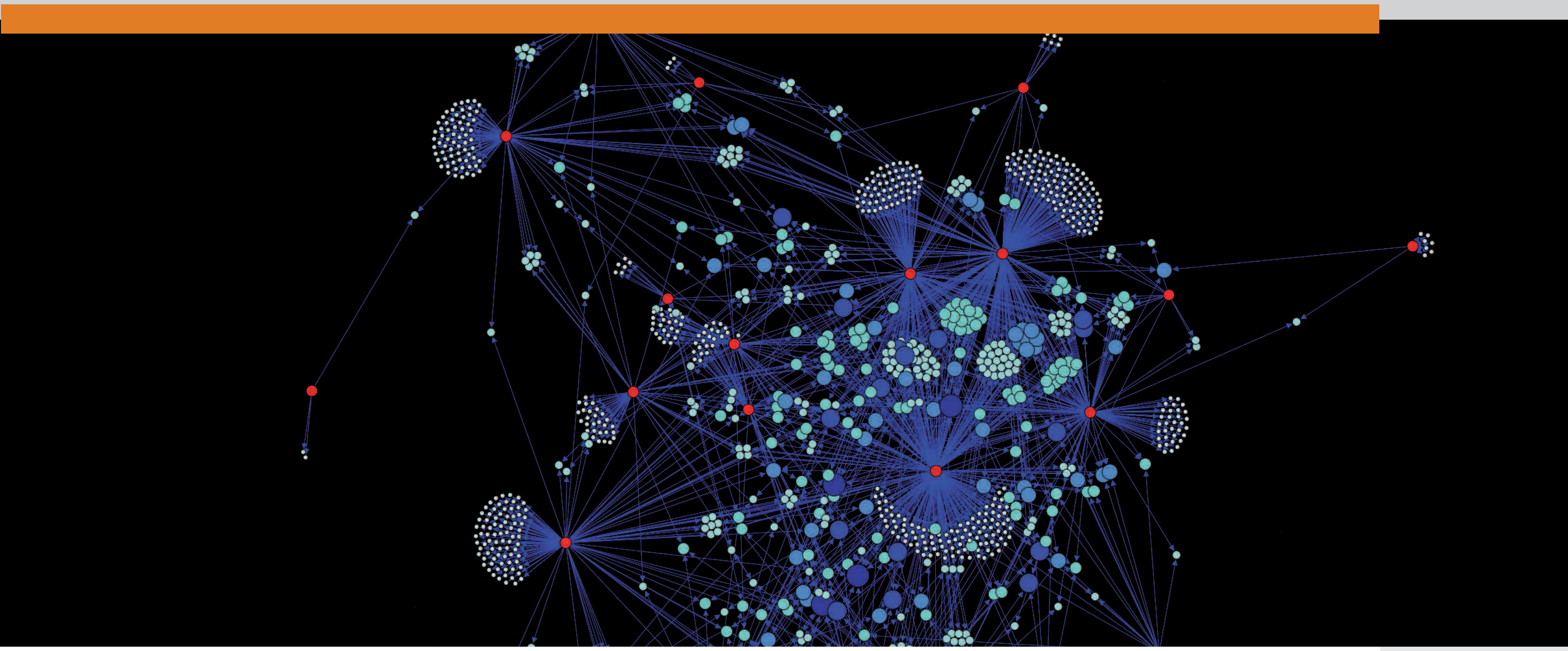
German
Research Center
for Artificial
Intelligence

Document D-15-01



Proceedings of the RIC Project Day

Workgroup *'Framework & Standardization'*

Frank Kirchner (Editor)
Thomas M. Roehr (Associate Editor)

03/2015

Document D-15-01
German Research Center for Artificial Intelligence (DFKI) GmbH

Bibliographic information published by the German National Library

The German National Library lists this publication in the German National Biography; detailed bibliographic data are available in the internet at http://dnb.ddb.de.

© German Research Center for Artificial Intelligence (DFKI) GmbH, 2015

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the German Research Center for Artificial Intelligence (DFKI) GmbH, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to German Research Center for Artificial Intelligence (DFKI) GmbH.

Issue D-15-01 (2015)
ISSN 0946-0098

German Research Center for Artificial Intelligence
Deutsches Forschungszentrum für Künstliche Intelligenz
DFKI GmbH

Founded in 1988, DFKI today is one of the largest nonprofit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation – from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialization.

Based in Kaiserslautern, Saarbrücken and Bremen, the German Research Center for Artificial Intelligence ranks among the important ‘Centers of Excellence’ worldwide. An important element of DFKI’s mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science DFKI has the strength to meet its technology transfer goals.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO). DFKI’s research departments are directed by internationally recognized research scientists:

- Knowledge Management (Prof. A. Dengel)
- Cyber-Physical Systems (Prof. R. Drechsler)
- Robotics Innovation Center (Prof. F. Kirchner)
- Innovative Retail Laboratory (Prof. A. Krüger)
- Institute for Information Systems (Prof. P. Loos)
- Embedded Intelligence (Prof. P. Lukowicz)
- Agents and Simulated Reality (Prof. P. Slusallek)
- Augmented Vision (Prof. D. Stricker)
- Language Technology (Prof. H. Uszkoreit)
- Intelligent User Interfaces (Prof. W. Wahlster)
- Innovative Factory Systems (Prof. D. Zühlke)

In this series, DFKI publishes research reports, technical memos, documents (eg. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster
Director

Proceedings of the RIC Project Day

Workgroup 'Framework & Standardization'

Frank Kirchner [1,2] (Editor)

Thomas M. Roehr [1] (Associate Editor)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

(2) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

03/2015

Document D-15-01 des

Deutschen Forschungszentrums für Künstliche Intelligenz (DFKI)

Abstract

This document is the current edition of a publication series which records the topics, discussions and efforts of the workgroups at the DFKI Robotics Innovation Center (RIC). Each edition contains presentation slides and posters of a project day which is organized by two workgroups.

Workgroups provide a platform for cross-project communication and knowledge transfer. They are formed by peers dedicated to a specific topic. Each workgroup has one administrator. In 2008, the workgroups started to present their results and efforts in an open presentation format called brown-bag talk. From 2009 onwards, these presentation were held at so-called project days. Since 2014, a project day consists of two main parts: an oral session and a poster session. Both sessions are documented in a proceedings using the DFKI Document format.

Zusammenfassung

Dieses Dokument enthält die aktuelle Ausgabe einer Tagungsbandserie, welche die Themen, Diskussionen und Bemühungen der Arbeitsgruppen am DFKI Robotics Innovation Center (RIC) protokolliert. Jede Ausgabe enthält Vortragsfolien und Poster eines Projekttages, der von je zwei Arbeitsgruppen gestaltet wird.

Arbeitsgruppen widmen sich einem bestimmten Themengebiet und stellen eine Plattform dar, um über Projekte hinaus zu kommunizieren und Wissen zu transferieren. Jede Arbeitsgruppe wird von einem sogenannten Kümmerer administriert. Im Jahr 2008 begannen die Arbeitsgruppen ihre Ergebnisse und Arbeiten in einem offenen Vortragsformat – dem sogenannten ‘Brown Bag Talk’ – vorzustellen, welches ein Jahr später in die Form von Projekttagen überführt wurde. Seit 2014 besteht ein Projekttag nicht nur aus Vorträgen, sondern beinhaltet zudem Posterpräsentationen. Beide Formate werden seitdem in einem Tagungsband in Form eines ‘DFKI Document’ festgehalten.

Contents

1 Editorial

This is the first edition of 2015 to document the efforts of the DFKI-RIC thematic workgroups on a deep content level and facilitate knowledge transfer amongst the peers. In 2008 we first started forming workgroups on specific topics around robotics and AI research. Among them were topics as ‘system design & engineering’, ‘machine learning’, ‘planning & representation’ as well as ‘frameworks & architectures’ and ‘man-machine interaction’. These workgroups intend to provide a platform for interested DFKI-RIC personnel for discussing the start of the art, recent achievements, and future developments in the respective fields.

This year’s project day season has been opened by the workgroup ’Framework & Standardization’. This workgroup shows a continuous effort to establish a shared software basis to facilitate the software development for complex robotic systems. To reach this goal it fosters knowledge sharing and code reuse, and establishes standards that lead to workflow optimizations. One intermediate and publicly visible results of this workgroup is its significant contribution to the Robot Construction Kit (Rock) – a framework that is gaining increasing attention even outside of this institute.

Frank Kirchner

This year’s first project day presented the material of the workgroup ‘Framework & Standardization’.

The workgroup ‘Framework & Standardization’ focuses its efforts on continuously improving the software development workflow and aims at supporting a software framework which fulfills the special needs of developers and systems in the domain of robotics. The workgroup’s primary motivation is to facilitate and accelerate routine tasks and to increase the robustness of the developed software. The workgroup has successfully established the Robot Construction Kit (Rock) as the main in-house development framework which can coexist with the well-known Robot Operating System (ROS).

The presentations of this year deal with technology adoption such as the application of the clang compiler and introduction of HTML5-based interfaces, while presenting the continuous effort of workflow optimization across the toolchain used for robotic software development. While the AG had introduced gitorious to account for a contemporary change towards git software repositories, the need arose for a better workflow for performing pull-requests. This requirement triggered the transition of the internal infrastructure from gitorious to gitlab. Improving user experience has been a driving factor in the past year, i.e., resulting in a robot UI using HTML5 as well as providing a C++-interface as alternative to the existing Ruby-scripting layer for managing Rock software modules. The presentations on ’constraint-based planning for component networks’ and on ’a generic description of manipulation behaviour’ illustrate the edge of robotic software development and at the same time the challenges of complex robotic system. The project day concluded with a review of the first workshop series for education of the inhouse Rock community.

I would like to thank all contributors of the first project day 2015 for creating an interesting and informative event.

Thomas M. Roehr

2 ‘Framework & Standardization’

2.1 ‘Introduction’ (FW-T-01)

Thomas M. Roehr[(1)]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany
(2) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `thomas.roehr@dfki.de`

Abstract

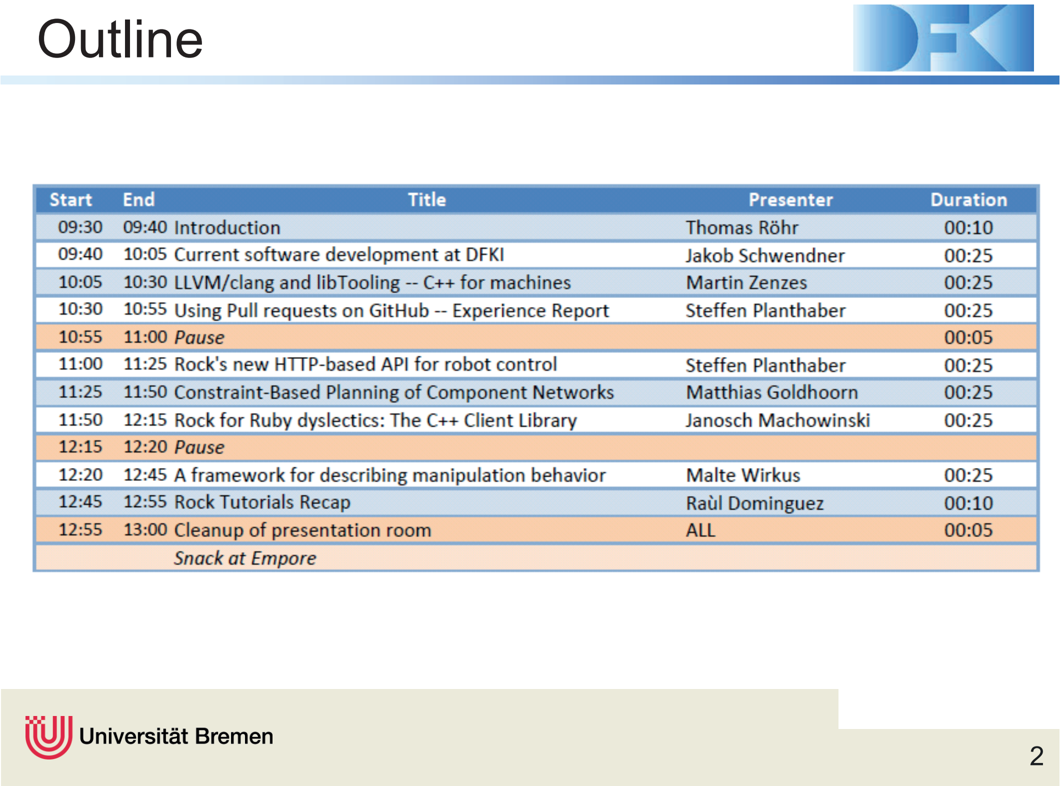
The introduction of this years project day presents the ongoing activities and highlights the past transition from the gitorious-based internal infrastructure to gitlab. Furthermore, a significant achievement has been made with the automated generation of Debian packages for Rock.

Project Day 2015
AG Framework and Standardization

Introduction by ‚Kümmerer' Thomas M. Roehr

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



Outline

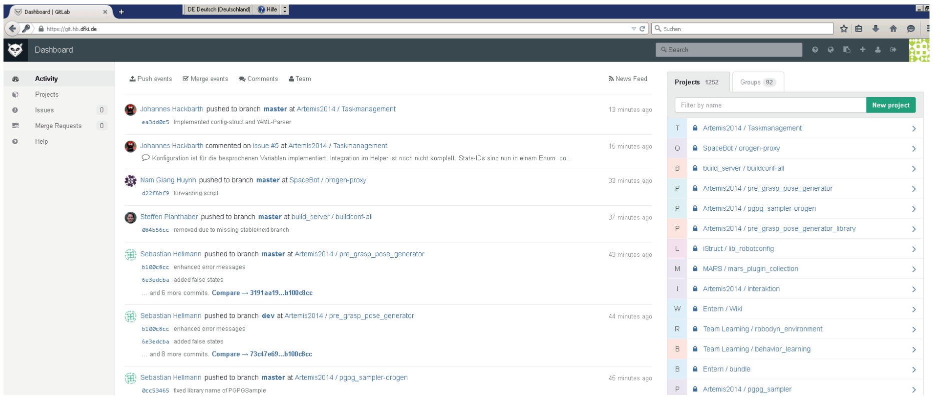
Start	End	Title	Presenter	Duration
09:30	09:40	Introduction	Thomas Röhr	00:10
09:40	10:05	Current software development at DFKI	Jakob Schwendner	00:25
10:05	10:30	LLVM/clang and libTooling -- C++ for machines	Martin Zenzes	00:25
10:30	10:55	Using Pull requests on GitHub -- Experience Report	Steffen Planthaber	00:25
10:55	11:00	*Pause*		00:05
11:00	11:25	Rock's new HTTP-based API for robot control	Steffen Planthaber	00:25
11:25	11:50	Constraint-Based Planning of Component Networks	Matthias Goldhoorn	00:25
11:50	12:15	Rock for Ruby dyslectics: The C++ Client Library	Janosch Machowinski	00:25
12:15	12:20	*Pause*		
12:20	12:45	A framework for describing manipulation behavior	Malte Wirkus	00:25
12:45	12:55	Rock Tutorials Recap	Raùl Dominguez	00:10
12:55	13:00	Cleanup of presentation room	ALL	00:05
		Snack at Empore		

Universität Bremen

2

Infrastructure changes

- Rock: moved from gitorious -> github
- Inhouse: moved from gitorious-based infrastructure to gitlab
  - spacegit -> git.hb.dfki.de



Johannes Hackbarth pushed to branch **master** at Artemis2014 / Taskmanagement — 13 minutes ago
ea3dd0c5 Implemented config-struct and YAML-Parser

Johannes Hackbarth commented on issue #5 at Artemis2014 / Taskmanagement — 15 minutes ago
Konfiguration ist für die besprochenen Variablen implementiert. Integration im Helper ist noch nicht komplett. State-IDs sind nun in einem Enum. co...

Nam Giang Huynh pushed to branch **master** at SpaceBot / orogen-proxy — 33 minutes ago
d22f6bf9 forwarding script

Steffen Planthaber pushed to branch **master** at build_server / buildconf-all — 37 minutes ago
084b56cc removed due to missing stable/next branch

Sebastian Hellmann pushed to branch **master** at Artemis2014 / pre_grasp_pose_generator — 43 minutes ago
b100c8cc enhanced error messages
6e3edcba added false states
... and 6 more commits. **Compare → 3191aa19...b100c8cc**

Sebastian Hellmann pushed to branch **dev** at Artemis2014 / pre_grasp_pose_generator — 44 minutes ago
b100c8cc enhanced error messages
6e3edcba added false states
... and 8 more commits. **Compare → 73c47e69...b100c8cc**

Sebastian Hellmann pushed to branch **master** at Artemis2014 / pgpg_sampler-orogen — 45 minutes ago
0cc53465 fixed library name of PGPGSample

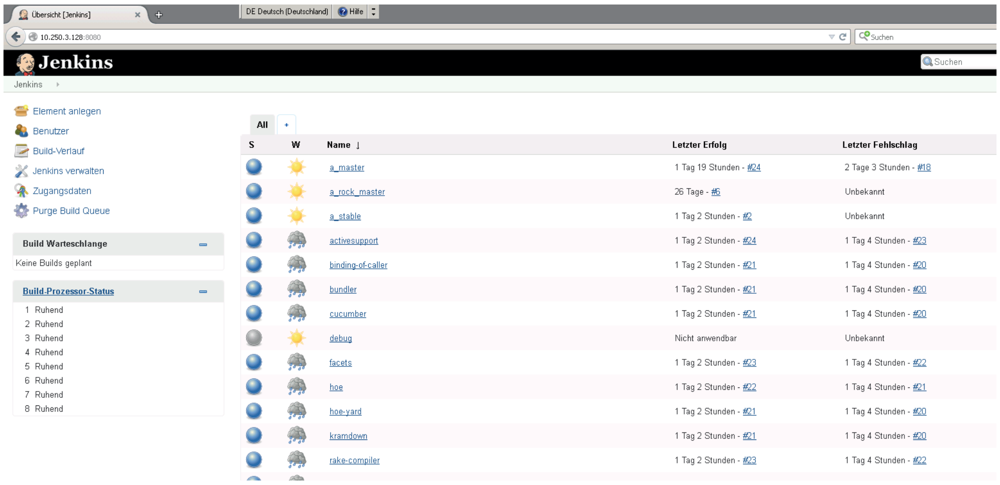
T Artemis2014 / Taskmanagement
O SpaceBot / orogen-proxy
B build_server / buildconf-all
P Artemis2014 / pre_grasp_pose_generator
P Artemis2014 / pgpg_sampler-orogen
P Artemis2014 / pre_grasp_pose_generator_library
L iStruct / lib_robotconfig
M MARS / mars_plugin_collection
I Artemis2014 / Interaktion
W Entern / Wiki
R Team Learning / robodyn_environment
B Team Learning / behavior_learning
B Entern / bundle
P Artemis2014 / pgpg_sampler

Universität Bremen

3

Testing phase

- Debian packages for Rock
  - Setup: Jenkins based build server



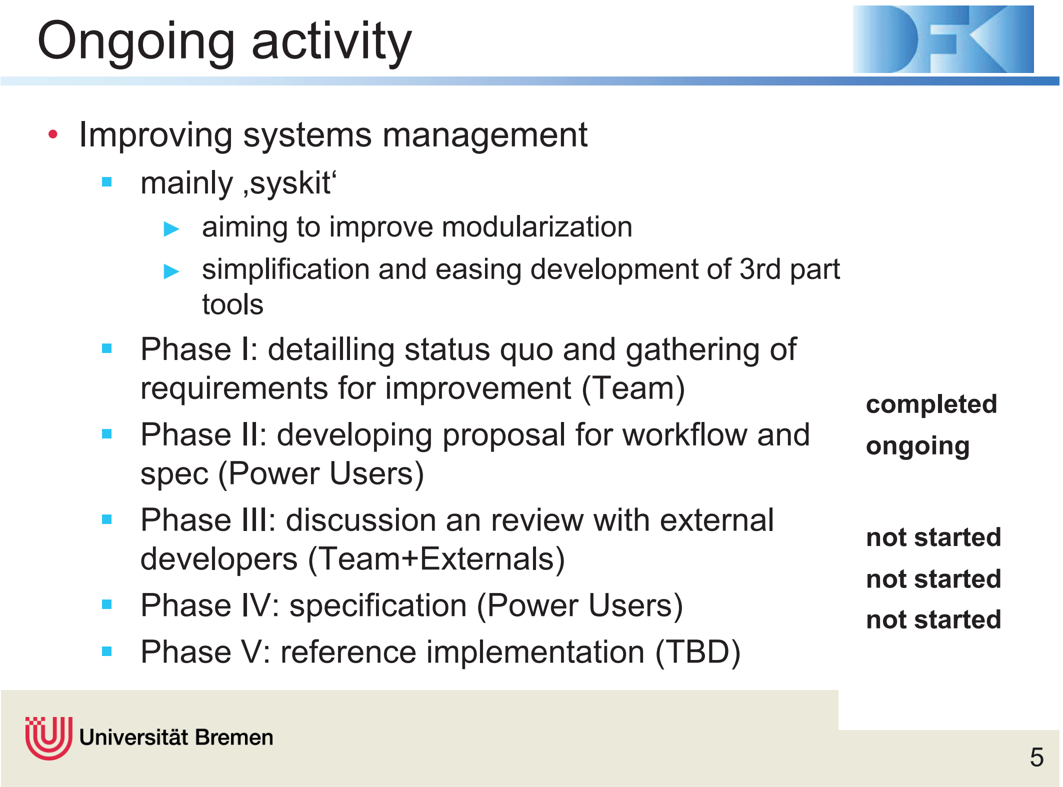
Build Warteschlange
Keine Builds geplant

Build-Prozessor-Status
1 Ruhend
2 Ruhend
3 Ruhend
4 Ruhend
5 Ruhend
6 Ruhend
7 Ruhend
8 Ruhend

S	W	Name ↓	Letzter Erfolg	Letzter Fehlschlag
●	☀	a_master	1 Tag 19 Stunden - #24	2 Tage 3 Stunden - #18
●	☀	a_rock_master	26 Tage - #6	Unbekannt
●	☀	a_stable	1 Tag 2 Stunden - #2	Unbekannt
●	☂	activesupport	1 Tag 2 Stunden - #24	1 Tag 4 Stunden - #23
●	☂	binding-of-caller	1 Tag 2 Stunden - #21	1 Tag 4 Stunden - #20
●	☂	bundler	1 Tag 2 Stunden - #21	1 Tag 4 Stunden - #20
●	☂	cucumber	1 Tag 2 Stunden - #21	1 Tag 4 Stunden - #20
○	☀	debug	Nicht anwendbar	Unbekannt
●	☂	facets	1 Tag 2 Stunden - #23	1 Tag 4 Stunden - #22
●	☂	hoe	1 Tag 2 Stunden - #22	1 Tag 4 Stunden - #21
●	☂	hoe-yard	1 Tag 2 Stunden - #21	1 Tag 4 Stunden - #20
●	☂	kramdown	1 Tag 2 Stunden - #21	1 Tag 4 Stunden - #20
●	☂	rake-compiler	1 Tag 2 Stunden - #23	1 Tag 4 Stunden - #22

Universität Bremen

4

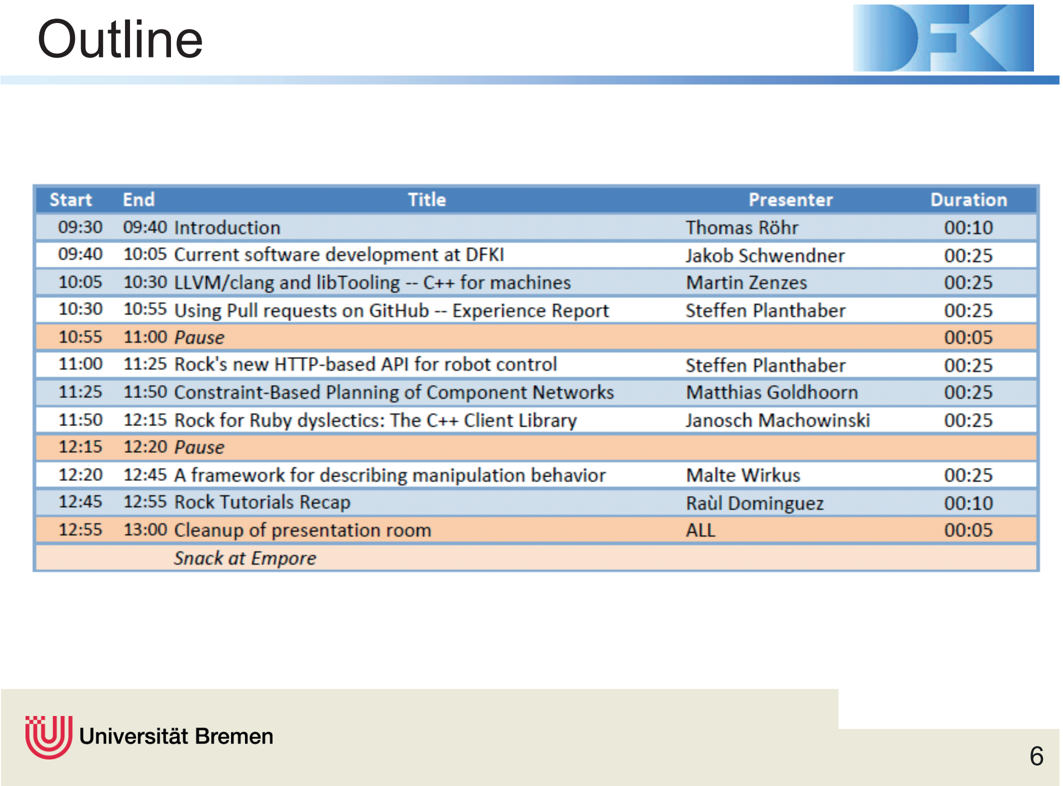


Ongoing activity

- Improving systems management
  - mainly ‚syskit'
    - aiming to improve modularization
    - simplification and easing development of 3rd part tools
  - Phase I: detailling status quo and gathering of requirements for improvement (Team) — **completed**
  - Phase II: developing proposal for workflow and spec (Power Users) — **ongoing**
  - Phase III: discussion an review with external developers (Team+Externals) — **not started**
  - Phase IV: specification (Power Users) — **not started**
  - Phase V: reference implementation (TBD) — **not started**

Universität Bremen

5



Outline

Start	End	Title	Presenter	Duration
09:30	09:40	Introduction	Thomas Röhr	00:10
09:40	10:05	Current software development at DFKI	Jakob Schwendner	00:25
10:05	10:30	LLVM/clang and libTooling -- C++ for machines	Martin Zenzes	00:25
10:30	10:55	Using Pull requests on GitHub -- Experience Report	Steffen Planthaber	00:25
10:55	11:00	*Pause*		00:05
11:00	11:25	Rock's new HTTP-based API for robot control	Steffen Planthaber	00:25
11:25	11:50	Constraint-Based Planning of Component Networks	Matthias Goldhoorn	00:25
11:50	12:15	Rock for Ruby dyslectics: The C++ Client Library	Janosch Machowinski	00:25
12:15	12:20	*Pause*		
12:20	12:45	A framework for describing manipulation behavior	Malte Wirkus	00:25
12:45	12:55	Rock Tutorials Recap	Raùl Dominguez	00:10
12:55	13:00	Cleanup of presentation room	ALL	00:05
		Snack at Empore		

Universität Bremen

6

2.2 ‘Current software development at DFKI’ (FW-T-02)

Jakob Schwendner[1]

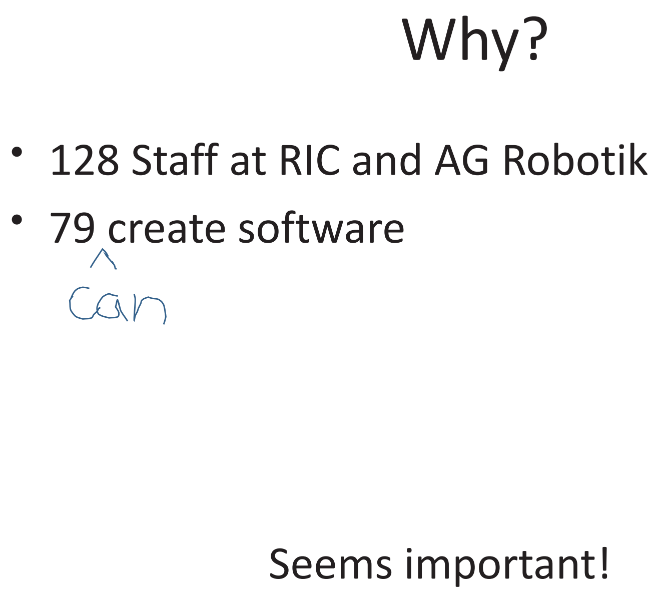
(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: jakob.schwendner@dfki.de

Abstract

The talk gives an overview of the current software development activities at the DFKI. A vision is formulated for the direction of the collaborative development efforts. The current tools in use are presented, and social aspects highlighted. Finally, a summary of future activities is given which will likely play an important role in development activities of the RIC.

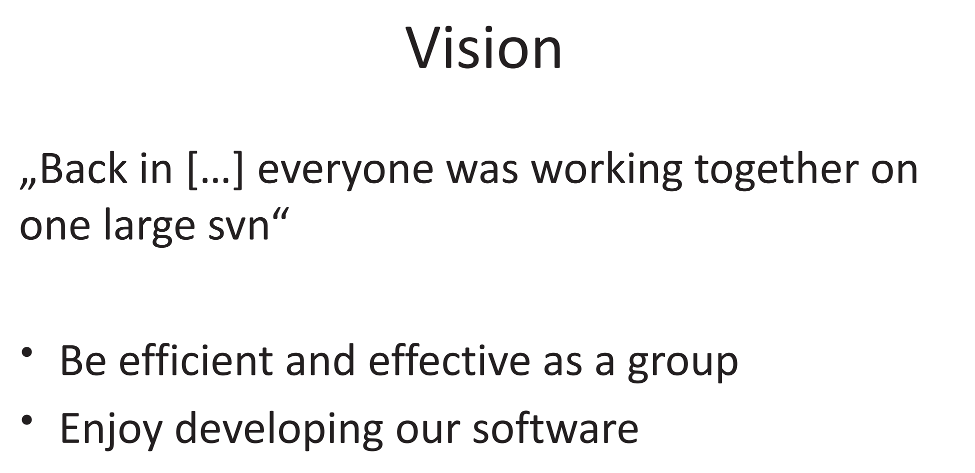
Software Development As I see it at the DFKI RIC



Why?

- 128 Staff at RIC and AG Robotik
- 79 can create software

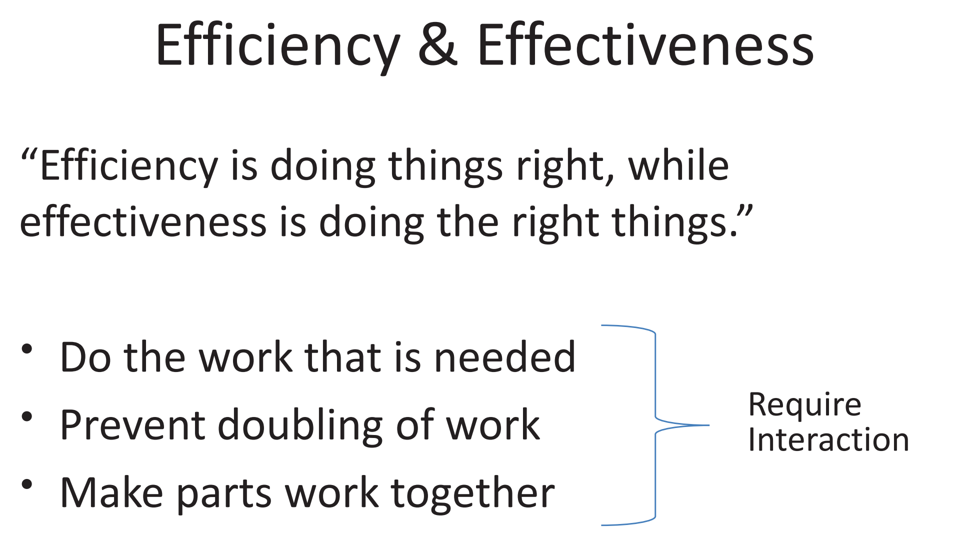
Seems important!



Vision

„Back in […] everyone was working together on one large svn“

- Be efficient and effective as a group
- Enjoy developing our software

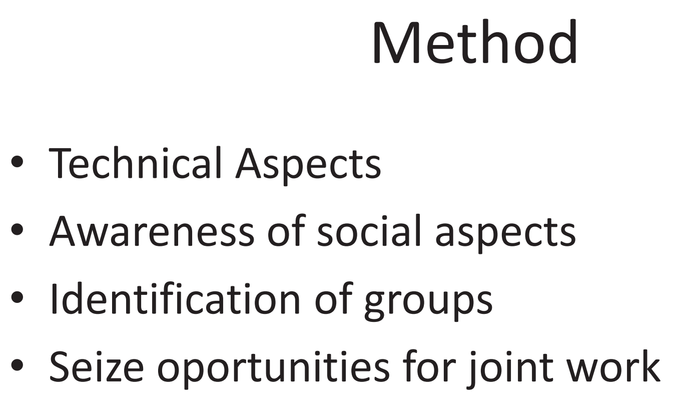


Efficiency & Effectiveness

“Efficiency is doing things right, while effectiveness is doing the right things.”

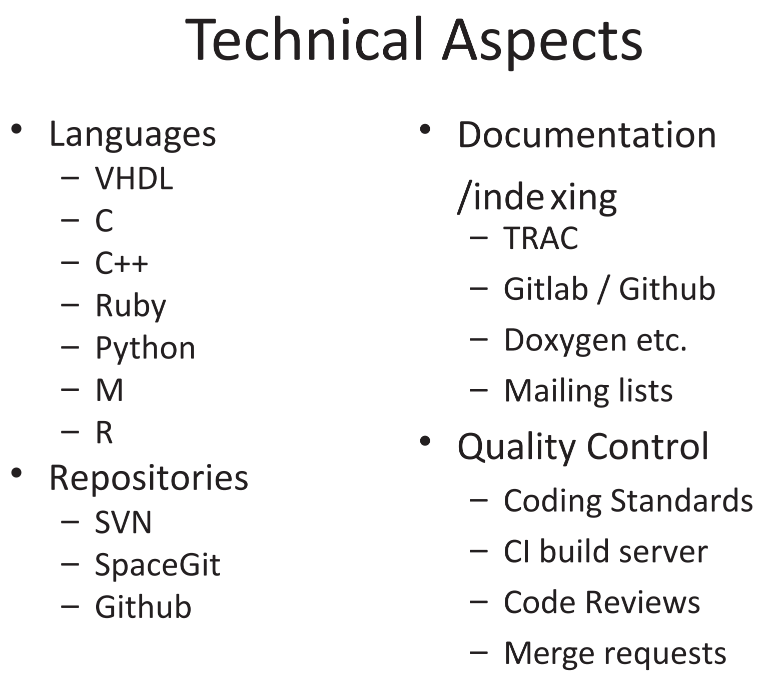
- Do the work that is needed
- Prevent doubling of work
- Make parts work together

Require Interaction



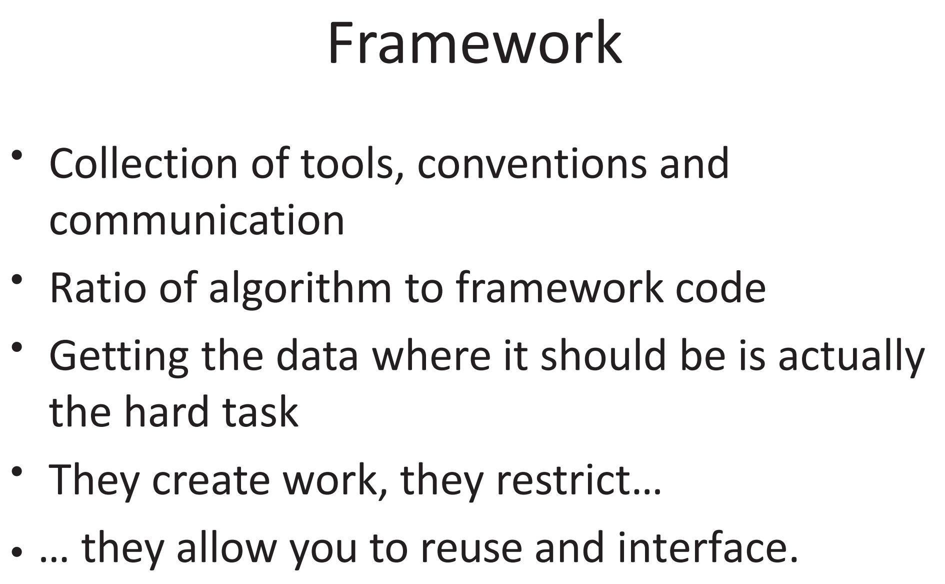
Method

- Technical Aspects
- Awareness of social aspects
- Identification of groups
- Seize oportunities for joint work



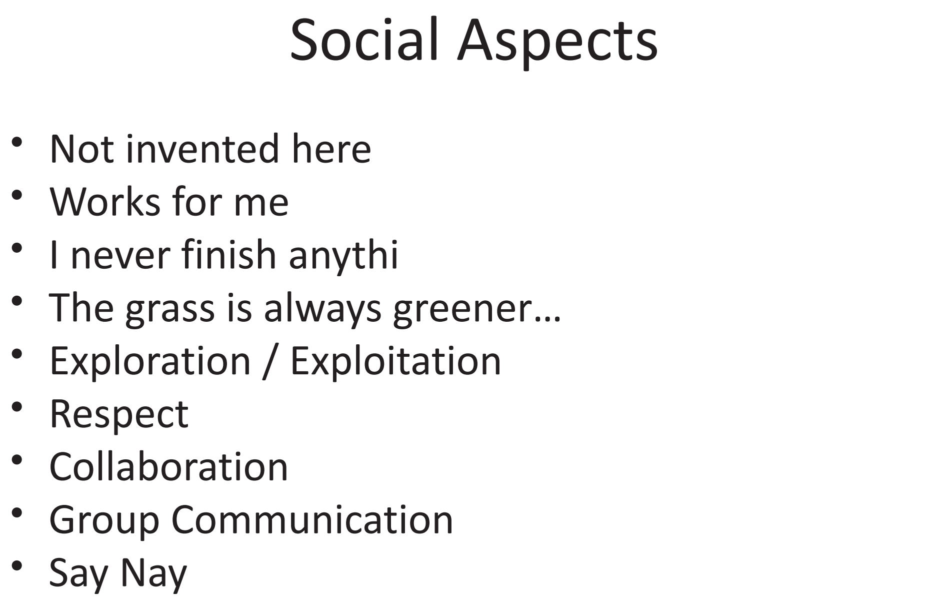
Technical Aspects

- Languages
  - VHDL
  - C
  - C++
  - Ruby
  - Python
  - M
  - R
- Repositories
  - SVN
  - SpaceGit
  - Github
- Documentation /inde xing
  - TRAC
  - Gitlab / Github
  - Doxygen etc.
  - Mailing lists
- Quality Control
  - Coding Standards
  - CI build server
  - Code Reviews
  - Merge requests



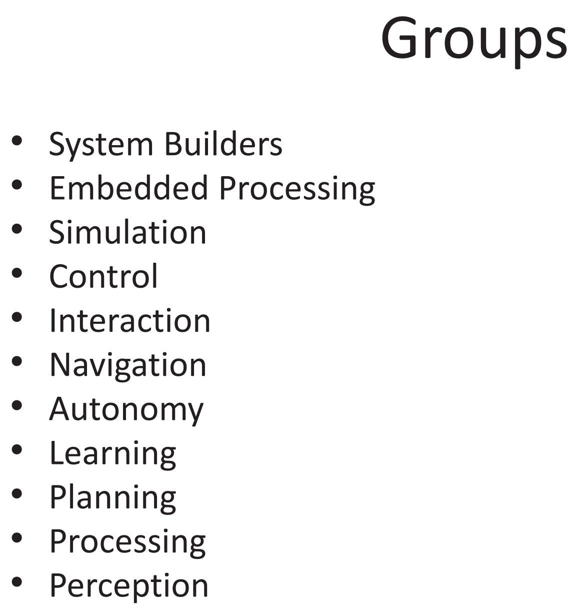
Framework

- Collection of tools, conventions and communication
- Ratio of algorithm to framework code
- Getting the data where it should be is actually the hard task
- They create work, they restrict…
- … they allow you to reuse and interface.



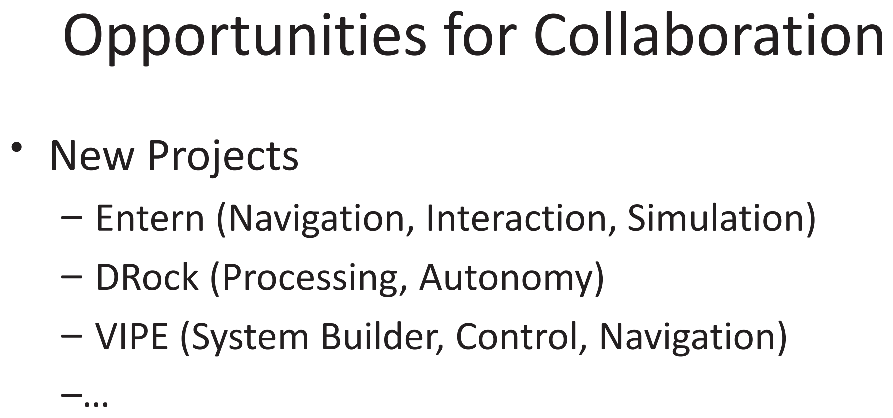
Social Aspects

- Not invented here
- Works for me
- I never finish anythi
- The grass is always greener…
- Exploration / Exploitation
- Respect
- Collaboration
- Group Communication
- Say Nay



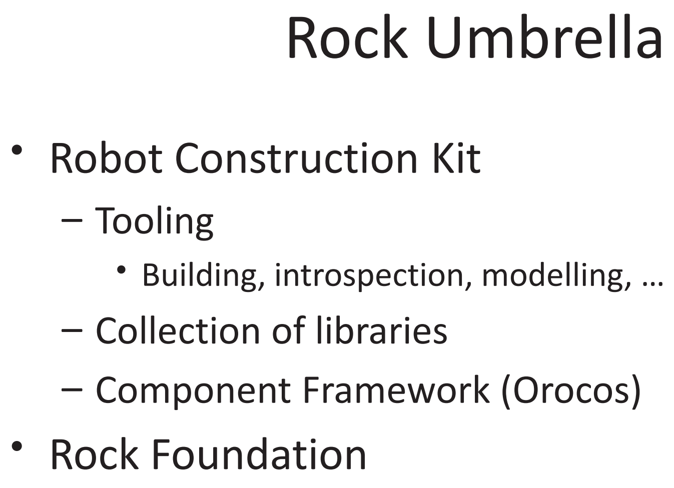
Groups

- System Builders
- Embedded Processing
- Simulation
- Control
- Interaction
- Navigation
- Autonomy
- Learning
- Planning
- Processing
- Perception



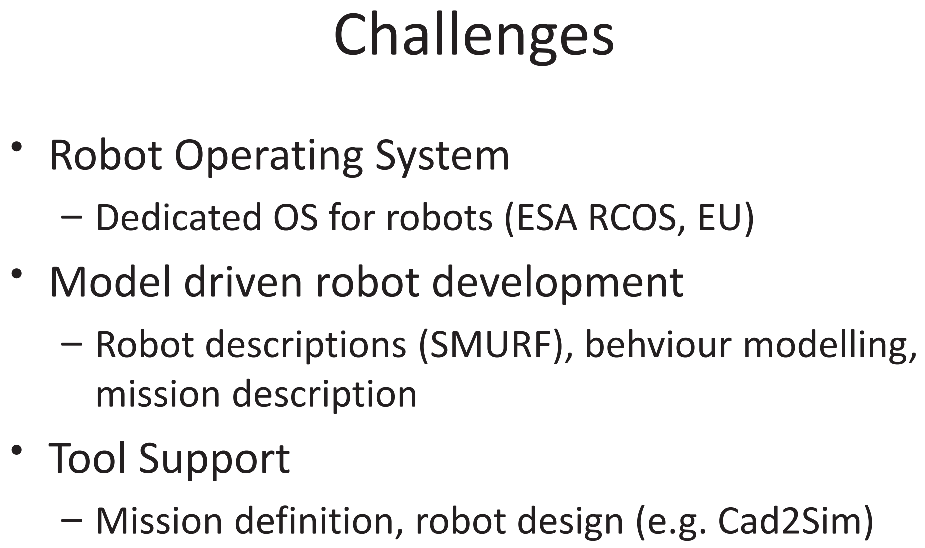
Opportunities for Collaboration

- New Projects
  - Entern (Navigation, Interaction, Simulation)
  - DRock (Processing, Autonomy)
  - VIPE (System Builder, Control, Navigation)
  - …



Rock Umbrella

- Robot Construction Kit
  - Tooling
    - Building, introspection, modelling, …
  - Collection of libraries
  - Component Framework (Orocos)
- Rock Foundation



Challenges

- Robot Operating System
  - Dedicated OS for robots (ESA RCOS, EU)
- Model driven robot development
  - Robot descriptions (SMURF), behviour modelling, mission description
- Tool Support
  - Mission definition, robot design (e.g. Cad2Sim)

2.3 ‘LLVM/clang and libTooling – C++ for machines’ (FW-T-03)

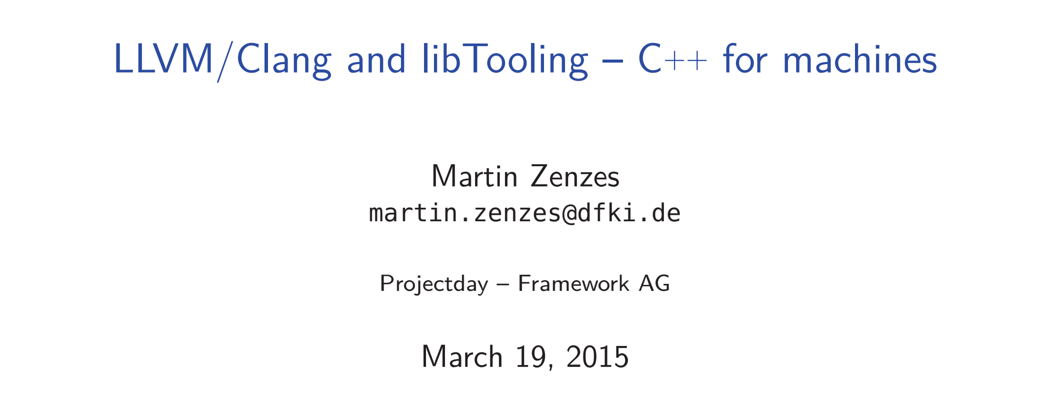
Martin Zenzes(1)

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: martin.zenzes@dfki.de

Abstract

LLVM/clang and the associated libTooling provide a flexible API for semantic processing of C/C++ source-code. After introduction of the LLVM/clang ecosystem, this talk will present an overview of existing tools available in Debian/Ubuntu as well as means to create new ones.



LLVM/Clang and libTooling – C++ for machines

Martin Zenzes
`martin.zenzes@dfki.de`

Projectday – Framework AG

March 19, 2015

Contents

Introduction

Existing Tools
- Source code analysis
- Automatic modifications

Abstract Syntax Tree

libTooling
- Compiling LLVM/Clang from source
- Refactoring Tool
- Compiler Plugin

2 / 46

Contents

Introduction

Existing Tools
- Source code analysis
- Automatic modifications

Abstract Syntax Tree

libTooling
- Compiling LLVM/Clang from source
- Refactoring Tool
- Compiler Plugin

Motivation

yet another talk on frameworks

- C/C++ is very complex, and it evolves
- Mortal humans need centuries to grasp it
- Wouldn’t it be nice to get help from machines?

Disclaimer:

- Sorry for too much Shell and C++ ; -)

Motivation

yet another talk on frameworks

- C/C++ is very complex, and it evolves
- Mortal humans need centuries to grasp it
- Wouldn't it be nice to get help from machines?

Disclaimer:

- Sorry for too much Shell and C++ ;-)

LLVM/Clang

introduction

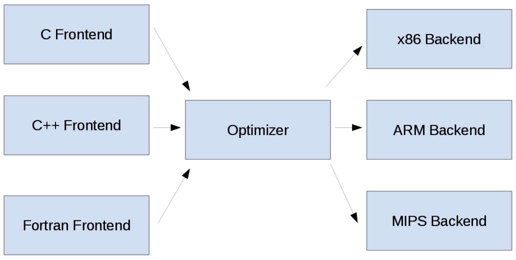
- Extensive framework for processing and compiling Code
- Modern, object-oriented C++ with public API
- Permissive license: *MIT/BSD*
- Roughly 6 month release cycle:

2007	First public release
2010	Self hosting
2012	Primary compiler in FreeBSD
2013	C++14 feature complete
February 27, 2015	LLVM/Clang 3.6

🔗 `RMS is pissed`

LLVM/Clang

ecosystem



- Frontend, Optimizer, Backend...
- Framework of complementary libraries: *libTooling*
- Dynamic plugin interface
- Used in many free and commercial tools
- Releases available in distributions:

Debian Wheezy	3.1
Debian Jessie	3.4 – 3.7 (co-installable!)
Ubuntu 12.04	3.0, 3.3, 3.4
Ubuntu 14.04	3.3 – 3.5
Windows	*3.4 – 3.7*

🔗 LLVM Weekly

Contents

Introduction

Existing Tools

Source code analysis

Automatic modifications

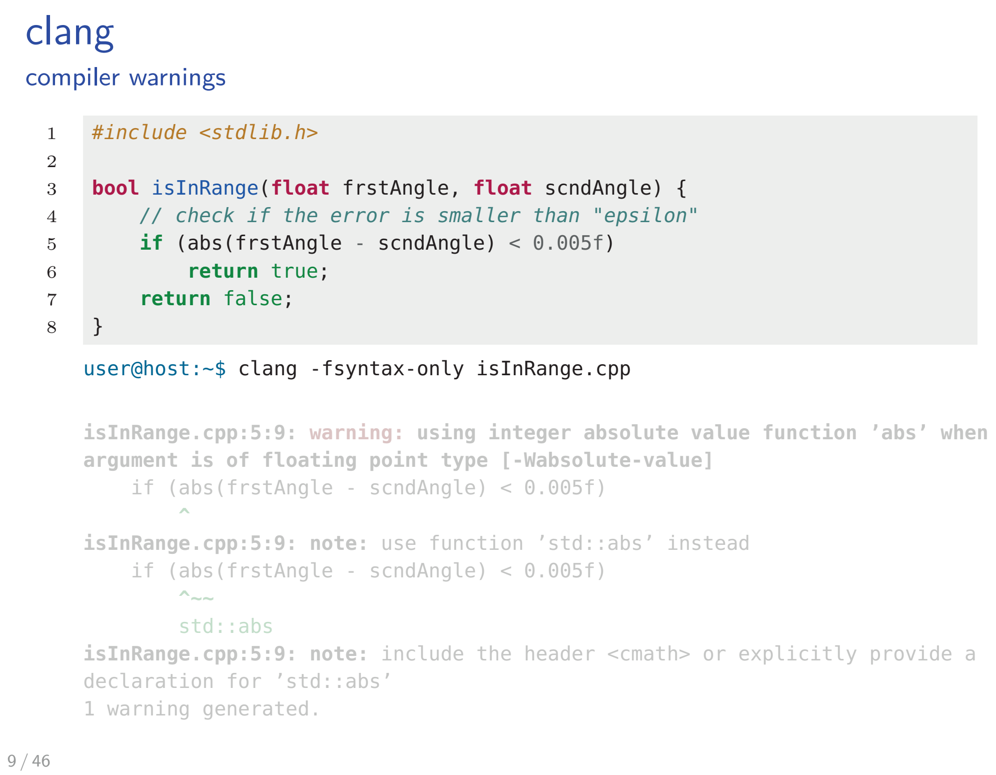
Abstract Syntax Tree

libTooling

Compiling LLVM/Clang from source

Refactoring Tool

Compiler Plugin

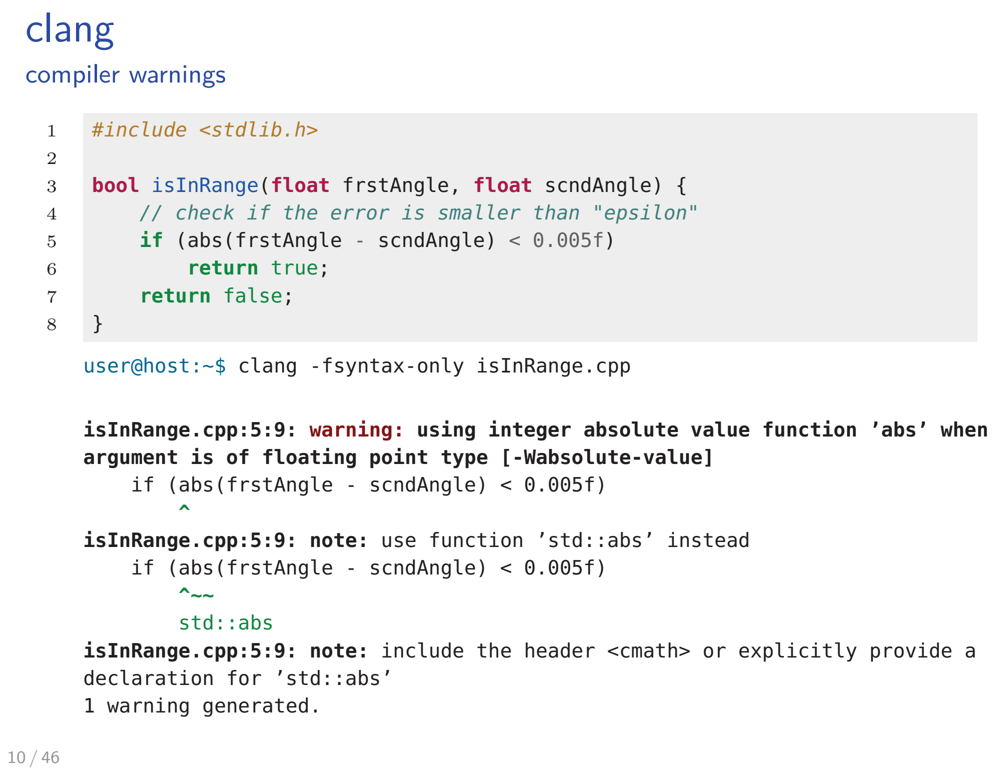


clang

compiler warnings

```
#include <stdlib.h>

bool isInRange(float frstAngle, float scndAngle) {
    // check if the error is smaller than "epsilon"
    if (abs(frstAngle - scndAngle) < 0.005f)
        return true;
    return false;
}
```

```
user@host:~$ clang -fsyntax-only isInRange.cpp

isInRange.cpp:5:9: warning: using integer absolute value function 'abs' when
argument is of floating point type [-Wabsolute-value]
    if (abs(frstAngle - scndAngle) < 0.005f)
        ^
isInRange.cpp:5:9: note: use function 'std::abs' instead
    if (abs(frstAngle - scndAngle) < 0.005f)
        ^~~
        std::abs
isInRange.cpp:5:9: note: include the header <cmath> or explicitly provide a
declaration for 'std::abs'
1 warning generated.
```

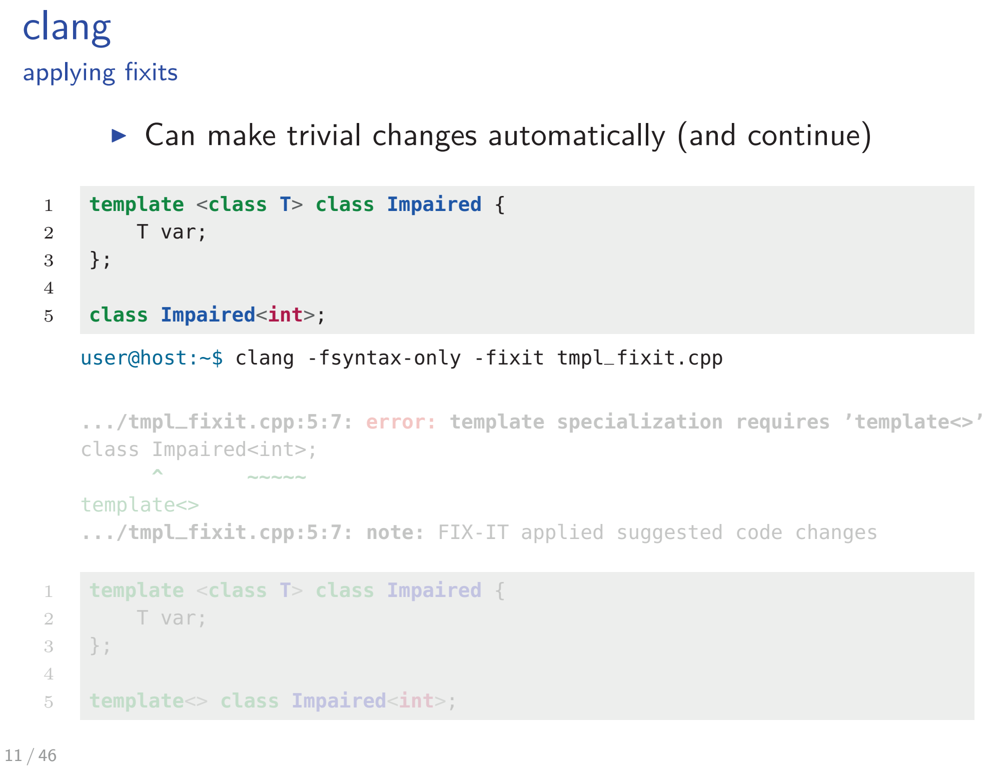


clang

compiler warnings

```
#include <stdlib.h>

bool isInRange(float frstAngle, float scndAngle) {
    // check if the error is smaller than "epsilon"
    if (abs(frstAngle - scndAngle) < 0.005f)
        return true;
    return false;
}
```

```
user@host:~$ clang -fsyntax-only isInRange.cpp

isInRange.cpp:5:9: warning: using integer absolute value function 'abs' when
argument is of floating point type [-Wabsolute-value]
    if (abs(frstAngle - scndAngle) < 0.005f)
        ^
isInRange.cpp:5:9: note: use function 'std::abs' instead
    if (abs(frstAngle - scndAngle) < 0.005f)
        ^~~
        std::abs
isInRange.cpp:5:9: note: include the header <cmath> or explicitly provide a
declaration for 'std::abs'
1 warning generated.
```



clang

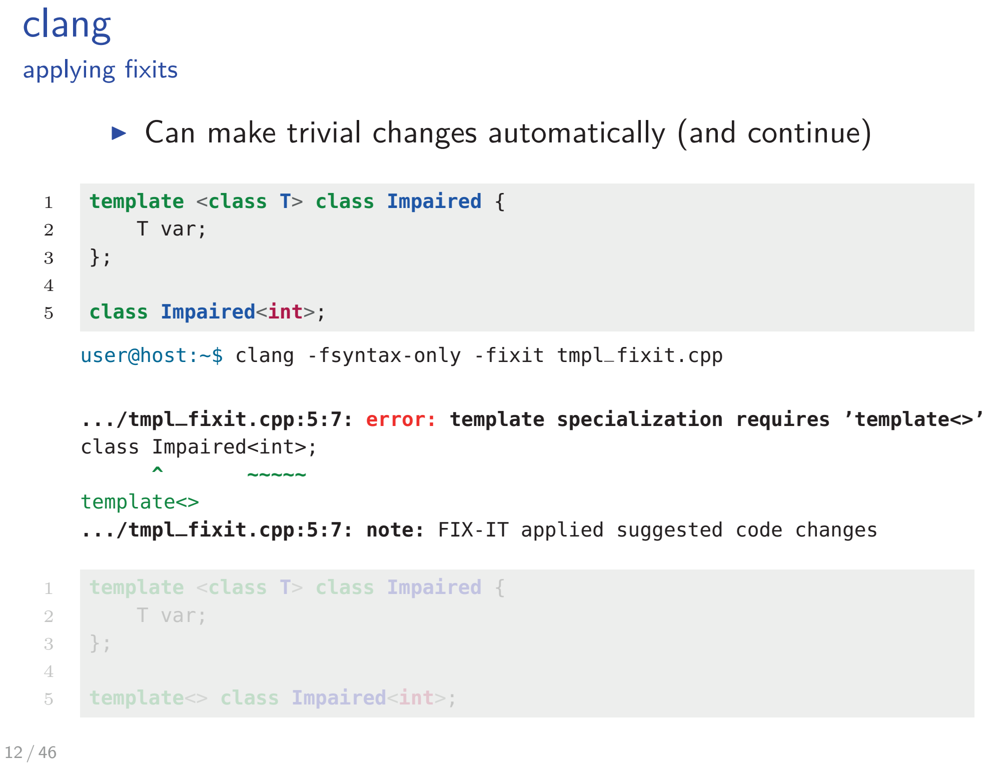
applying fixits

- Can make trivial changes automatically (and continue)

```cpp
template <class T> class Impaired {
    T var;
};

class Impaired<int>;
```

```
user@host:~$ clang -fsyntax-only -fixit tmpl_fixit.cpp

.../tmpl_fixit.cpp:5:7: error: template specialization requires 'template<>'
class Impaired<int>;
      ^        ~~~~~
template<>
.../tmpl_fixit.cpp:5:7: note: FIX-IT applied suggested code changes
```

```cpp
template <class T> class Impaired {
    T var;
};

template<> class Impaired<int>;
```



clang

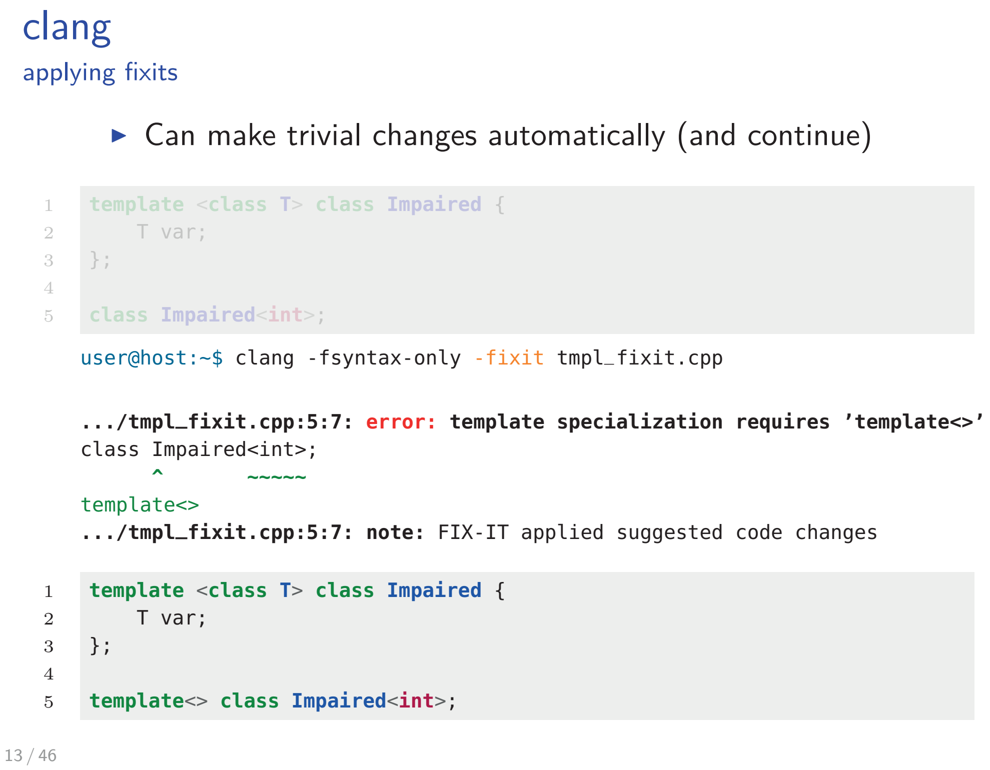
applying fixits

- Can make trivial changes automatically (and continue)

```cpp
template <class T> class Impaired {
    T var;
};

class Impaired<int>;
```

```
user@host:~$ clang -fsyntax-only -fixit tmpl_fixit.cpp

.../tmpl_fixit.cpp:5:7: error: template specialization requires 'template<>'
class Impaired<int>;
      ^        ~~~~~
template<>
.../tmpl_fixit.cpp:5:7: note: FIX-IT applied suggested code changes
```

```cpp
template <class T> class Impaired {
    T var;
};

template<> class Impaired<int>;
```



clang

applying fixits

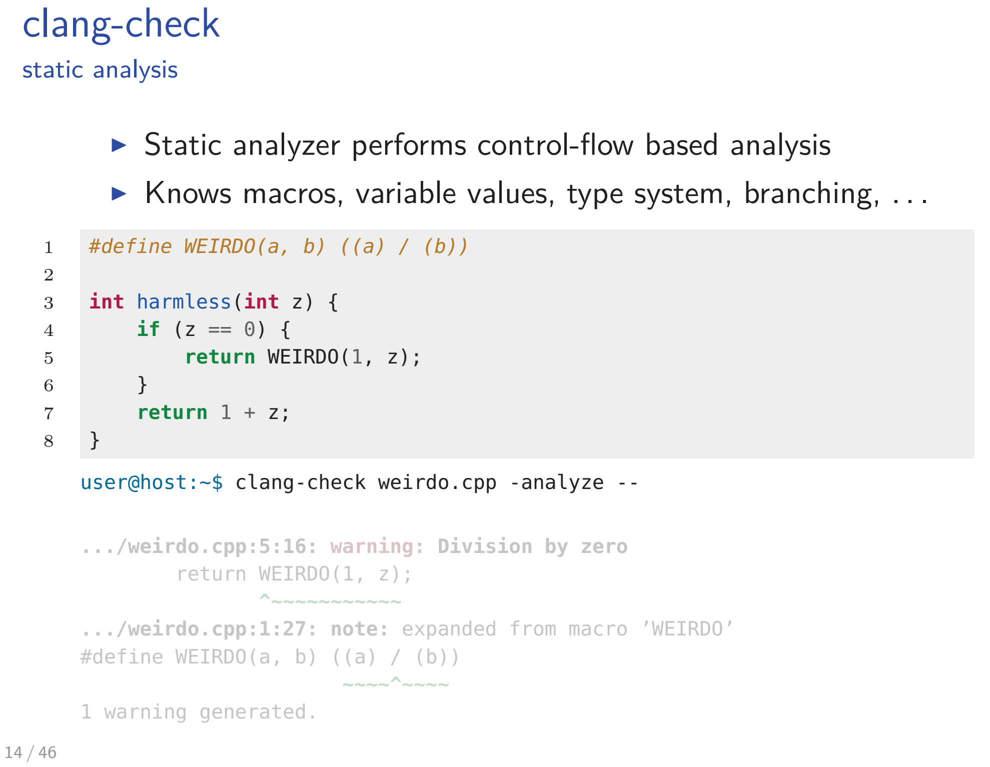
- Can make trivial changes automatically (and continue)

```
template <class T> class Impaired {
    T var;
};

class Impaired<int>;
```

```
user@host:~$ clang -fsyntax-only -fixit tmpl_fixit.cpp

.../tmpl_fixit.cpp:5:7: error: template specialization requires 'template<>'
class Impaired<int>;
      ^       ~~~~~
template<>
.../tmpl_fixit.cpp:5:7: note: FIX-IT applied suggested code changes
```

```
template <class T> class Impaired {
    T var;
};

template<> class Impaired<int>;
```

13 / 46



clang-check

static analysis

- Static analyzer performs control-flow based analysis
- Knows macros, variable values, type system, branching, . . .

```
#define WEIRDO(a, b) ((a) / (b))

int harmless(int z) {
    if (z == 0) {
        return WEIRDO(1, z);
    }
    return 1 + z;
}
```

```
user@host:~$ clang-check weirdo.cpp -analyze --

.../weirdo.cpp:5:16: warning: Division by zero
        return WEIRDO(1, z);
               ^~~~~~~~~~~~
.../weirdo.cpp:1:27: note: expanded from macro 'WEIRDO'
#define WEIRDO(a, b) ((a) / (b))
                      ~~~~^~~~~
1 warning generated.
```

14 / 46

clang-check

static analysis

- Static analyzer performs control-flow based analysis
- Knows macros, variable values, type system, branching, ...

```cpp
#define WEIRDO(a, b) ((a) / (b))

int harmless(int z) {
    if (z == 0) {
        return WEIRDO(1, z);
    }
    return 1 + z;
}
```

```
user@host:~$ clang-check weirdo.cpp -analyze --

.../weirdo.cpp:5:16: warning: Division by zero
        return WEIRDO(1, z);
               ^~~~~~~~~~~
.../weirdo.cpp:1:27: note: expanded from macro 'WEIRDO'
#define WEIRDO(a, b) ((a) / (b))
                      ~~~~^~~~~
1 warning generated.
```

15 / 46

clang-modernize

- Migrating source code to C++11
- `UseNullptr`, `PassByValue`, `ReplaceAutoPtr`, ...
- Risk-levels: *safe*, *reasonable*, *risky*

```cpp
#include <vector>
#include <iostream>

void nullptr_assignment() {
    char *a = NULL;
    char *b = 0;
    char c = 0;
}
void loop_convert(std::vector<int> v) {
    for (std::vector<int>::iterator it = v.begin(); it != v.end(); ++it)
        std::cout << *it;
}
```

16 / 46

clang-modernize

- Migrating source code to C++11
- `UseNullptr`, `PassByValue`, `ReplaceAutoPtr`, ...
- Risk-levels: *safe*, *reasonable*, *risky*

```
#include <vector>
#include <iostream>

void nullptr_assignment() {
    char *a = nullptr;
    char *b = nullptr;
    char c = 0;
}
void loop_convert(std::vector<int> v) {
    for (auto & elem : v)
        std::cout << elem;
}
```

clang-format

whitespace fixes

- Automatic formatting based on semantic analysis
- Different coding styles, configurable `.clang-format`

```
user@host:~$ cat kaputt.cpp
```

```
#include <sstream>
#define PURE_HORROR(scare) \
        std::stringstream ss; \
        ss << "omg, this '"   << scare<<"' is horrible. run!"; \
        throw(ss.str())

class TC { double interestingName(const   int *rumba, int * bumba)
    {    if (bumba)
 return *rumba**bumba-1.f;
else PURE_HORROR( * rumba);}};
```

clang-format

whitespace fixes

- Automatic formatting based on semantic analysis
- Different coding styles, configurable `.clang-format`

```
user@host:~$ clang-format kaputt.cpp -style=Google
```

```
#include <sstream>
#define PURE_HORROR(scare)                             \
  std::stringstream ss;                                \
  ss << "omg, this ’" << scare << "’ is horrible. run!"; \
  throw(ss.str())

class TC {
  double interestingName(const int *rumba, int *bumba) {
    if (bumba)
      return *rumba * *bumba - 1.f;
    else
      PURE_HORROR(*rumba);
  }
};
```

19 / 46

clang-format

whitespace fixes

- Automatic formatting based on semantic analysis
- Different coding styles, configurable `.clang-format`

```
user@host:~$ clang-format kaputt.cpp -style='{BasedOnStyle:
      Google,SpacesInParentheses:  true,IndentWidth:  3}'
```

```
#include <sstream>
#define PURE_HORROR( scare )                            \
   std::stringstream ss;                                \
   ss << "omg, this ’" << scare << "’ is horrible. run!"; \
   throw( ss.str() )

class TC {
   double interestingName( const int *rumba, int *bumba ) {
      if ( bumba )
         return *rumba * *bumba - 1.f;
      else
         PURE_HORROR( *rumba );
   }
};
```

20 / 46

Contents

Introduction

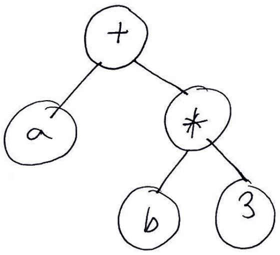
Existing Tools
- Source code analysis
- Automatic modifications

Abstract Syntax Tree

libTooling
- Compiling LLVM/Clang from source
- Refactoring Tool
- Compiler Plugin

Abstract Syntax Tree

not GIMPLE



- Clangs intermediary data structure, e.g. for optimizations
- Closely resembles written C++ source
- Central part of libTooling
- Traverseable via API: `getTranslationUnitDecl()`
- AST Nodes: `DeclStmt`, `BinaryOperator`, `ParenExpr`, `IntegerLiteral`, `ImplicitCastExpr`, ...

🔗 `Introduction to Clang AST`

clang-check

inspecting the AST

- Dumping coloured AST (pipe into `less -R`)
- Helps debugging clang-related tooling

```
/** with doxygen comment attached */
int doubleItUp(int myArg) {
    return 2 * myArg;
}
```

```
user@host:~$ clang-check simple-func.cpp -ast-dump --

TranslationUnitDecl 0x2d889f0 <<invalid sloc>> <invalid sloc>
|-TypedefDecl 0x2d88f30 <<invalid sloc>> <invalid sloc> implicit __int128_t ’__int128’
|-TypedefDecl 0x2d88f90 <<invalid sloc>> <invalid sloc> implicit __uint128_t ’unsigned __int128’
|-TypedefDecl 0x2d89390 <<invalid sloc>> <invalid sloc> implicit __builtin_va_list ’__va_list_tag [1]’
‘-FunctionDecl 0x2d894e0 <.../simple-func.cpp:2:1, line:4:1> line:2:5 doubleItUp ’int (int)’
  |-ParmVarDecl 0x2d89420 <col:16, col:20> col:20 used myArg ’int’
  |-CompoundStmt 0x2d89630 <col:27, line:4:1>
  | ‘-ReturnStmt 0x2d89610 <line:3:5, col:16>
  |   ‘-BinaryOperator 0x2d895e8 <col:12, col:16> ’int’ ’*’
  |     |-IntegerLiteral 0x2d89588 <col:12> ’int’ 2
  |     ‘-ImplicitCastExpr 0x2d895d0 <col:16> ’int’ <LValueToRValue>
  |       ‘-DeclRefExpr 0x2d895a8 <col:16> ’int’ lvalue ParmVar 0x2d89420 ’myArg’ ’int’
  ‘-FullComment 0x2d896f0 <line:1:4, col:34>
    ‘-ParagraphComment 0x2d896c0 <col:4, col:34>
      ‘-TextComment 0x2d89690 <col:4, col:34> Text=" with doxygen comment attached "
```

23 / 46

clang-check

inspecting the AST

- Dumping coloured AST (pipe into `less -R`)
- Helps debugging clang-related tooling

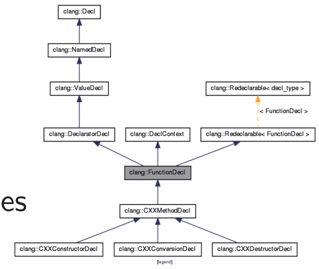
```
/** with doxygen comment attached */
int doubleItUp(int myArg) {
    return 2 * myArg;
}
```

```
user@host:~$ clang-check simple-func.cpp -ast-dump --

TranslationUnitDecl 0x2d889f0 <<invalid sloc>> <invalid sloc>
|-TypedefDecl 0x2d88f30 <<invalid sloc>> <invalid sloc> implicit __int128_t ’__int128’
|-TypedefDecl 0x2d88f90 <<invalid sloc>> <invalid sloc> implicit __uint128_t ’unsigned __int128’
|-TypedefDecl 0x2d89390 <<invalid sloc>> <invalid sloc> implicit __builtin_va_list ’__va_list_tag [1]’
‘-FunctionDecl 0x2d894e0 <.../simple-func.cpp:2:1, line:4:1> line:2:5 doubleItUp ’int (int)’
  |-ParmVarDecl 0x2d89420 <col:16, col:20> col:20 used myArg ’int’
  |-CompoundStmt 0x2d89630 <col:27, line:4:1>
  | ‘-ReturnStmt 0x2d89610 <line:3:5, col:16>
  |   ‘-BinaryOperator 0x2d895e8 <col:12, col:16> ’int’ ’*’
  |     |-IntegerLiteral 0x2d89588 <col:12> ’int’ 2
  |     ‘-ImplicitCastExpr 0x2d895d0 <col:16> ’int’ <LValueToRValue>
  |       ‘-DeclRefExpr 0x2d895a8 <col:16> ’int’ lvalue ParmVar 0x2d89420 ’myArg’ ’int’
  ‘-FullComment 0x2d896f0 <line:1:4, col:34>
    ‘-ParagraphComment 0x2d896c0 <col:4, col:34>
      ‘-TextComment 0x2d89690 <col:4, col:34> Text=" with doxygen comment attached "
```

24 / 46

AST Matcher

magic template-based magic



- Hierarchy modelled via inheritance of Classes → usage of C++ type system
- Template based API to locate specific Nodes
- Very powerful; very complex

```
// matches 'int b[7]' for example
arrayType(hasElementType(builtinType()))

// match records named 'Foo' that are derived from 'Bar'
recordDecl(hasName("Foo"), isDerivedFrom("Bar")).

// matches 'int x = 0' in
//     for (int x = 0; x < N; ++x) { }
forStmt(hasLoopInit(declStmt()))
```

🔗 AST Matcher Reference

clang-query

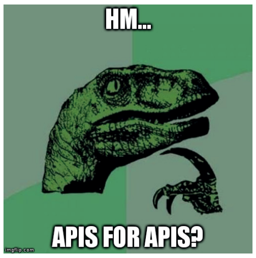
- Interactive interpreter for AST Matcher expressions
- Just discovered while preparing this talk...

```
int foo(int* p, int v) {
  if (*p == 0) {
    return v + 1;
  } else {
    return v - 1;
  }
}
```

```
clang-query> match varDecl().bind("var")

Match #1:

ifexx.c:1:9: note: "var" binds here
int foo(int* p, int v) {
        ^~~~~~

Match #2:

ifexx.c:1:17: note: "var" binds here
int foo(int* p, int v) {
                ^~~~~
2 matches.
```

libTooling

what can we do with this?



LLVM/Clang and libTooling for programmatic access to C++!

- Policy: reformatting, generating documentation, verification
- IDE: code completion, moving around, pulling in definitions
- Refactoring: Renaming variables, evolving APIs
- Dynamic tools: JIT, interpreter, generating code, language-to-language translation
- Less boilerplate: Writing one-off tools to scratch *your* itch

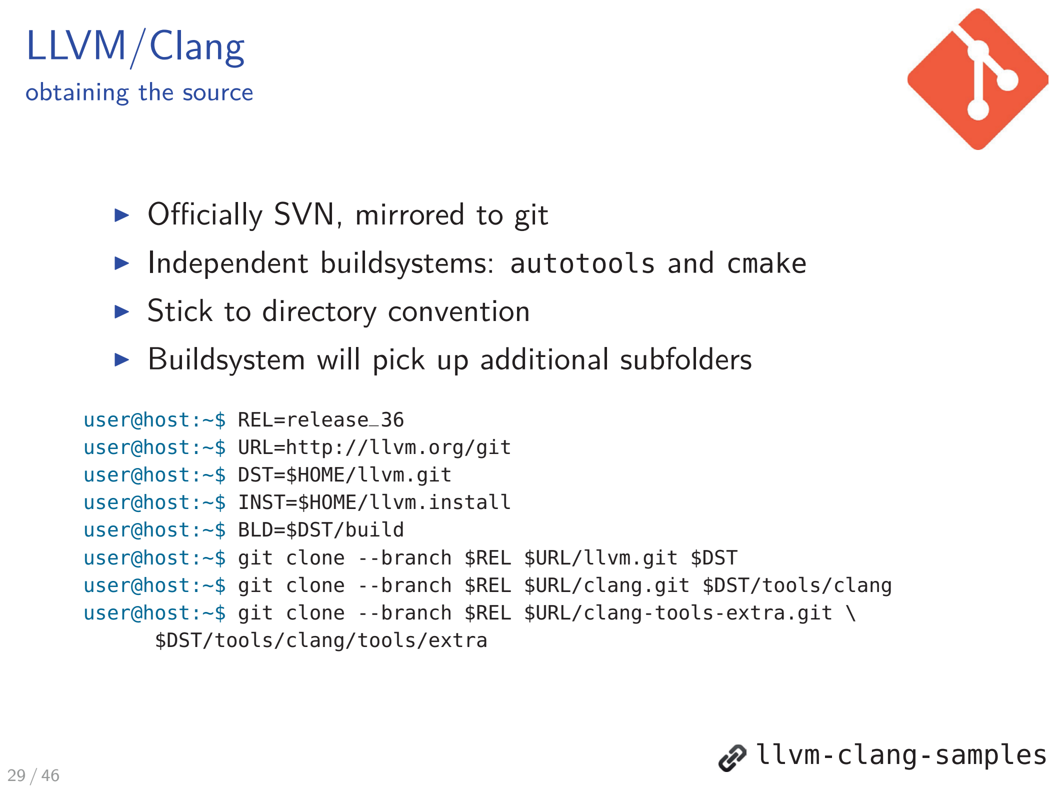
Contents

Introduction

Existing Tools
- Source code analysis
- Automatic modifications

Abstract Syntax Tree

libTooling
- Compiling LLVM/Clang from source
- Refactoring Tool
- Compiler Plugin



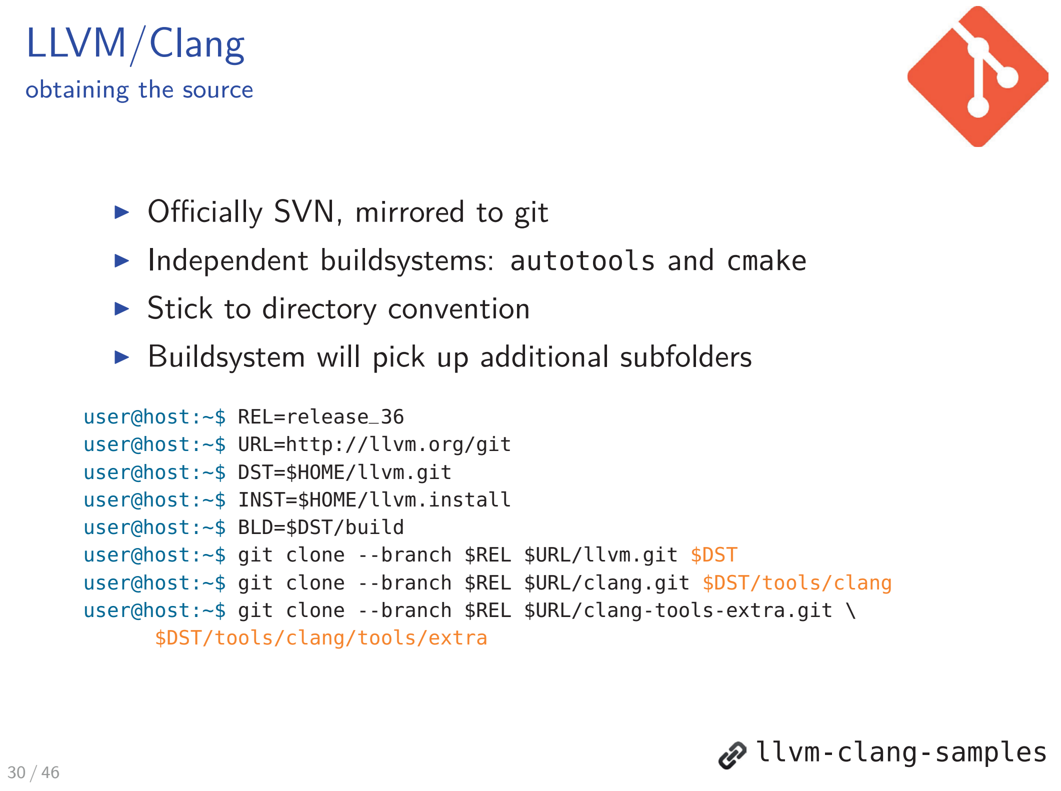
LLVM/Clang

obtaining the source

- Officially SVN, mirrored to git
- Independent buildsystems: `autotools` and `cmake`
- Stick to directory convention
- Buildsystem will pick up additional subfolders

```
user@host:~$ REL=release_36
user@host:~$ URL=http://llvm.org/git
user@host:~$ DST=$HOME/llvm.git
user@host:~$ INST=$HOME/llvm.install
user@host:~$ BLD=$DST/build
user@host:~$ git clone --branch $REL $URL/llvm.git $DST
user@host:~$ git clone --branch $REL $URL/clang.git $DST/tools/clang
user@host:~$ git clone --branch $REL $URL/clang-tools-extra.git \
      $DST/tools/clang/tools/extra
```

🔗 llvm-clang-samples



LLVM/Clang

obtaining the source

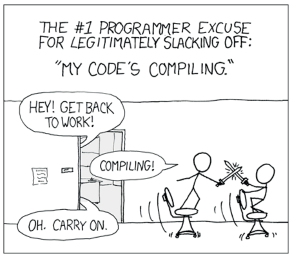
- Officially SVN, mirrored to git
- Independent buildsystems: `autotools` and `cmake`
- Stick to directory convention
- Buildsystem will pick up additional subfolders

```
user@host:~$ REL=release_36
user@host:~$ URL=http://llvm.org/git
user@host:~$ DST=$HOME/llvm.git
user@host:~$ INST=$HOME/llvm.install
user@host:~$ BLD=$DST/build
user@host:~$ git clone --branch $REL $URL/llvm.git $DST
user@host:~$ git clone --branch $REL $URL/clang.git $DST/tools/clang
user@host:~$ git clone --branch $REL $URL/clang-tools-extra.git \
      $DST/tools/clang/tools/extra
```

🔗 llvm-clang-samples

LLVM/Clang

testing the result



- Configure optimized build, install to $HOME

```
user@host:~$ mkdir -p $BLD && cd $BLD
user@host:~/llvm.git/build$ cmake ..  -DCMAKE_INSTALL_PREFIX=$INST \
      -DCMAKE_BUILD_TYPE=Release -DLLVM_TARGETS_TO_BUILD="X86"
user@host:~/llvm.git/build$ make all install
```

- Check that it actually works

```
user@host:~$ export PATH=$INST/bin:$PATH
user@host:~$ which clang
.../llvm.install/bin/clang
user@host:~$ clang --version
clang version 3.6.1 (http://llvm.org/git/clang.git 1d3945d684dbc51b...)
Target:  x86_64-unknown-linux-gnu
Thread model:  posix
```

LLVM/Clang

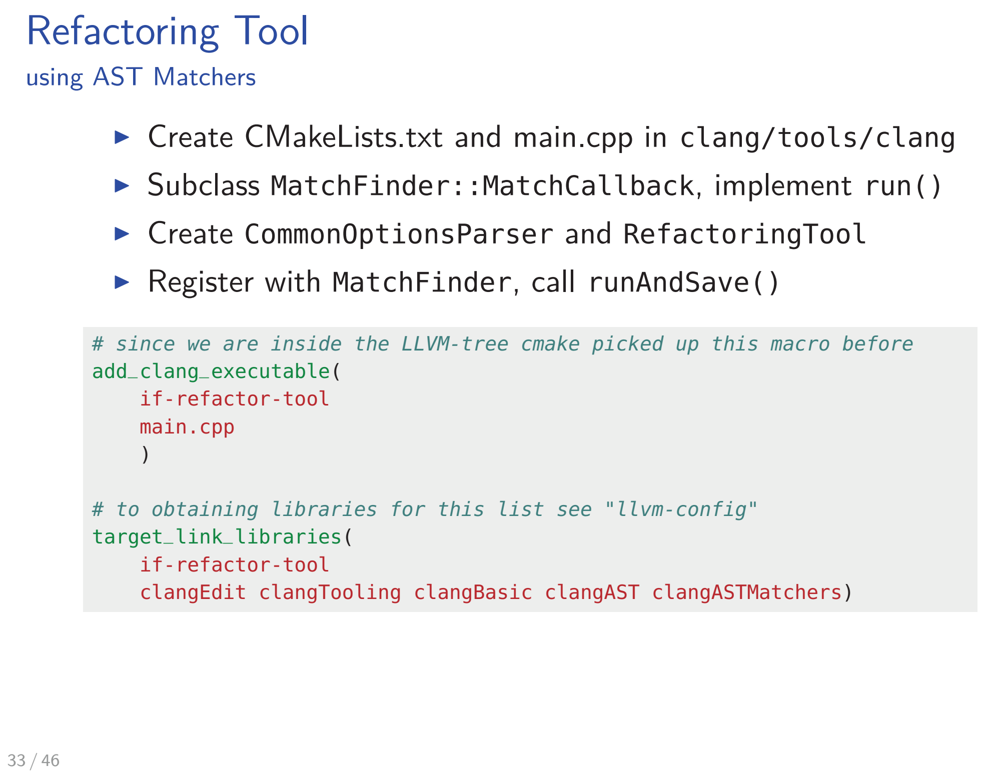
testing the result

- Configure optimized build, install to $HOME

```
user@host:~$ mkdir -p $BLD && cd $BLD
user@host:~/llvm.git/build$ cmake ..  -DCMAKE_INSTALL_PREFIX=$INST \
      -DCMAKE_BUILD_TYPE=Release -DLLVM_TARGETS_TO_BUILD="X86"
user@host:~/llvm.git/build$ make all install
```

- Check that it actually works

```
user@host:~$ export PATH=$INST/bin:$PATH
user@host:~$ which clang
.../llvm.install/bin/clang
user@host:~$ clang --version
clang version 3.6.1 (http://llvm.org/git/clang.git 1d3945d684dbc51b...)
Target:  x86_64-unknown-linux-gnu
Thread model:  posix
```



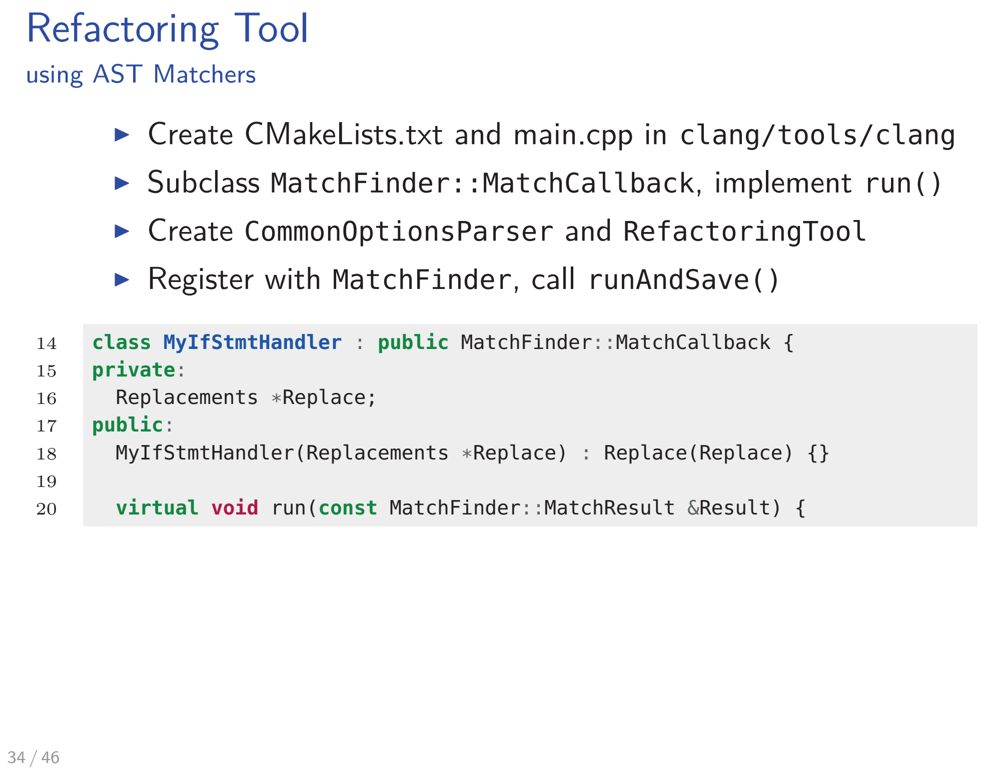
Refactoring Tool

using AST Matchers

- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```
# since we are inside the LLVM-tree cmake picked up this macro before
add_clang_executable(
    if-refactor-tool
    main.cpp
    )

# to obtaining libraries for this list see "llvm-config"
target_link_libraries(
    if-refactor-tool
    clangEdit clangTooling clangBasic clangAST clangASTMatchers)
```

33 / 46



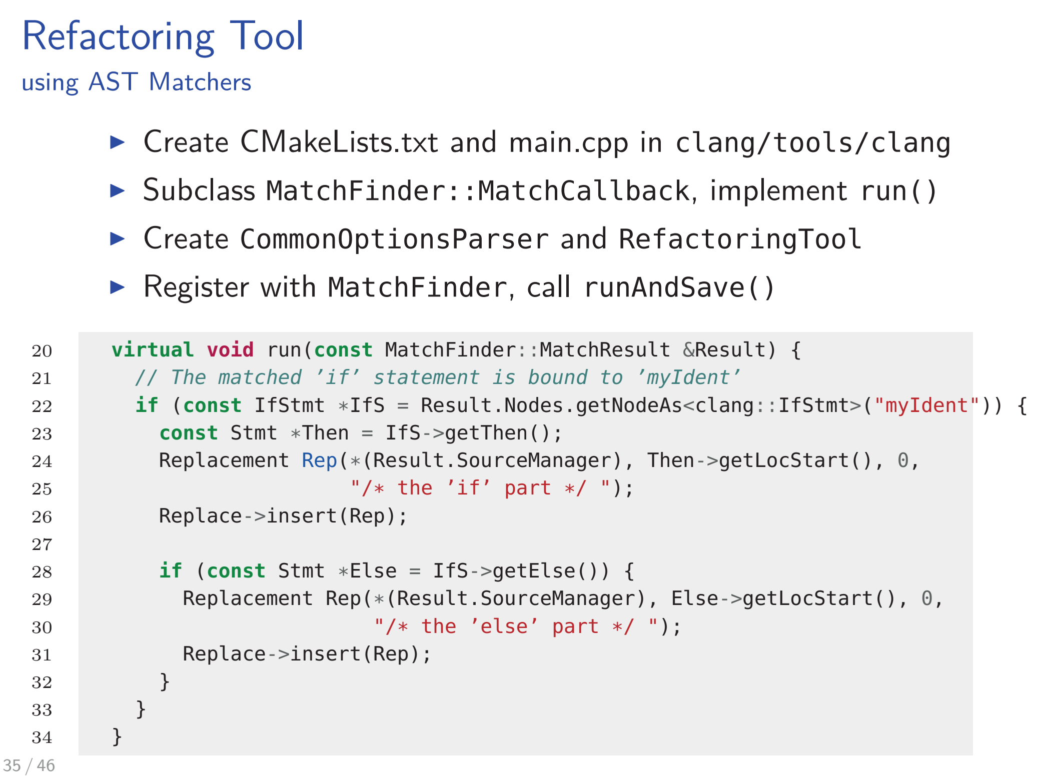
Refactoring Tool

using AST Matchers

- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```
class MyIfStmtHandler : public MatchFinder::MatchCallback {
private:
  Replacements *Replace;
public:
  MyIfStmtHandler(Replacements *Replace) : Replace(Replace) {}

  virtual void run(const MatchFinder::MatchResult &Result) {
```

34 / 46



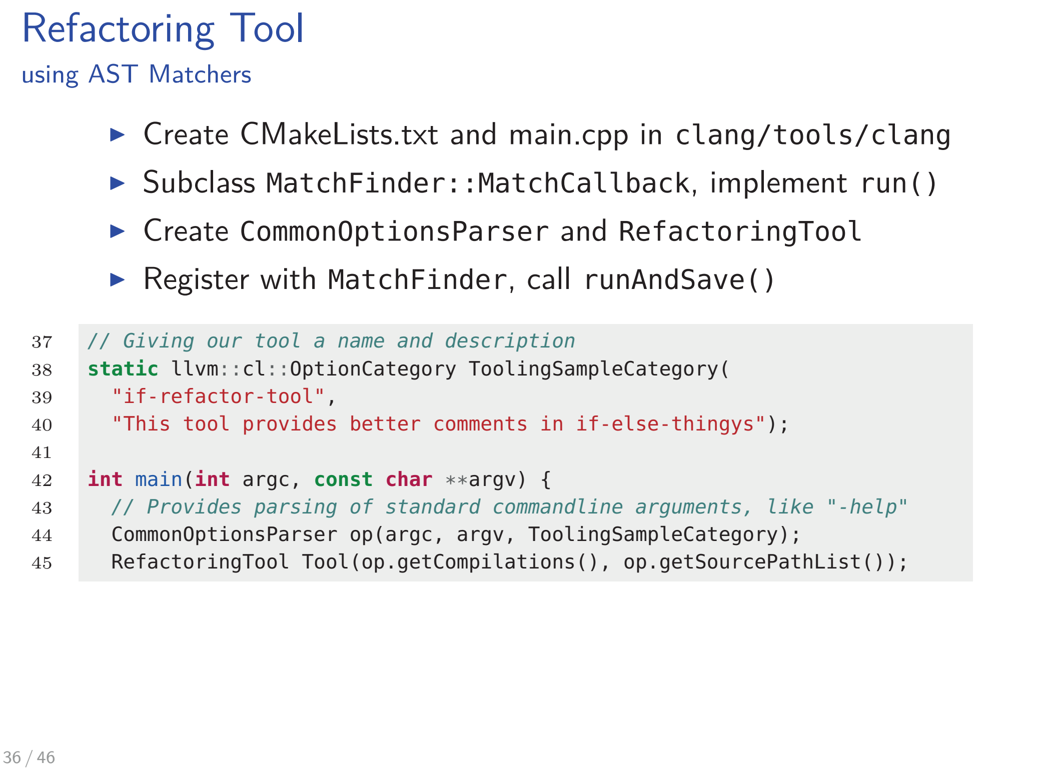
Refactoring Tool

using AST Matchers

- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```
  virtual void run(const MatchFinder::MatchResult &Result) {
    // The matched ’if’ statement is bound to ’myIdent’
    if (const IfStmt *IfS = Result.Nodes.getNodeAs<clang::IfStmt>("myIdent")) {
      const Stmt *Then = IfS->getThen();
      Replacement Rep(*(Result.SourceManager), Then->getLocStart(), 0,
                      "/* the ’if’ part */ ");
      Replace->insert(Rep);

      if (const Stmt *Else = IfS->getElse()) {
        Replacement Rep(*(Result.SourceManager), Else->getLocStart(), 0,
                        "/* the ’else’ part */ ");
        Replace->insert(Rep);
      }
    }
  }
```

35 / 46



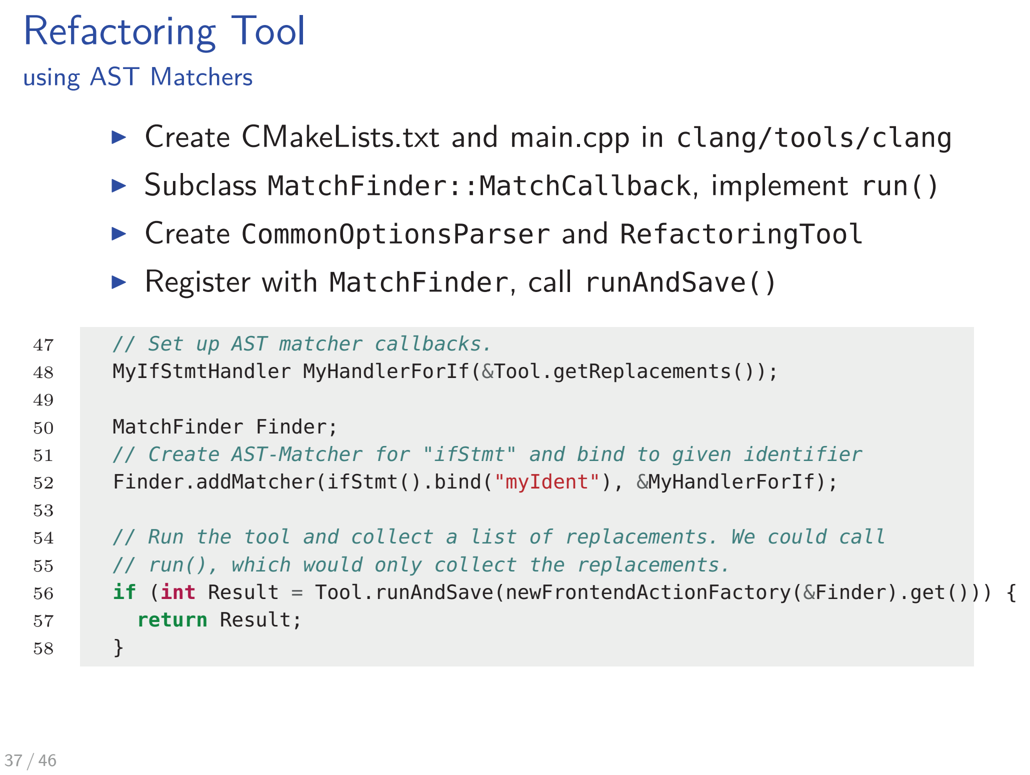
Refactoring Tool

using AST Matchers

- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```
// Giving our tool a name and description
static llvm::cl::OptionCategory ToolingSampleCategory(
  "if-refactor-tool",
  "This tool provides better comments in if-else-thingys");

int main(int argc, const char **argv) {
  // Provides parsing of standard commandline arguments, like "-help"
  CommonOptionsParser op(argc, argv, ToolingSampleCategory);
  RefactoringTool Tool(op.getCompilations(), op.getSourcePathList());
```

36 / 46

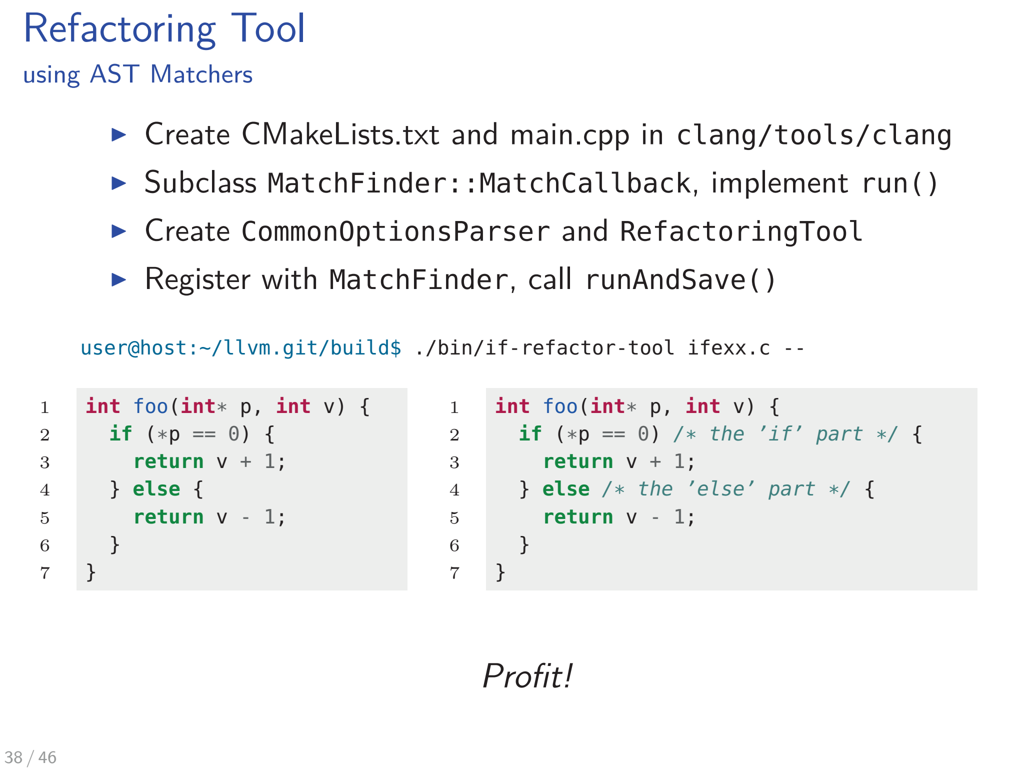


Refactoring Tool

using AST Matchers

- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```cpp
  // Set up AST matcher callbacks.
  MyIfStmtHandler MyHandlerForIf(&Tool.getReplacements());

  MatchFinder Finder;
  // Create AST-Matcher for "ifStmt" and bind to given identifier
  Finder.addMatcher(ifStmt().bind("myIdent"), &MyHandlerForIf);

  // Run the tool and collect a list of replacements. We could call
  // run(), which would only collect the replacements.
  if (int Result = Tool.runAndSave(newFrontendActionFactory(&Finder).get())) {
    return Result;
  }
```



Refactoring Tool

using AST Matchers

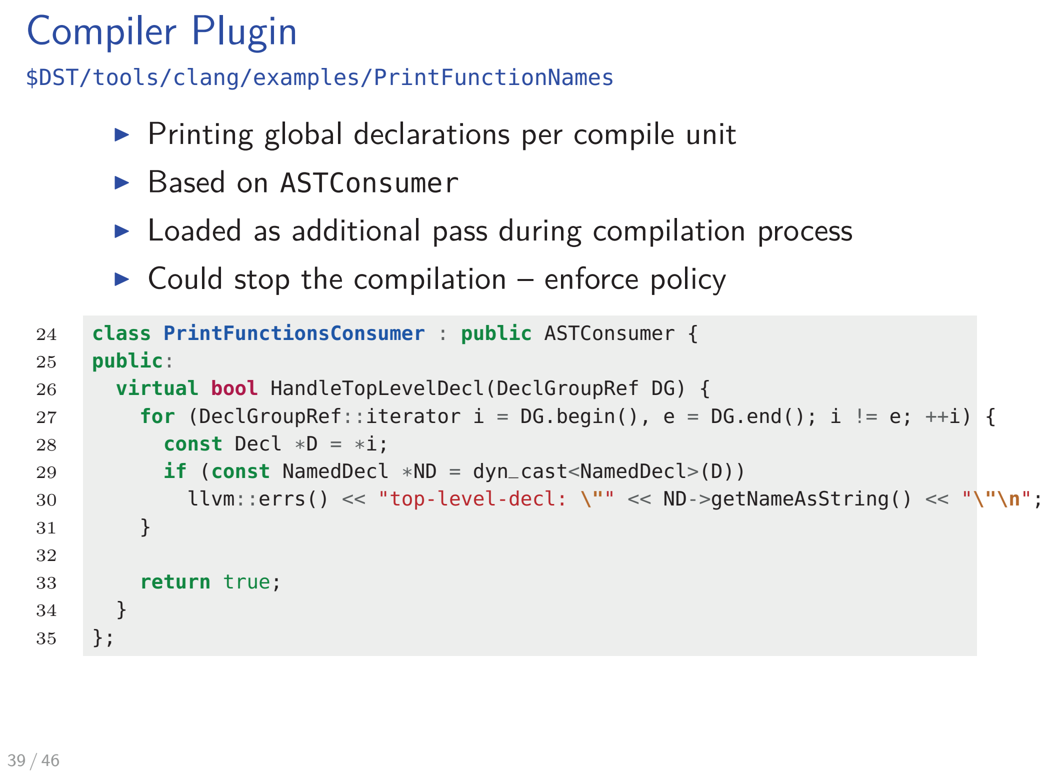
- Create CMakeLists.txt and main.cpp in clang/tools/clang
- Subclass MatchFinder::MatchCallback, implement run()
- Create CommonOptionsParser and RefactoringTool
- Register with MatchFinder, call runAndSave()

```
user@host:~/llvm.git/build$ ./bin/if-refactor-tool ifexx.c --
```

```c
int foo(int* p, int v) {
  if (*p == 0) {
    return v + 1;
  } else {
    return v - 1;
  }
}
```

```c
int foo(int* p, int v) {
  if (*p == 0) /* the 'if' part */ {
    return v + 1;
  } else /* the 'else' part */ {
    return v - 1;
  }
}
```

Profit!

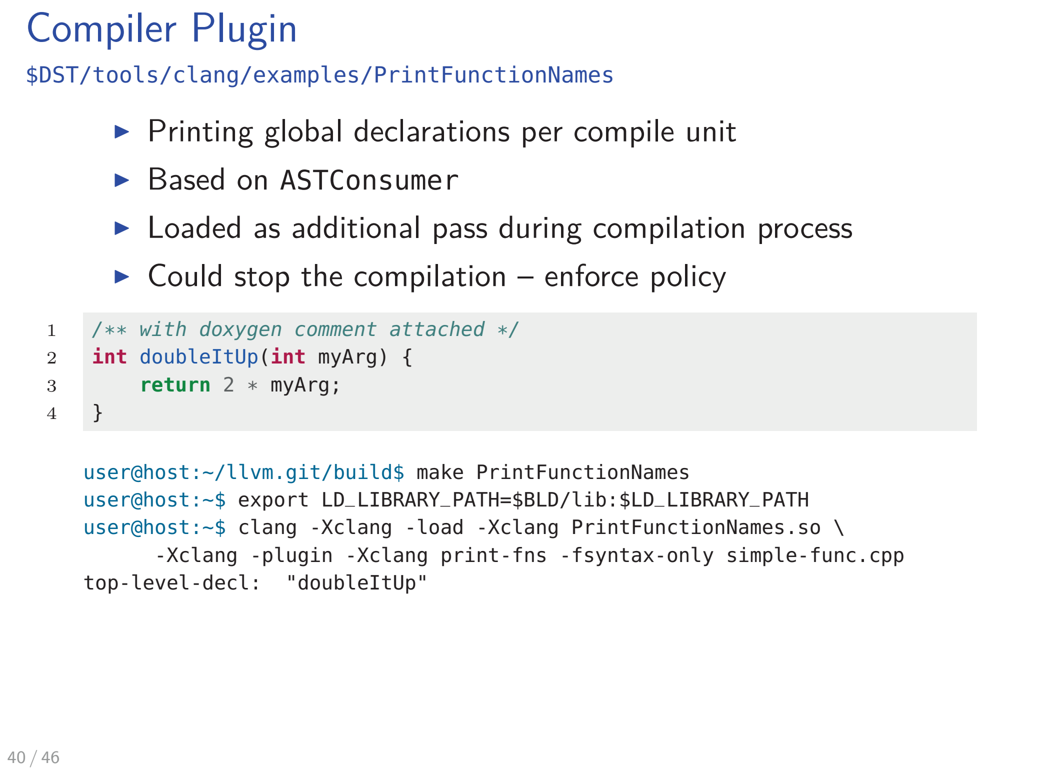


Compiler Plugin

`$DST/tools/clang/examples/PrintFunctionNames`

- Printing global declarations per compile unit
- Based on ASTConsumer
- Loaded as additional pass during compilation process
- Could stop the compilation – enforce policy

```
class PrintFunctionsConsumer : public ASTConsumer {
public:
  virtual bool HandleTopLevelDecl(DeclGroupRef DG) {
    for (DeclGroupRef::iterator i = DG.begin(), e = DG.end(); i != e; ++i) {
      const Decl *D = *i;
      if (const NamedDecl *ND = dyn_cast<NamedDecl>(D))
        llvm::errs() << "top-level-decl: \"" << ND->getNameAsString() << "\"\n";
    }

    return true;
  }
};
```



Compiler Plugin

`$DST/tools/clang/examples/PrintFunctionNames`

- Printing global declarations per compile unit
- Based on ASTConsumer
- Loaded as additional pass during compilation process
- Could stop the compilation – enforce policy

```
/** with doxygen comment attached */
int doubleItUp(int myArg) {
    return 2 * myArg;
}
```

```
user@host:~/llvm.git/build$ make PrintFunctionNames
user@host:~$ export LD_LIBRARY_PATH=$BLD/lib:$LD_LIBRARY_PATH
user@host:~$ clang -Xclang -load -Xclang PrintFunctionNames.so \
      -Xclang -plugin -Xclang print-fns -fsyntax-only simple-func.cpp
top-level-decl:  "doubleItUp"
```

Questions? Remarks?

Questions? Remarks?

Videos

Some full-hour talks on YouTube:

- 🔗 `The Clang AST - a tutorial`
- 🔗 `clang-format - Automatic formatting for C++`
- 🔗 `Clang MapReduce - Automatic C++ Refactoring at Google Scale`
- 🔗 `Interactive Metaprogramming Shell based on Clang`
- 🔗 `Rebuilding Debian with LLVM/Clang`
- 🔗 `Integrating LLVM into FreeBSD`

LLVM/Clang

Runtime tooling

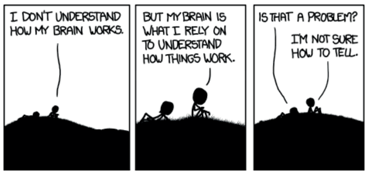
Runtime instrumentation by clang with additional *Sanitizer*:

- AddressSanitizer – Fast memory error detector
- ThreadSanitizer – Detects data races
- LeakSanitizer – Memory leak detector
- MemorySanitizer – Detects reads of uninitialized variables
- UBSanitizer – Detects undefined behaviour

Other runtime tooling:

- lldb – LLVM-based debugger
- *lld – early stage of a new Linker*

lldb


I DON'T UNDERSTAND HOW MY BRAIN WORKS.
BUT MY BRAIN IS WHAT I RELY ON TO UNDERSTAND HOW THINGS WORK.
IS THAT A PROBLEM?
I'M NOT SURE HOW TO TELL.

- Replacement for GDB, based on libTooling
- Standard in OS X, mostly supports Linux/FreeBSD
- Python and C++API
- Much faster with similar syntax/features

```
user@host:~$ lldb ./build/myTest

# start execution with argument
(lldb) process launch -- -myArg=2
# display 'this' when stopping in 'class myClass'
(lldb) target stop-hook add --classname myClass \
        --one-liner "frame variable *this"
# Show mixed source and disassembly of current line
(lldb) disassemble --frame --mixed
```

🔗 `gdb-to-lldb cheat set`

clang-tidy

- Enforce coding style, more tests than `clang-check`
- Compile with own checks, including `-fixits`
- Config file: `.clang-tidy`, in closest parent-dir of src-file
- Allows to ignore specific lines, enable/disable checks

🔗 `add your own checks`

2.4 ‘Using Pull Requests on GitHub - Experience report’ (FW-T-04)

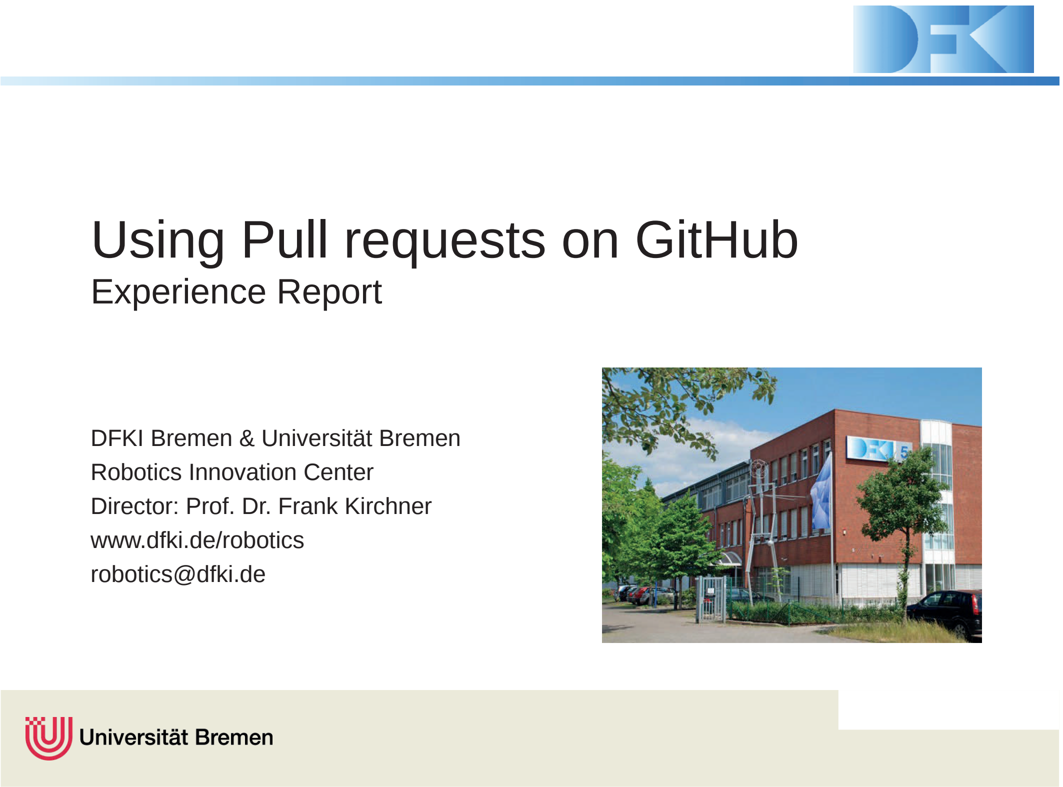
Steffen Planthaber[1]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `steffen.planthaber@dfki.de`

Abstract

Pull request are an essential part of a developer workflow when interacting with git software repositories. This presentation contains an experience report on how to work with pull requests on github.com. Furthermore, it shows to use this functionality to improve the overall code quality by using it for continuous code reviews.

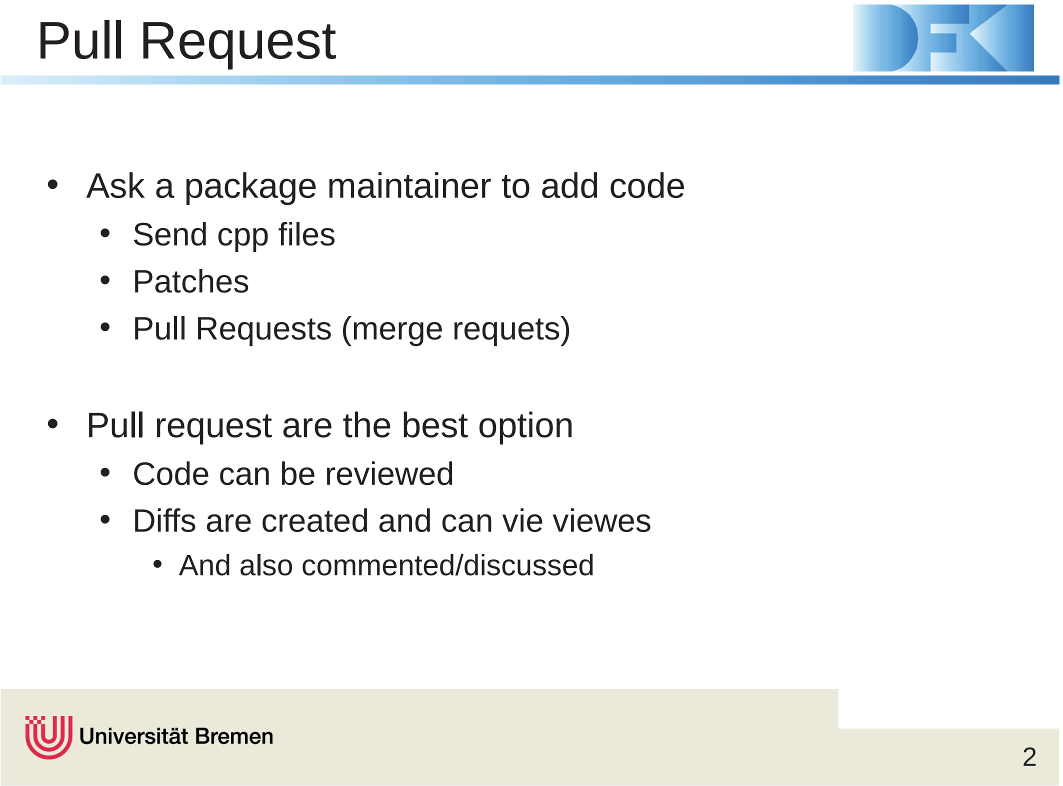


Using Pull requests on GitHub

Experience Report

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



Pull Request

- Ask a package maintainer to add code
  - Send cpp files
  - Patches
  - Pull Requests (merge requets)

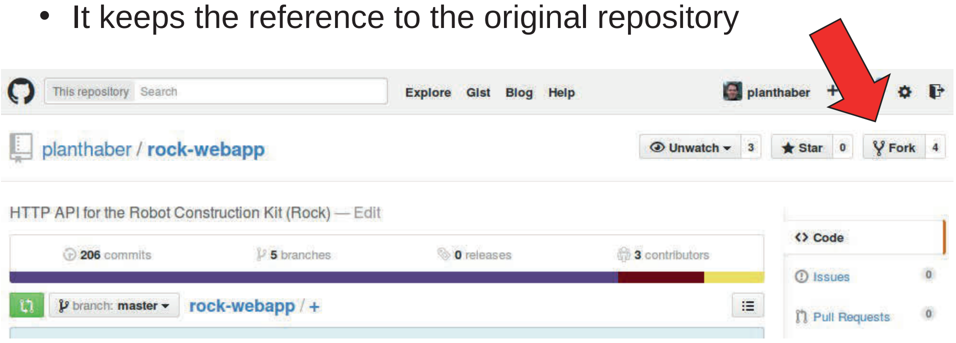
- Pull request are the best option
  - Code can be reviewed
  - Diffs are created and can vie viewes
    - And also commented/discussed

Universität Bremen

2

How to create one

- You need a branch!
  - In the main repository
  - Or in a „fork“
- A fork copies a repository into another account
  - It keeps the reference to the original repository



planthaber / rock-webapp

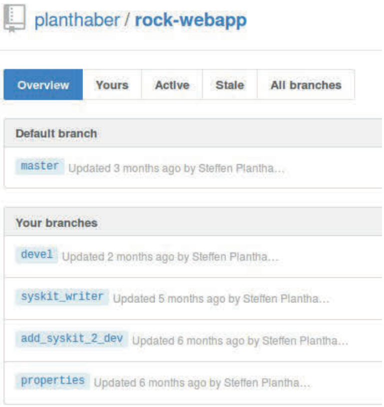
HTTP API for the Robot Construction Kit (Rock) — Edit

206 commits · 5 branches · 0 releases · 3 contributors

branch: master · rock-webapp / +

Work on the branch

- Add your fork as remote
  - git add remote myfork PUSH_URL
  - git checkout –b myPRbranch
  - git commit –m“added feature“ -a
  - git push myfork myPRbranch
- One branch/pull request per Feature
  - „master“ branch <- do not use
  - „devel“ for development
    - Several pull request branches
    - One branch per pull request!



planthaber / rock-webapp

Default branch

master Updated 3 months ago by Steffen Plantha…

Your branches

devel Updated 2 months ago by Steffen Plantha…

syskit_writer Updated 5 months ago by Steffen Plantha…

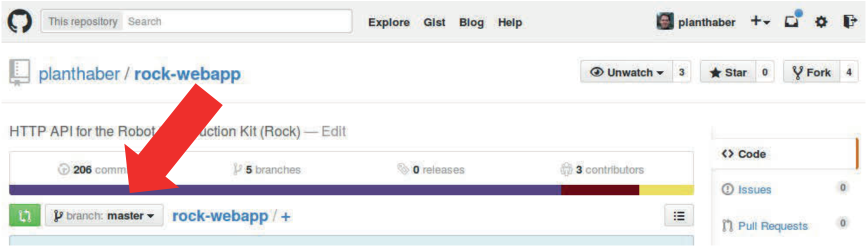
add_syskit_2_dev Updated 6 months ago by Steffen Plantha…

properties Updated 6 months ago by Steffen Plantha…

Start Pull Request

- You need one branch per pull request!

- Select the PR branch in the main view
- Hit the green button



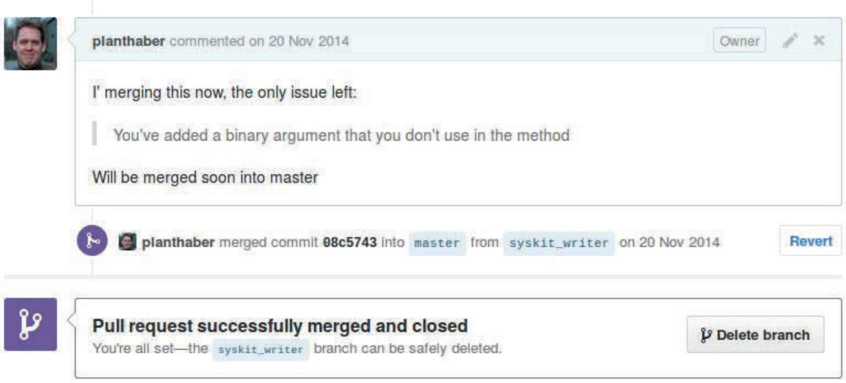
- Select branches to merge (if not correct)

Discuss the Changes

- The maintainer can now:
  - Comment or ask questions
    - On the PR in general
    - On specific code lines
    - Accept the PR and merge via the PR page on github
- You can
  - Reply
  - Push to the PR branch to update the PR
    - Important: You need to comment on the PR to notify
    - No mail is send on branch update!

- https://github.com/planthaber/rock-webapp/pull/17

Merged!



planthaber commented on 20 Nov 2014

I' merging this now, the only issue left:

> You've added a binary argument that you don't use in the method

Will be merged soon into master

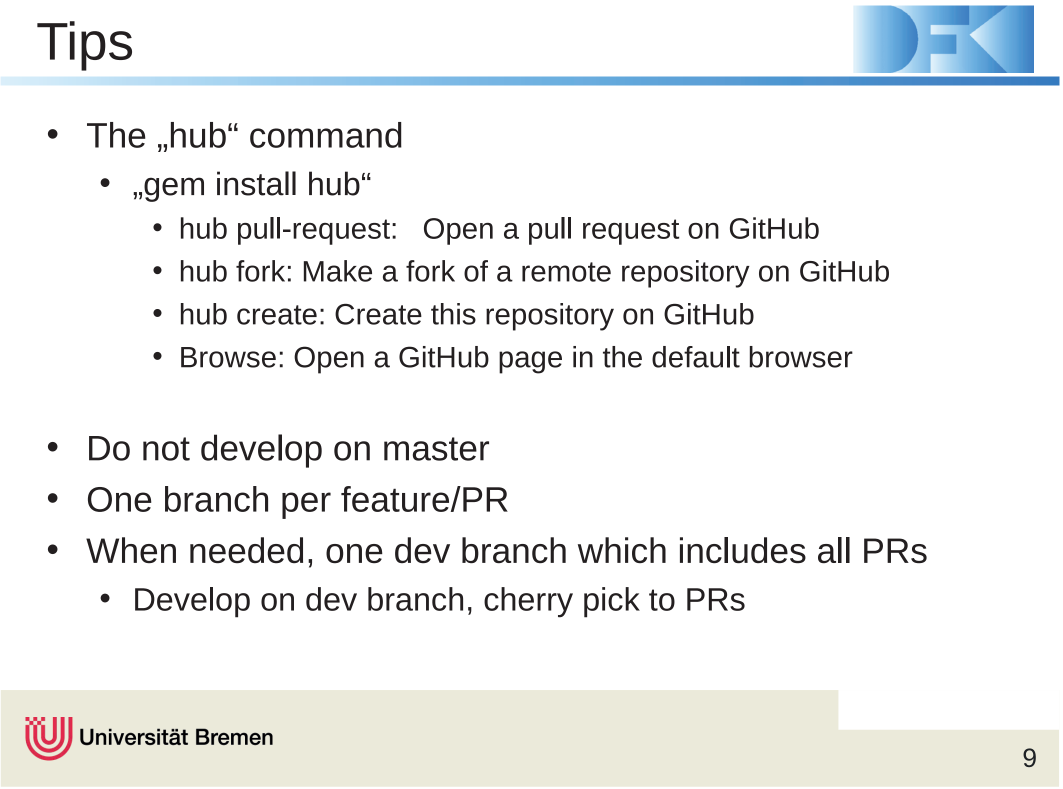
planthaber merged commit 08c5743 into `master` from `syskit_writer` on 20 Nov 2014

Pull request successfully merged and closed
You're all set—the `syskit_writer` branch can be safely deleted.

Merged myself!

Possibilities

- PRs are offering the possibility for code reviews
- They can be used for code reviews
  - „When in doubt, use a PR“
  - Or by default
    - Two users with write access on one repository
    - Only pull requests used for changing master
    - Shared dev branch, or seperate ones

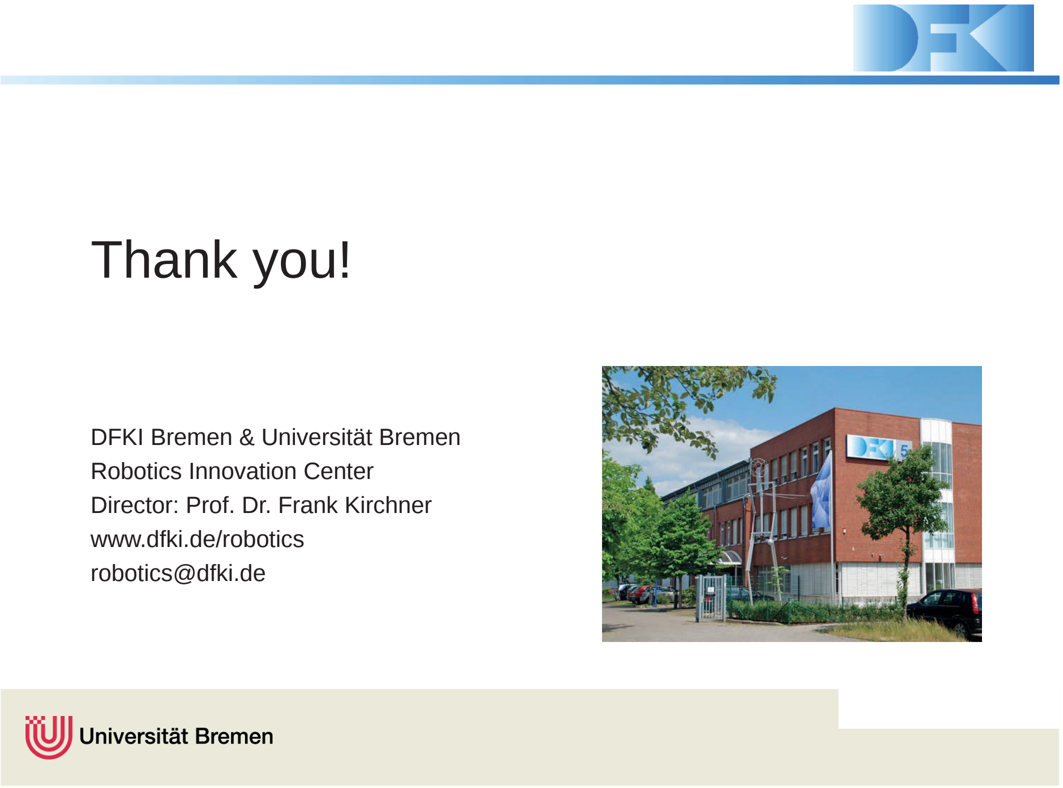


Tips

- The „hub“ command
  - „gem install hub“
    - hub pull-request: Open a pull request on GitHub
    - hub fork: Make a fork of a remote repository on GitHub
    - hub create: Create this repository on GitHub
    - Browse: Open a GitHub page in the default browser
- Do not develop on master
- One branch per feature/PR
- When needed, one dev branch which includes all PRs
  - Develop on dev branch, cherry pick to PRs

Universität Bremen

9



Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

2.5 ‘Rocks new http-based API for robot control’ (FW-T-05)

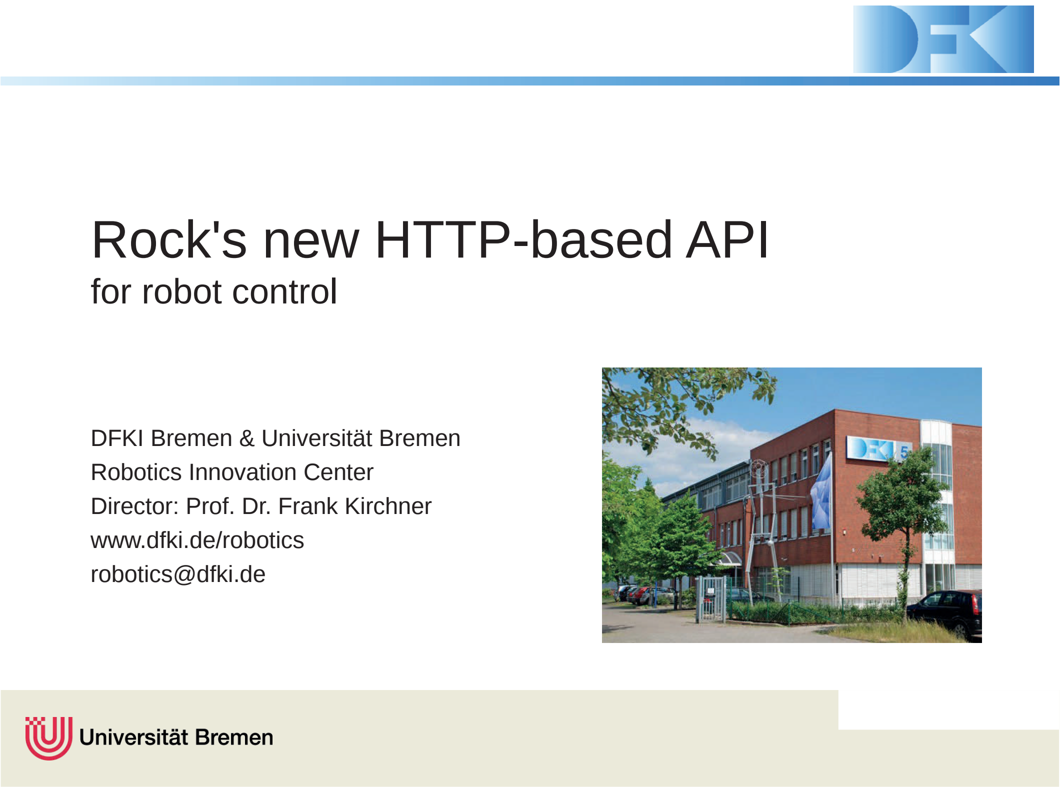
Steffen Planthaber[(1)]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: steffen.planthaber@dfki.de

Abstract

This presentation introduces the new HTTP API of Rock, which provides dependency-less control of rock-based robots since no rock installation needed for contol. It also introduces HTML5-based software using the new API to interact with robots from any smart device or tablet.

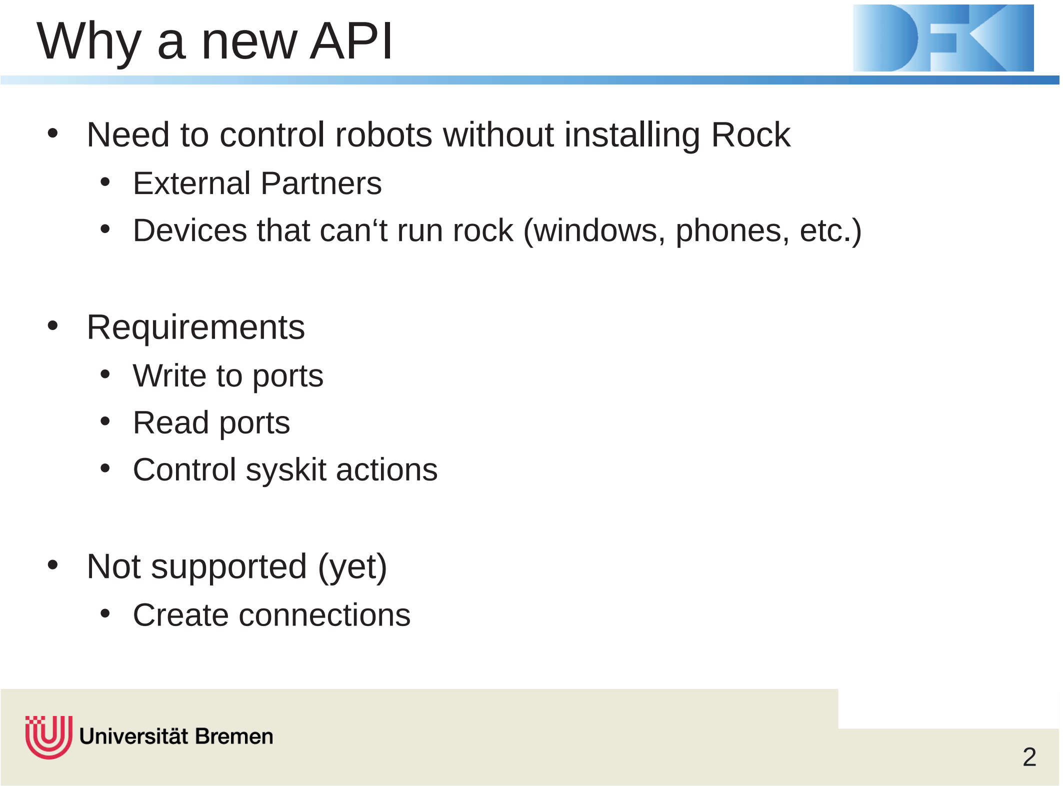


Rock's new HTTP-based API

for robot control

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



Why a new API

- Need to control robots without installing Rock
  - External Partners
  - Devices that can‘t run rock (windows, phones, etc.)
- Requirements
  - Write to ports
  - Read ports
  - Control syskit actions
- Not supported (yet)
  - Create connections

Universität Bremen

2

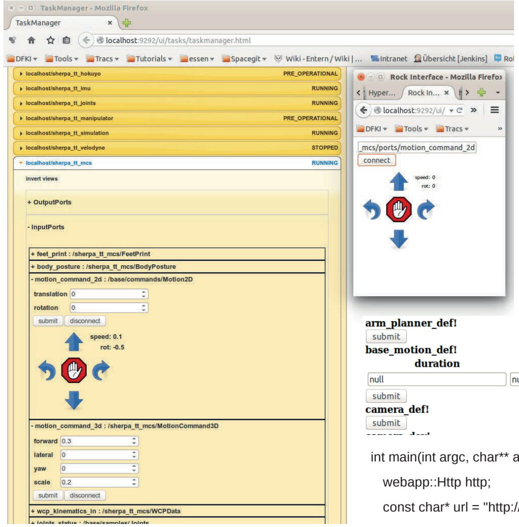
Types and Protocol Selection

- Types need to be transferred via custom connection
  - Several specifications
    - Prococol Buffers
    - Corba IDL
    - JSON
- The connections need a protocol
  - Options:
    - TCP/IP
    - Websockets
    - Higher level protocol with return values?
- Selected JSON, http, and optional Websockets
  - Any browser can use the API!

Universität Bremen

3

Examples


TaskManager - Mozilla Firefox
localhost:9292/ui/tasks/taskmanager.html

localhost/sherpa_tt_hokuyo	PRE_OPERATIONAL
localhost/sherpa_tt_imu	RUNNING
localhost/sherpa_tt_joints	RUNNING
localhost/sherpa_tt_manipulator	PRE_OPERATIONAL
localhost/sherpa_tt_simulation	RUNNING
localhost/sherpa_tt_velodyne	STOPPED
localhost/sherpa_tt_mcs	RUNNING

invert views

+ OutputPorts

- InputPorts

+ feet_print : /sherpa_tt_mcs/FeetPrint
+ body_posture : /sherpa_tt_mcs/BodyPosture
- motion_command_2d : /base/commands/Motion2D

translation 0
rotation 0

speed: 0.1
rot: -0.5

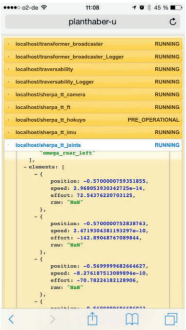
- motion_command_3d : /sherpa_tt_mcs/MotionCommand3D

forward 0.3
lateral 0
yaw 0
scale 0.2

+ wcp_kinematics_in : /sherpa_tt_mcs/WCPData
+ joints_status : /base/samples/Joints

Rock Interface - Mozilla Firefox
localhost:9292/ui/
_mcs/ports/motion_command_2d

speed: 0
rot: 0


●●●●○ o2-de 11:08 45 %
planthaber-u

localhost/transformer_broadcaster	RUNNING
localhost/transformer_broadcaster_Logger	RUNNING
localhost/traversability	RUNNING
localhost/traversability_Logger	RUNNING
localhost/sherpa_tt_camera	RUNNING
localhost/sherpa_tt_ft	RUNNING
localhost/sherpa_tt_hokuyo	PRE_OPERATIONAL
localhost/sherpa_tt_imu	RUNNING
localhost/sherpa_tt_joints	RUNNING

```
        "omega_rear_left"
    ],
  - elements: [
      - {
            position: -0.5700000759351855,
            speed: 2.968053930342725e-14,
            effort: 72.54376220703125,
            raw: "NaN"
        },
      - {
            position: -0.5700000752838763,
            speed: 2.4719304381193297e-10,
            effort: -142.89048767089844,
            raw: "NaN"
        },
      - {
            position: -0.5699999682664627,
            speed: -8.276187513089894e-10,
            effort: -70.78224182128906,
            raw: "NaN"
        },
      - {
```

arm_planner_def!
submit
base_motion_def!

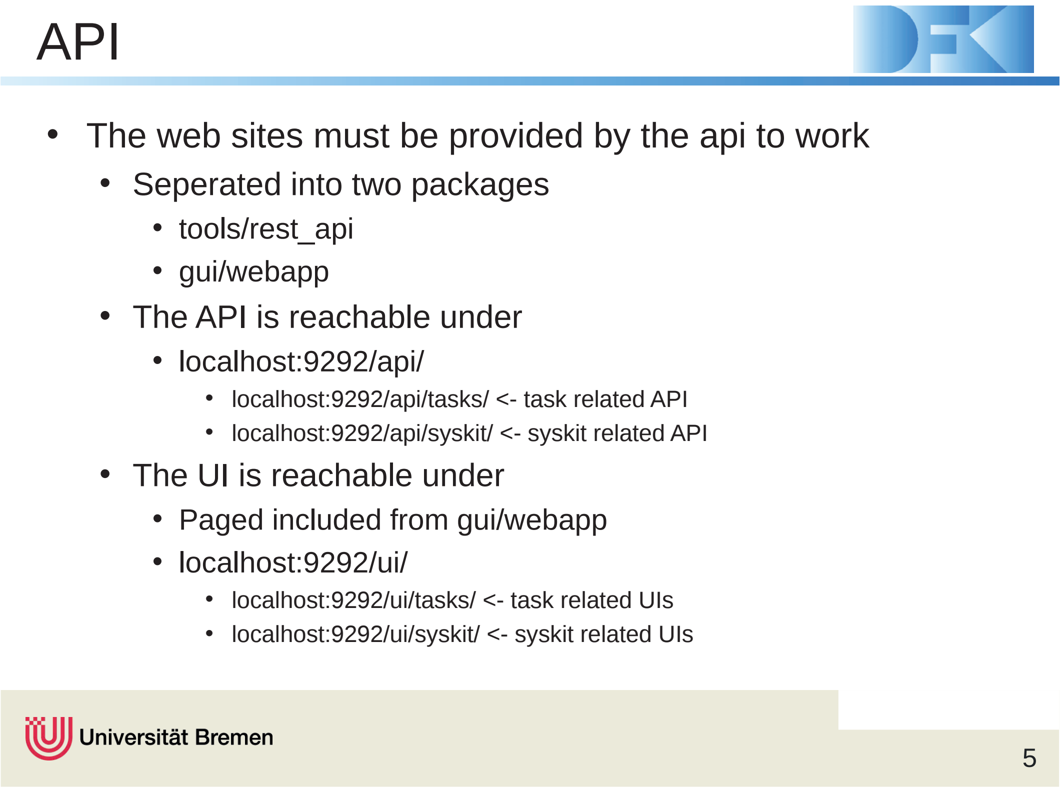
duration	lateral	velocity	yaw
null	null	null	null

submit
camera_def!
submit

```
int main(int argc, char** argv){
   webapp::Http http;
   const char* url = "http://192.168.151.171:9292/api/syskit/actions/move_to_def!/start";
   std::cout << http.post(url,"value={\"x\":1,\"y\":1}") << std::endl;
   return 0;
}
```

Universität Bremen

4

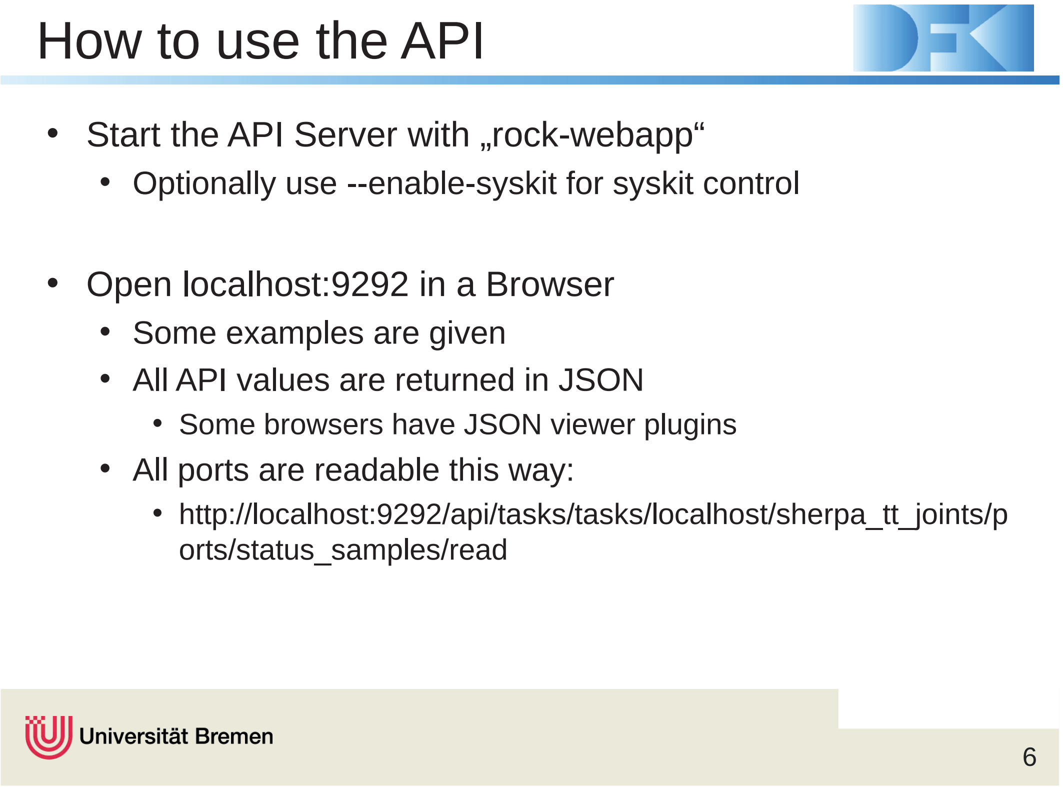


API

- The web sites must be provided by the api to work
  - Seperated into two packages
    - tools/rest_api
    - gui/webapp
  - The API is reachable under
    - localhost:9292/api/
      - localhost:9292/api/tasks/ <- task related API
      - localhost:9292/api/syskit/ <- syskit related API
  - The UI is reachable under
    - Paged included from gui/webapp
    - localhost:9292/ui/
      - localhost:9292/ui/tasks/ <- task related UIs
      - localhost:9292/ui/syskit/ <- syskit related UIs

Universität Bremen

5



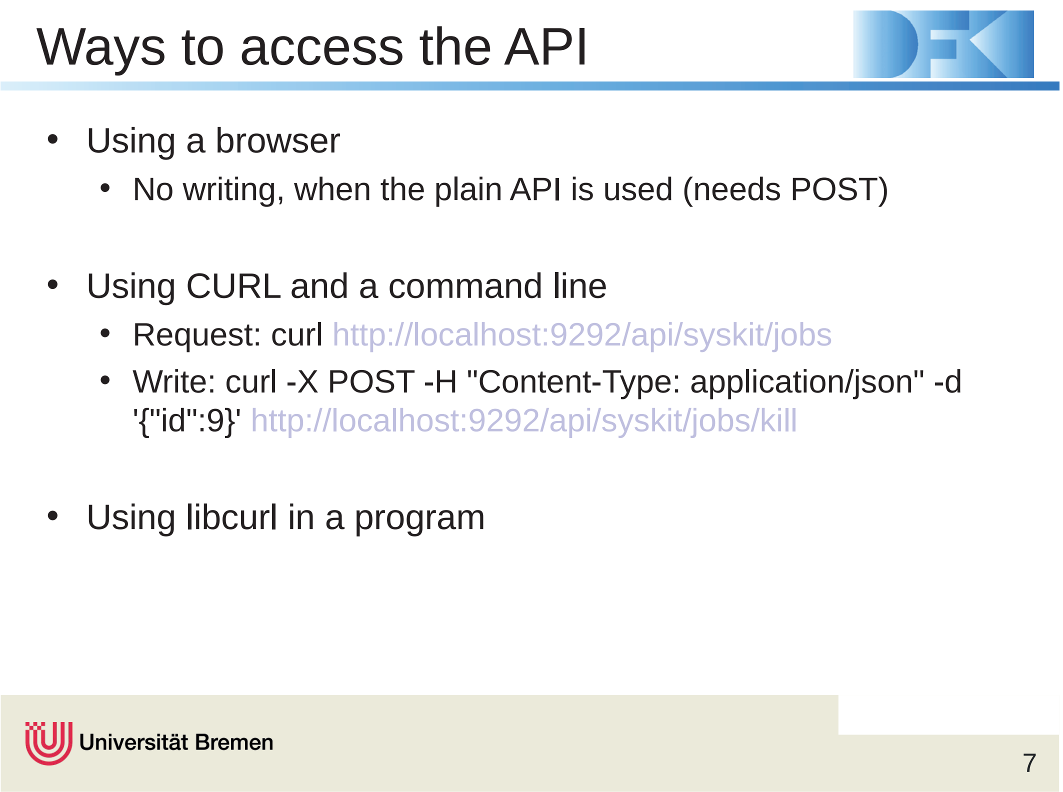
How to use the API

- Start the API Server with „rock-webapp“
  - Optionally use --enable-syskit for syskit control

- Open localhost:9292 in a Browser
  - Some examples are given
  - All API values are returned in JSON
    - Some browsers have JSON viewer plugins
  - All ports are readable this way:
    - http://localhost:9292/api/tasks/tasks/localhost/sherpa_tt_joints/ports/status_samples/read

Universität Bremen

6

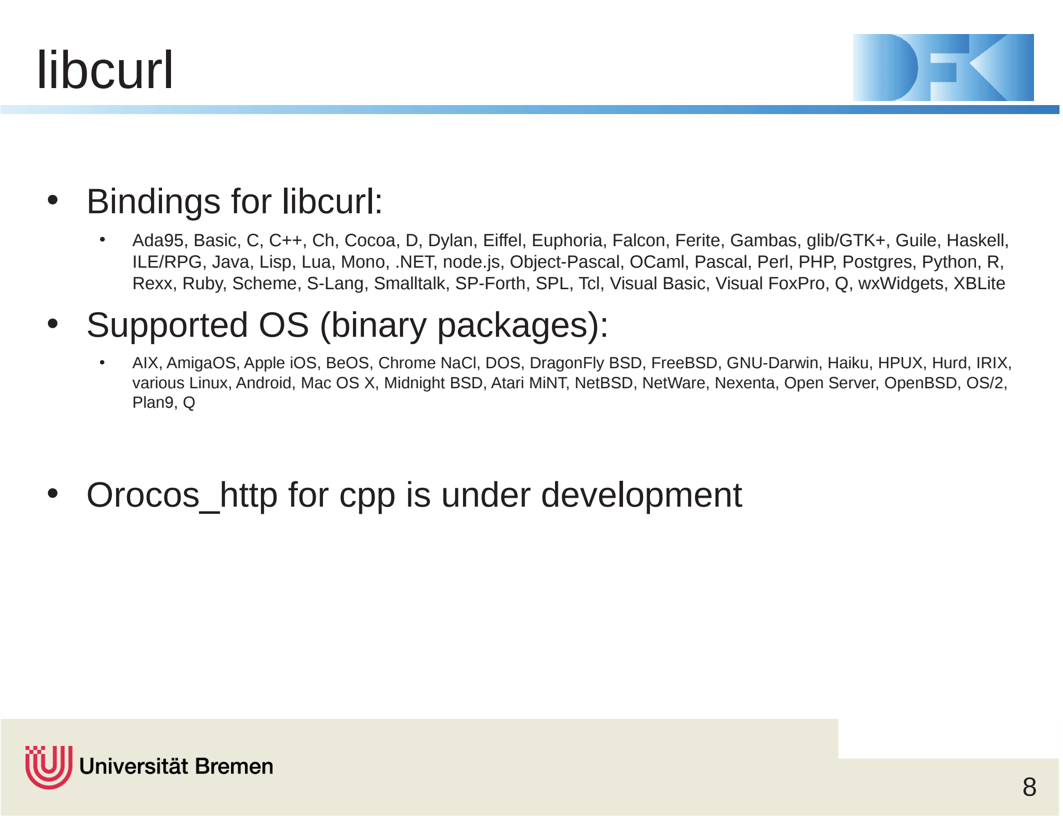


Ways to access the API

- Using a browser
  - No writing, when the plain API is used (needs POST)
- Using CURL and a command line
  - Request: curl http://localhost:9292/api/syskit/jobs
  - Write: curl -X POST -H "Content-Type: application/json" -d '{"id":9}' http://localhost:9292/api/syskit/jobs/kill
- Using libcurl in a program

Universität Bremen

7

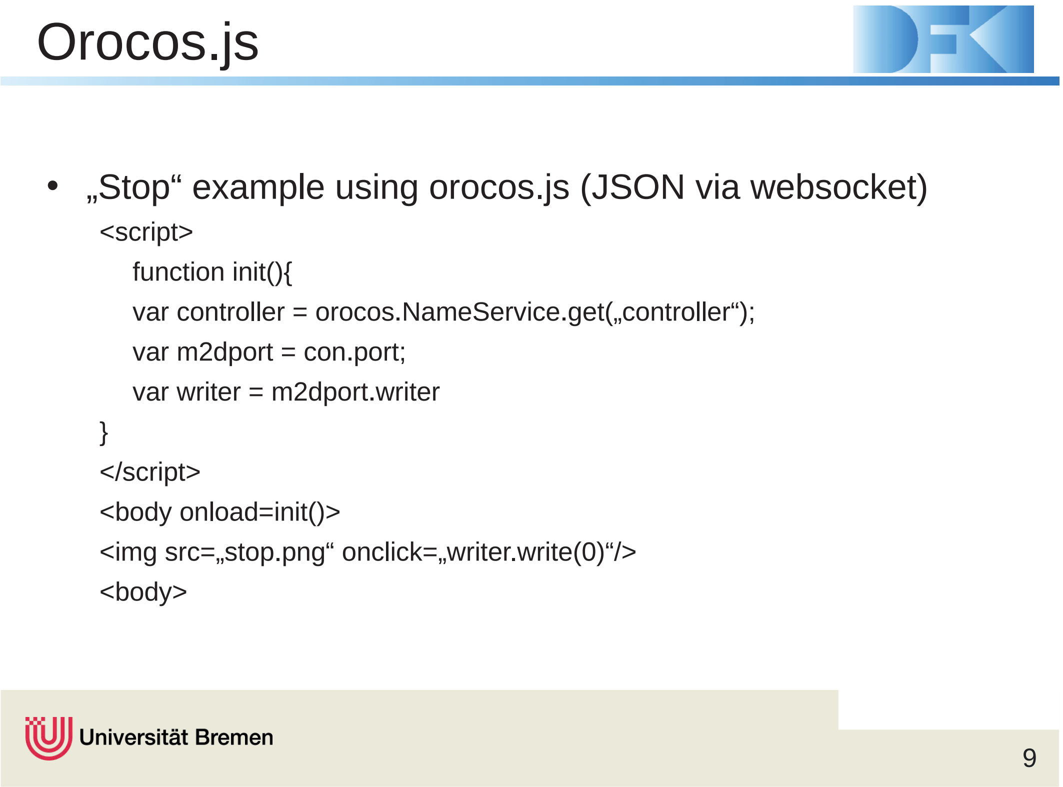


libcurl

- Bindings for libcurl:
  - Ada95, Basic, C, C++, Ch, Cocoa, D, Dylan, Eiffel, Euphoria, Falcon, Ferite, Gambas, glib/GTK+, Guile, Haskell, ILE/RPG, Java, Lisp, Lua, Mono, .NET, node.js, Object-Pascal, OCaml, Pascal, Perl, PHP, Postgres, Python, R, Rexx, Ruby, Scheme, S-Lang, Smalltalk, SP-Forth, SPL, Tcl, Visual Basic, Visual FoxPro, Q, wxWidgets, XBLite
- Supported OS (binary packages):
  - AIX, AmigaOS, Apple iOS, BeOS, Chrome NaCl, DOS, DragonFly BSD, FreeBSD, GNU-Darwin, Haiku, HPUX, Hurd, IRIX, various Linux, Android, Mac OS X, Midnight BSD, Atari MiNT, NetBSD, NetWare, Nexenta, Open Server, OpenBSD, OS/2, Plan9, Q
- Orocos_http for cpp is under development

Universität Bremen

8



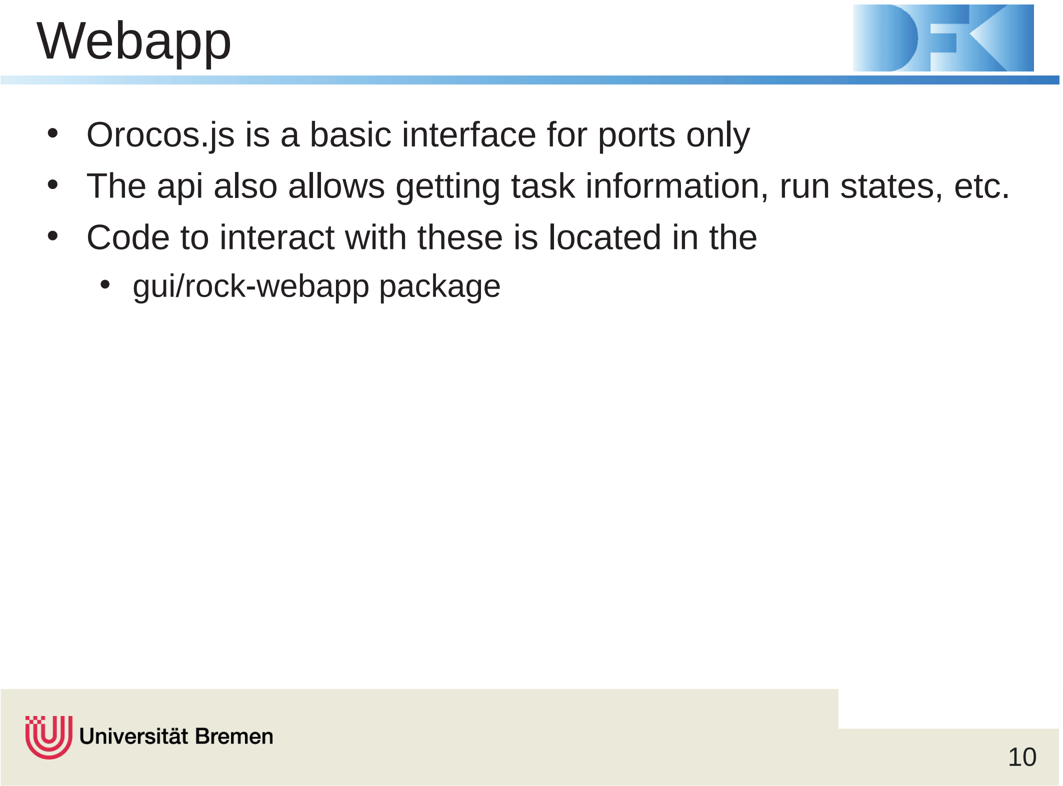
Orocos.js

- „Stop“ example using orocos.js (JSON via websocket)

```
<script>
   function init(){
   var controller = orocos.NameService.get(„controller“);
   var m2dport = con.port;
   var writer = m2dport.writer
}
</script>
<body onload=init()>
<img src=„stop.png“ onclick=„writer.write(0)“/>
<body>
```

Universität Bremen

9

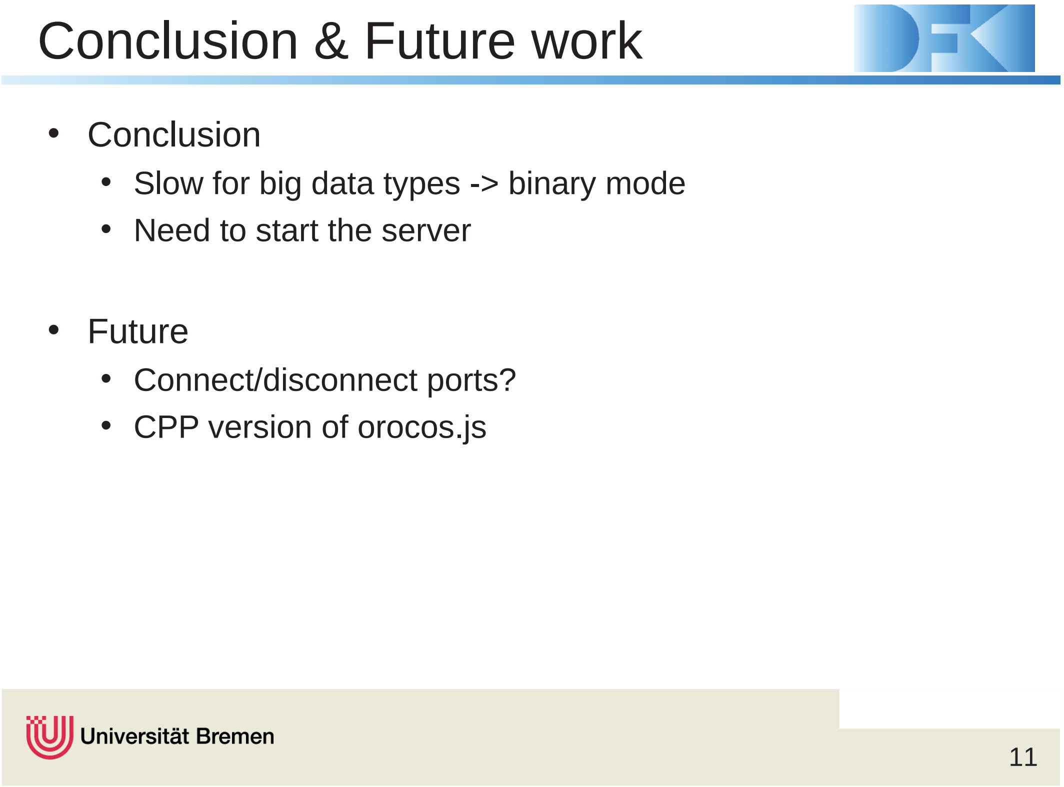


Webapp

- Orocos.js is a basic interface for ports only
- The api also allows getting task information, run states, etc.
- Code to interact with these is located in the
  - gui/rock-webapp package

Universität Bremen

10



Conclusion & Future work

- Conclusion
  - Slow for big data types -> binary mode
  - Need to start the server
- Future
  - Connect/disconnect ports?
  - CPP version of orocos.js

Universität Bremen

11

Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

2.6 'Constraint-based planning of component networks' (FW-T-06)

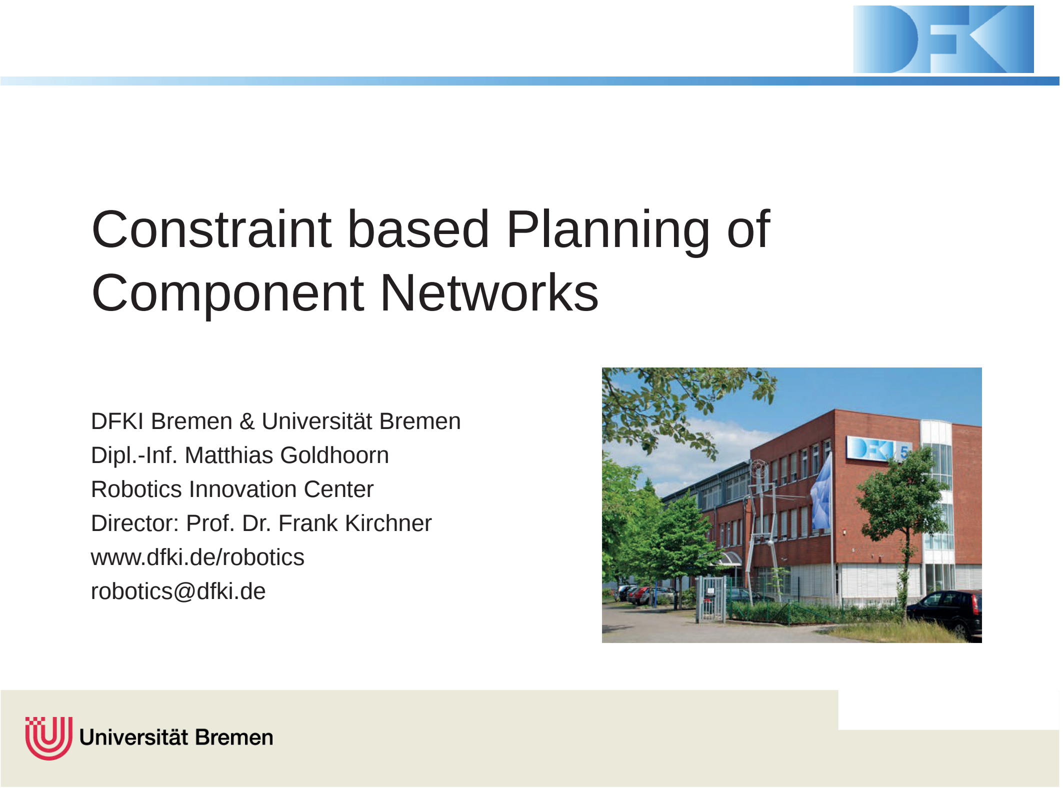
Matthias Goldhoorn[1]

(1) Universität Bremen, Arbeitsgruppe Robotik, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `matthias.goldhoorn@uni-bremen.de`

Abstract

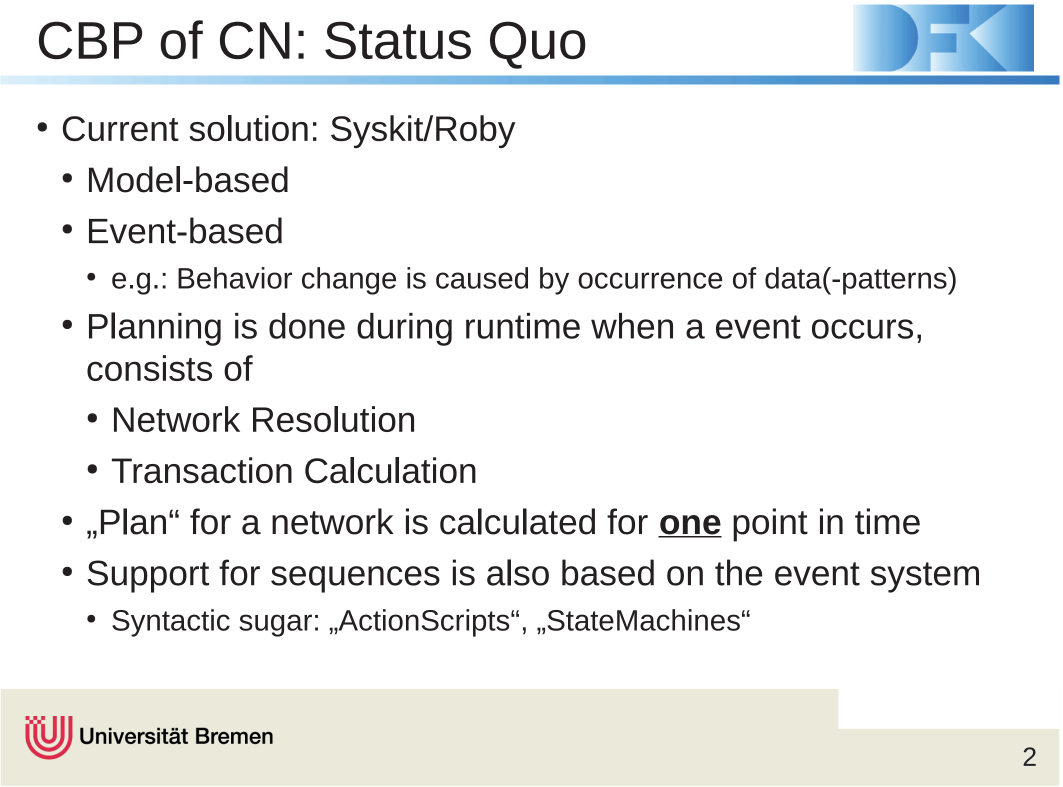
This presentation gives a overview of the ongoing work regarding constrained-based planning of component networks. This work aims at a better introspection and a more formalized way to handle component networks.



Constraint based Planning of Component Networks

DFKI Bremen & Universität Bremen
Dipl.-Inf. Matthias Goldhoorn
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

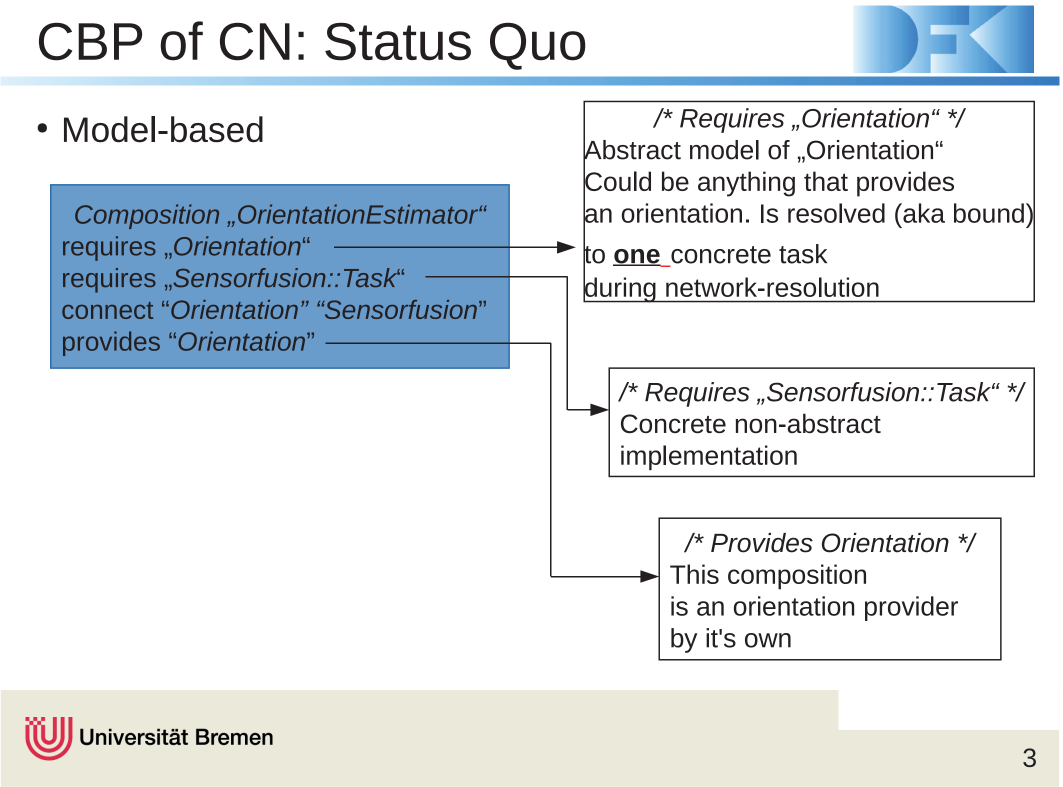
Universität Bremen



CBP of CN: Status Quo

- Current solution: Syskit/Roby
  - Model-based
  - Event-based
    - e.g.: Behavior change is caused by occurrence of data(-patterns)
  - Planning is done during runtime when a event occurs, consists of
    - Network Resolution
    - Transaction Calculation
  - „Plan“ for a network is calculated for **one** point in time
  - Support for sequences is also based on the event system
    - Syntactic sugar: „ActionScripts“, „StateMachines“

Universität Bremen 2



CBP of CN: Status Quo

- Model-based

Composition „OrientationEstimator"
requires „*Orientation*"
requires „*Sensorfusion::Task*"
connect "*Orientation*" "*Sensorfusion*"
provides "*Orientation*"

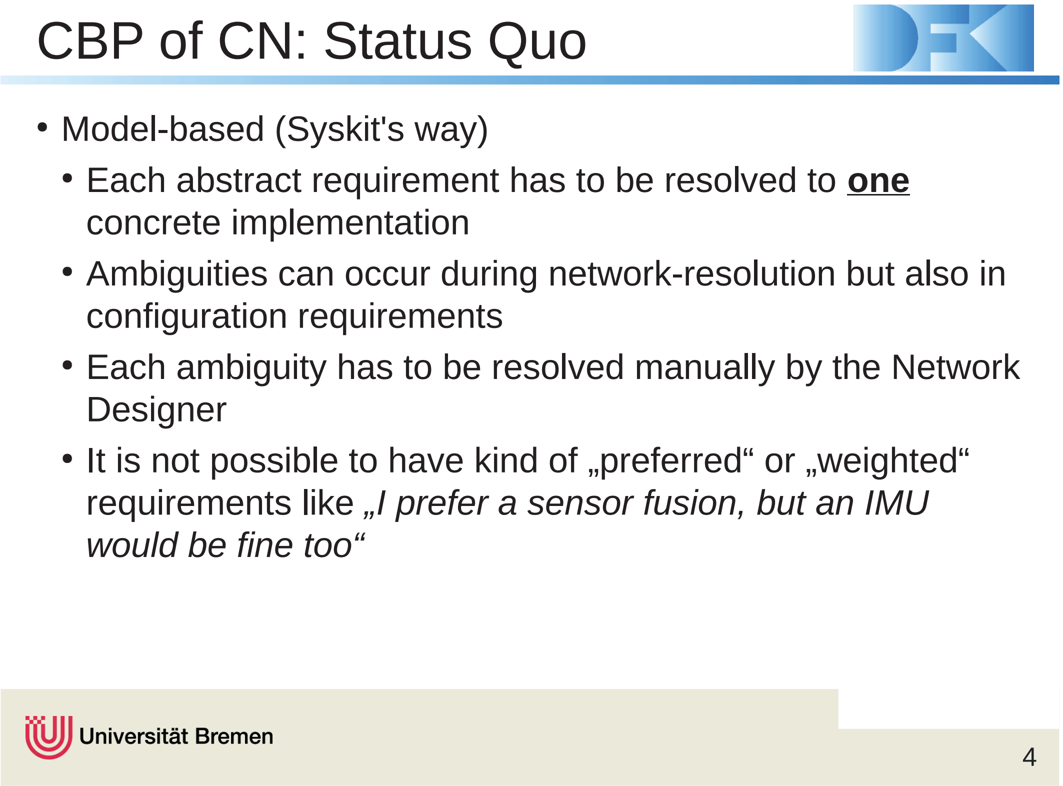
/ Requires „Orientation" */*
Abstract model of „Orientation"
Could be anything that provides
an orientation. Is resolved (aka bound)
to **one** concrete task
during network-resolution

/ Requires „Sensorfusion::Task" */*
Concrete non-abstract
implementation

/ Provides Orientation */*
This composition
is an orientation provider
by it's own

Universität Bremen

3

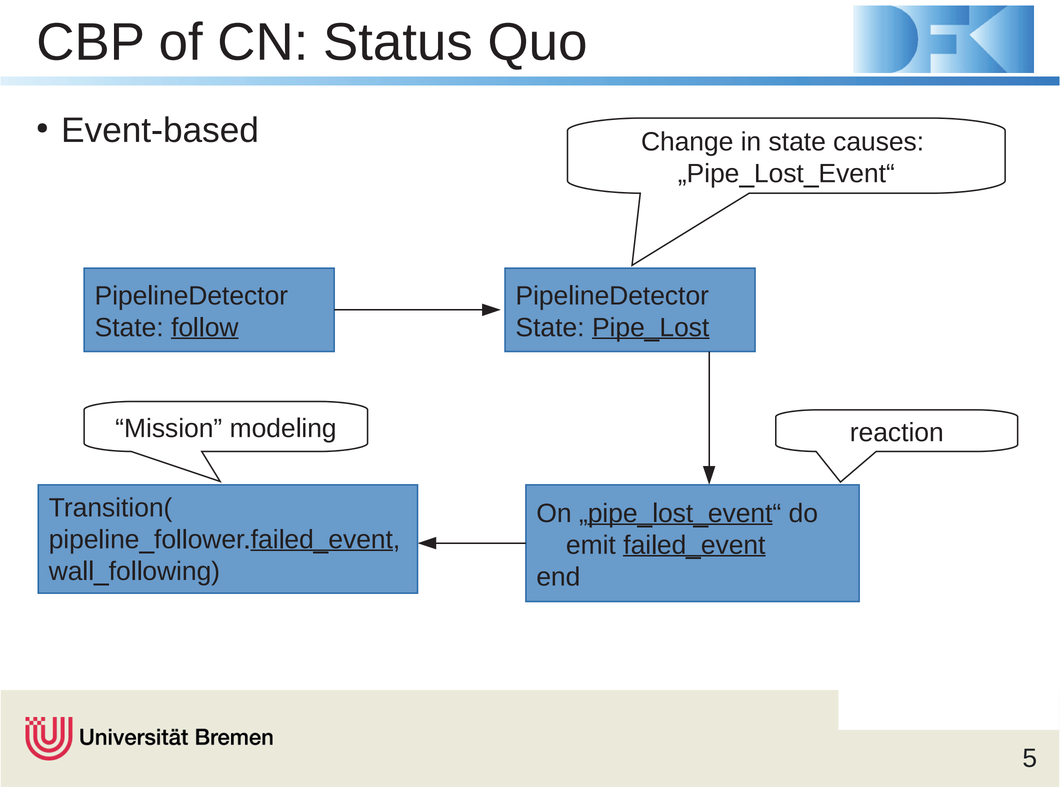


CBP of CN: Status Quo

- Model-based (Syskit's way)
  - Each abstract requirement has to be resolved to **one** concrete implementation
  - Ambiguities can occur during network-resolution but also in configuration requirements
  - Each ambiguity has to be resolved manually by the Network Designer
  - It is not possible to have kind of „preferred" or „weighted" requirements like *„I prefer a sensor fusion, but an IMU would be fine too"*

Universität Bremen

4



CBP of CN: Status Quo

- Event-based

Change in state causes: „Pipe_Lost_Event“

PipelineDetector
State: follow

PipelineDetector
State: Pipe_Lost

“Mission” modeling

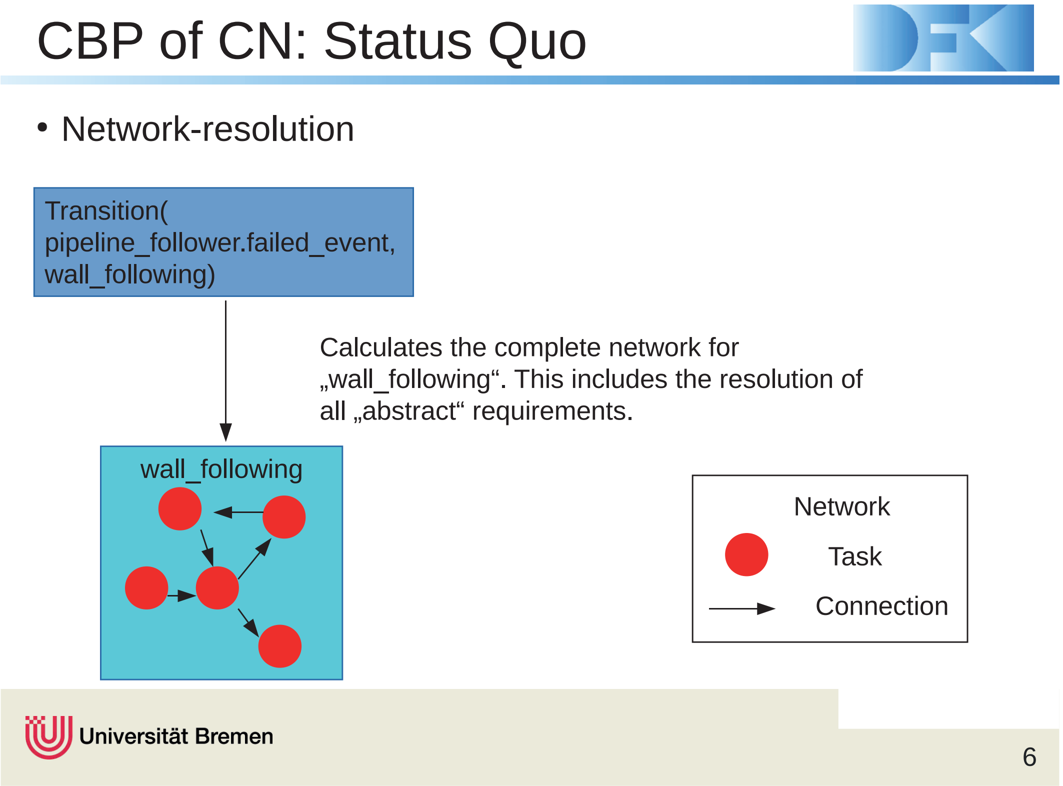
reaction

Transition(
pipeline_follower.failed_event,
wall_following)

On „pipe_lost_event“ do
emit failed_event
end

Universität Bremen

5



CBP of CN: Status Quo

- Network-resolution

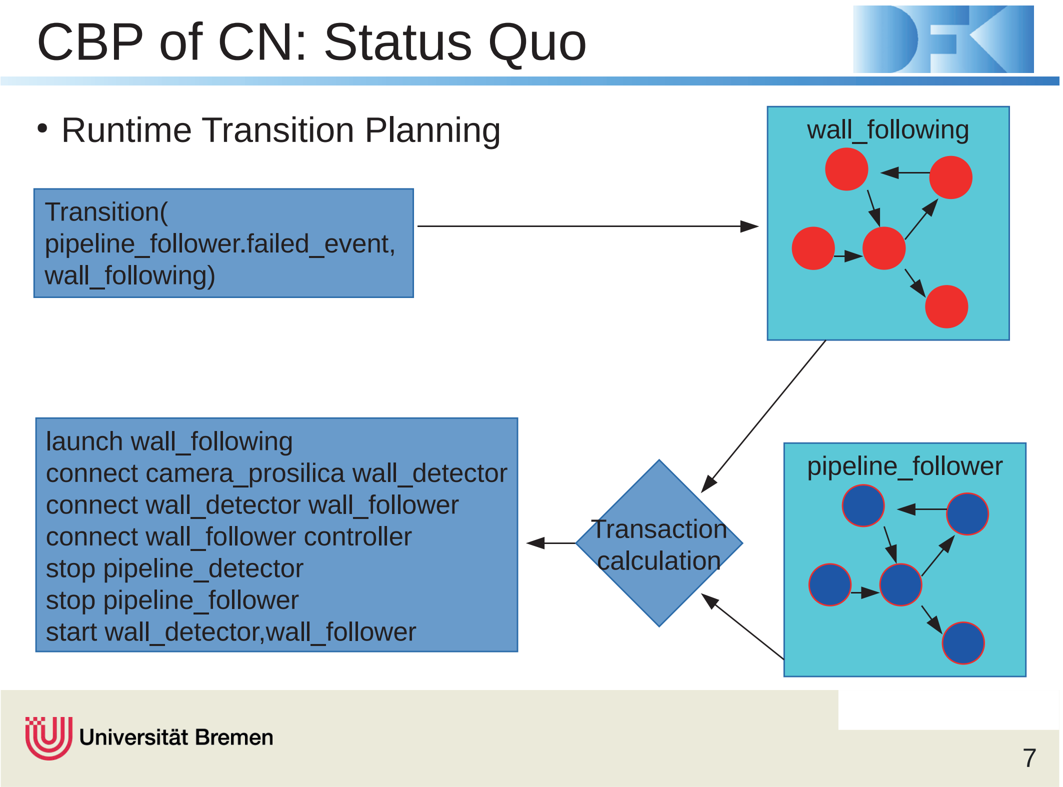
Transition(
pipeline_follower.failed_event,
wall_following)

Calculates the complete network for „wall_following“. This includes the resolution of all „abstract“ requirements.

Network
● Task
→ Connection

Universität Bremen

6



CBP of CN: Status Quo

- Runtime Transition Planning

Transition(
pipeline_follower.failed_event,
wall_following)

wall_following

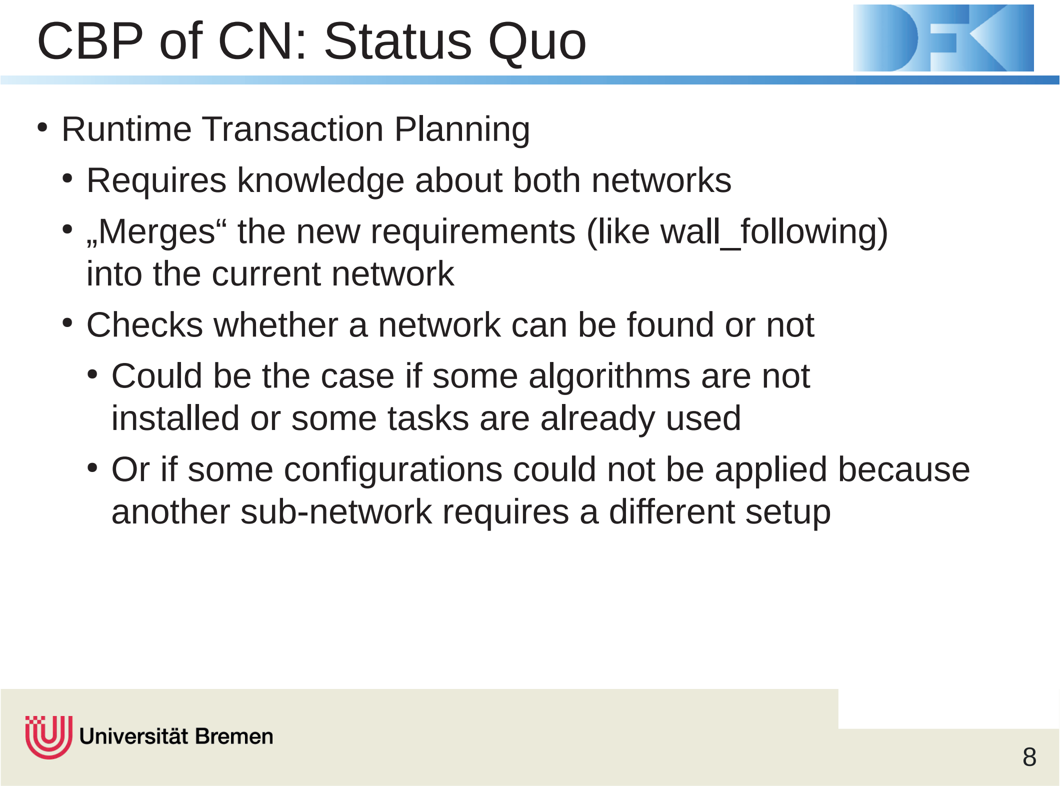
launch wall_following
connect camera_prosilica wall_detector
connect wall_detector wall_follower
connect wall_follower controller
stop pipeline_detector
stop pipeline_follower
start wall_detector,wall_follower

Transaction calculation

pipeline_follower

Universität Bremen

7

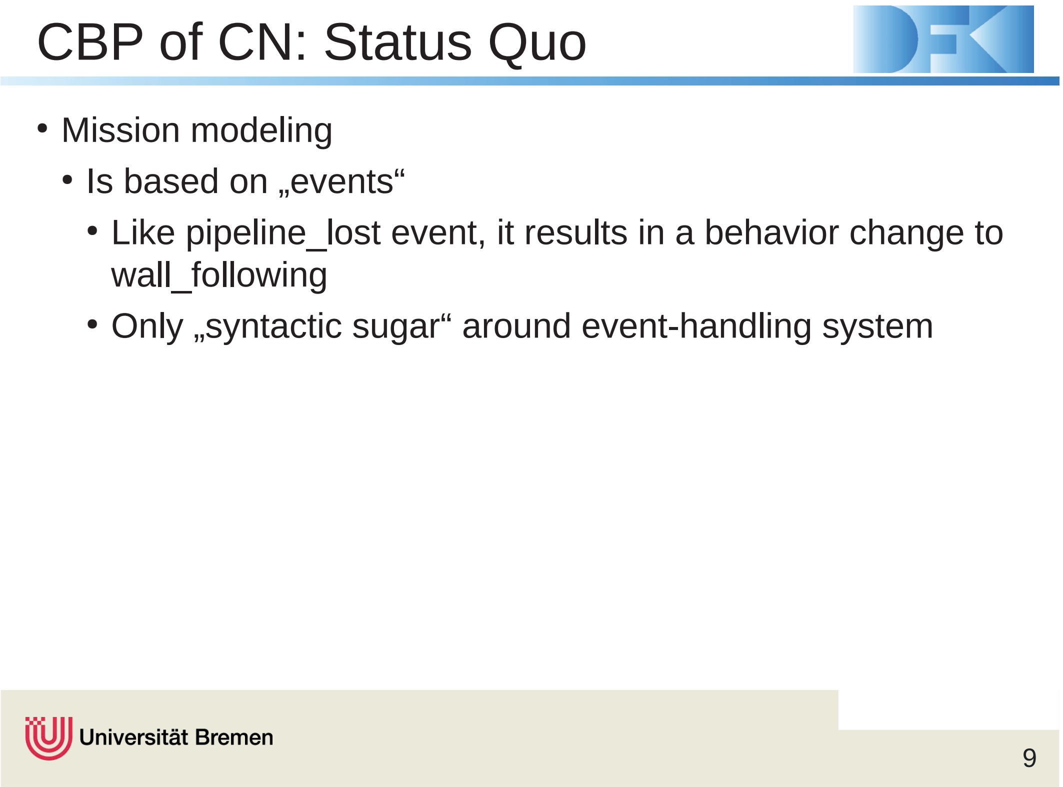


CBP of CN: Status Quo

- Runtime Transaction Planning
  - Requires knowledge about both networks
  - „Merges“ the new requirements (like wall_following) into the current network
  - Checks whether a network can be found or not
    - Could be the case if some algorithms are not installed or some tasks are already used
    - Or if some configurations could not be applied because another sub-network requires a different setup

Universität Bremen

8

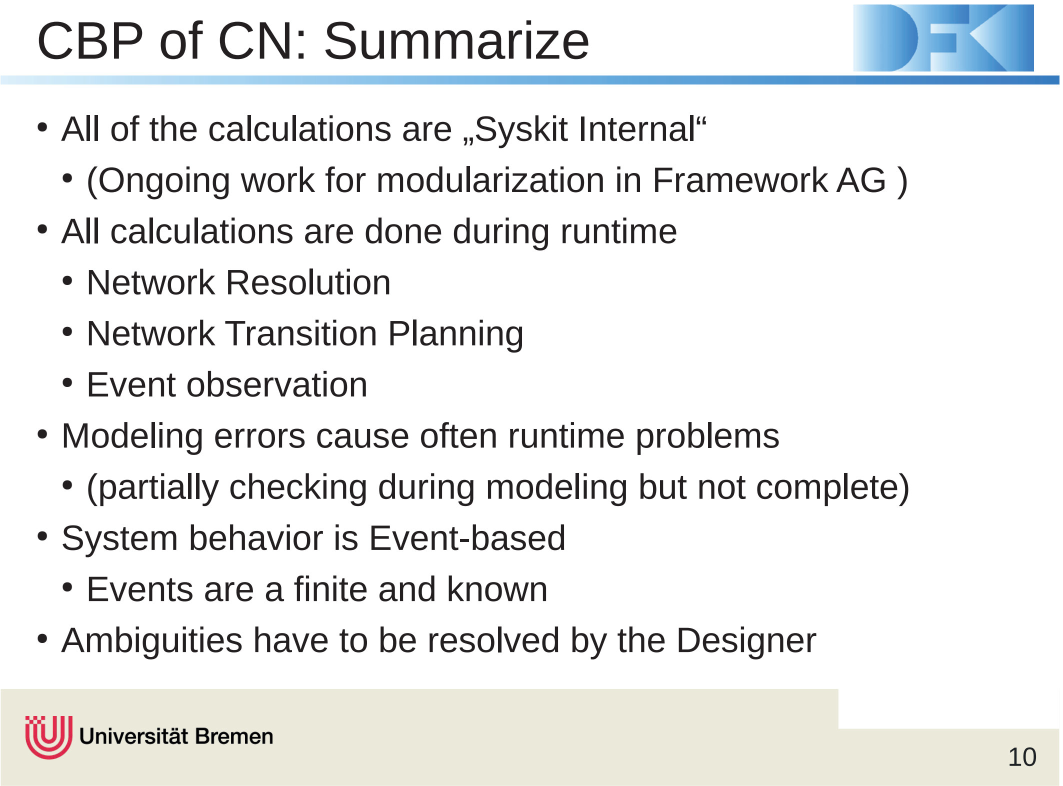


CBP of CN: Status Quo

- Mission modeling
  - Is based on „events“
    - Like pipeline_lost event, it results in a behavior change to wall_following
    - Only „syntactic sugar“ around event-handling system

Universität Bremen

9

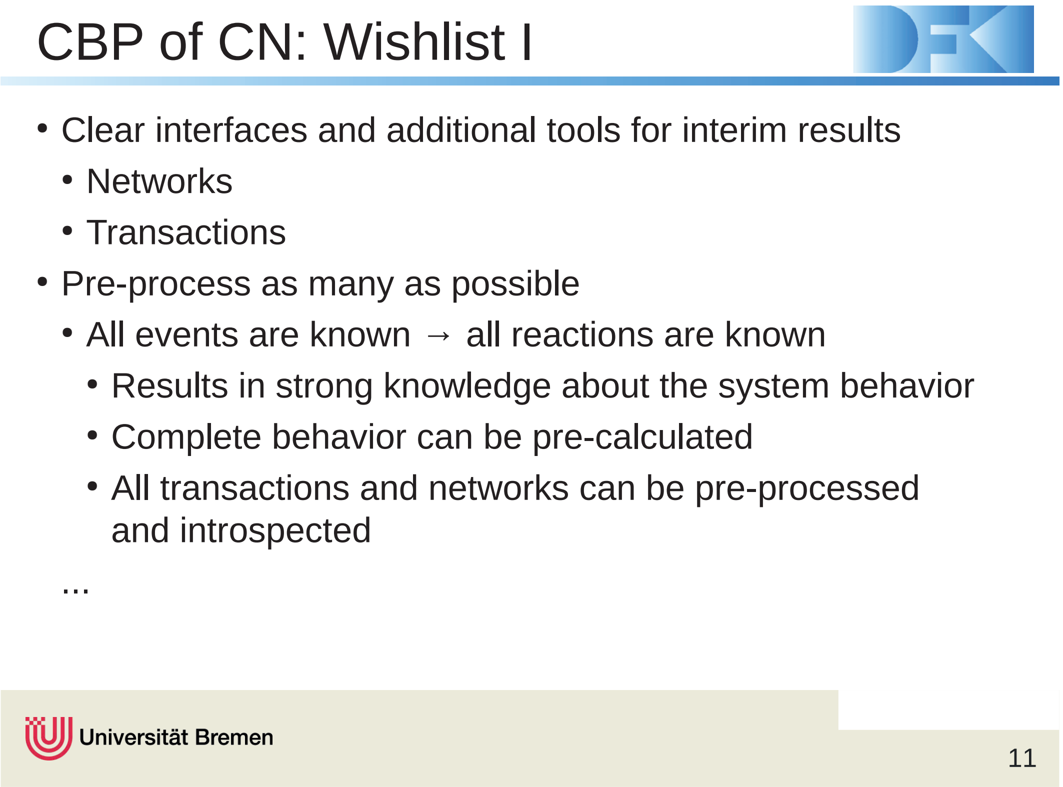


CBP of CN: Summarize

- All of the calculations are „Syskit Internal“
  - (Ongoing work for modularization in Framework AG)
- All calculations are done during runtime
  - Network Resolution
  - Network Transition Planning
  - Event observation
- Modeling errors cause often runtime problems
  - (partially checking during modeling but not complete)
- System behavior is Event-based
  - Events are a finite and known
- Ambiguities have to be resolved by the Designer

Universität Bremen

10



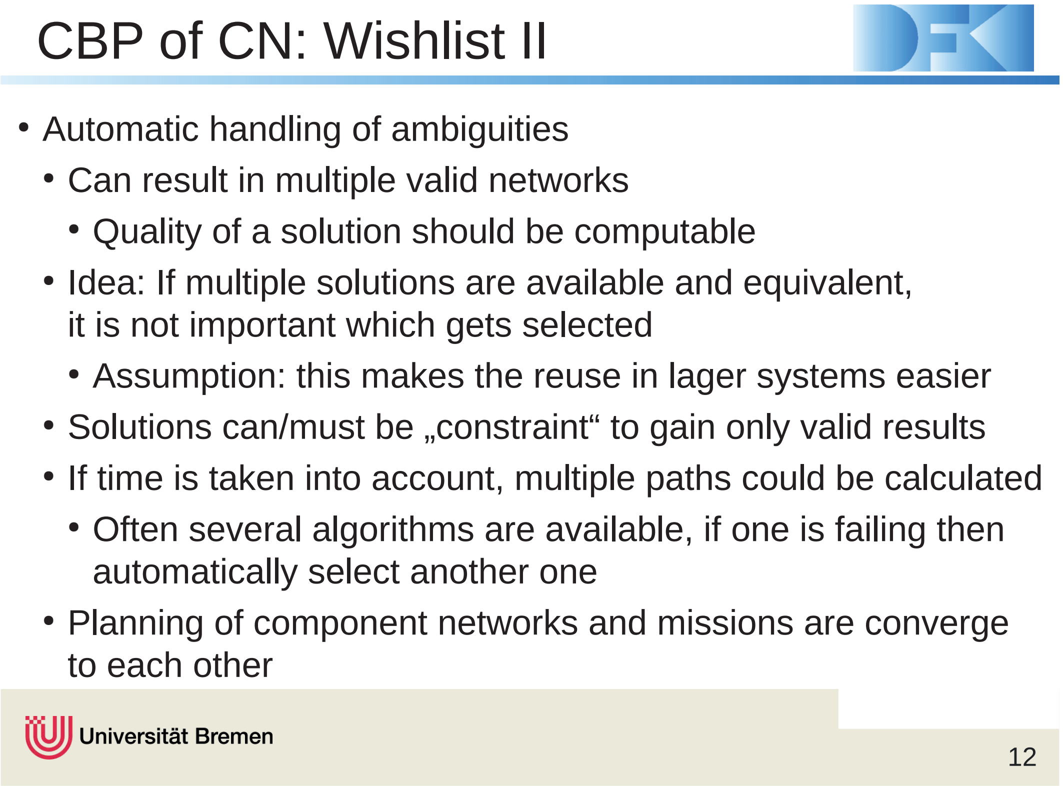
CBP of CN: Wishlist I

- Clear interfaces and additional tools for interim results
  - Networks
  - Transactions
- Pre-process as many as possible
  - All events are known → all reactions are known
    - Results in strong knowledge about the system behavior
    - Complete behavior can be pre-calculated
    - All transactions and networks can be pre-processed and introspected

...

Universität Bremen

11

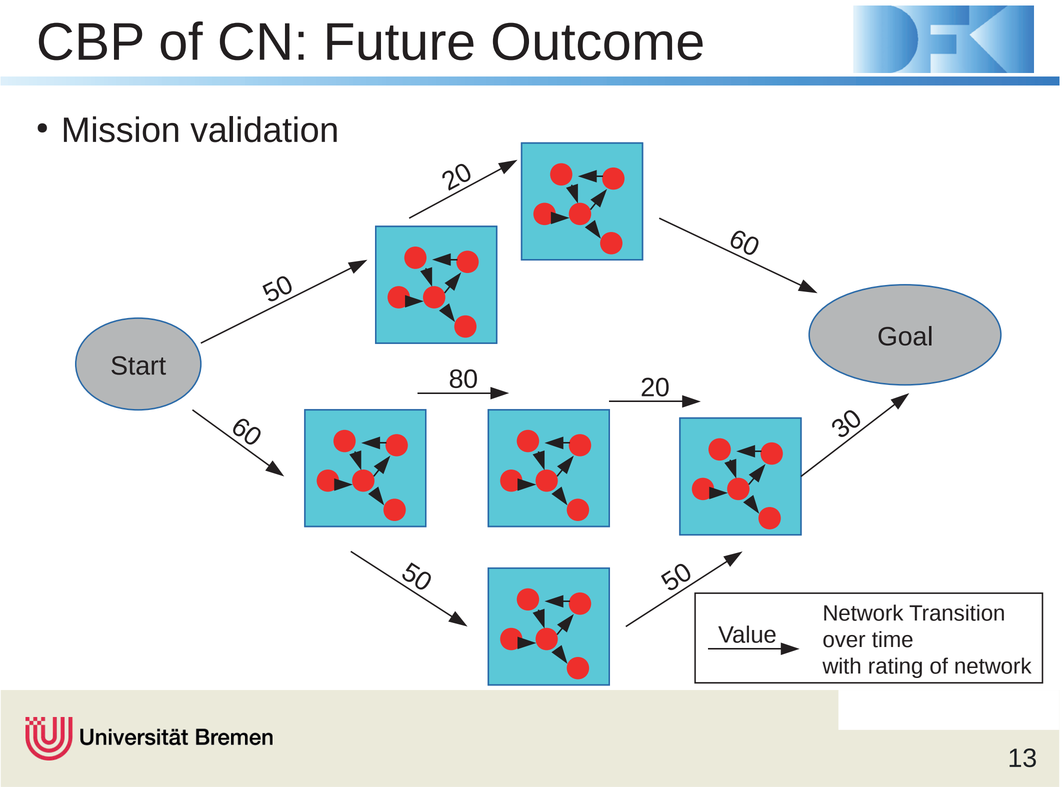


CBP of CN: Wishlist II

- Automatic handling of ambiguities
  - Can result in multiple valid networks
    - Quality of a solution should be computable
  - Idea: If multiple solutions are available and equivalent, it is not important which gets selected
    - Assumption: this makes the reuse in lager systems easier
  - Solutions can/must be „constraint“ to gain only valid results
  - If time is taken into account, multiple paths could be calculated
    - Often several algorithms are available, if one is failing then automatically select another one
  - Planning of component networks and missions are converge to each other

Universität Bremen

12



CBP of CN: Future Outcome

- Mission validation

Start

50

20

60

Goal

60

80

20

30

50

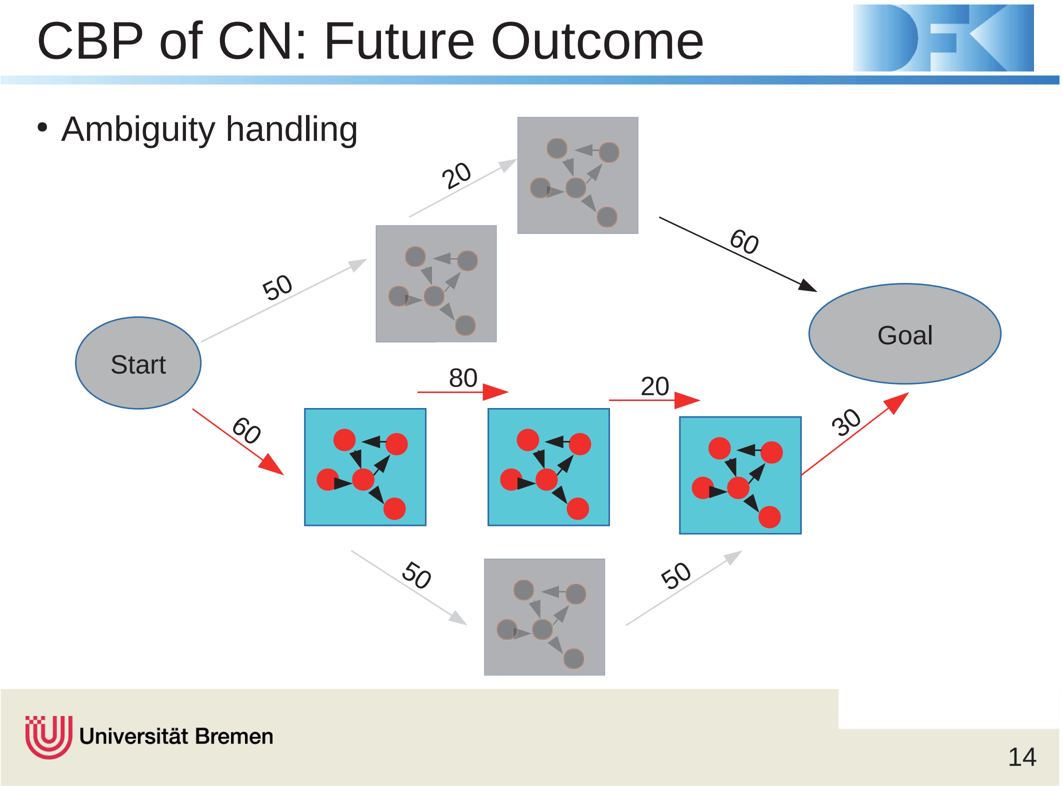
50

Value

Network Transition over time with rating of network

Universität Bremen

13



CBP of CN: Future Outcome

- Ambiguity handling

Start

50

20

60

Goal

60

80

20

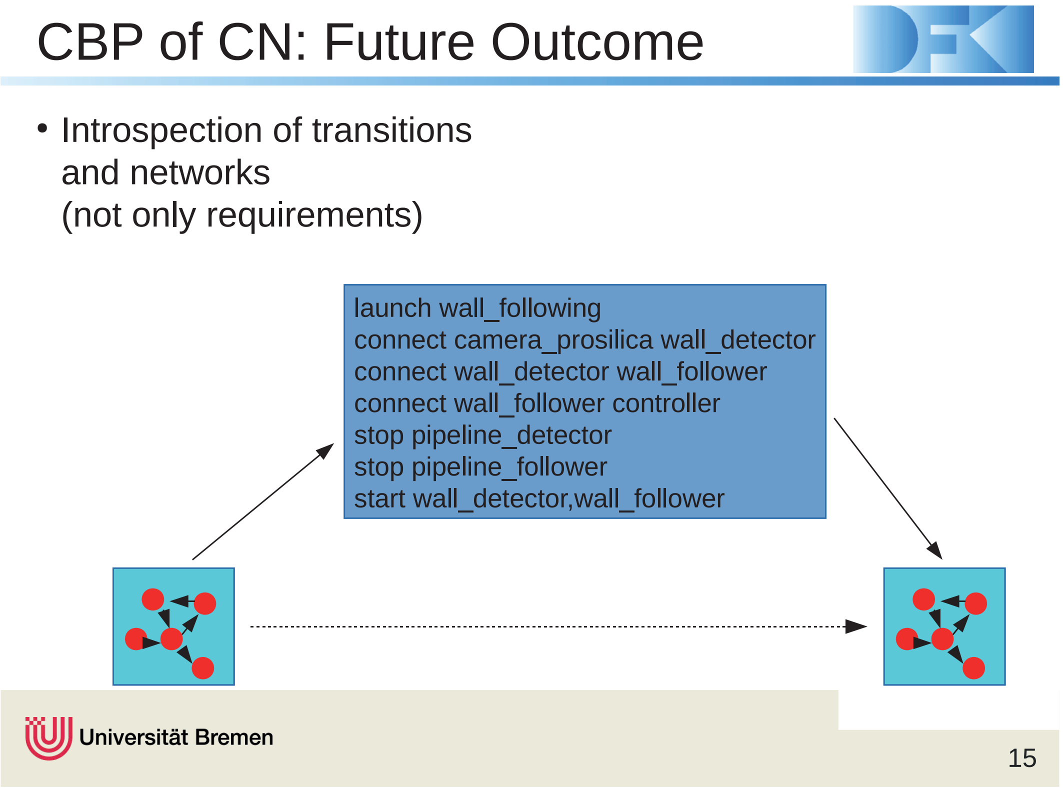
30

50

50

Universität Bremen

14



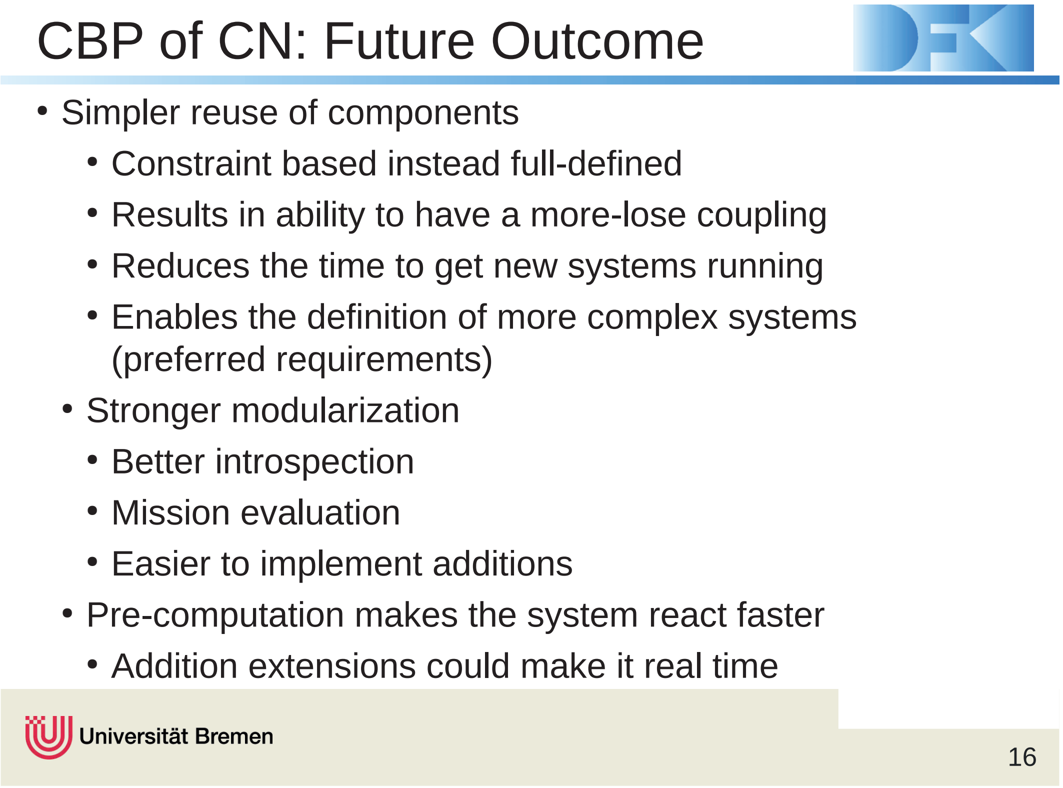
CBP of CN: Future Outcome

- Introspection of transitions and networks (not only requirements)

```
launch wall_following
connect camera_prosilica wall_detector
connect wall_detector wall_follower
connect wall_follower controller
stop pipeline_detector
stop pipeline_follower
start wall_detector,wall_follower
```

Universität Bremen

15

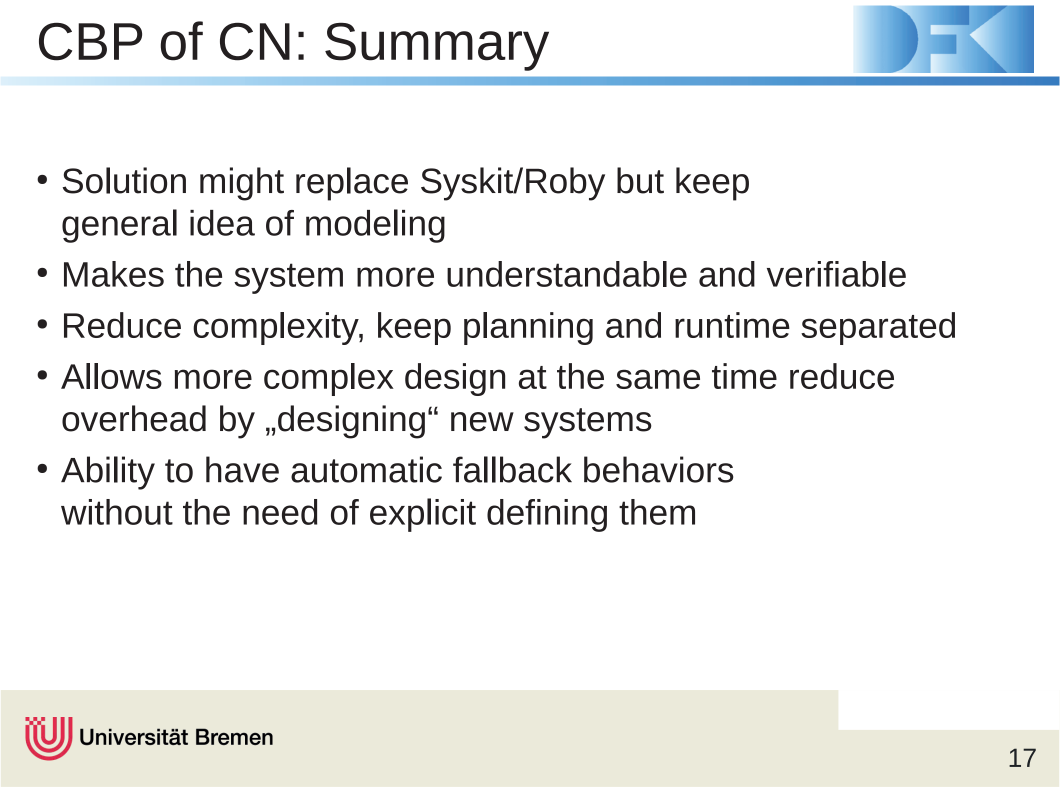


CBP of CN: Future Outcome

- Simpler reuse of components
  - Constraint based instead full-defined
  - Results in ability to have a more-lose coupling
  - Reduces the time to get new systems running
  - Enables the definition of more complex systems (preferred requirements)
- Stronger modularization
  - Better introspection
  - Mission evaluation
  - Easier to implement additions
- Pre-computation makes the system react faster
  - Addition extensions could make it real time

Universität Bremen

16

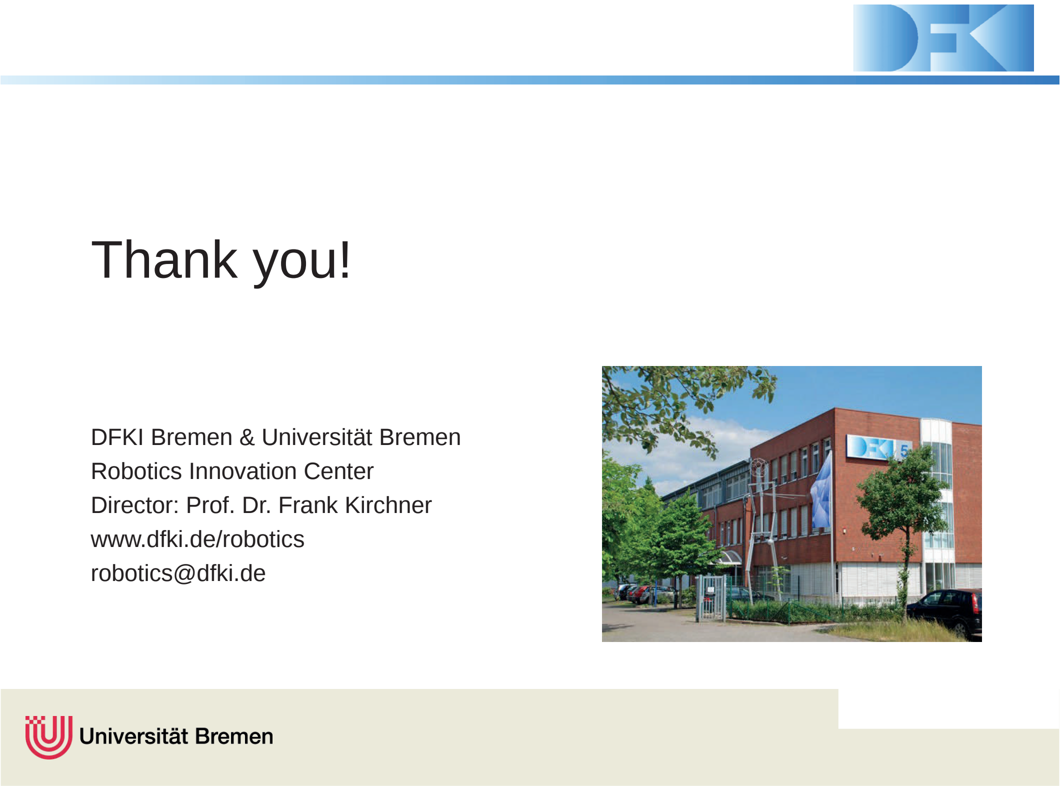


CBP of CN: Summary

- Solution might replace Syskit/Roby but keep general idea of modeling
- Makes the system more understandable and verifiable
- Reduce complexity, keep planning and runtime separated
- Allows more complex design at the same time reduce overhead by „designing“ new systems
- Ability to have automatic fallback behaviors without the need of explicit defining them

Universität Bremen

17



Thank you!

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

2.7 'Orocos CPP: A C++ client layer for RTT' (FW-T-07)

Janosch Machowinski[1]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `janosch.machowinski@dfki.de`

Abstract

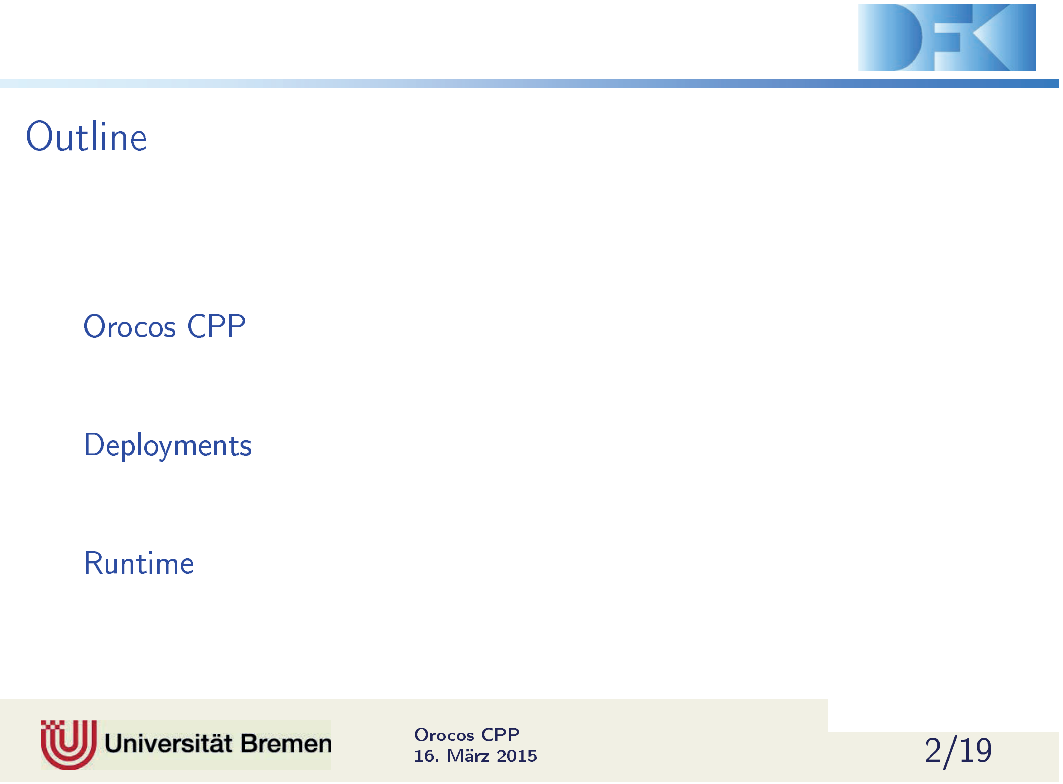
This presentation gives an overview of the Orocos CPP client library. The library allow direct access and management of oroGen-based components from C++. It intends to provide an alternative to the existing Ruby-Interpreter-based orocos.rb interface.

Orocos CPP: A C++ client layer for RTT

Janosch Machowinski

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen



Outline

Orocos CPP

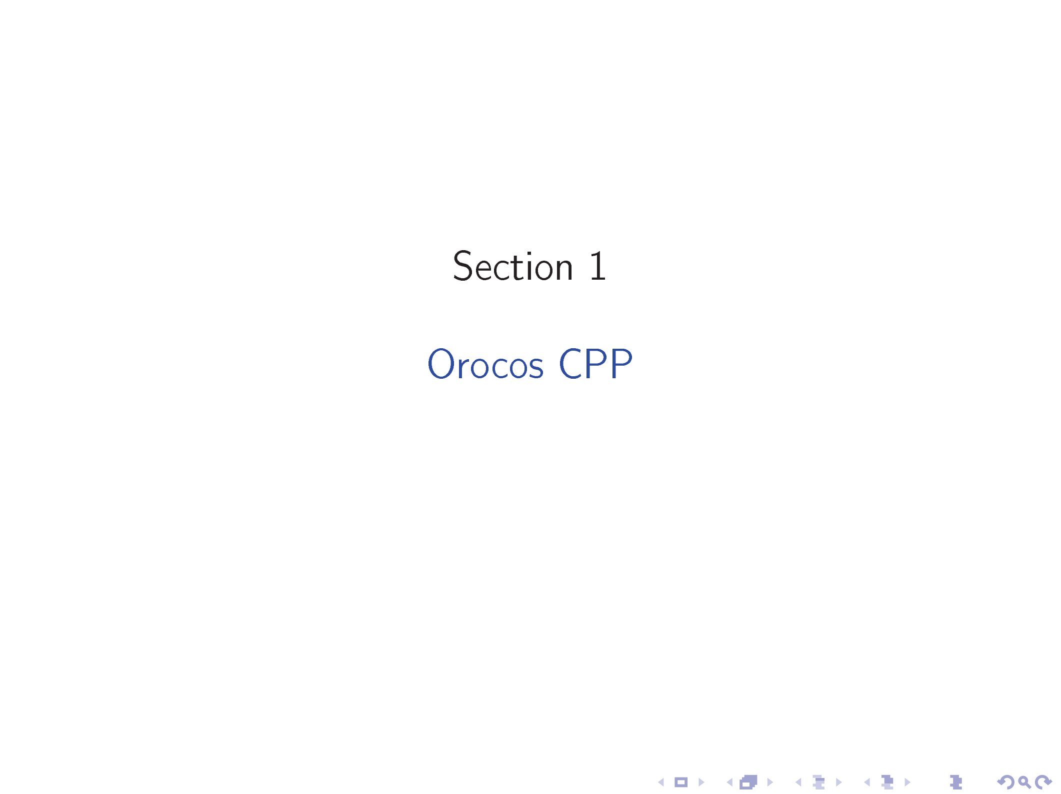
Deployments

Runtime

Universität Bremen

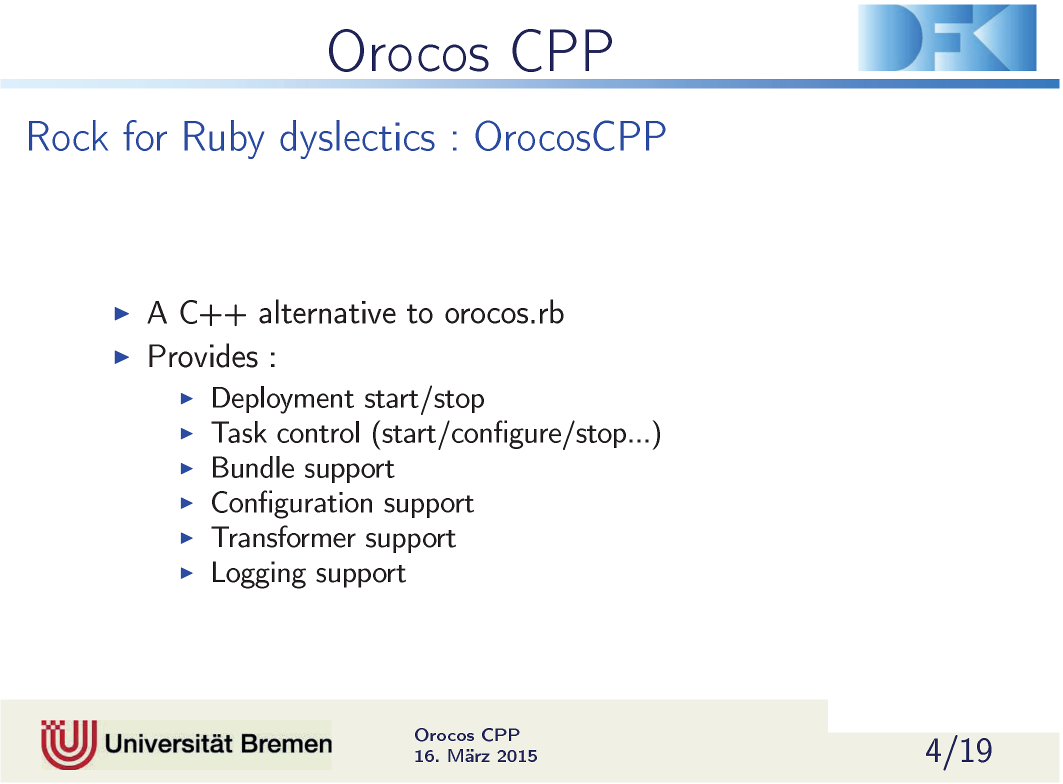
Orocos CPP
16. März 2015

2/19



Section 1

Orocos CPP



Orocos CPP

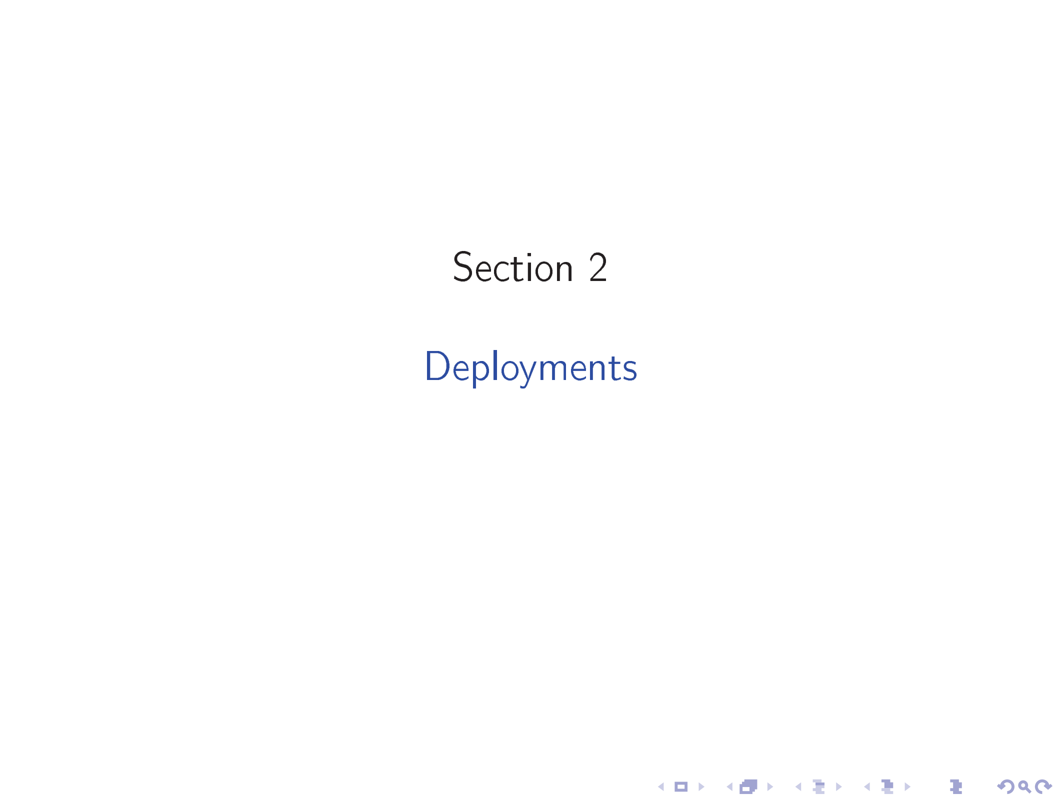
Rock for Ruby dyslectics : OrocosCPP

- A C++ alternative to orocos.rb
- Provides :
  - Deployment start/stop
  - Task control (start/configure/stop...)
  - Bundle support
  - Configuration support
  - Transformer support
  - Logging support

Universität Bremen

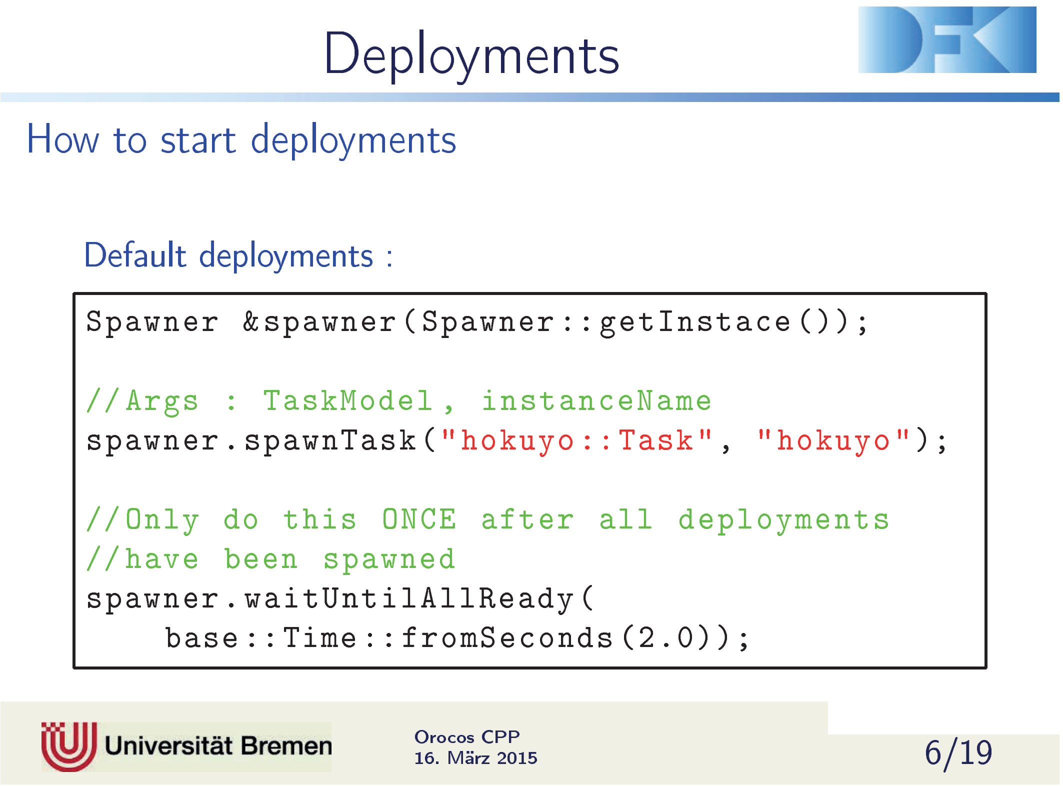
Orocos CPP
16. März 2015

4/19



Section 2

Deployments



Deployments

How to start deployments

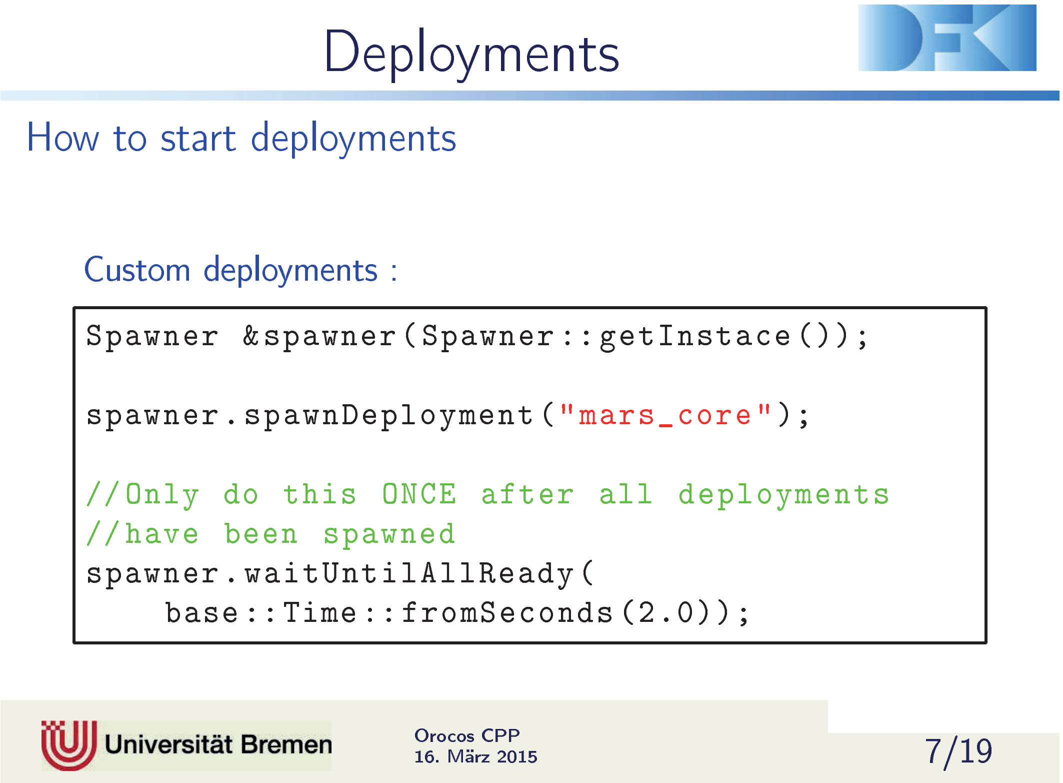
Default deployments :

```
Spawner &spawner(Spawner::getInstace());

//Args : TaskModel, instanceName
spawner.spawnTask("hokuyo::Task", "hokuyo");

//Only do this ONCE after all deployments
//have been spawned
spawner.waitUntilAllReady(
    base::Time::fromSeconds(2.0));
```

Universität Bremen

Orocos CPP
16. März 2015

6/19



Deployments

How to start deployments

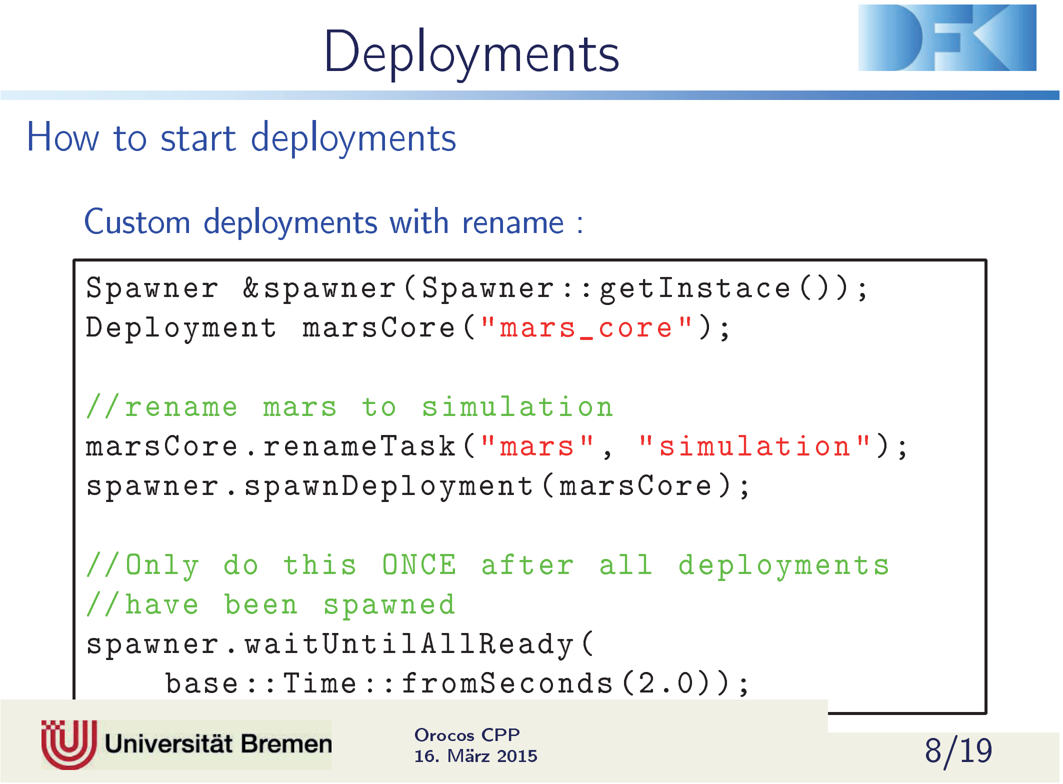
Custom deployments :

```
Spawner &spawner(Spawner::getInstace());

spawner.spawnDeployment("mars_core");

//Only do this ONCE after all deployments
//have been spawned
spawner.waitUntilAllReady(
    base::Time::fromSeconds(2.0));
```

Universität Bremen

Orocos CPP
16. März 2015

7/19



Deployments

How to start deployments

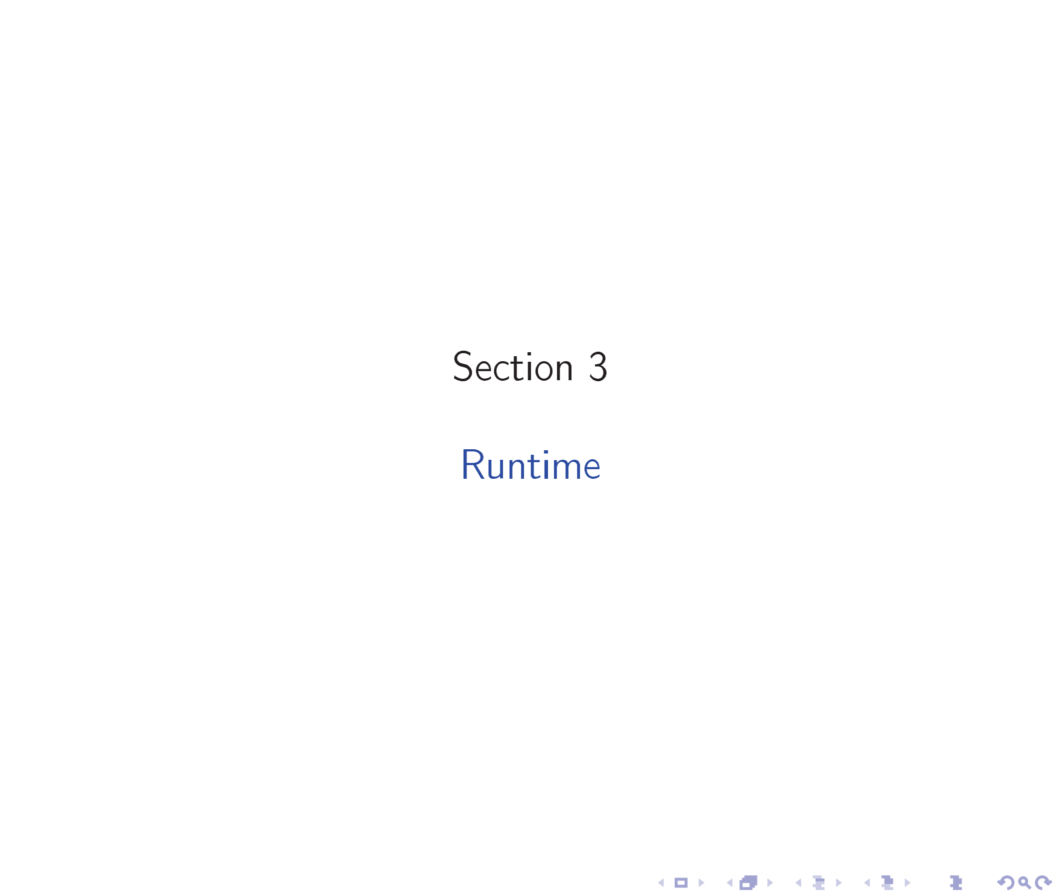
Custom deployments with rename :

```
Spawner &spawner(Spawner::getInstace());
Deployment marsCore("mars_core");

//rename mars to simulation
marsCore.renameTask("mars", "simulation");
spawner.spawnDeployment(marsCore);

//Only do this ONCE after all deployments
//have been spawned
spawner.waitUntilAllReady(
    base::Time::fromSeconds(2.0));
```

Universität Bremen

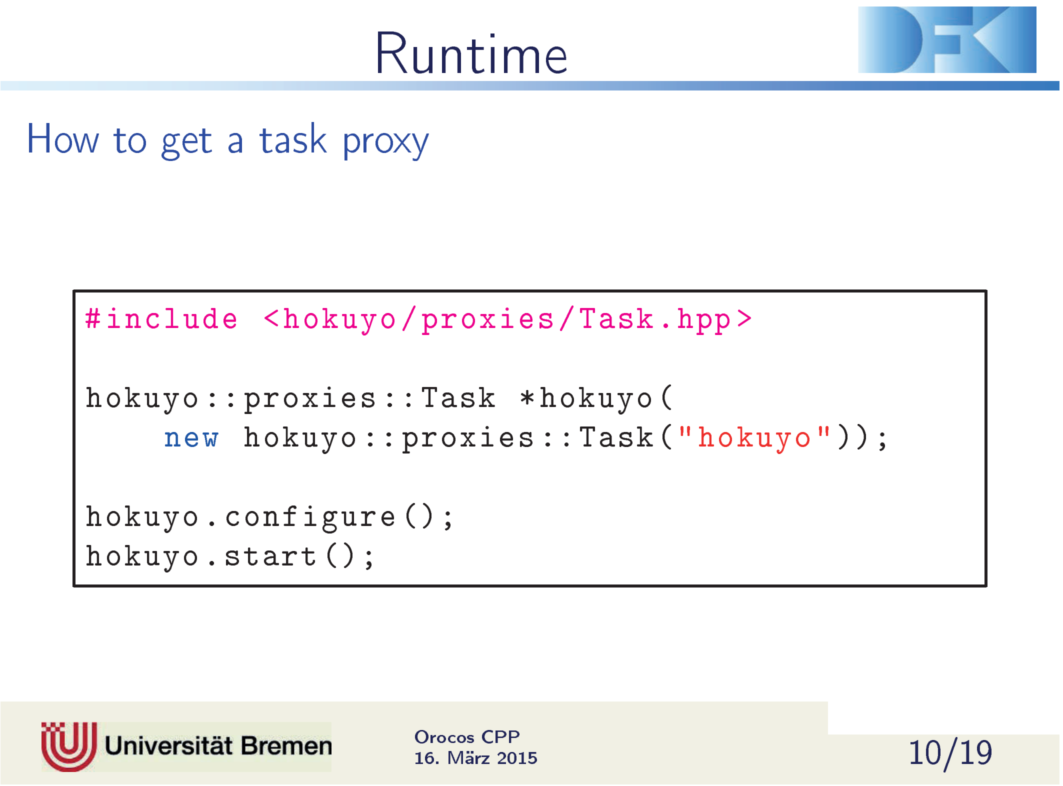
Orocos CPP
16. März 2015

8/19



Section 3

Runtime



Runtime

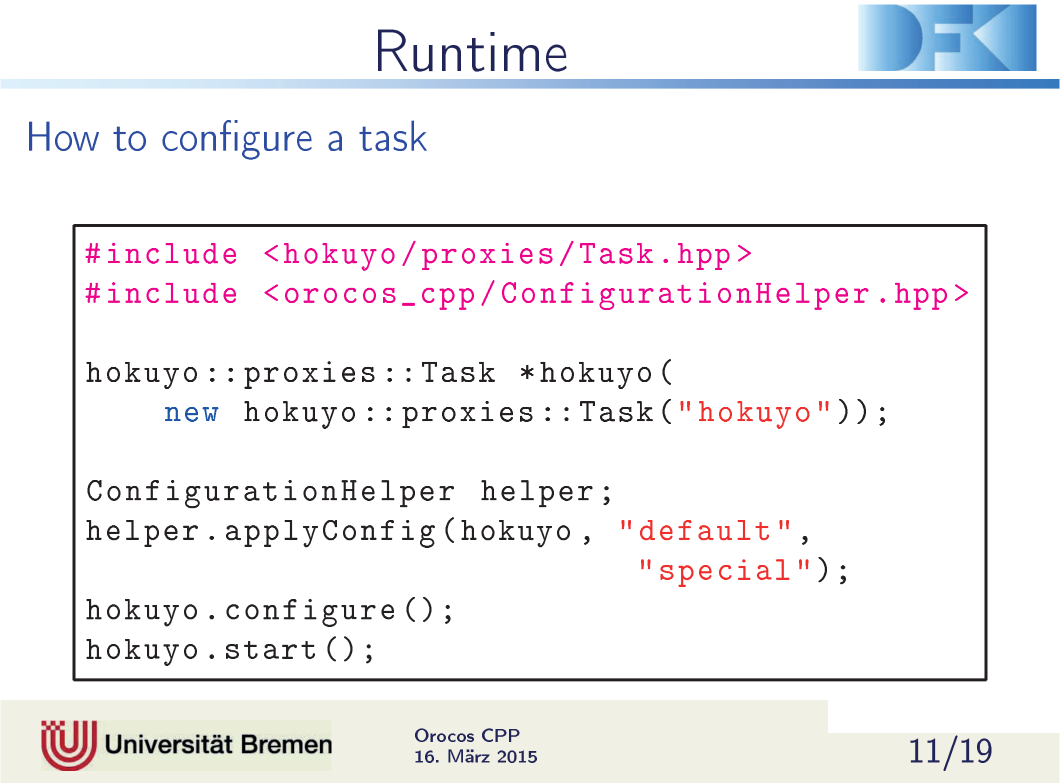
How to get a task proxy

```
#include <hokuyo/proxies/Task.hpp>

hokuyo::proxies::Task *hokuyo(
    new hokuyo::proxies::Task("hokuyo"));

hokuyo.configure();
hokuyo.start();
```

Universität Bremen

Orocos CPP
16. März 2015

10/19



Runtime

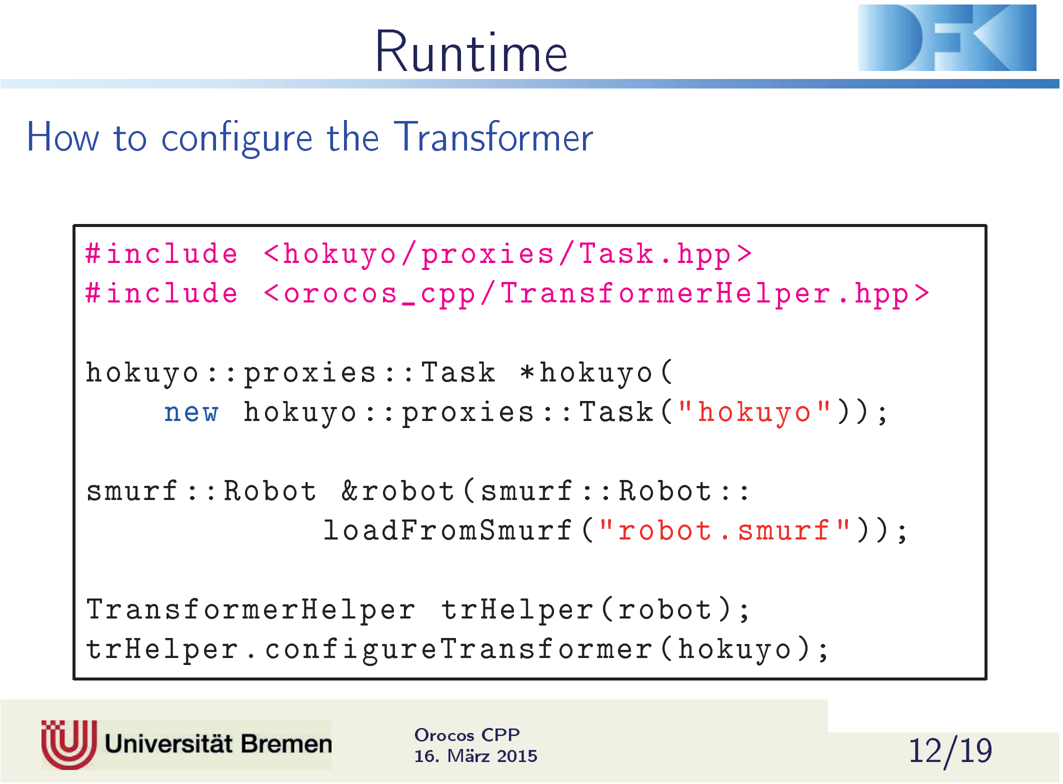
How to configure a task

```
#include <hokuyo/proxies/Task.hpp>
#include <orocos_cpp/ConfigurationHelper.hpp>

hokuyo::proxies::Task *hokuyo(
    new hokuyo::proxies::Task("hokuyo"));

ConfigurationHelper helper;
helper.applyConfig(hokuyo, "default",
                            "special");
hokuyo.configure();
hokuyo.start();
```

Universität Bremen

Orocos CPP
16. März 2015

11/19



Runtime

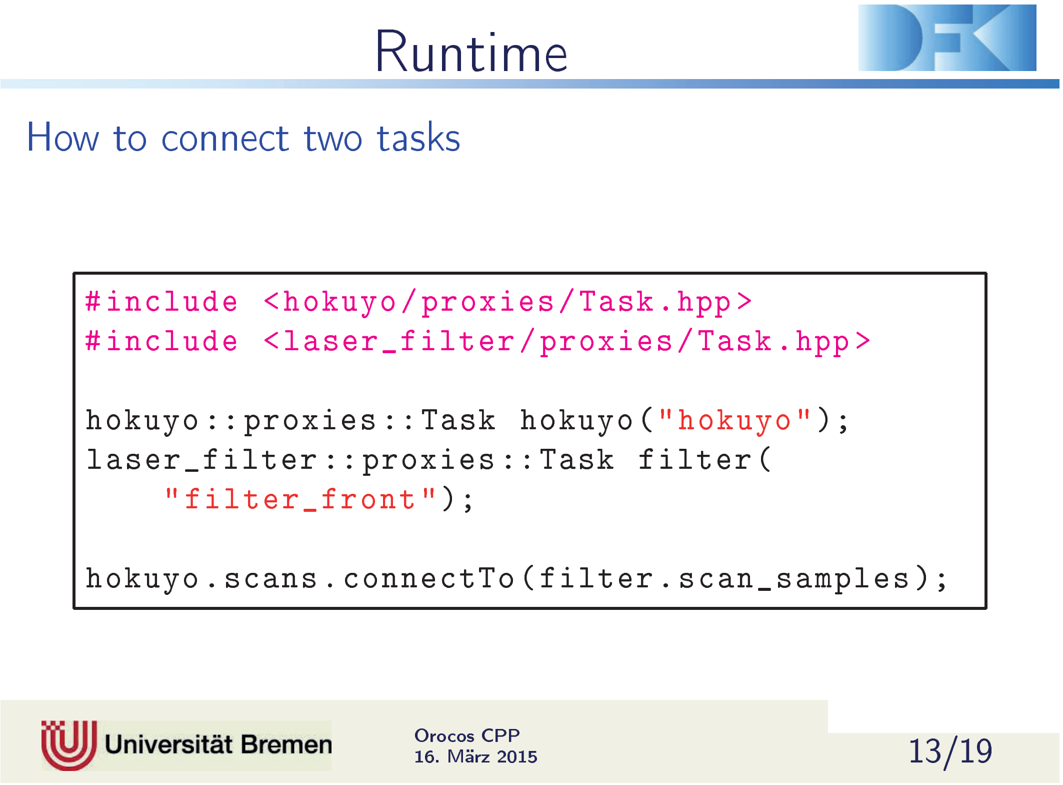
How to configure the Transformer

```
#include <hokuyo/proxies/Task.hpp>
#include <orocos_cpp/TransformerHelper.hpp>

hokuyo::proxies::Task *hokuyo(
    new hokuyo::proxies::Task("hokuyo"));

smurf::Robot &robot(smurf::Robot::
            loadFromSmurf("robot.smurf"));

TransformerHelper trHelper(robot);
trHelper.configureTransformer(hokuyo);
```

Universität Bremen

Orocos CPP
16. März 2015

12/19



Runtime

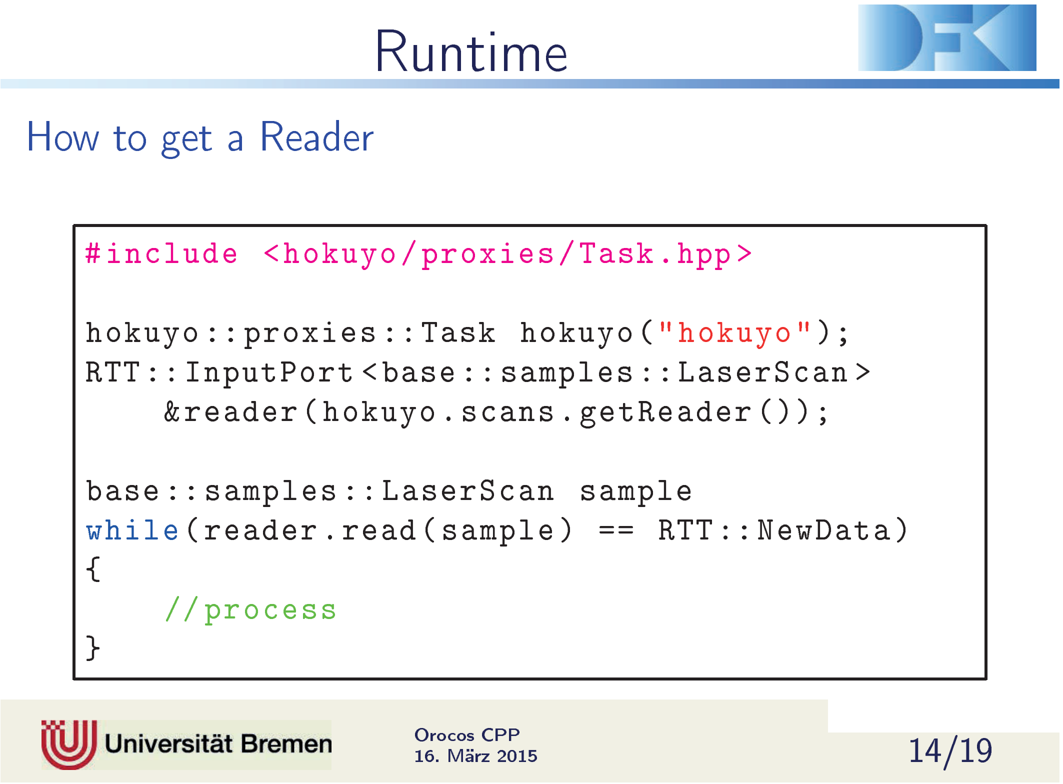
How to connect two tasks

```
#include <hokuyo/proxies/Task.hpp>
#include <laser_filter/proxies/Task.hpp>

hokuyo::proxies::Task hokuyo("hokuyo");
laser_filter::proxies::Task filter(
    "filter_front");

hokuyo.scans.connectTo(filter.scan_samples);
```

Universität Bremen

Orocos CPP
16. März 2015

13/19



Runtime

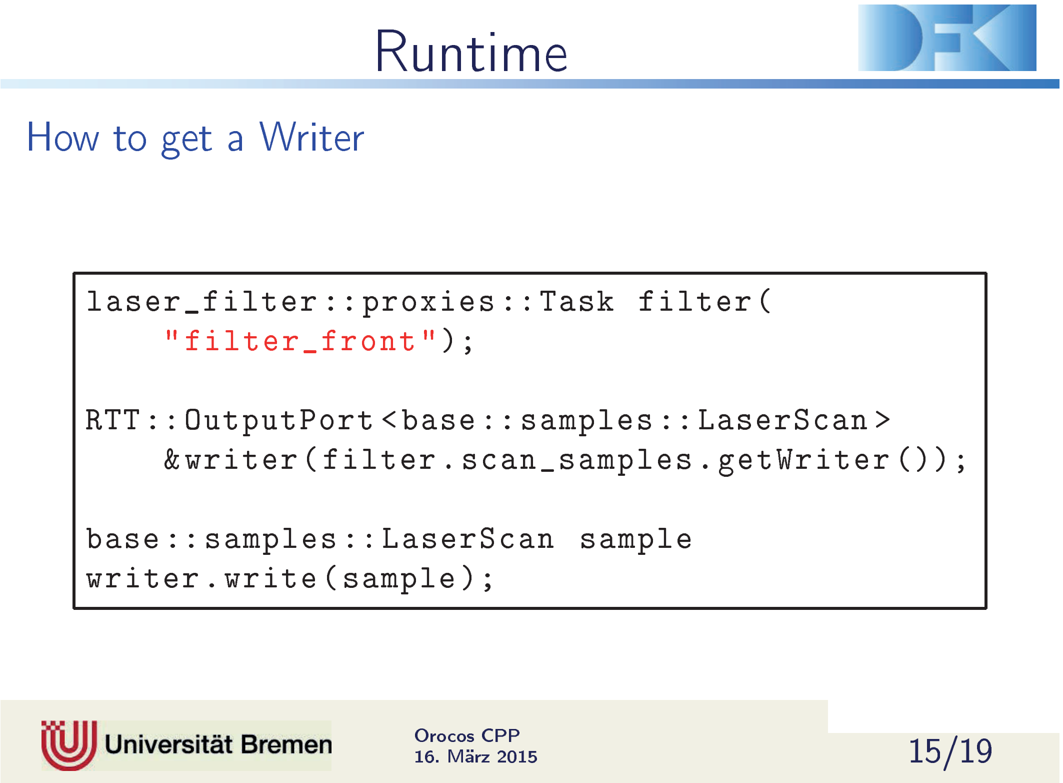
How to get a Reader

```
#include <hokuyo/proxies/Task.hpp>

hokuyo::proxies::Task hokuyo("hokuyo");
RTT::InputPort<base::samples::LaserScan>
    &reader(hokuyo.scans.getReader());

base::samples::LaserScan sample
while(reader.read(sample) == RTT::NewData)
{
    //process
}
```

Universität Bremen

Orocos CPP
16. März 2015

14/19



Runtime

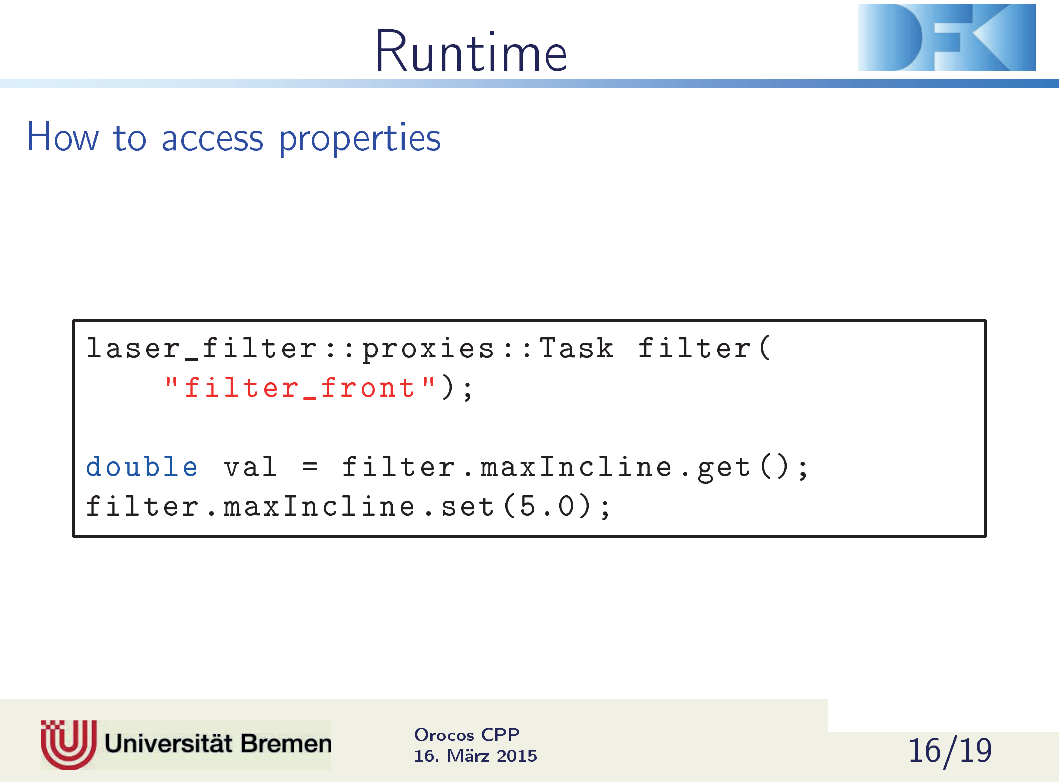
How to get a Writer

```
laser_filter::proxies::Task filter(
    "filter_front");

RTT::OutputPort<base::samples::LaserScan>
    &writer(filter.scan_samples.getWriter());

base::samples::LaserScan sample
writer.write(sample);
```

Universität Bremen

Orocos CPP
16. März 2015

15/19



Runtime

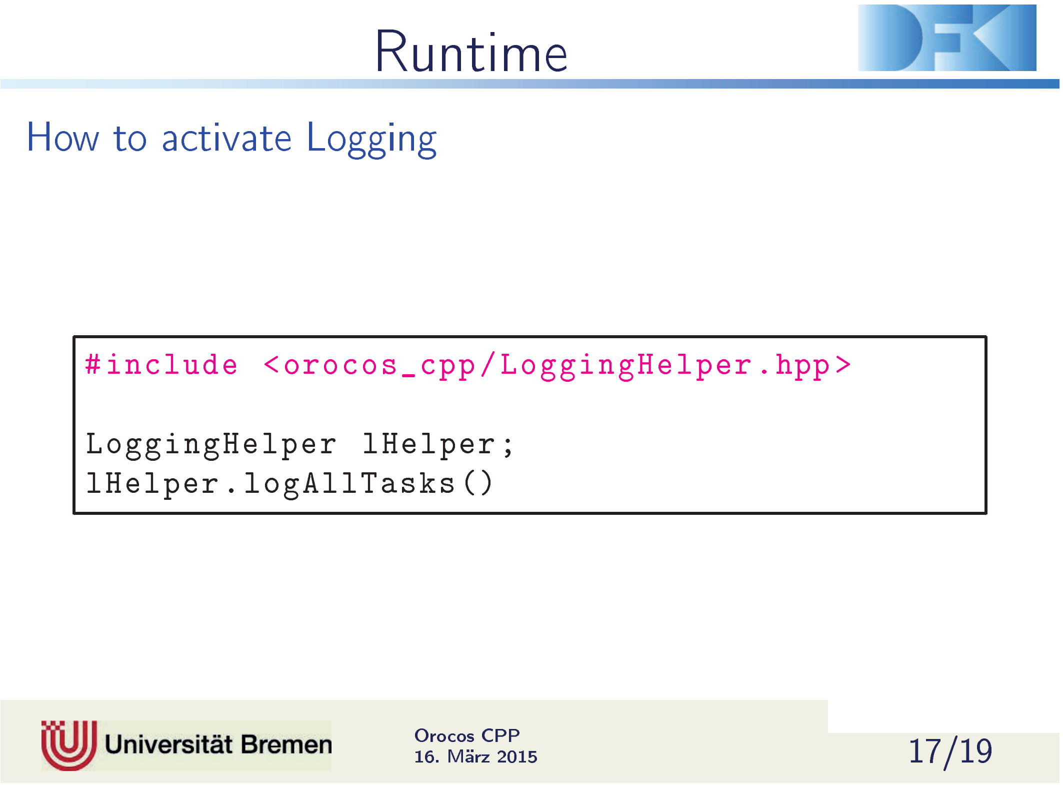
How to access properties

```
laser_filter::proxies::Task filter(
    "filter_front");

double val = filter.maxIncline.get();
filter.maxIncline.set(5.0);
```

Universität Bremen

Orocos CPP
16. März 2015

16/19



Runtime

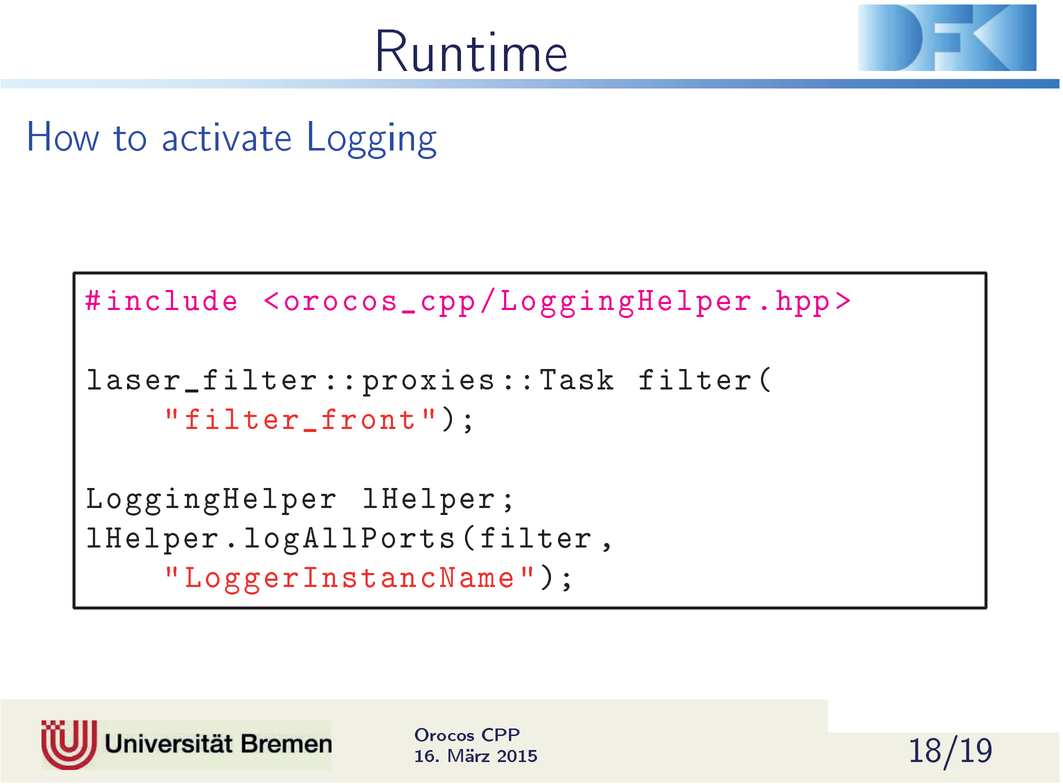
How to activate Logging

```
#include <orocos_cpp/LoggingHelper.hpp>

LoggingHelper lHelper;
lHelper.logAllTasks()
```

Universität Bremen

Orocos CPP
16. März 2015

17/19



Runtime

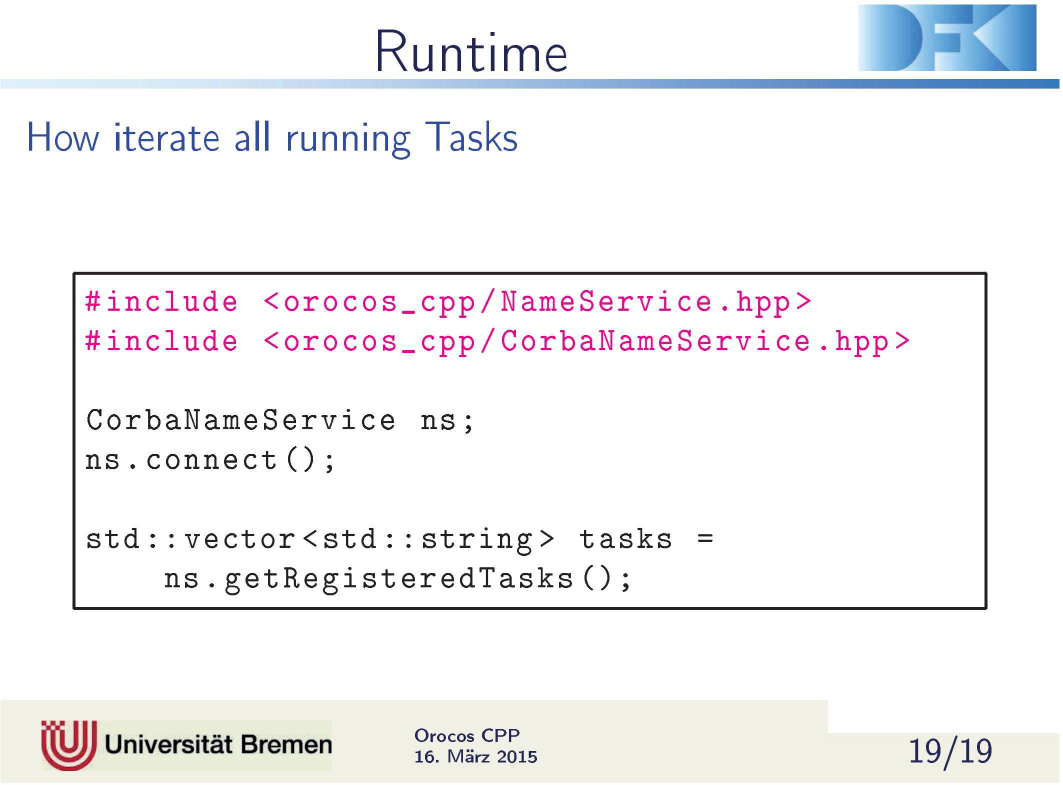
How to activate Logging

```
#include <orocos_cpp/LoggingHelper.hpp>

laser_filter::proxies::Task filter(
    "filter_front");

LoggingHelper lHelper;
lHelper.logAllPorts(filter,
    "LoggerInstancName");
```

Universität Bremen

Orocos CPP
16. März 2015

18/19



Runtime

How iterate all running Tasks

```
#include <orocos_cpp/NameService.hpp>
#include <orocos_cpp/CorbaNameService.hpp>

CorbaNameService ns;
ns.connect();

std::vector<std::string> tasks =
    ns.getRegisteredTasks();
```

Universität Bremen

Orocos CPP
16. März 2015

19/19

2.8 'A framework for describing manipulation behavior' (FW-T-08)

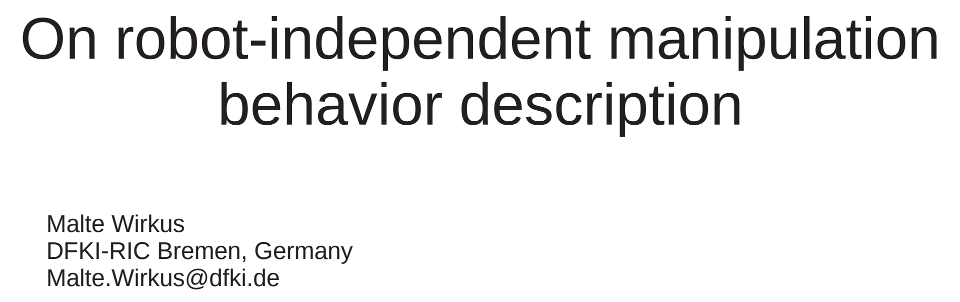
Malte Wirkus[(1)]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: malte.wirkus@dfki.de

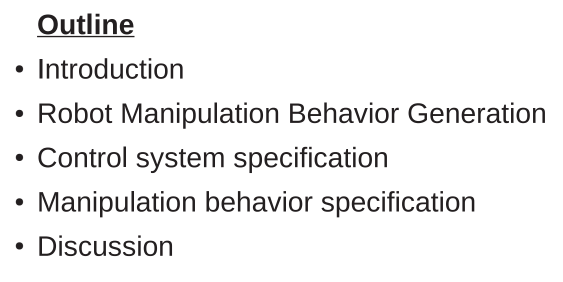
Abstract

In this talk a framework to design and control robot manipulation behavior is presented. To remain independent from particular robot hardware and an explicit area of application, an embedded domain specific language (eDSL) is applied to describe the particular robot and a controller network that drives the robot. We make use of a) a component-based software framework (Rock), b) model-based algorithms for motion- and sensor processing representations, c) an abstract model of the control system, and d) a plan management software, to describe a sequence of software component networks that generate the desired robot behavior.



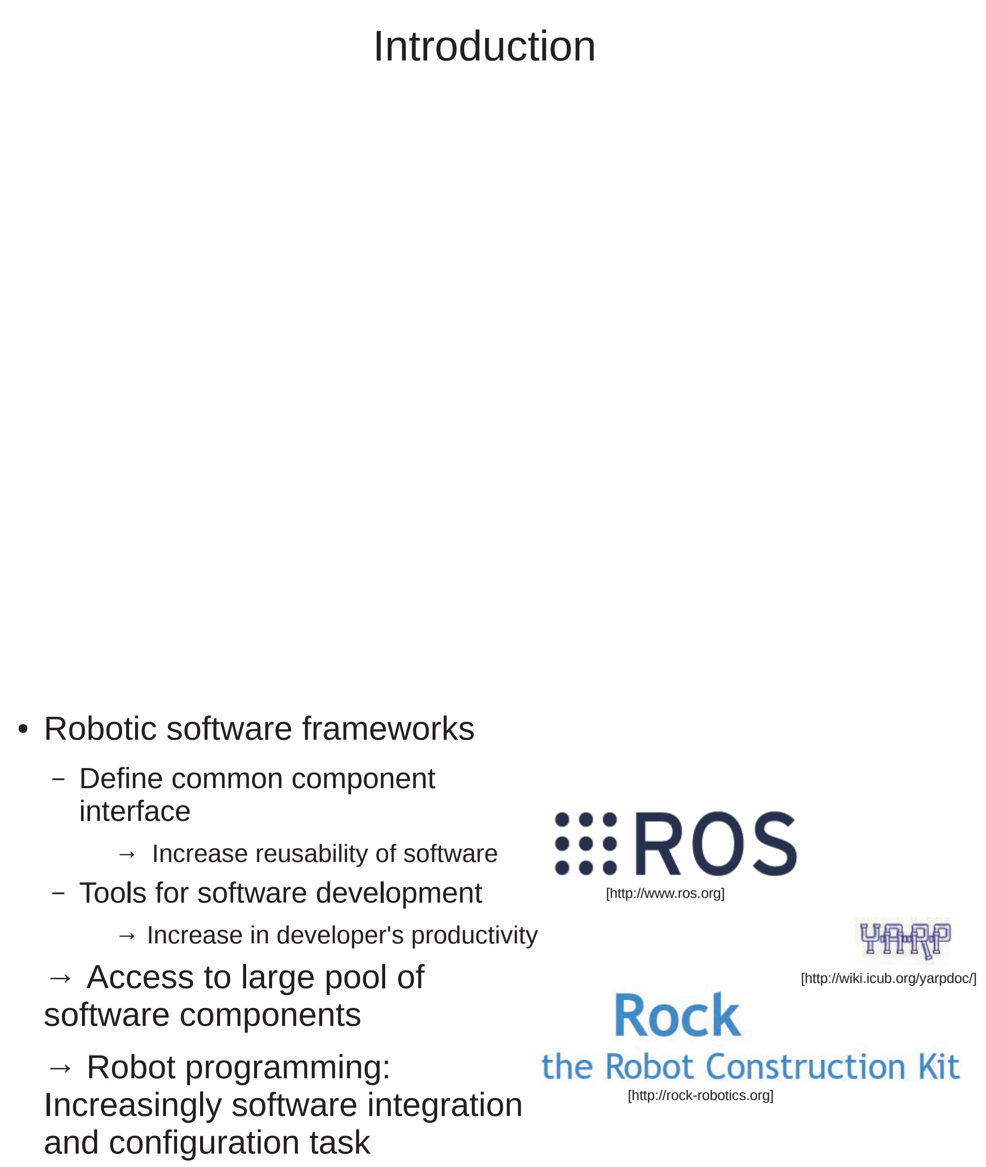
On robot-independent manipulation behavior description

Malte Wirkus
DFKI-RIC Bremen, Germany
Malte.Wirkus@dfki.de



Outline

- Introduction
- Robot Manipulation Behavior Generation
- Control system specification
- Manipulation behavior specification
- Discussion



Introduction

- Robotic software frameworks
  - Define common component interface
    - → Increase reusability of software
  - Tools for software development
    - → Increase in developer's productivity

→ Access to large pool of software components

→ Robot programming: Increasingly software integration and configuration task

ROS
YARP
Rock
the Robot Construction Kit

[http://www.ros.org]

[http://wiki.icub.org/yarpdoc/]

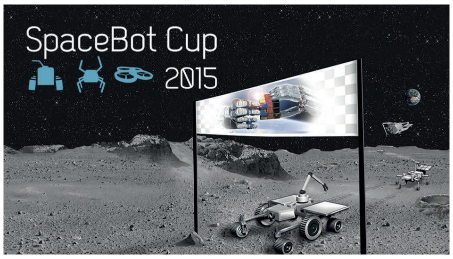
[http://rock-robotics.org]

- Expectations on robots increase
  - More complex tasks
  - Complex missions
    - Different modes of operation / behaviors

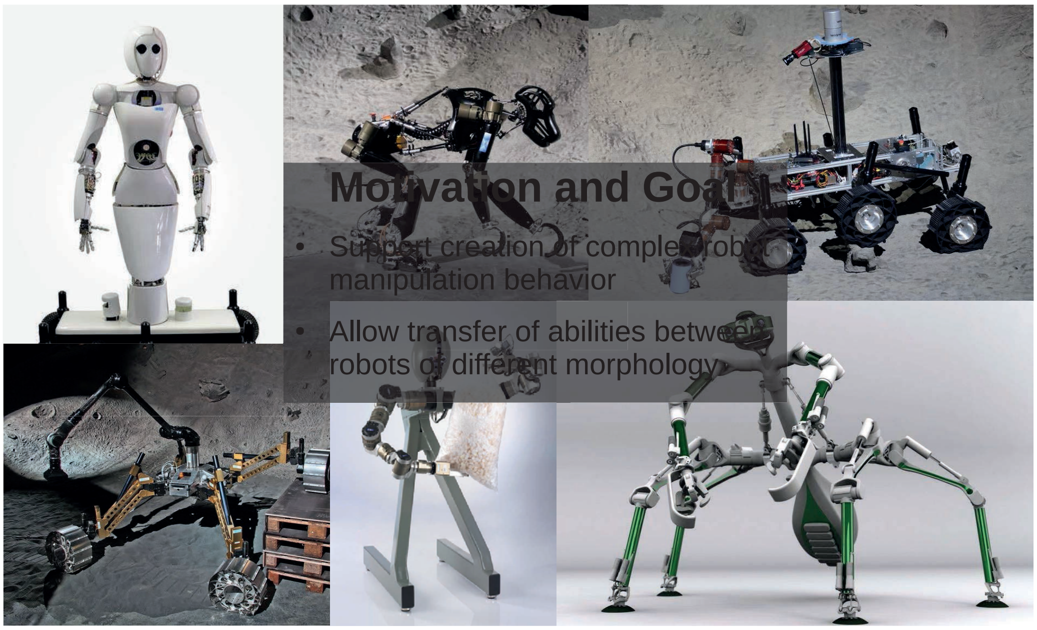
“Robot programming increasingly becomes a software integration and configuration task”

→ .. but it's still complex

[DARPA Robotics Grand Challenge]



[DLR SpaceBot Cup]

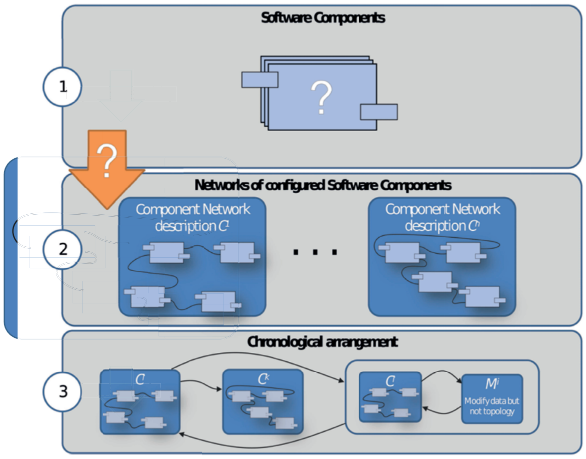


Motivation and Goals

- Support creation of complex robot manipulation behavior
- Allow transfer of abilities between robots of different morphology

[all Images: http://robotik.dfki-bremen.de]

Robot Manipulation Behavior Generation

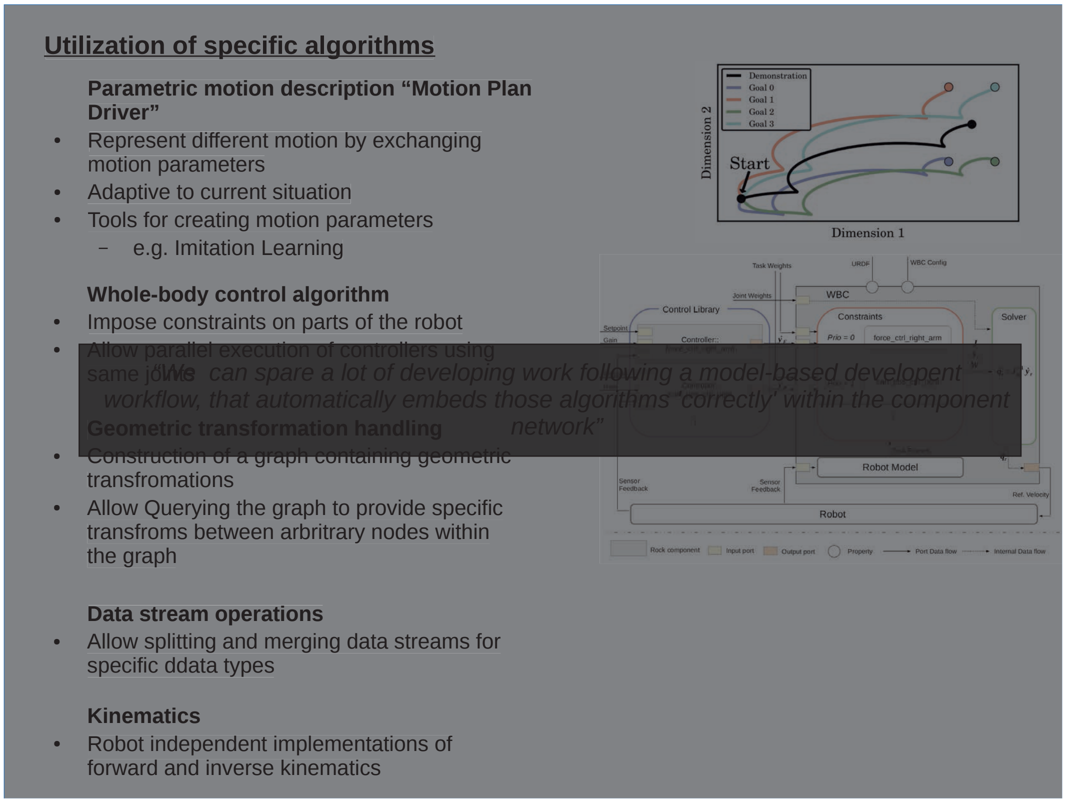

Software Components
1
?
?
Networks of configured Software Components
2
Component Network description C^1
. . .
Component Network description C^n
Chronological arrangement
3
C^j
C^k
C^l
M^i
Modify data but not topology

Contribution

Workflow for robot manipulation behavior design

- easy to work with
- supports transfer of behavior

- Utilization of specific algorithms
- Data processing for control
- eDSL to support development



Utilization of specific algorithms

Parametric motion description “Motion Plan Driver”

- Represent different motion by exchanging motion parameters
- Adaptive to current situation
- Tools for creating motion parameters
  - e.g. Imitation Learning

Whole-body control algorithm

- Impose constraints on parts of the robot
- Allow parallel execution of controllers using same joints

“We can spare a lot of developing work following a model-based developent workflow, that automatically embeds those algorithms ‘correctly’ within the component network”

Geometric transformation handling

- Construction of a graph containing geometric transfromations
- Allow Querying the graph to provide specific transfroms between arbitrary nodes within the graph

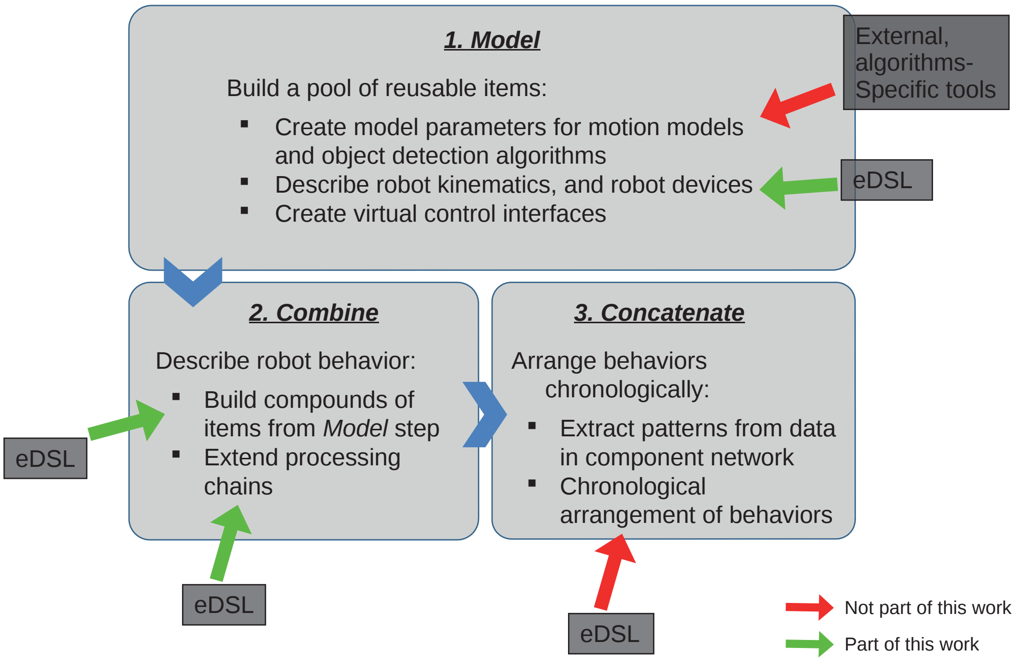
Data stream operations

- Allow splitting and merging data streams for specific ddata types

Kinematics

- Robot independent implementations of forward and inverse kinematics

The development workflow



1. Model

Build a pool of reusable items:

- Create model parameters for motion models and object detection algorithms
- Describe robot kinematics, and robot devices
- Create virtual control interfaces

2. Combine

Describe robot behavior:

- Build compounds of items from *Model* step
- Extend processing chains

3. Concatenate

Arrange behaviors chronologically:

- Extract patterns from data in component network
- Chronological arrangement of behaviors

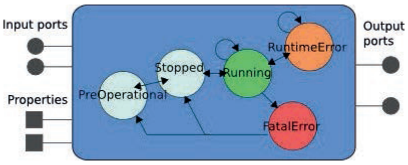
Control Network Specification

Rock

the Robot Construction Kit

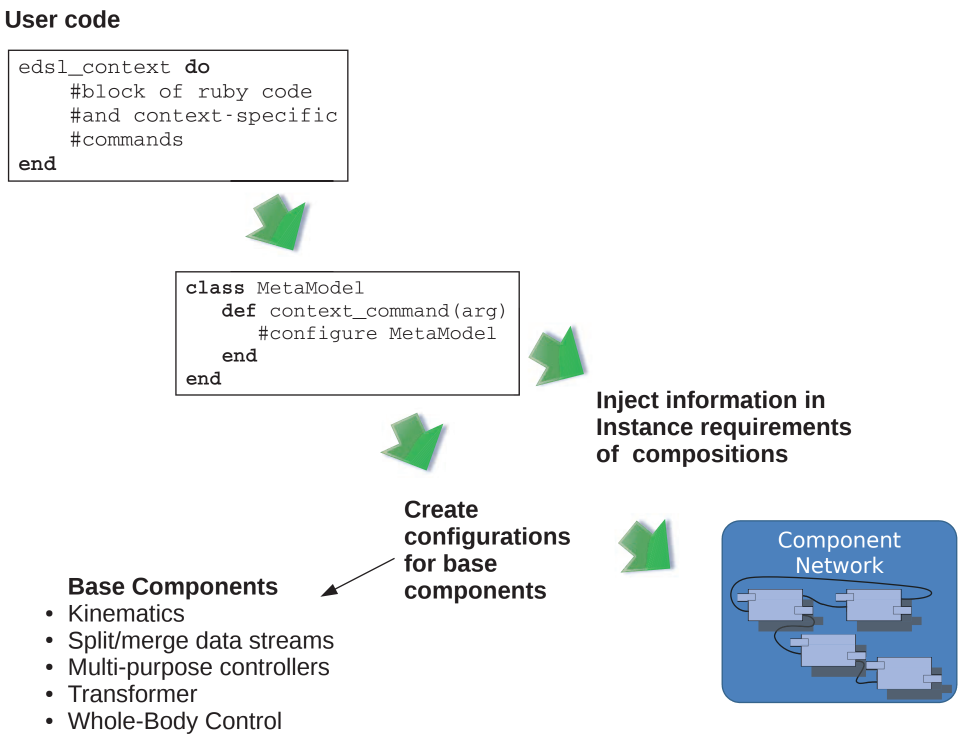
[http://rock-robotics.org]

- **Component model**
  - Orocos RTT
  - Configuration interface
  - Data flow interface
  - Life-cycle
  - Single-purpose



[http://rock-robotics.org]

- **System modeling**
  - Data Service: Semantic labels → abstract data flow interface
  - Compositions: Functional subnetworks of actual components, Data Services, already modeled subnetworks
  - Instantiation requirements: Selection of actual components for Data Services. Choosing of configurations for component.
- **Plan management**
  - Represent and execute plans ("missions")
  - Component network models can be used as tasks
    - Component network instantiation
    - Supervision



User code

```
edsl_context do
    #block of ruby code
    #and context-specific
    #commands
end
```

```
class MetaModel
   def context_command(arg)
      #configure MetaModel
   end
end
```

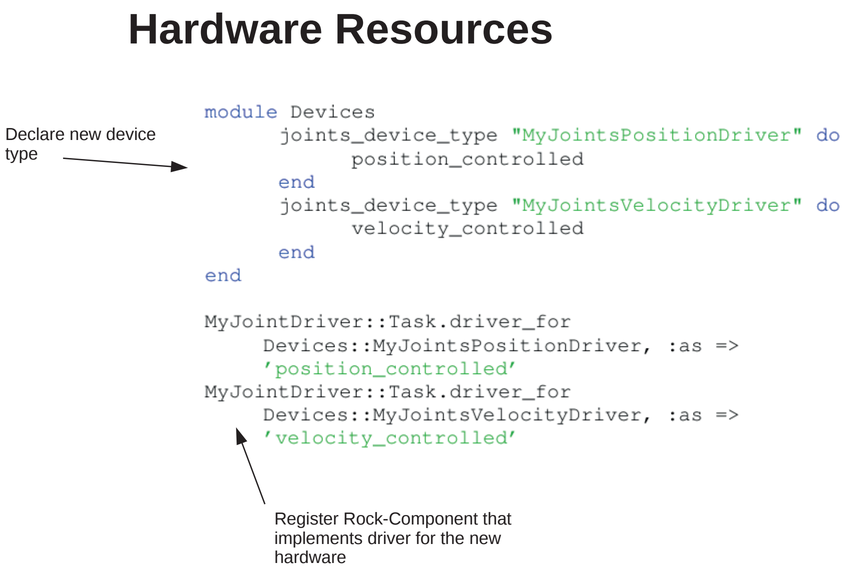
Inject information in
Instance requirements
of compositions

Create
configurations
for base
components

Component
Network

Base Components

- Kinematics
- Split/merge data streams
- Multi-purpose controllers
- Transformer
- Whole-Body Control



Hardware Resources

Declare new device type

```
module Devices
      joints_device_type "MyJointsPositionDriver" do
            position_controlled
      end
      joints_device_type "MyJointsVelocityDriver" do
            velocity_controlled
      end
end

MyJointDriver::Task.driver_for
     Devices::MyJointsPositionDriver, :as =>
     'position_controlled'
MyJointDriver::Task.driver_for
     Devices::MyJointsVelocityDriver, :as =>
     'velocity_controlled'
```

Register Rock-Component that implements driver for the new hardware

Robot

```
robot do
      kinematic_description
           "/path/to/my/kinematic_description.urdf"
      device(Devices::JointsPositionDriver, :as =>
           ’armr’).joint_names(’ar’, ’br’,
           ’cr’).with_conf(’armr’)
      device(Devices::JointsPositionDriver, :as =>
           ’arml’).joint_names(’al’, ’bl’,
           ’cl’).with_conf(’arml’)
      device(Devices::JointsPositionDriver, :as =>
           ’hr’).joint_names(’wr’,’gr’).with_conf(’hr’)
      device(Devices::JointsPositionDriver, :as =>
           ’hl’).joint_names(’wl’,’gl’).with_conf(’hl’)
      device(Devices::JointsVelocityDriver, :as =>
           ’head’).joint_names(’p’, ’t’).
           with_conf(’head’)
end
```

Provide kinematic description.

Compose robot of device models

Relate devices to the robot's body by unique names for joints and structure parts

Load specific config for Driver.
Give additional information.

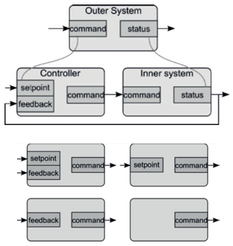
Control Networks

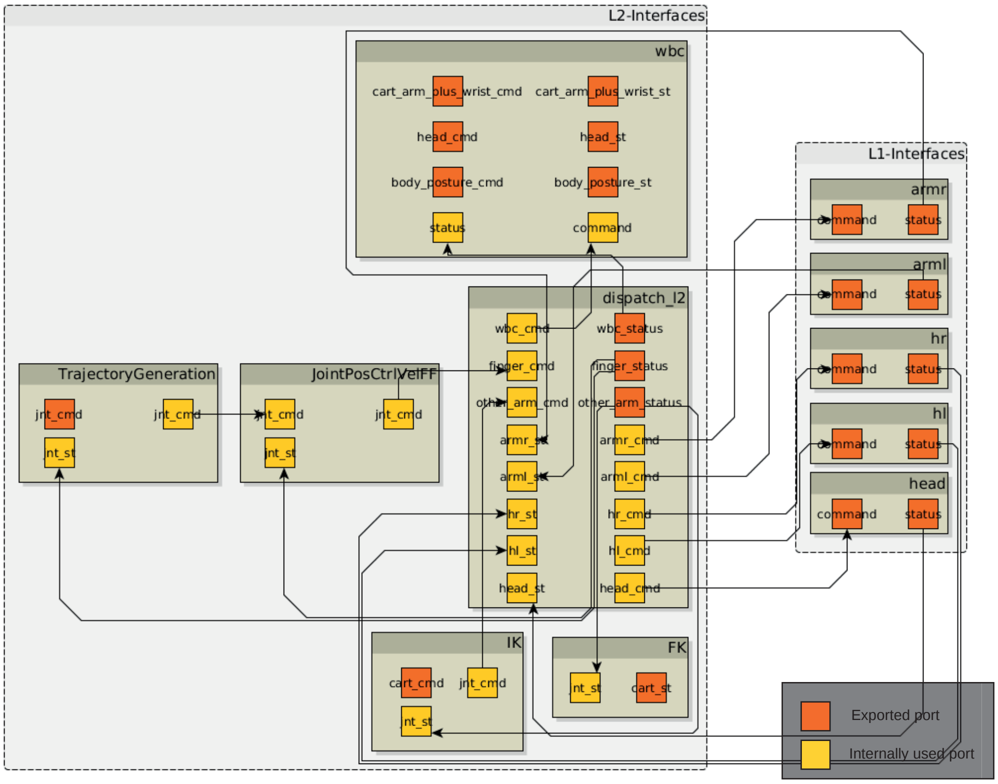
```
control_collection "l2" do
      used_joints = [’ar’,’br’,’cr’,’wr’,’p’,’t’]
      wbc_interface used_joints, :as => "wbc",
           :initial_joint_weights => [1]*used_joints.size
           do
            cartesian_control_interface [’O’,’WR’],
                  :as => "cart_arm_plus_wrist",
                  :joint_names => [’ar’,’br’,’cr’,’wr’],
                  :priority => 1, :weights => [1,1,1,0.5]

            control_interface [’p’,’t’], :as => "head",
                 :priority => 2

            control_interface [’ar’,’br’,’cr’,’wr’],
                  :as => "body_posture", :priority => 3
      end

      control_interface [’gr’],
            :control_mode => :position,
            :as => ’finger’
      cartesian_control_interface ’O’, ’WL’,
            :joint_names => [’al’,’bl’,’cl’,’wl’],
            :control_mode => :velocity,
            :as => ’other_arm’
      end

      cascade_control finger_interface do
            push TrajectoryGeneration::Task
                        .with_conf(’arm_with_hand’)
      end
end
```

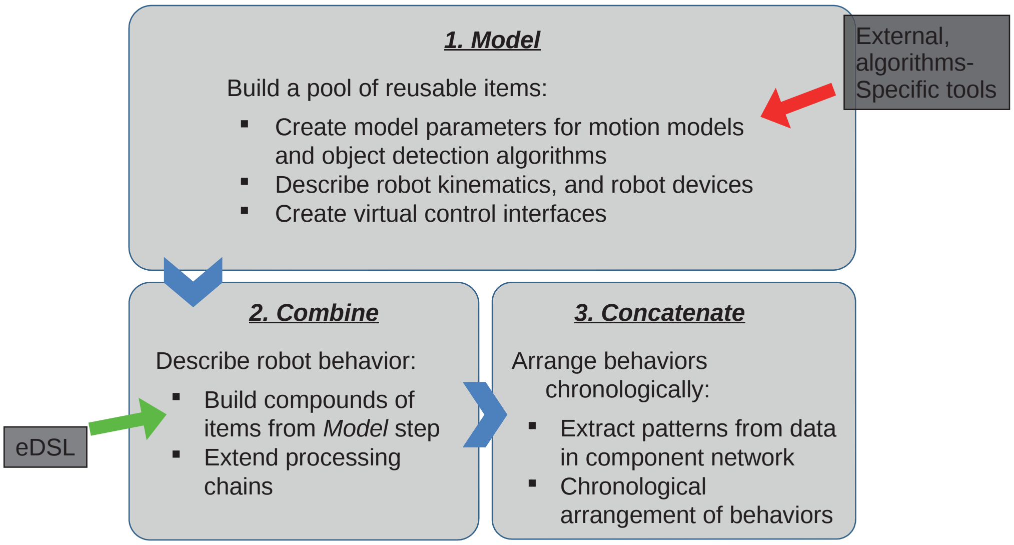
One stage in multi-stage control network

Different control interface types

Extend control interfaces with control chains






1. Model

Build a pool of reusable items:

- Create model parameters for motion models and object detection algorithms
- Describe robot kinematics, and robot devices
- Create virtual control interfaces

External, algorithms-Specific tools

2. Combine

Describe robot behavior:

- Build compounds of items from *Model* step
- Extend processing chains

eDSL

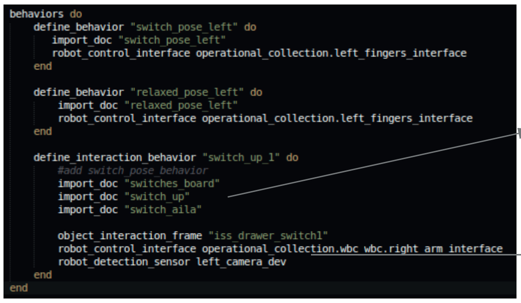
3. Concatenate

Arrange behaviors chronologically:

- Extract patterns from data in component network
- Chronological arrangement of behaviors

Information required to describe an cartesian interaction

- Motion plan driver task
- Object structure representation (urdf)
- Object detector task
- Name of object pose reconstruction frame
- Name of object interaction frame
- Action-specific transform applied to interaction frame
- Sensor to use for detection
- Cartesian control interface
- Action specific transform applied to tip of control chain



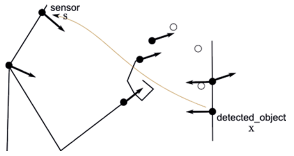
```
behaviors do
    define_behavior "switch_pose_left" do
       import_doc "switch_pose_left"
       robot_control_interface operational_collection.left_fingers_interface
    end

    define_behavior "relaxed_pose_left" do
        import_doc "relaxed_pose_left"
        robot_control_interface operational_collection.left_fingers_interface
    end

    define_interaction_behavior "switch_up_1" do
        #add switch pose behavior
        import_doc "switches_board"
        import_doc "switch_up"
        import_doc "switch_aila"

        object_interaction_frame "iss_drawer_switch1"
        robot_control_interface operational_collection.wbc_wbc.right_arm_interface
        robot_detection_sensor left_camera_dev
    end
end
```

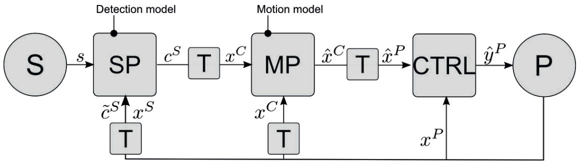
include from yaml,
for interfacing with external tools

Specify information via eDSL

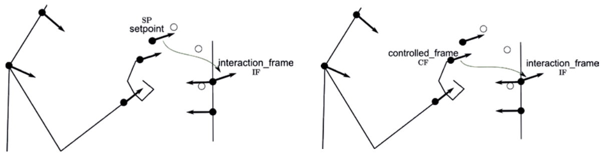
Data processing for Cartesian control

Perception



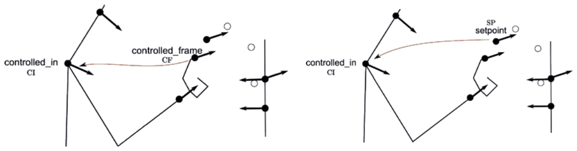


Motion Plan Driver

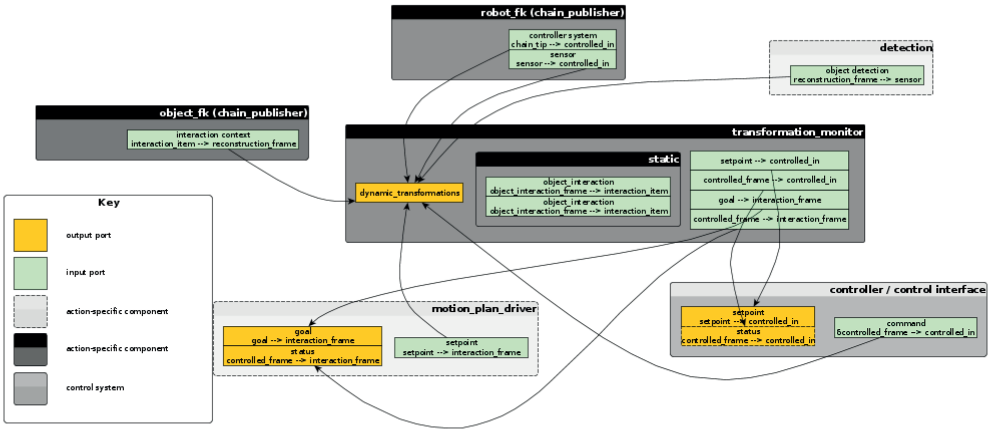


- Decouples robot morphology from task motion and sensor processing
- Motion description can be applied to different context

Control



Automatic transformation resolution and component interconnections


robot_fk (chain_publisher)
controller system
chain_tip --> controlled_in
sensor
sensor --> controlled_in
detection
object detection
reconstruction_frame --> sensor
object_fk (chain_publisher)
interaction context
interaction_item --> reconstruction_frame
transformation_monitor
dynamic_transformations
static
object_interaction
object_interaction_frame --> interaction_item
object_interaction
object_interaction_frame --> interaction_item
setpoint --> controlled_in
controlled_frame --> controlled_in
goal --> interaction_frame
controlled_frame --> interaction_frame
Key
output port
input port
action-specific component
action-specific component
control system
motion_plan_driver
goal
goal --> interaction_frame
status
controlled_frame --> interaction_frame
setpoint
setpoint --> interaction_frame
controller / control interface
setpoint
setpoint --> controlled_in
status
controlled_frame --> controlled_in
command
6controlled_frame –> controlled_in

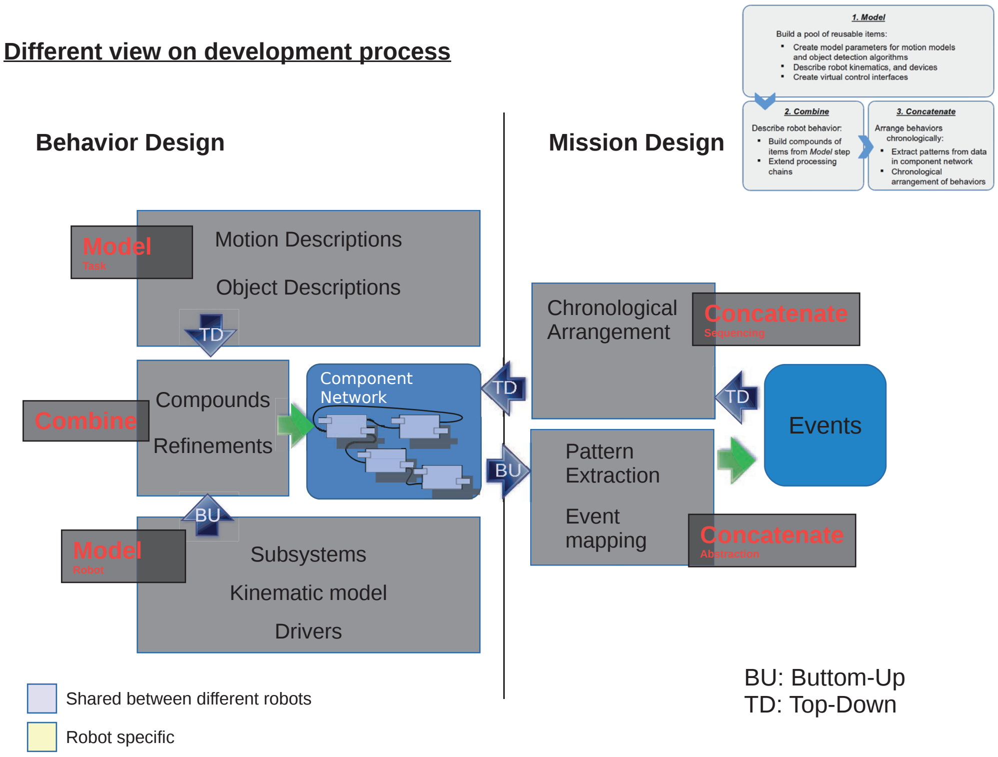
Callable “action”, that when executed, attaches a motion command generating component network to the “main” component network

Discussion

Summary

- Workflow for robot manipulation behavior design
  - Supports transfer of behavior between robots and application contexts
    - Allow specification of multi-stage control networks
    - Attach robot-independent task description
  - Support developers by providing domain-specific high-level commands and allows integration of additional tooling

Different view on development process



1. Model

Build a pool of reusable items:

- Create model parameters for motion models and object detection algorithms
- Describe robot kinematics, and devices
- Create virtual control interfaces

2. Combine

Describe robot behavior:

- Build compounds of items from *Model* step
- Extend processing chains

3. Concatenate

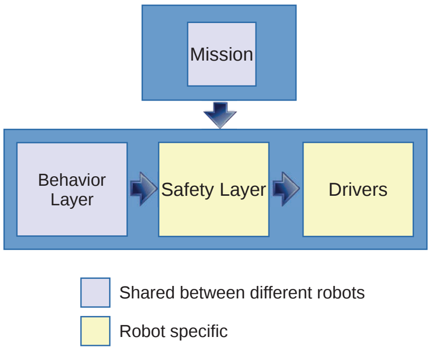
Arrange behaviors chronologically:

- Extract patterns from data in component network
- Chronological arrangement of behaviors

BU: Buttom-Up
TD: Top-Down

Next steps

- Evaluation with
  - Control network
    - operational, saftety layer and driver layer
    - containing joint devices with different control modes
  - Mission consists of different manipulation behaviors



2.9 'Rock Tutorials Recap' (FW-T-09)

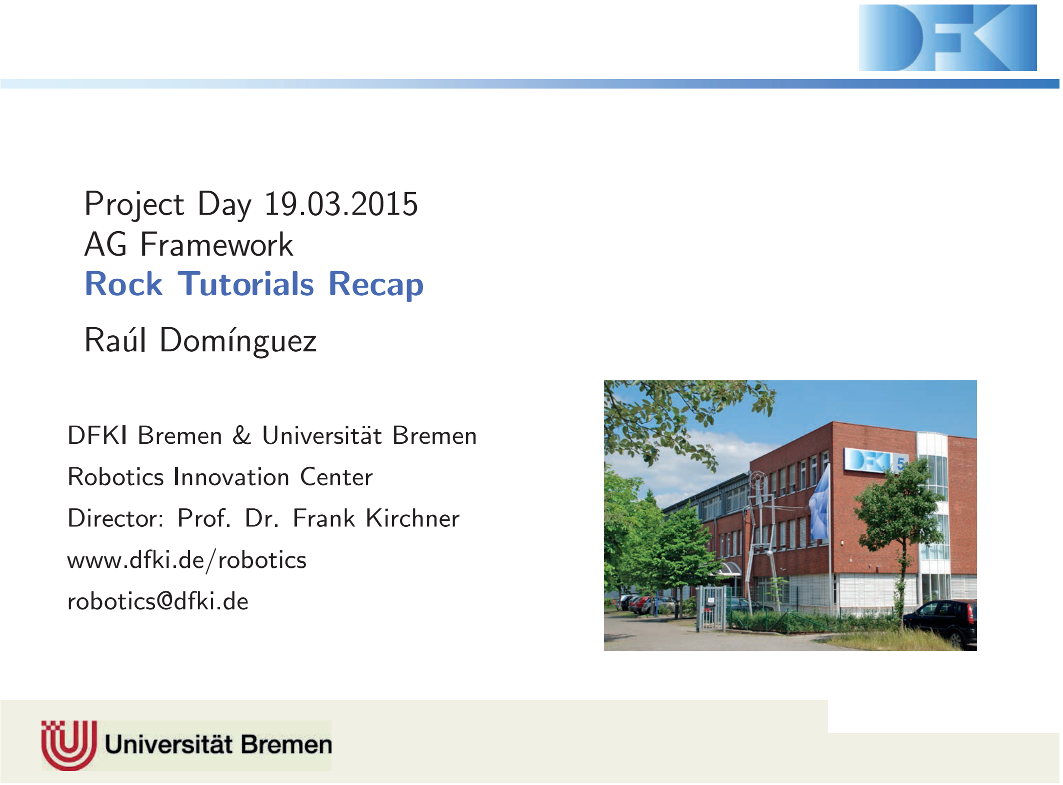
Raúl Domínguez[(1)]

(1) DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 1, 28359 Bremen, Germany

Contact: `raul.dominguez@dfki.de`

Abstract

The presentation starts with a review on the workshops that were done at DFKI during the last year. The workshops focused on hands-on work with other rock users in following the tutorials. Next, a summary of the feedback received from the attenders will be presented. Finally, lessons learned and next steps towards helping new users learn rock efficiently can be discussed.



Project Day 19.03.2015
AG Framework
Rock Tutorials Recap

Raúl Domínguez

DFKI Bremen & Universität Bremen
Robotics Innovation Center
Director: Prof. Dr. Frank Kirchner
www.dfki.de/robotics
robotics@dfki.de

Universität Bremen

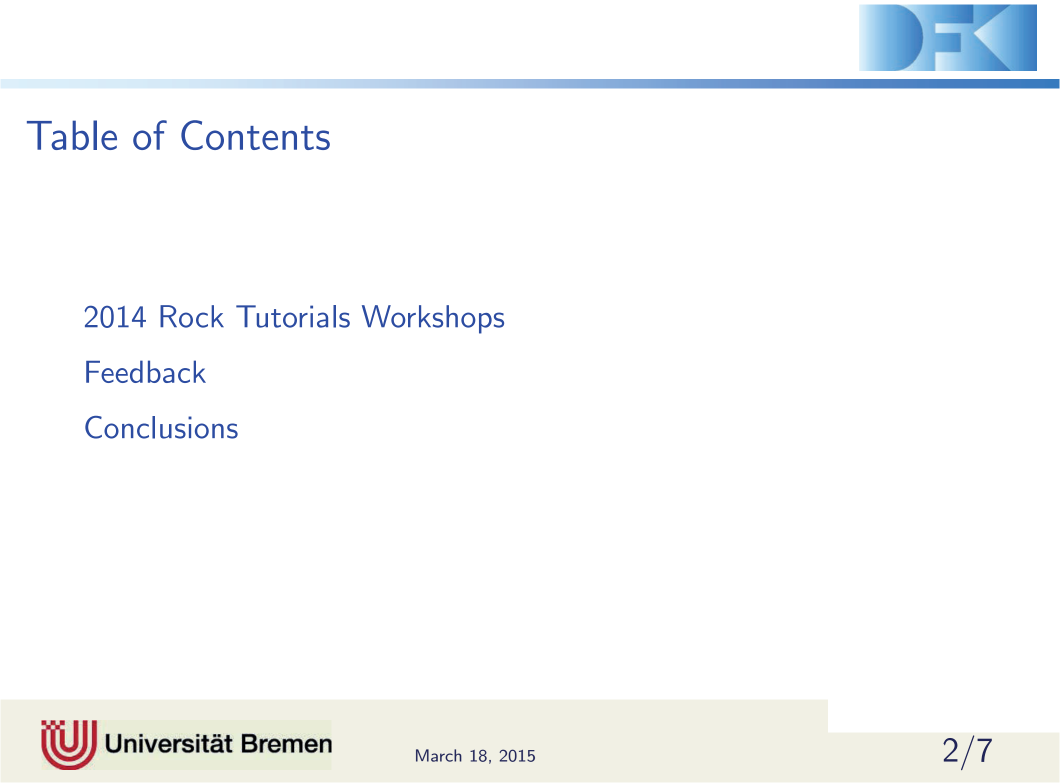


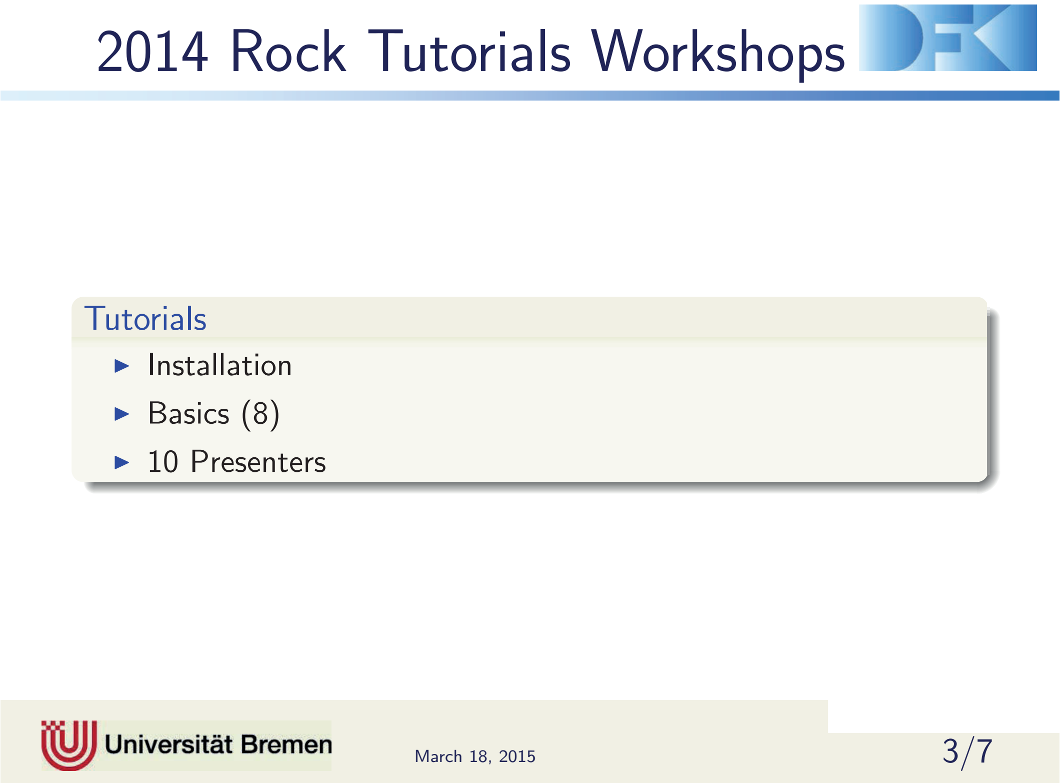
Table of Contents

2014 Rock Tutorials Workshops

Feedback

Conclusions

Universität Bremen March 18, 2015 2/7

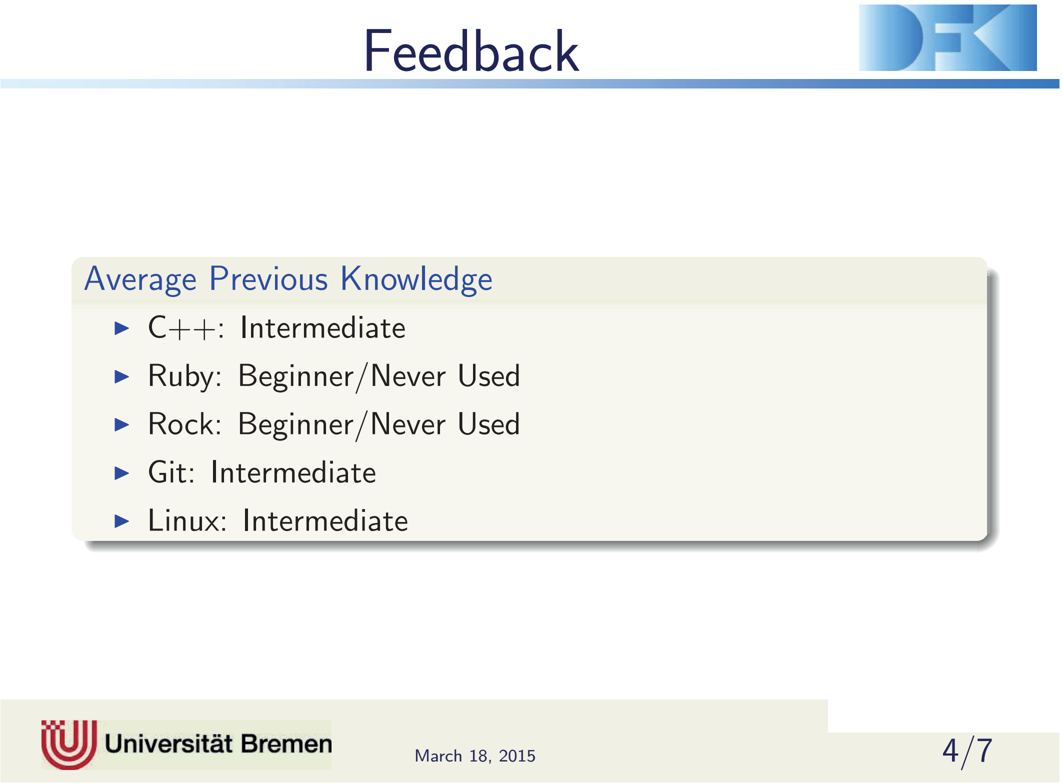


2014 Rock Tutorials Workshops

Tutorials

- Installation
- Basics (8)
- 10 Presenters

Universität Bremen March 18, 2015 3/7

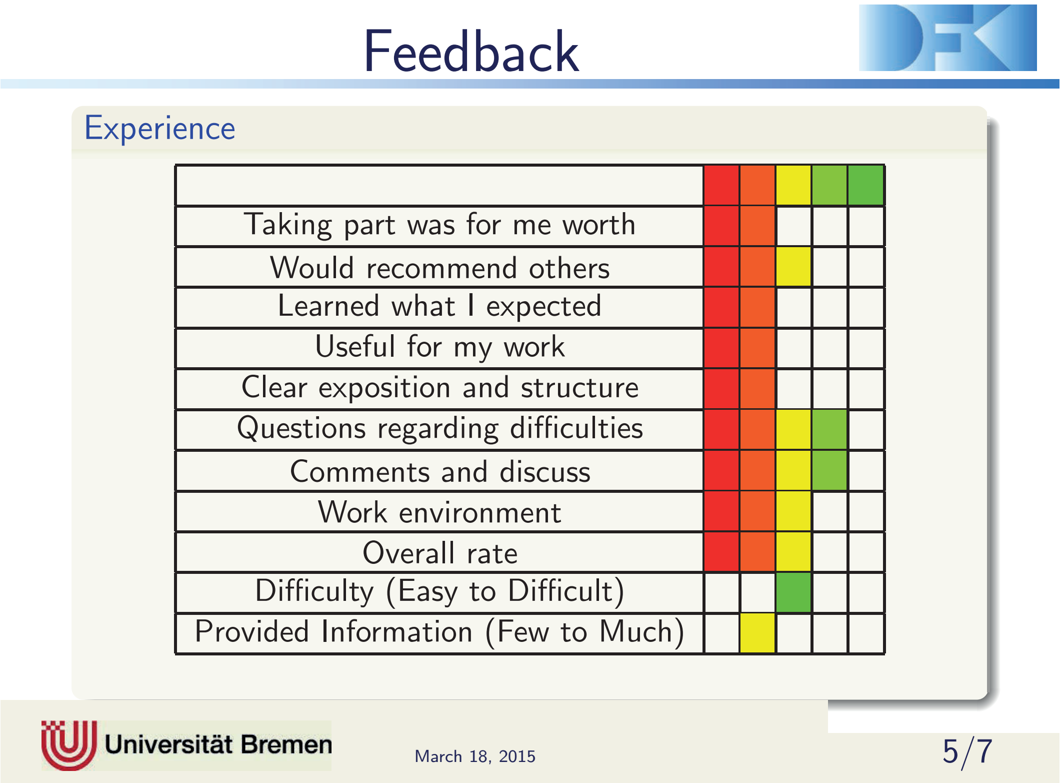


Feedback

Average Previous Knowledge

- C++: Intermediate
- Ruby: Beginner/Never Used
- Rock: Beginner/Never Used
- Git: Intermediate
- Linux: Intermediate

Universität Bremen March 18, 2015 4/7

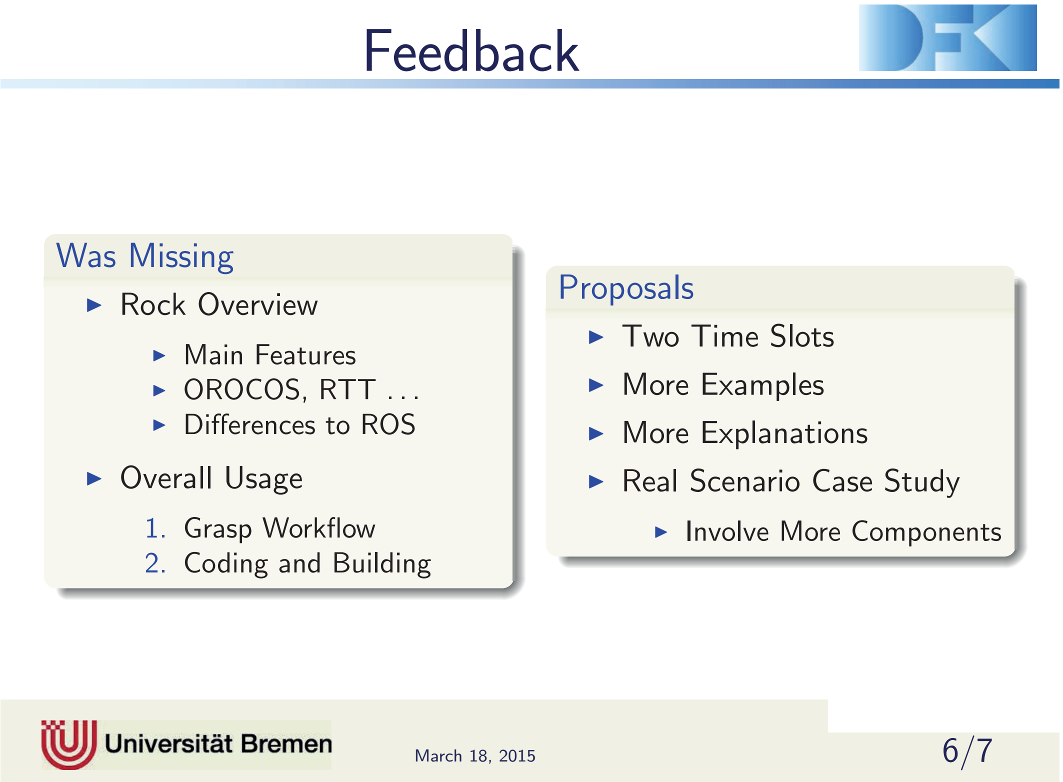


Feedback

Experience

	🟥	🟧	🟨	🟩	🟩
Taking part was for me worth	🟥	🟧			
Would recommend others	🟥	🟧	🟨		
Learned what I expected	🟥	🟧			
Useful for my work	🟥	🟧			
Clear exposition and structure	🟥	🟧			
Questions regarding difficulties	🟥	🟧	🟨	🟩	
Comments and discuss	🟥	🟧	🟨	🟩	
Work environment	🟥	🟧	🟨		
Overall rate	🟥	🟧	🟨		
Difficulty (Easy to Difficult)			🟩		
Provided Information (Few to Much)		🟨			

Universität Bremen March 18, 2015 5/7



Feedback

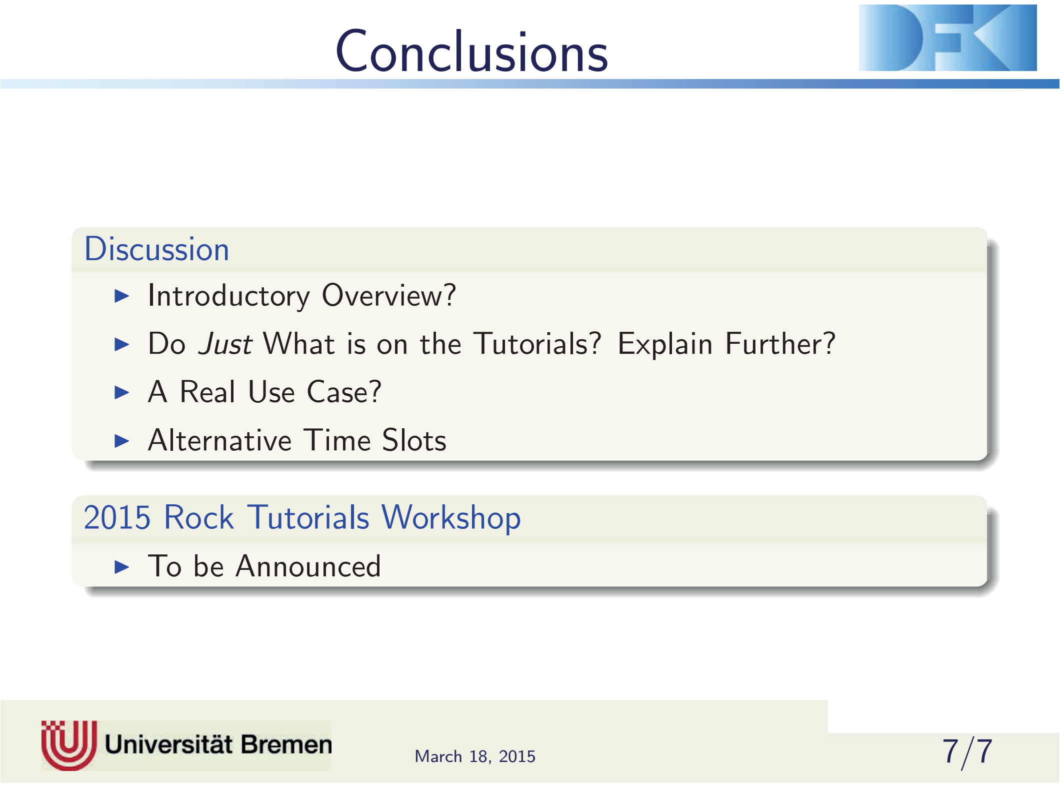
Was Missing

- Rock Overview
  - Main Features
  - OROCOS, RTT . . .
  - Differences to ROS
- Overall Usage
  1. Grasp Workflow
  2. Coding and Building

Proposals

- Two Time Slots
- More Examples
- More Explanations
- Real Scenario Case Study
  - Involve More Components

Universität Bremen March 18, 2015 6/7



Conclusions

Discussion

- Introductory Overview?
- Do *Just* What is on the Tutorials? Explain Further?
- A Real Use Case?
- Alternative Time Slots

2015 Rock Tutorials Workshop

- To be Announced

Universität Bremen

March 18, 2015

7/7

German Research Center for Artificial Intelligence (DFKI) GmbH

DFKI Bremen
Robert-Hooke-Straße 1
28359 Bremen
Germany
Phone: +49 421 178 45 0
Fax: +49 421 178 45 4150

DFKI Saarbrücken
Stuhlsatzenhausweg 3
Campus D3 2
66123 Saarbrücken
Germany
Phone:+49 681 875 75 0
Fax: +49 681 857 75 5341

DFKI Kaiserslautern
Trippstadter Straße 122
67608 Kaiserslautern
Germany
Phone:+49 631 205 75 0
Fax: +49 631 205 75 5030

DFKI Projektbüro Berlin
Alt-Moabit 91c
10559 Berlin
Germany
Phone:+49 30 238 95 0

E-mail:
reports@dfki.de

Further information:
http://www.dfki.de