FONDA

Foundations of Workflows for Large-Scale Scientific Data Analysis

CRC 1404



Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

ßMACH

—

A Software Management Guidance

Marcus Hilbrich, Fabian Lehmann

marcus.hilbrich@informatik.hu-berlin.de

FONDA

Foundations of Workflows for Large-Scale Scientific Data Analysis (Collaborative Research Center 1404)

Humboldt-Universität zu Berlin

Technische Universität zu Berlin

Freie Universität Berlin

Charité – Universitätsmedizin Berlin

Max-Delbrück Center for Molecular Medicine

Hasso-Plattner Institut an der Universität Potsdam

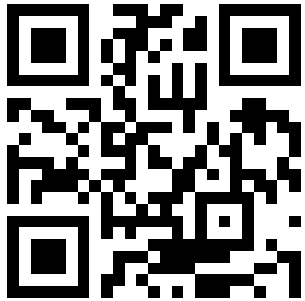
Zuse Institut Berlin

Fraunhofer Heinrich-Hertz-Institut Berlin

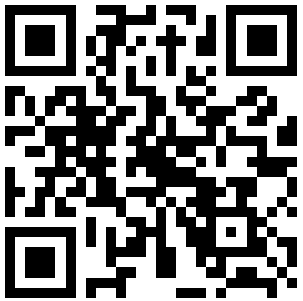
FONDA

- Mailing Lists
- Lecture Series
- Guest Program
- Focus (1st Phase)
  - Portability
  - Adaptability
  - Dependability

https://fonda.hu-berlin.de



marcus.hilbrich@informatik.hu-berlin.de



FONDA

Foundations of Workflows for Large-Scale Scientific Data Analysis

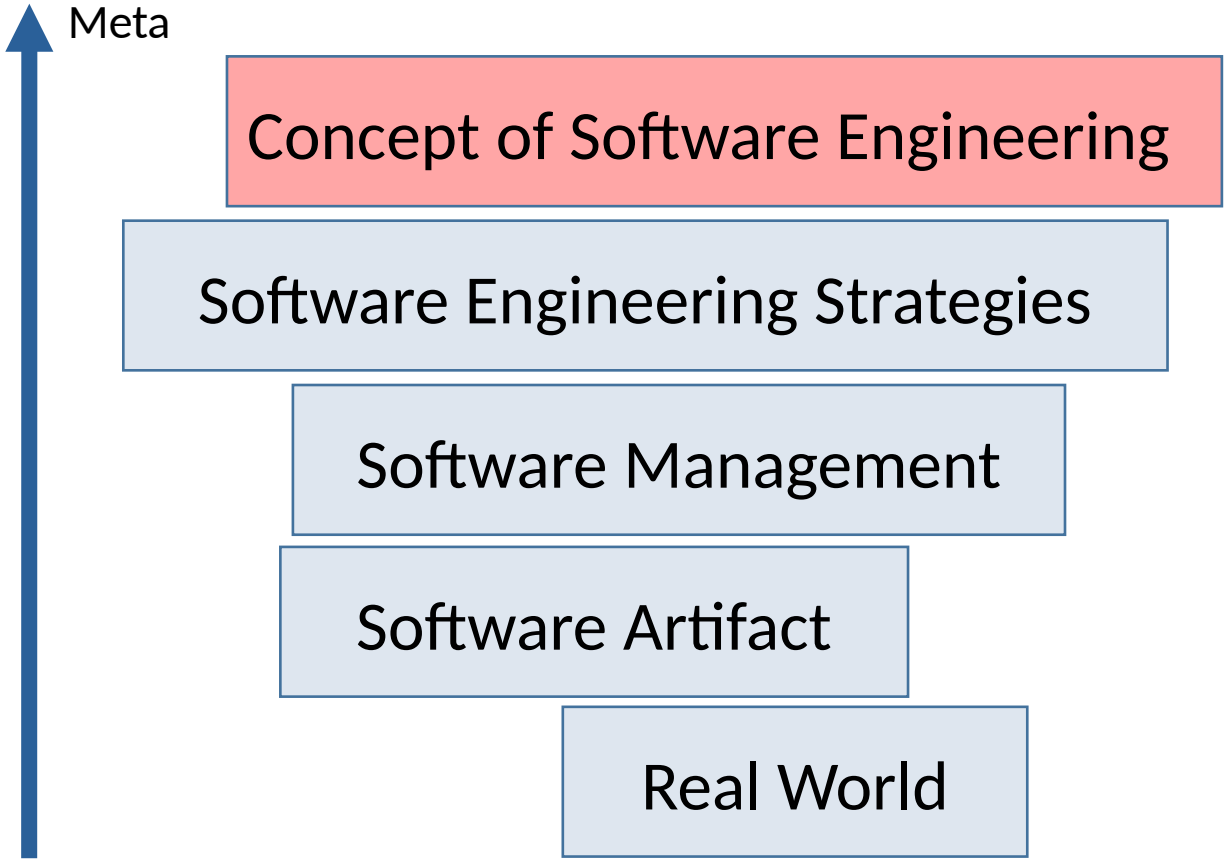
CRC 1404

Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

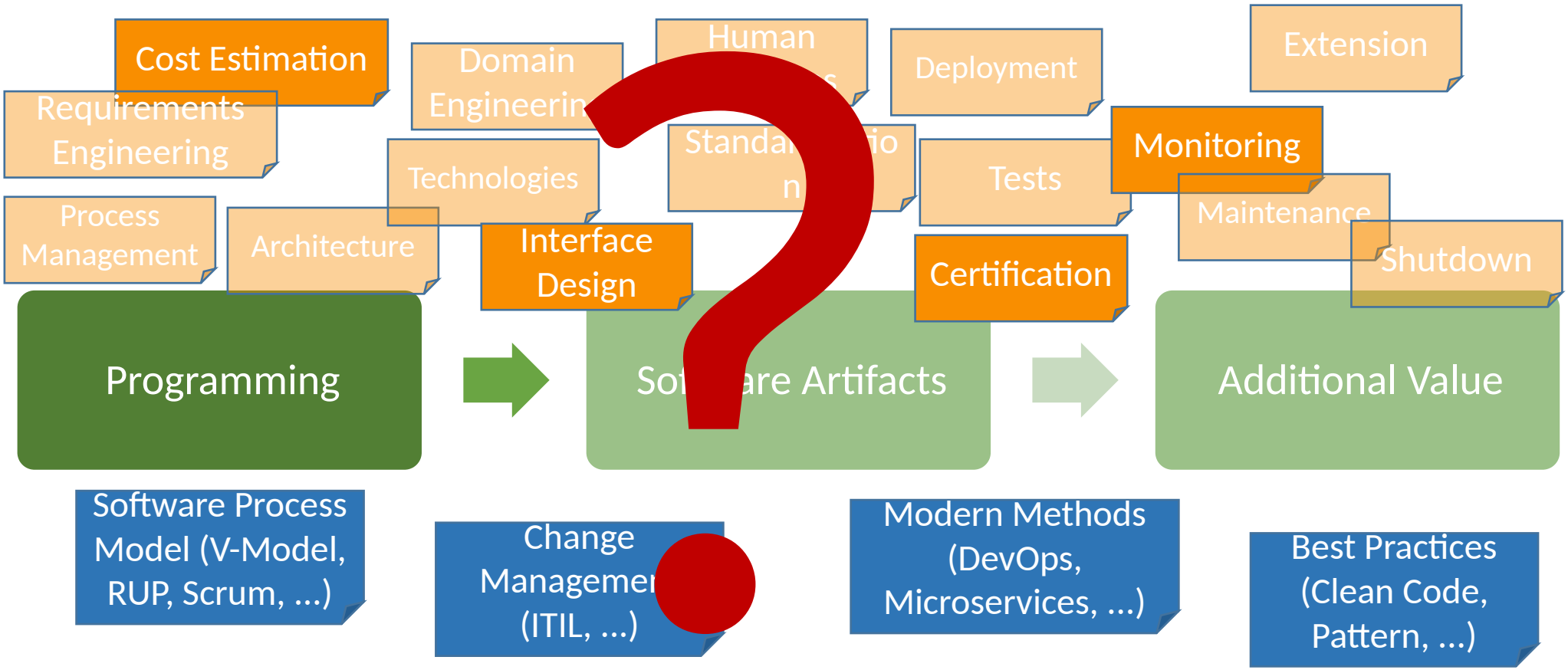
ßMACH

Systematic Software Management Approaches Characterization Helper

Abstraction



Computer Science (simplified)



Management Description (V-Model and SCRUM)

The list of requirements is created by requirements engineering and is the **definition** of the product to produce.

The use cases are continuously discussed with the **buyer** of the product and are documented in backlogs.

No direct comparison but:

- Product aspects are needed (defining the product)
- Product knowledge, demanded knowledge, roles, process knowledge, process information (defining the development)
- Development, maintenance, improvement (work package definition)

Management Description (general)

- Terms, Language?
- Similarities?
- Differentiation points?

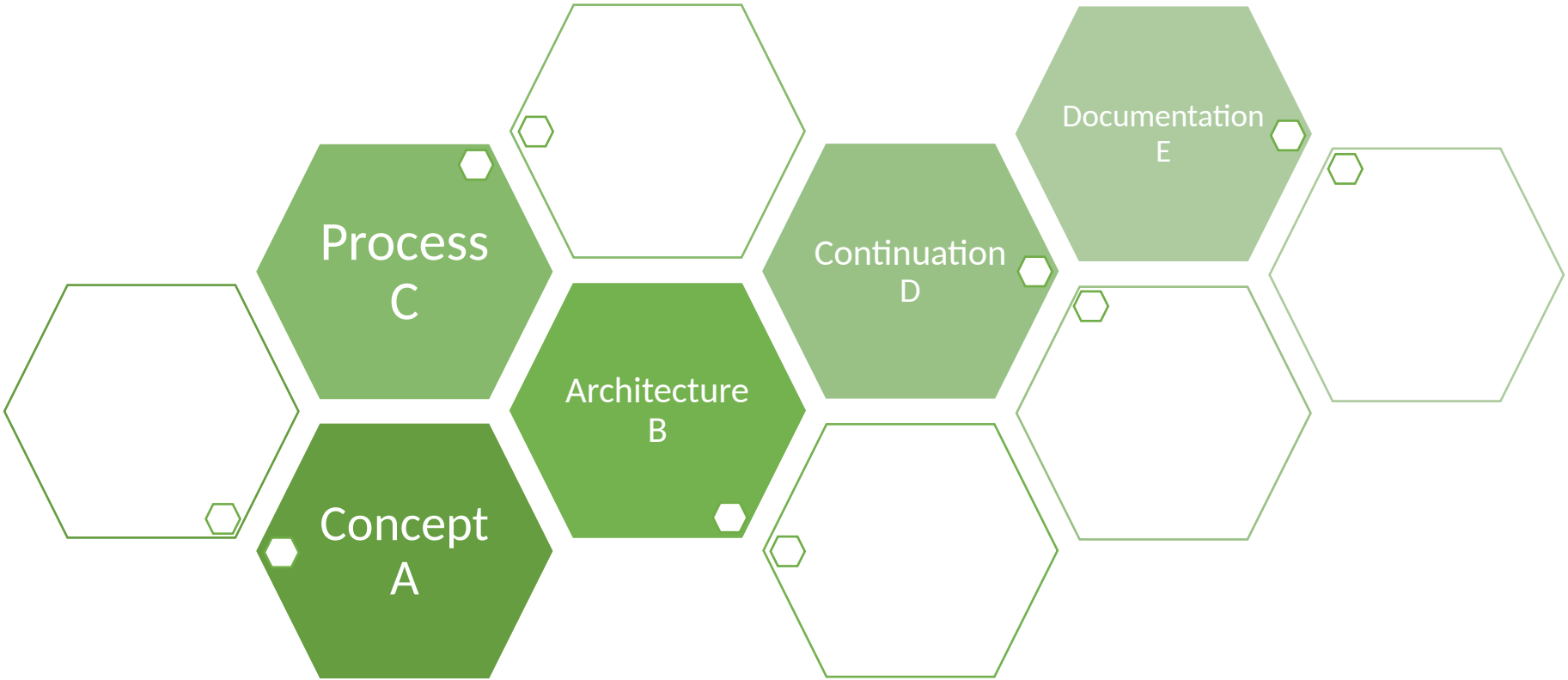
- Selection?
- Combination of ...?
- Tailoring?

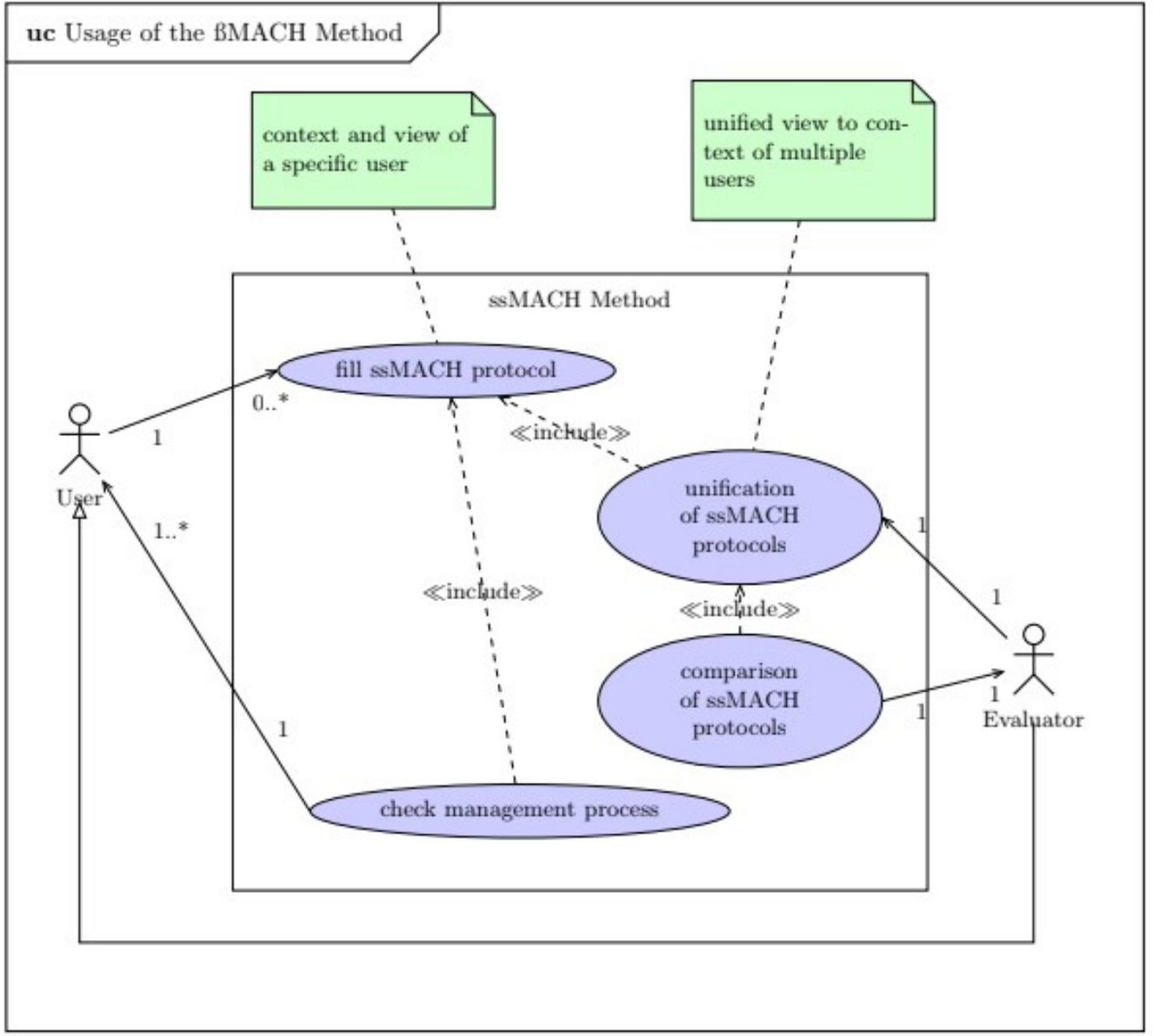
- Integration of:
  - Actual processes
  - Concepts
  - Architectures
  - …

Definition of the Management Process

(complete and minimal)

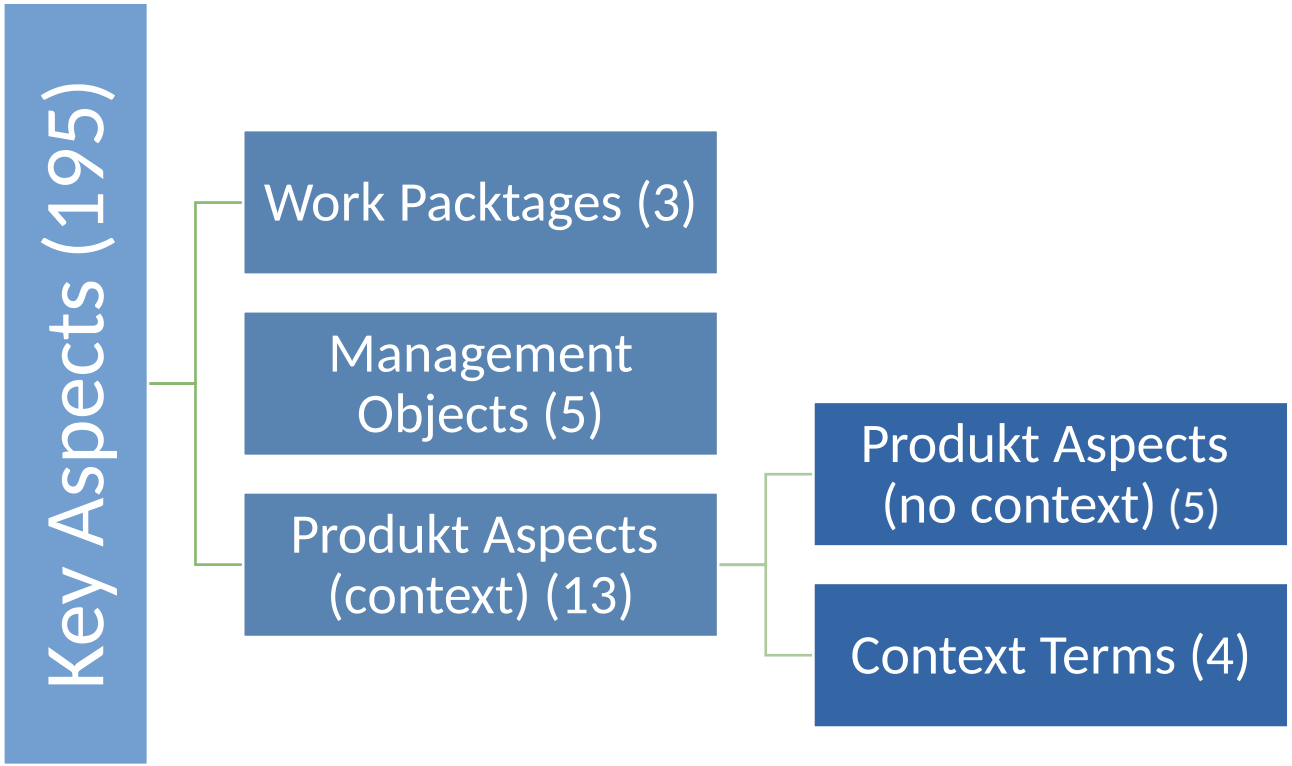
Flexible Management Definition




uc Usage of the ßMACH Method
context and view of a specific user
unified view to context of multiple users
ssMACH Method
fill ssMACH protocol
0..*
1
User
1..*
≪include≫
≪include≫
≪include≫
unification of ssMACH protocols
comparison of ssMACH protocols
check management process
1
1
1
1
1
1
Evaluator

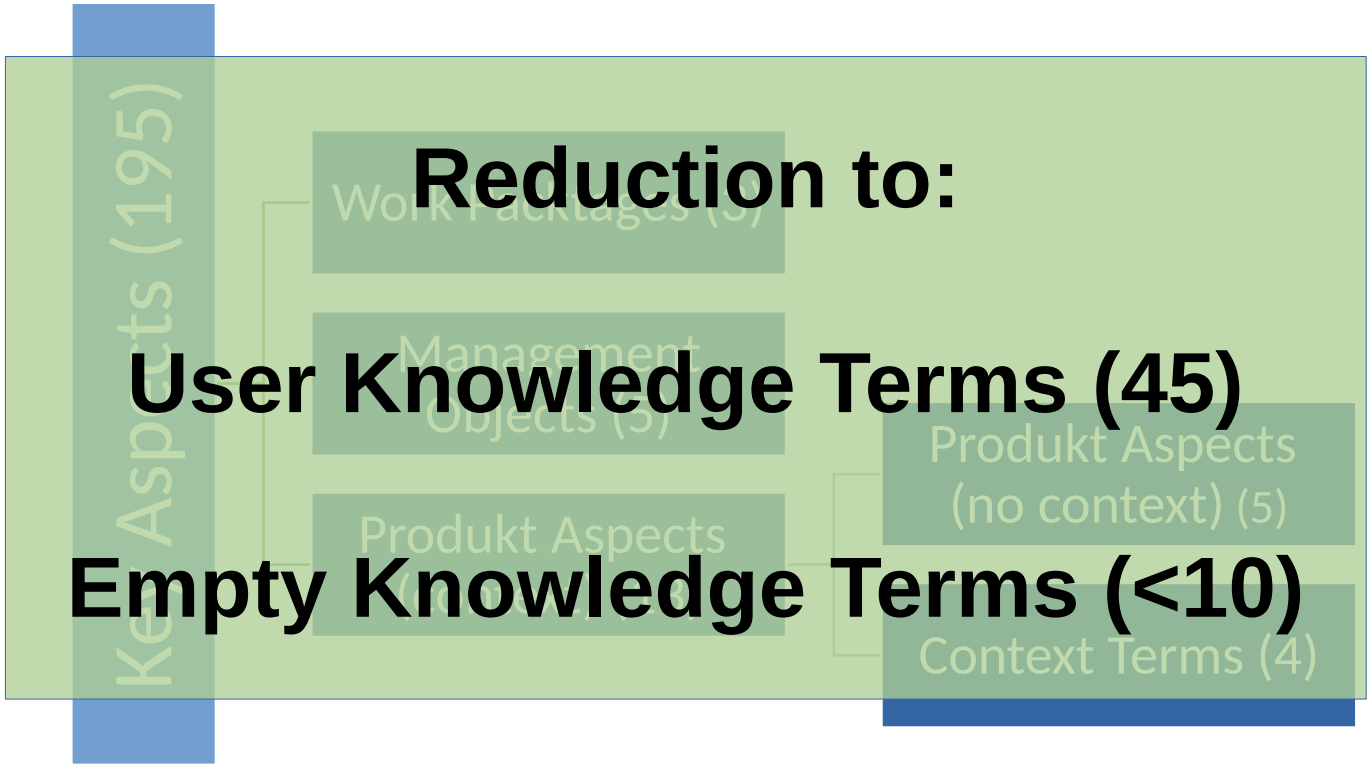
Software Engineering Level

- Ontology of Key Aspects
- Abstract description of software artifact management



Software Engineering Level

- Ontology of Key Aspects
- Abstract description of software artifact management



ßMACH Protocol (Example)

ßMACH Protocol to Evaluate the Management of Software Artifacts

Work Package Responsibilities:

- ☑ Product Development
- ☑ Product Maintains
- ☑ Product Improvement

Definition of the Management Concept:

1. The microservice system consists of microservices, microservices have no (or minimal) dependences to each other.
2. Microservices represent encapsulated system concerns that are delivered via interfaces to endusers.
3. A microservice is managed by a DevOps team that provides all needed knowledge and manages itself.
4. The separation of system concerns to microservices has to be realized, how to do so is not covered by the microservice concept.

Context (User/Team/Context Information):

Name of the Filler:	Marcus Hilbrich
Represented Team (Internal Border):	Microservice Team A: one of the teams managing microservices of the overall microservice system
Cooperating Teams (External Border):	All teams working on the same microservice system.
Managed Artifacts:	A subset of the microservices of the overall microservice system.
Date:	2020/06/11
Version of Document:	1.0

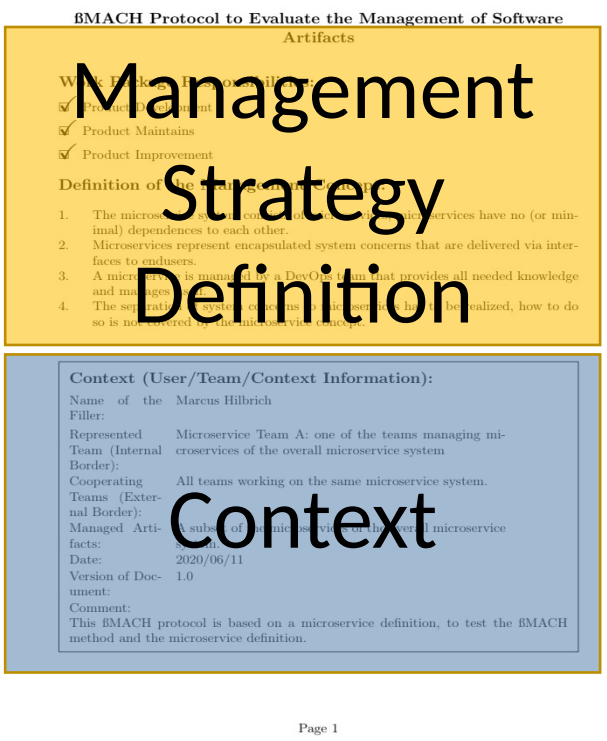
Comment:
This ßMACH protocol is based on a microservice definition, to test the ßMACH method and the microservice definition.

Page 1

Page 2

	+1exProduct Knowledge	+1exDemanded Knowledge	+1exRoles	+1exProcess Knowledge	+1exProcess Information
Explanation for Aspects the Team is Not Responsible for:					
Product Development	——— –	——— –	——— –	——— –	——— –
Product Maintains	——— –	——— –	——— –	——— –	——— –
Product Improvement	——— –	——— –	——— –	——— –	——— –
Explanation for Aspects the Team is Responsible for:					
Inside Product Properties	Dependencies of microservices are minimised and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Product Properties	Defined by a subset of the system concerns (via the interfaces), managed by DevOps. 2	The DevOps team has the knowledge how to do, the interface definition.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Defined by a subset of the system concerns (via the interfaces), Managed by DevOps. 2
Inside Dependencies	Dependencies of microservices are minimized and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Dependencies	Can be given by system concerns or by the DevOps team. 2	No additional knowledge needed.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Can be requested by the system concerns covered in this row. 2
Inside Interfaces	Dependencies of microservices are minimized and do not need to be managed. 1 2	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Interfaces	Representation of a subset of the system concerns delivered to the enduser. 2	Given by the outside responsibilities, managed by DevOps, so no open demands.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Representation of a subset of the system concerns delivered to the enduser. 2
Inside Responsibilities	Dependencies of microservices are minimized and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Responsibilities	Given by a subset of the system concerns. 2	The separation of system concerns to microservices. 4	??? 4	??? 4	Given by a subset of the system concerns. 2
External Artifacts	Can be given by system concerns by the DevOps team. 2	No additional knowledge needed.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Can be requested by the system concerns covered in this row. 2

ßMACH Protocol (Example)



ßMACH Protocol to Evaluate the Management of Software Artifacts

Management Strategy

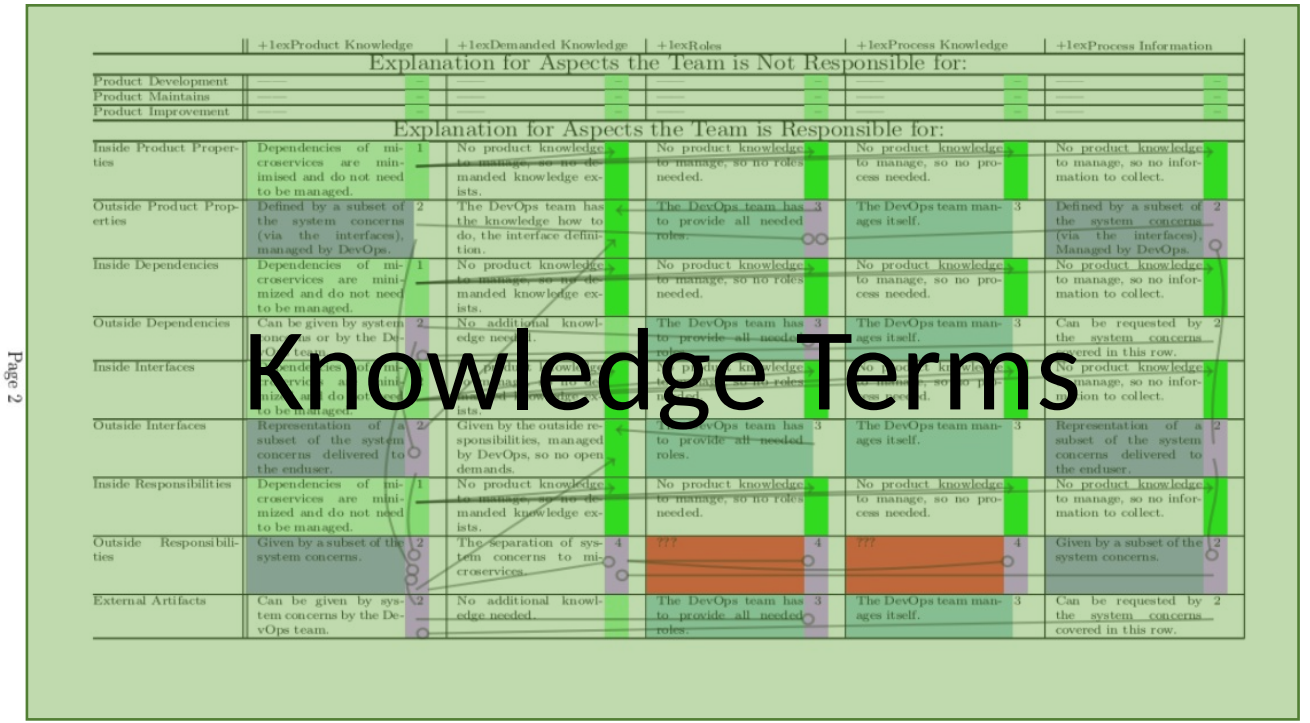
Definition

Context

Context (User/Team/Context Information):

Name of the Filler:	Marcus Hilbrich
Represented Team (Internal Border):	Microservice Team A: one of the teams managing microservices of the overall microservice system
Cooperating Teams (External Border):	All teams working on the same microservice system.
Managed Artifacts:	A subset of the microservice-internal microservice
Date:	2020/06/11
Version of Document:	1.0
Comment:	This ßMACH protocol is based on a microservice definition, to test the ßMACH method and the microservice definition.

Page 1



Knowledge Terms

Explanation for Aspects the Team is Not Responsible for:

	+1exProduct Knowledge	+1exDemanded Knowledge	+1exRoles	+1exProcess Knowledge	+1exProcess Information
Product Development	———	———	———	———	———
Product Maintains	———	———	———	———	———
Product Improvement	———	———	———	———	———

Explanation for Aspects the Team is Responsible for:

Inside Product Properties	Dependencies of microservices are minimised and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Product Properties	Defined by a subset of the system concerns (via the interfaces), managed by DevOps. 2	The DevOps team has the knowledge how to do, the interface definition.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Defined by a subset of the system concerns (via the interfaces), Managed by DevOps. 2
Inside Dependencies	Dependencies of microservices are minimized and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Dependencies	Can be given by system concerns or by the DevOps team. 2	No additional knowledge needed.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Can be requested by 2 the system concerns covered in this row.
Inside Interfaces	Dependencies of microservices are minimized and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Interfaces	Representation of a subset of the system concerns delivered to the enduser. 2	Given by the outside responsibilities, managed by DevOps, so no open demands.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Representation of a subset of the system concerns delivered to the enduser. 2
Inside Responsibilities	Dependencies of microservices are minimized and do not need to be managed. 1	No product knowledge to manage, so no demanded knowledge exists.	No product knowledge to manage, so no roles needed.	No product knowledge to manage, so no process needed.	No product knowledge to manage, so no information to collect.
Outside Responsibilities	Given by a subset of the system concerns. 2	The separation of system concerns to microservices. 4	??? 4	??? 4	Given by a subset of the system concerns. 2
External Artifacts	Can be given by system concerns by the DevOps team. 2	No additional knowledge needed.	The DevOps team has to provide all needed roles. 3	The DevOps team manages itself. 3	Can be requested by 2 the system concerns covered in this row.

Page 2

Thanks!

Questions