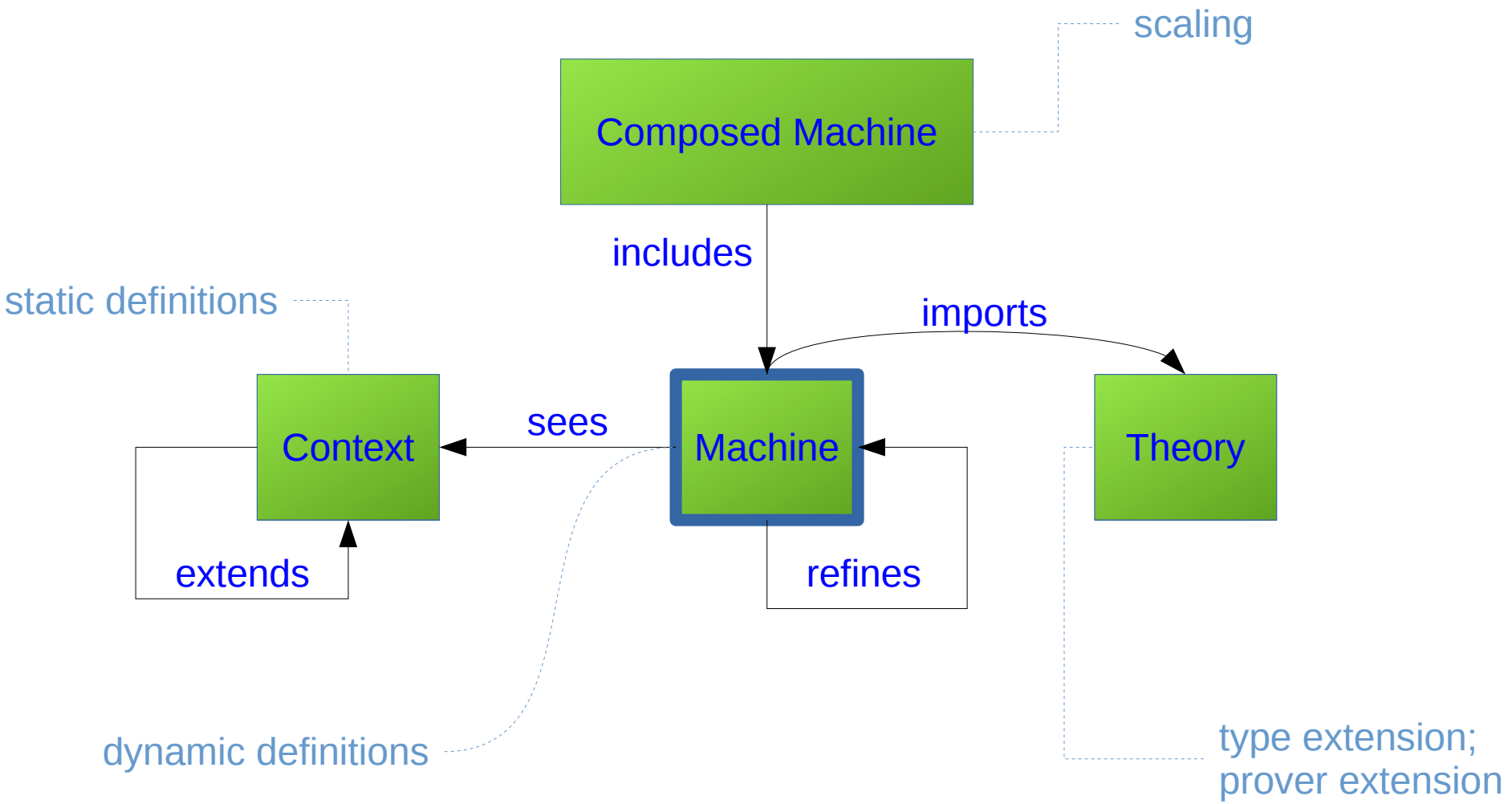
- HI systems engineering requires,
  - more 'upfront' activities.
  - traceability of requirements (safety, functional, non-functional).
  - usually requires 'certified' deliveries.

- Use of formal methods must be justified,
  - **Event-B** is our approach.
  - models discrete systems, but used with continuous simulations.
  - but, it is just part of the engineering process.

Event-B

- Systems Modelling.
  - Specify important 'invariant' properties
    - make sure they hold as development progresses
    - Uses proof and/or model checking
- Various interfaces
    - text editor
    - class diagrams
    - state-machine diagrams
    - animations

Event-B Artefacts



Event-B Context

```
context C0
sets
 pointState
constants
 lastKnown  updated
axioms
 @axm1 "partition(pointState, {lastKnown}, {updated})"
end
```

Event-B Machine

```
machine M0
sees
 C0
variables
pointPos
invariants
@inv1 "pointPos ∈ pointState"
events
 event INITIALISATION  ordinary
 then @act1 "pointPos :∈ pointState"
 end
 event movePoint
 when @grd0_1 "pointPos = lastKnown"
 then @act0_1 "pointPos ≔ updated"
 end

 event reset ordinary
  when @grd0_1 "pointPos = updated"
  then @act0_1 "pointPos ≔ lastKnown"
 end

end
```

specify properties

atomic, guarded state updates

Agile Development

- Using the **DAD Book** for meta-analysis.
- DAD,
  - has elements of Scrum, XP, Lean, etc.
  - is process goal-driven, but not prescriptive.
  - provides an adaptable, framework.

- Research Questions
  - How can agile techniques improve Event-B?
  - How can Event-B be used in an agile development?
  - How can we best use **metrics** with Event-B?

Agile Influences on HI System Development with Event-B

- The process: a lot is obvious; but may be **difficult to do**,
  - short iterations.
  - TDD with Event-B, without automatic code generation.
  - continuous integration, without automatic code generation.
- Event-B: the obvious:
  - improve refactoring.
  - already has MDD, with executable models.
  - fits into a modified iterative approach.
  - use metrics, but code might not be the main measure.
  - **improve reuse with Components.**
- New for Event-B: **Process Goals**,
  - A major part of DAD.
  - Can modelling with Event-B benefit from this – we think so!

Agile Influences on HI System Development

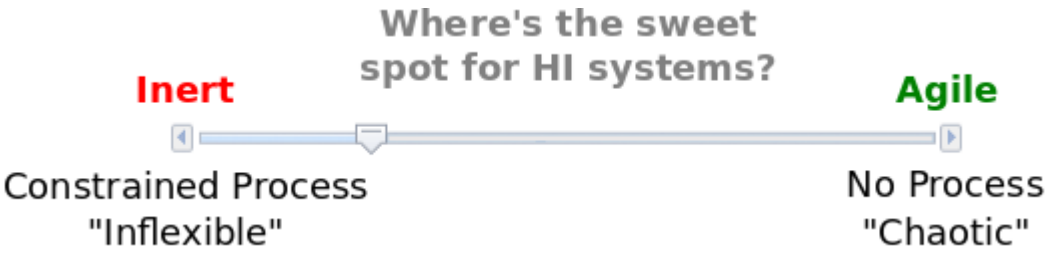
- The process: a lot is obvious, but may be difficult to do,
  - short iterations.
  - TDD with Event-B, without automatic code generation.
  - continuous integration, without automatic code generation.
- Event-B: the obvious:
  - improve refactoring.
  - already has MDD, with executable models.
  - fits into a modified iterative approach.
  - use metrics, but code might not be the main measure.
  - **improve reuse - develop Components.**
- New for Event-B: **Process Goals**,
  - A major part of DAD.
  - Can modelling with Event-B benefit from this – we think so!

Agile Influences on HI System Development

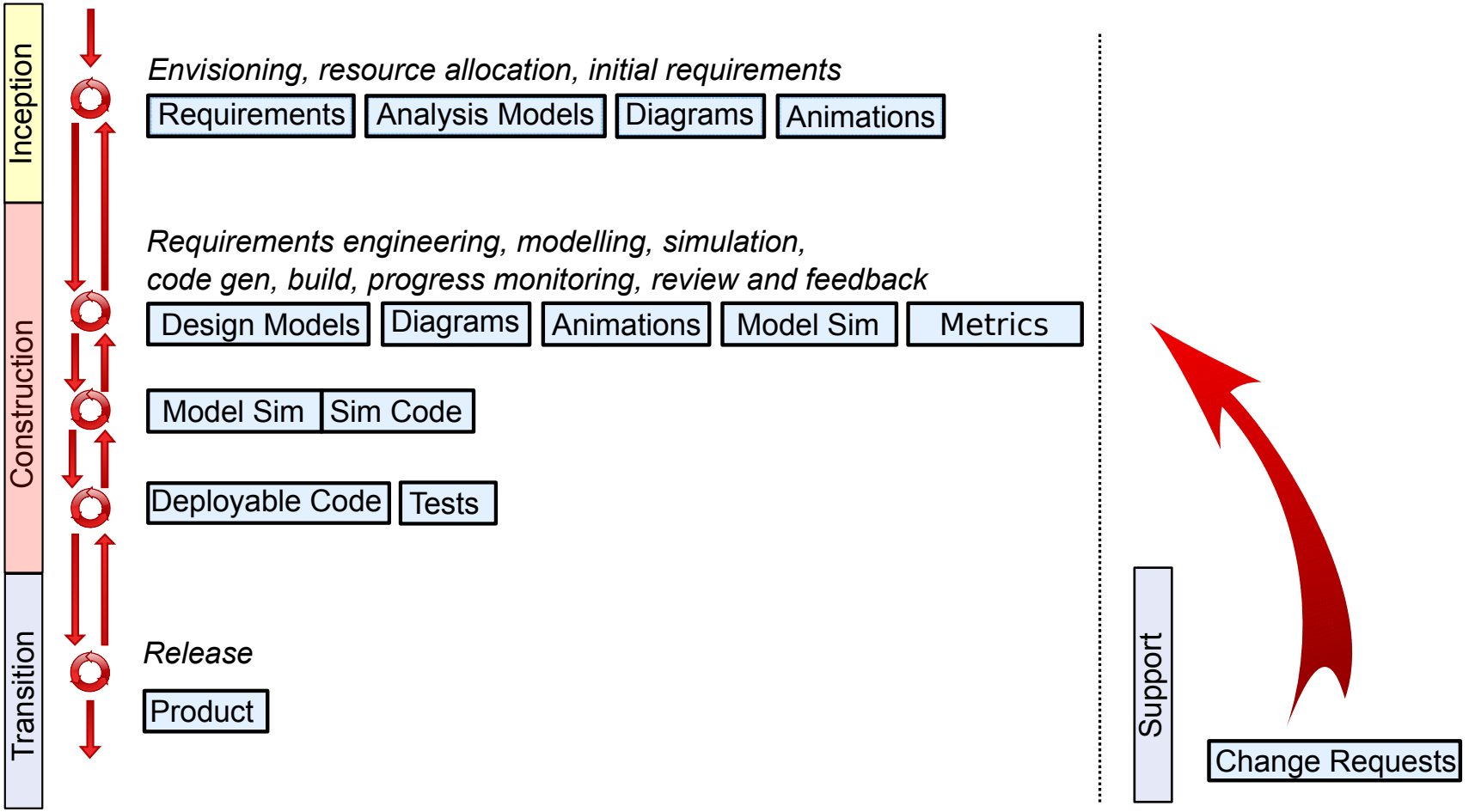
- The process: a lot is obvious, but may be difficult to do,
  - short iterations.
  - TDD with Event-B, without automatic code generation.
  - continuous integration, without automatic code generation.
- Event-B: the obvious:
  - improve refactoring.
  - already has MDD, with executable models.
  - fits into a modified iterative approach.
  - use metrics, but code might not be the main measure.
  - **improve reuse with Components.**
- New for Event-B: **Process Goals**,
  - A major part of DAD.
  - Can modelling with Event-B benefit from this – we think so!

Changing DAD, for HI Systems Development?

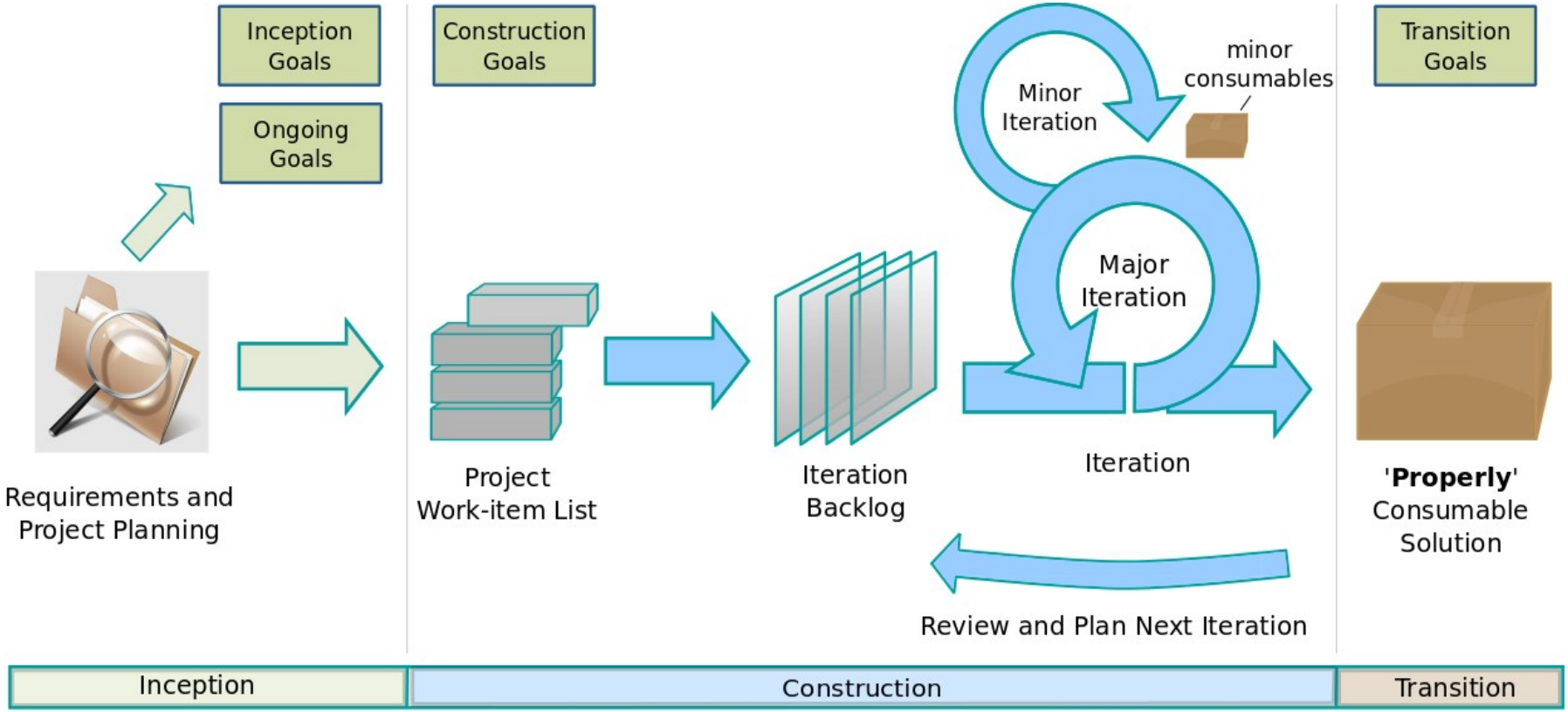
- A different view of agile,
  - but that's also true within the agile community.
  - we redefine consumable solutions.
  - We need major/minor iterations.



Event-B (Consumables) in DAD



A DAD Life-cycle Diagram



Process Goals with Event-B

- Process goal derivation,
  - Using a list of considerations (process factors).
  - extend goals for use with HI system development.
  - leads to requirement for guidelines, patterns, and components.
  - guidelines should take into account new/expert users.
  - existing guidelines and patterns are widely dispersed, can we provide linked data?
  - requires development of component library.

Finally: Agile in HI Systems Development

- Previous work -
  - R. Paige, R. Charalambous et al. (York)
  - F. Redmill (Newcastle)
  - Safety Critical Systems Club (UK)

- Acknowledgements
  - Åbo Akademi (ADVICeS Project)
  - Marina Walden, Marta Olszewksa