

The Task-Resource View

A domain specific view for manufacturing systems

IPA Ientedagen 2005, Maastricht
Arie van Deuren, Bas Graaf
b.s.graaf@ewi.tudelft.nl
March 31, 2005

Contents

- Motivation
- Architectural views
- Supervisory control and task resource systems
- Domain specific viewpoint: Task-resource viewpoint
- Intermezzo: architecture reconstruction (Symphony)
- Task-resource view applications
 - Conformance checking
 - Migration support
 - Forward engineering
- Conclusion

Motivation

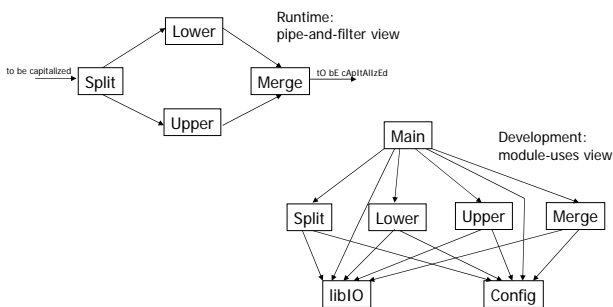
- Scheduling in manufacturing machines
- ASML is studying TRS-approach [Nieuwelaar 2004]
- Tasks and resources are central concepts

- What are the architectural implications?
 - Architectural views
 - Migration support
 - Conformance

Architectural Views

- Architectural views are used to specify relevant design decisions
- Views are associated with viewpoints
- Different types of views:
 - Runtime views (pipe-and-filter, client-server, etc...)
→ *How does the system work? Runtime structure*
 - Development views (uses, decomposition, etc...)
→ *How is the system developed? Code structure*
 - Configuration views (deployment, work assignment, etc...)
→ *How are views related to each other and their context?*

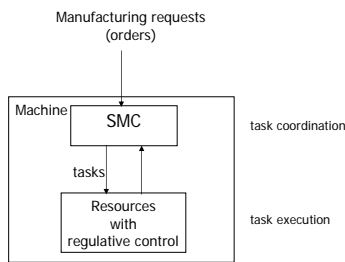
Example: capitalize



Research Questions

- Which of the traditional views can be used to understand task-resource systems?
- Are traditional views sufficient or do we need a new type of view?

Supervisory Machine Control (SMC)



[Nieuwelaar 2004]

b.s.graaf@ewi.tudelft.nl

7

Task-Resource System (TRS)

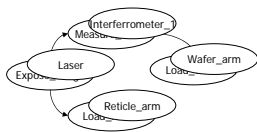
- Mathematical model
- Manufacturing request (order) → TRS
- Manufacturing processes → tasks
- Electromechanical subsystems → resources
- 3 transformations from order to timed schedule:
 - Instantiating → Partially ordered set of tasks (plan)
 - Selecting → Plan + resources
 - Timing → Plan + resources + start/end times (schedule)

b.s.graaf@ewi.tudelft.nl

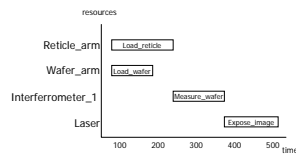
8

TRS Example

Expose_wafer



Order ...
Instantiation ...
Selecting ...
Timing ...



b.s.graaf@ewi.tudelft.nl

9

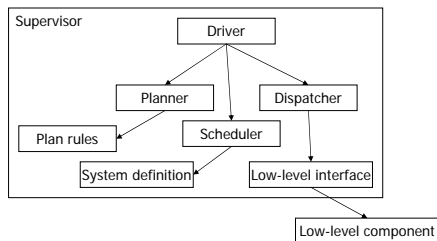
SMC: Complexity

- Multiple orders
- Orders comprising multiple tasks
- Task precedence
- Multiple tasks having the same result
- Multiple resources capable of executing the same task
- Resources capable of executing multiple tasks
- Material logistics
- Resource interference

b.s.graaf@ewi.tudelft.nl

10

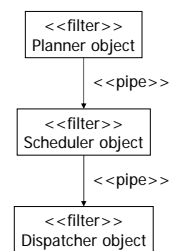
SMC: Implementation (uses view)



b.s.graaf@ewi.tudelft.nl

11

SMC: Implementation (pipe-and-filter view)



b.s.graaf@ewi.tudelft.nl

12

Context

- These 'traditional' views are domain-independent and aimed at development
- Our case was different:
 - Flexible supervisory machine control
→ specific domain
 - Reference architecture + library
→ aimed at instantiating this reference architecture
 - Existing SMC-components
→ aimed at migration to new approach

b.s.graaf@ewi.tudelft.nl

13

Domain-Specific View

- 'Traditional' views show design decisions already taken
- Variability is modularized in `Plan rules` and `System definition`
- Need view expressive over variability to specify product instances: tasks, resources, capabilities, ...

→ a domain specific view: task-resource view

b.s.graaf@ewi.tudelft.nl

14

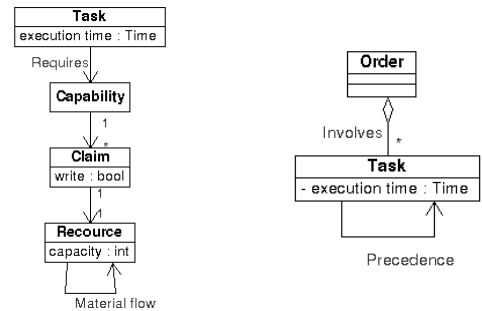
Viewpoint

- Conventions for constructing and using a view
 - Stakeholders
 - Concerns
 - Analyses
 - Modelling language / meta-model (primary presentation)
- Modelling language / meta-model
 - Elements
 - Relations
 - Properties of elements
 - Properties of relations
 - Topological constraints

b.s.graaf@ewi.tudelft.nl

15

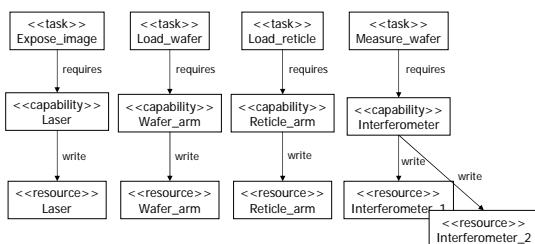
Task-Resource Viewpoint Meta-Model



b.s.graaf@ewi.tudelft.nl

16

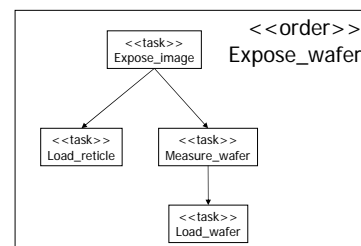
Task-Resource View Example (system-dependent part)



b.s.graaf@ewi.tudelft.nl

17

Task-Resource View Example (work-dependent part)



b.s.graaf@ewi.tudelft.nl

18

Summary

- Variability in terms of orders, tasks and resources (source of customer change requests)
- Reference architecture and library available
 - Fixed development and runtime views
- Defined viewpoint expressing undefined part

b.s.graaf@ewi.tudelft.nl

19

Architecture?

- Alexander Ran:
"design decisions that are hard to change"
→ NO
- Michel Chaudron:
"design decisions that affect system quality"
→ YES

b.s.graaf@ewi.tudelft.nl

20

Applications

- Reverse engineering
 - Migration support
 - Conformance checking (against reference architecture and product instance specification)
- Forward engineering
 - DSL design
 - Checklist, graphical presentation of product instance
 - Basis for reasoning framework over design decisions wrt, orders, tasks, and resources

b.s.graaf@ewi.tudelft.nl

21

Architecture reconstruction: motivation

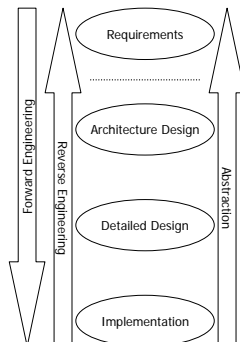
- No documentation available at all
 - Documentation not up-to-date
 - Relation between architecture and implementation unclear
- Recovering design knowledge: e.g. for migrations

b.s.graaf@ewi.tudelft.nl

22

Reverse engineering

- Create representation of a system at a higher level of abstraction using different sources, e.g.:
 - Program code
 - Documentation
 - Developer experience
 - Domain knowledge
 - Executing system
- Architecture reconstruction



b.s.graaf@ewi.tudelft.nl

23

Reconstruction viewpoints

- Target viewpoints (modules, components, connectors, uses, is-part-of, ...)
- Source viewpoints (classes, methods, declarations, inheritance, object creation, methods calls...)
- Mapping rules (source → target)
- *All very much depended on the problem at hand and need to be established for every reverse engineering effort!*

b.s.graaf@ewi.tudelft.nl

24

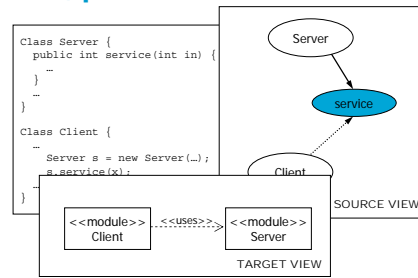
Very simple example: viewpoints definitions

- Source viewpoint
 - Elements: classes, methods
 - Relations: defines, uses
- Target viewpoint: module/uses (DSA)
- Mapping rules
 - Class → module
 - Definition, use → module/uses

b.s.graaf@ewi.tudelft.nl

25

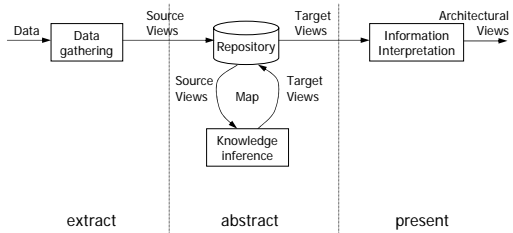
Very simple example: extract, abstract, present



b.s.graaf@ewi.tudelft.nl

26

Architecture reconstruction process: Symphony



[Van Deursen et al.]

b.s.graaf@ewi.tudelft.nl

27

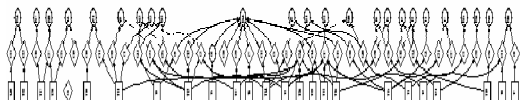
Applications: Migration Support

- Extract a task-resource view from existing SMC-controllers
- Symphony viewpoints:
 - Target viewpoint: task-resource viewpoint
 - Source viewpoint: callgraph of existing controller
 - Mapping rules: task function -> task
order function -> order
sub components -> resources
+ heuristics
- Symphony processes:
 - Data gathering: grep, sed, perl, bauhaus
 - Knowledge inference: croccopat, rigi standard format
 - Presentation: rigi, bauhaus, dotty

b.s.graaf@ewi.tudelft.nl

28

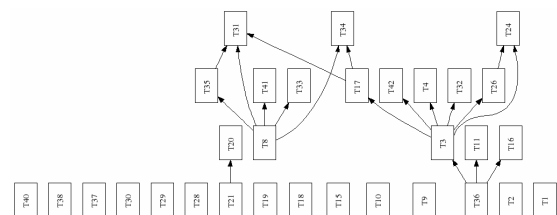
Task-Resource View Example (system-dependent part)



b.s.graaf@ewi.tudelft.nl

29

Task-Resource View Example (work-dependent part)



b.s.graaf@ewi.tudelft.nl

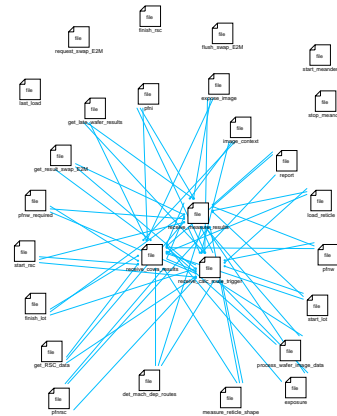
30

Applications: Conformance Checking

- Check if design rules imposed by reference architecture are not violated:
 - Tasks are only dispatched by dispatcher
 - Resources are only invoked by low-level interface modules
 - ...
- Check if the task-resource view itself is not violated
 - Tasks invoked for an order
 - Resources executing tasks
 - ...

b.s.graaf@ewi.tudelft.nl

31



b.s.graaf@ewi.tudelft.nl

32

Applications: Forward Engineering

- Domain specific language
 - UML, XMI, Perl
 - Eclipse, EMF (MOF), and code generation
- Explicit modelling using tasks and resources
 - Checklist
 - Reasoning framework (what-if analysis)

b.s.graaf@ewi.tudelft.nl

33

Conclusions

- Task-resource viewpoint
 - Manufacturing machines involving supervisory control
 - Other supervisory control applications
- Concept
 - Domain-specific view
 - Variability / configurability
 - Reference architectures

b.s.graaf@ewi.tudelft.nl

34

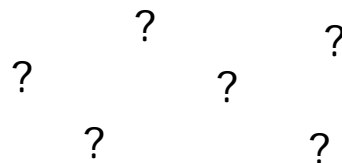
References

- Barend van den Nieuwelaar, "Supervisory control by predictive-reactive scheduling", TU/e, 2004.

b.s.graaf@ewi.tudelft.nl

35

Q & A



b.s.graaf@ewi.tudelft.nl

36