



Continuous Integration for Robotics Systems Development using Jenkins



Florian Lier, Johannes Wienke, Arne Nordmann, Michael Götting, Sebastian Wrede

**Bielefeld University
(CoR-Lab, CITEC)**

www.cor-lab.de | www.cit-ec.de



CoR-Lab CITEC





Talk Overview

- Robotics as an emerging research area
- Research context
 - Different applications and institutions
 - General requirements and problems
- What we are doing with Jenkins in detail. How does it help?
- Specific project examples:
HUMAVIPS and RoboCup using Jenkins
- Conclusion and outlook



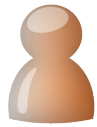


Robotics as a Research Area

A robot is a “re-programmable multi-functional manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks”. [Robotics Institute of America]



This is a very broad definition - what are some typical examples ?



Robotics as a Research Area

Typical Examples:

Industrial automation → Assembly lines

Search and rescue robots → Disaster scene

Entertainment robotics → Pets/toys

Household robots → Floor cleaning



KUKA Titan



iRobot Packbot

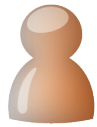


Sony AIBO



iRobot Roomba





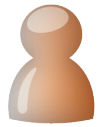
Robotics as a Research Area

**There are already a few commercial products, but...
...there are limitations:**

Tele-operated, pre-programmed, limited autonomy, constrained environments, lack of rich human-robot interaction

- Robots are often operated by domain experts.
- To extend the robots functionality you need an expert, or you need to become one.
- “Learning” is mostly uncovered. What is learning?
- Missing user friendly interaction/interfaces.





Robotics as a Research Area



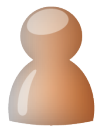


Research Context

Based on these statements researchers at **CoR-Lab** and **CITEC** pursue a mission:



- **Creating cognitive abilities in technical systems from everyday devices to humanoid robots to make them more useful, more friendly and easier to interact with.**
- What are the basic building blocks of cognition and learning? How can we endow robots with some social competence, to make them acceptable as assistants to humans?
- Creating bridges between the cultures of engineering and humanity to better shape tomorrow's technology according to human needs.



Research Context

Cognitive Robotics @ Bielefeld University



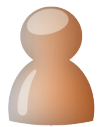
Industrial robots
focus on function



Cognitive robots also
focus on the interaction
interface

Important for social
interaction

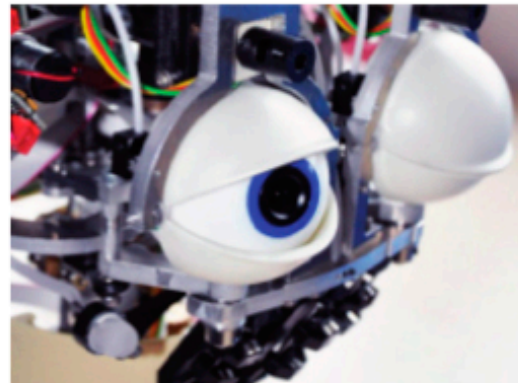
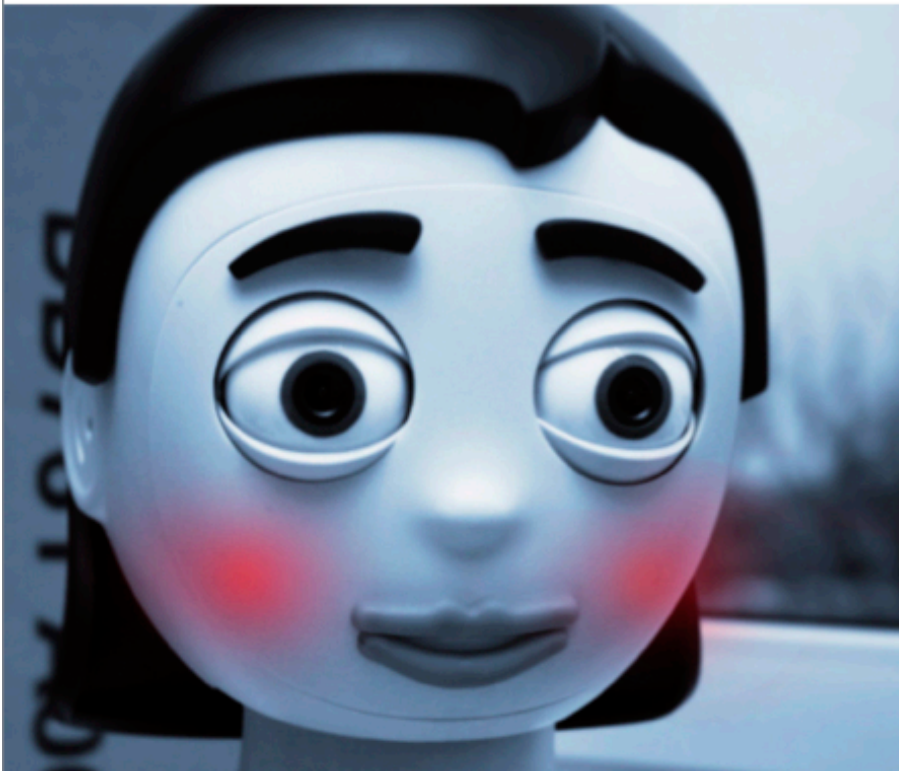
Determines flexibility
of interaction



Research Context

Cognitive Robotics @ Bielefeld University

Flobi



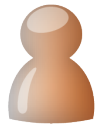
Anthropomorphic design

Flexible appearance through changeable parts e.g.: front mask, eyebrows, hair, lips)

Capable of Intuitive Human-Robot Interaction through:



Facial expressions
Signaling of emotions
Speech recognition and synthesis



Research Context





Research Context

Based on these statements researchers at **CoR-Lab** and **CITEC** pursue a mission:



- Creating cognitive abilities in technical systems from everyday devices to humanoid robots to make them more useful, more friendly and easier to interact with.
- **What are the basic building blocks of cognition and learning? How can we endow robots with some social competence, to make them acceptable as assistants to humans?**
- Creating bridges between the cultures of engineering and humanity to better shape tomorrow's technology according to human needs.



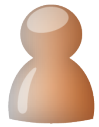
Research Context

Cognitive Robotics @ Bielefeld University

Questions robots could ask themselves:

- Where am I? [localization]
- Where do I want to be, and how do I get there? [path planning, navigation]
- How do I interpret my sensor feedback to determine my current state and surroundings? [perception]
- How do I make sense of noisy sensor readings? [uncertainty management]
- How do I fuse data from multiple sensors to improve estimates of the current situation? [sensor fusion]
- How do I know what to pay attention to? [focus-of-attention]
-





Research Context University



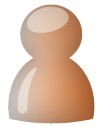


Research Context

Based on these statements researchers at **CoR-Lab** and **CITEC** pursue a mission:



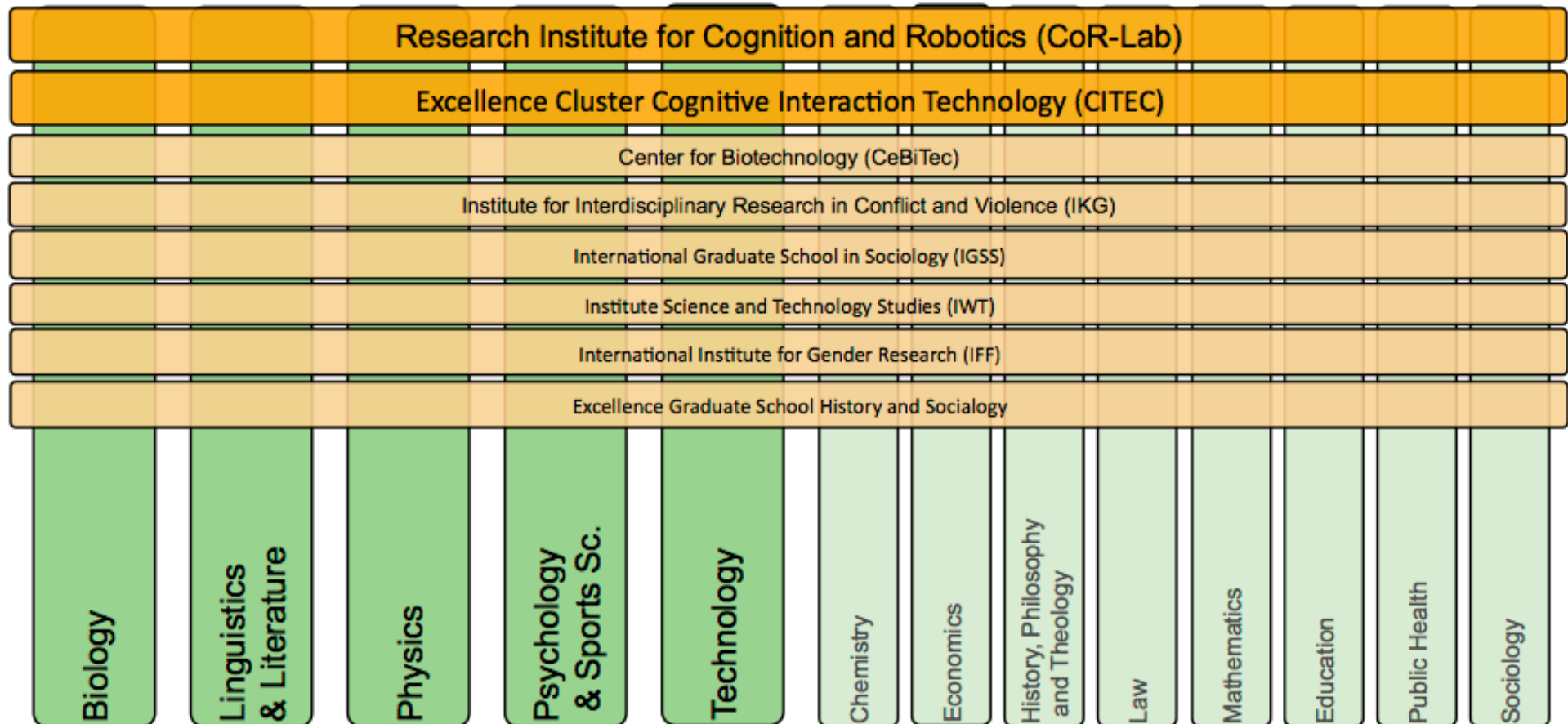
- Creating cognitive abilities in technical systems from everyday devices to humanoid robots to make them more useful, more friendly and easier to interact with.
- What are the basic building blocks of cognition and learning? How can we endow robots with some social competence, to make them acceptable as assistants to humans?
- **Creating bridges between the cultures of engineering and humanity to better shape tomorrow's technology according to human needs.**



Research Context



The mission can only be accomplished by interdisciplinarity!



Bringing together the ideas of researchers from computer science and robotics, from linguistics, from biology and physics [...] to develop systems with cognitive abilities.



Resulting (Technical) Challenges



- Developers with different skills and focus areas
 - System setup time must be reduced
 - Quality requirements on components need to be ensured
- Diversity of programming languages (e.g. Java, Python, **C++**, Matlab, Common Lisp) in one system
- Integration of legacy code outside of current research topics
- Difficulties in re-use of components (different use cases)
- Prototype hardware and hw. development
- Complex organization e.g.: scm and acm

Is Jenkins a useful tool to overcome these challenges ?



What are we doing in detail?



Outline:

- C++ project setup:
 - Relocatable projects (with CMake)
 - Building on diverse nodes
 - Code analysis and metrics
- Shared resources
- Common CMake job template
- Lessons learned with C++ and Jenkins
- Off-site development



C++: Relocatable Projects with CMake



Problem statement:

- Jenkins dependency tracking with artifacts **relocates** built projects to a different workspace path (through copy)
- With C++, usually, several paths are already fixed at configuration time, e.g. in config files for downstream projects.

An example from the Shared Desktop Ontologies

```
set(SHAREDDESKTOPONTOLOGIES_VERSION_MAJOR 0)
set(SHAREDDESKTOPONTOLOGIES_VERSION_MINOR 3)
set(SHAREDDESKTOPONTOLOGIES_VERSION 0.3)

set(SHAREDDESKTOPONTOLOGIES_ROOT_DIR
    /usr/share/ontology)
```



C++: Relocatable Projects with (CMake)



- Further occurrences of absolute paths:
 - Library rpath
 - External tool locations
 - pkg-config files
 - Shell scripts

Solution:

Make paths relative to the location of current file, e.g. the CMake config file



C++: Relocatable Projects with (CMake)



Exemplary CMake config file:

```
GET_FILENAME_COMPONENT(RSB_CONFIG_DIR
                        "${CMAKE_CURRENT_LIST_FILE}" PATH)

SET(RSB_INCLUDE_DIRS "${RSB_CONFIG_DIR}/../..../include")
SET(RSB_RUNTIME_LIBRARY_DIRS "${RSB_CONFIG_DIR}/../..../bin")

INCLUDE("${RSB_CONFIG_DIR}/RSBDepends.cmake")
SET(RSB_LIBRARIES rsbcore)
```

Similar techniques exists:

- pkg-config: provide definitions based on \${prefix},
command line tool offers
--define-variable=prefix=/relocated
- Avoid configuring source files with absolute
locations



C++: Building on Diverse Nodes



Problem statement:

- Software should be tested on different operating systems and distributions
- Build instructions vary for each different OSes
- Locations of dependencies vary across distributions

Solution:

- Different jobs for Unix-like systems and Windows (too many differences for multi-configuration projects between Win and Unix)
- Location information in node-specific environment variables

The screenshot shows the 'Node Properties' configuration page in Jenkins. It has a section for 'Environment variables' which is checked. Below this, there is a table-like structure for 'List of key-value pairs'. The first entry has 'name' as 'BOOSTUUIID_ROOT' and 'value' as '/vol/default'. The second entry has 'name' as 'GCOVR_PYTHONPATH' and 'value' as '/vol/default/lib/python2.6/site-packag'. Each entry has a 'Delete' button next to it.

List of key-value pairs	
name	BOOSTUUIID_ROOT
value	/vol/default
<button>Delete</button>	
name	GCOVR_PYTHONPATH
value	/vol/default/lib/python2.6/site-packag
<button>Delete</button>	

```
export PYTHONPATH=${GCOVR_PYTHONPATH}:${PYTHONPATH}

cmake -DCMAKE_BUILD_TYPE=coverage -DRSC_DIR="${RSC}/share/RSC" -
DRSBProtocol_DIR="${RSBPROTO}/share/RSBProtocol" -DBOOSTUUIID_ROOT=${BOOSTUUIID_ROOT} -
DSPREAD_ROOT=${SPREAD_ROOT} -DGCOVR_ROOT=${GCOVR_ROOT} -
DTEST_SPREAD_CONFIG_SYMLINK=/tmp/rsbmulti-test.conf -DTEST_SPREAD_PORT=4188 ..
```

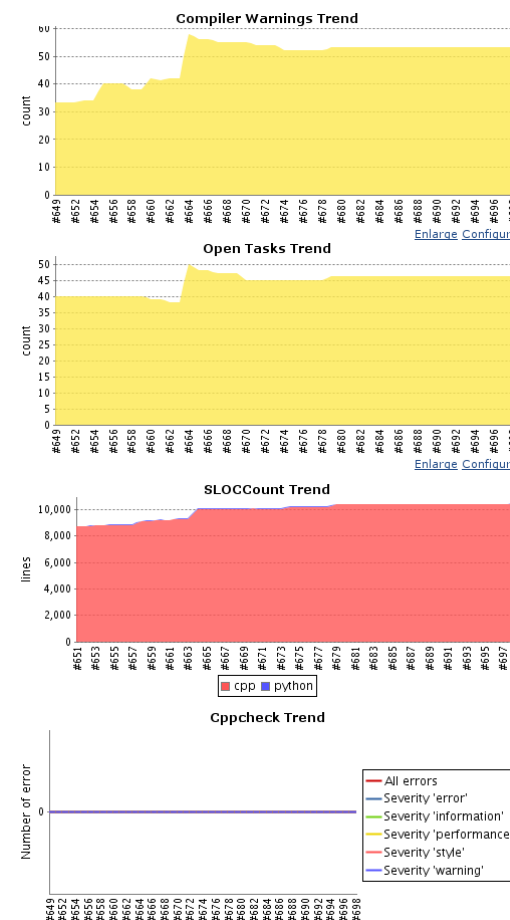


C++: Code Analysis and Metrics



Integrated tools:

- **Icov** and **gcovr**: Code coverage
 - HTML report
 - XML report tightly integrated with Jenkins
- **SLOCcount**: lines of code
- **GTest** and **GMock**: Unit Testing
- **Cppcheck**: Static code analysis
- Integration through CMake (easy availability of analysis outside Jenkins).
- Integration macros publicly available in RSC project: <https://code.cor-lab.de/projects/rsc>





Shared Resources



Problem statement:

- Some resources of the build environment are intrinsically shared, e.g. ports for testing

Solution:

- Builds can be parameterized for these resources
- Jenkins build configuration dictates the parameters to disambiguate jobs

E.g. in the job config:

```
cmake -DSPREAD_PORT=2324 ..
```



Common CMake Job Template



Some common techniques emerged

- Separated jobs:
 - Multi-conf projects for building and runtime-analysis (unit tests and code coverage)
 - Free-style job for static code analysis, other metrics and API documentation generation

●	☀️	RSB	14 hr (#550)	1 day 15 hr (#546)	10 min
●	⚡️	RSB Static Analysis	9 min 37 sec (#704)	11 min (#703)	1 min 32 sec
●	☁️	RSC	1 day 15 hr (#195)	2 days 16 hr (#193)	4 min 24 sec
●	☁️	RSC Static Analysis	7 min 33 sec (#309)	1 day 15 hr (#308)	21 sec



Common CMake Job Template



- Upstream projects:
 - One **upstream** folder for downloaded artifacts

```
# cleanup left-over artifacts
rm -rf upstream
mkdir upstream

# download artifacts to this folder

# extract
cd upstream
for name in *.tar.gz; do tar -xzf $name; done
```

- Using downloaded artifacts

```
UPSTR_A=`find "${WORKSPACE}/upstream" -maxdepth 1 \
    -type d -name "UPSTR_A*";`
cmake -DUPSTR_A_DIR="${UPSTR_A}/share/upa" -DFOO=...
```



Lessons Learned with C++ and Jenkins



- It works! But some things need to be done
- Relocatable projects essential requirement
- Generating fingerprints separately for every header file + library breaks dependency tracking:
 - Not all headers change with every build
 - Downstream projects suddenly depend on a lot of upstream versions
 - Use a single archive (e.g. *.tar.gz) file instead
- Conversion of analysis results to supported formats:
 - Existing tools sometimes hard to find, e.g. gcovr



Off-Site Development



Problem statement:

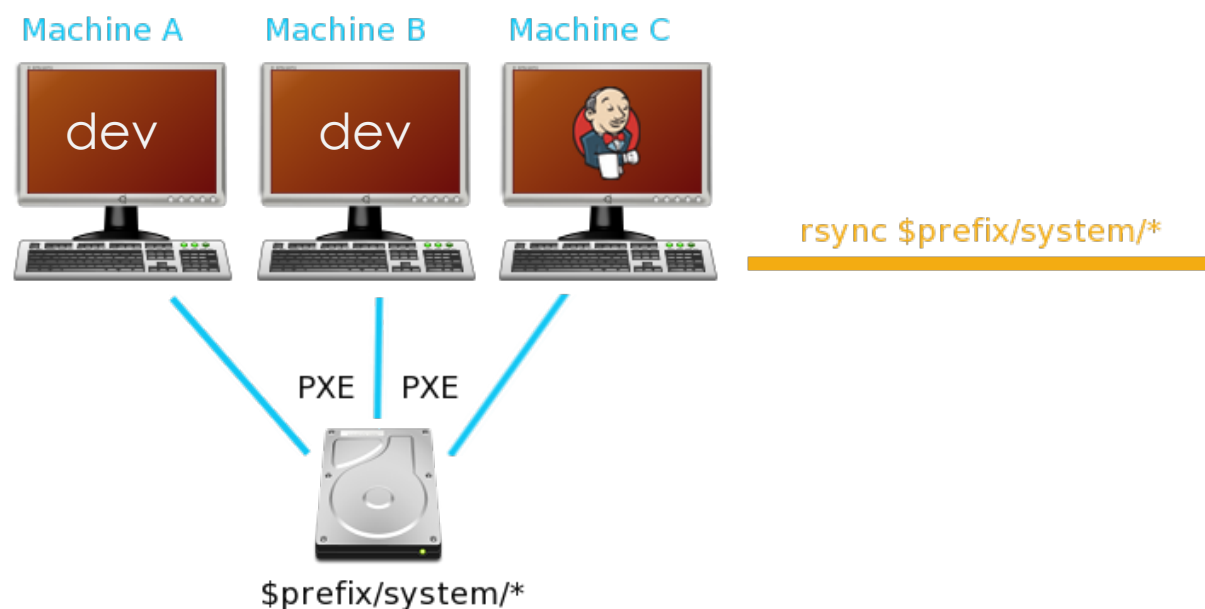
- Sometimes we do not only need to relocate build jobs but the whole project and infrastructure !
 - Live demos outside the university
 - Developer camps or courses
- New development environment
 - Diverse machines, e.g. private Laptops with possibly different system statuses
 - Unexperienced developers



Off-Site Development



Lab setup:



Each machine is served with a system image via PXE/netboot, which ensures a consistent system environment.

As soon as a system build becomes stable, the robot is synchronized automatically, triggered via Jenkins.



Off-Site Development

Off-site setup:

Laptop A



Laptop B



Build Server C



`rsync $prefix/system/*`

PXE PXE



`$prefix/system/*`



Off-site, developers need to synchronize and maintain their systems manually, possible risk of machine specific bugs caused by varying system statuses.



Off-Site Development



- After a successful system build Jenkins automatically checks all developer statuses.
- This is realized with a post-build step which triggers a shell script.
- The script creates a MD5 sum of the \$prefix, connects to each developer machine and derives another MD5 sum.
- Finally it compares the sums and determines the sync status.



Off-Site Development



S	W	Job ↓	12 hr (#33)	12 hr (#34)	12 hr (#19)	0.32
●	☀	BonSAI	12 hr (#34)	16 hr (#19)		0.81
●	☀	BTL	13 hr (#21)	36 min (#4)		0.87
●	☁	CheckSync Status Server vs Harlie	12 hr (#3)	12 hr (#6)		3.8 s
●	☁	CheckVolumeStatus -- Birte	36 min (#9)	12 hr (#2)		1.9 s
●	☁	CheckVolumeStatus -- Leon	36 min (#5)	12 hr (#1)		0.64
●	☁	CheckVolumeStatus -- Lukas	36 min (#3)	53 min (#5)		1.3 s
●	☁	CheckVolumeStatus -- Maikel	36 min (#7)	12 hr (#2)		1 sec
●	☀	CheckVolumeStatus -- Marcel	36 min (#4)	12 hr (#2)		0.74
●	☁	CheckVolumeStatus -- Marian	11 hr (#3)	12 hr (#1)		11 s
●	☁	CheckVolumeStatus -- Newton	36 min (#3)	36 min (#5)		0.76
●	☁	CheckVolumeStatus -- Norman	N/A	12 hr (#1)		12 s
●	☀	CheckVolumeStatus -- Phillip	11 hr (#2)	13 hr (#3)		59 s
●	☁	CheckVolumeStatus -- Viktor	13 hr (#5)	12 hr (#14)		17 s
●	☁	GPC	12 hr (#15)	16 hr (#79)		42 s
●	☀	PCA	11 hr (#104)	17 hr (#27)		13 s
●	☁	RoboCupAtHome	17 hr (#28)	1 day 17 hr (#8)		
●	☀	SpeechRec	36 min (#71)			

RSS for failures RSS for



Lessons Learned from Off-Site Development



- Off-site development is critical when it comes to manual system maintenance and synchronization
- Jenkins is a great tool to provide instant visual feedback of system statuses, which is extremely important when unexperienced developers are involved
- Easy realization with post-build shell scripts
- Works for build slaves too

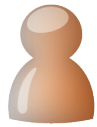
The check-sync script will soon be available at:
<https://ci.clf.cit-ec.de/citools>



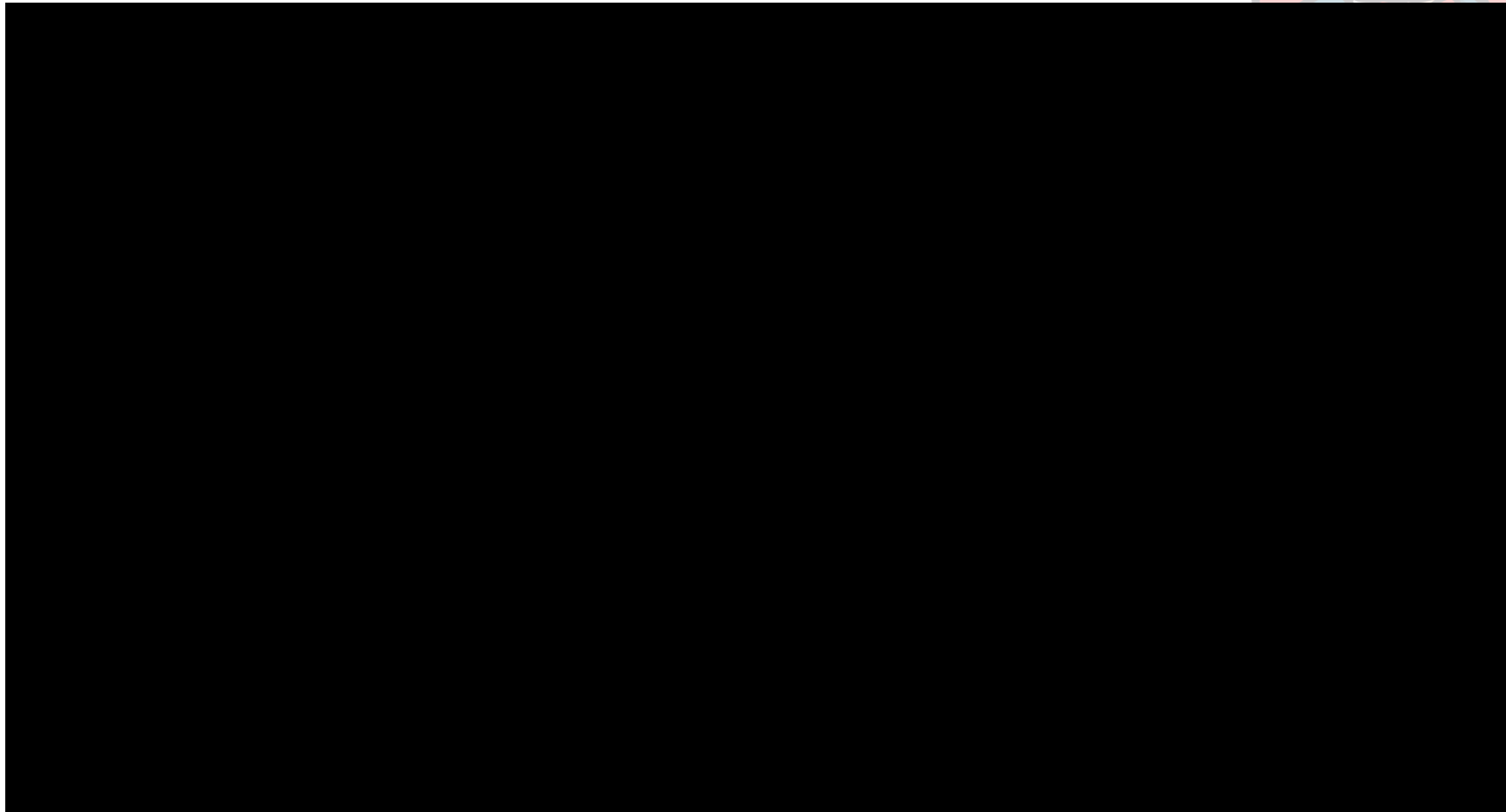
HUMAVIPS

- *“Humanoids with Auditory and Visual Abilities in Populated Spaces”*
- Goal: Endow the humanoid robot NAO with abilities to interact in groups
- Collaborative research project with 4 universities + robot manufacturer Aldebaran Robotics (ALD)
- Besides (ALD) and us partners work on algorithms, **not** integration
- Developed software partially used in other systems





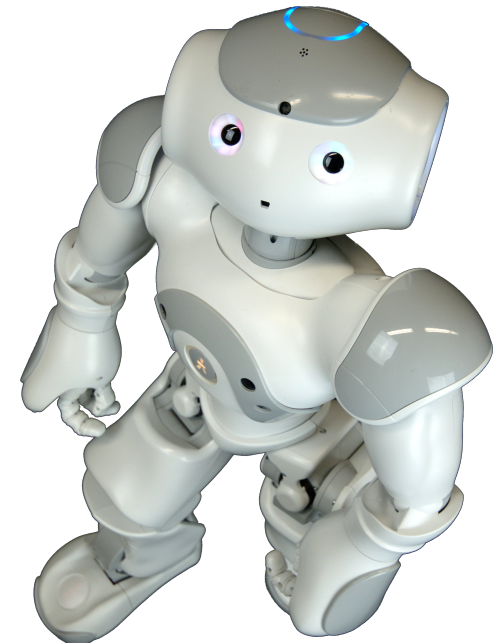
HUMAVIPS

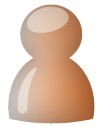




HUMAVIPS: Challenges

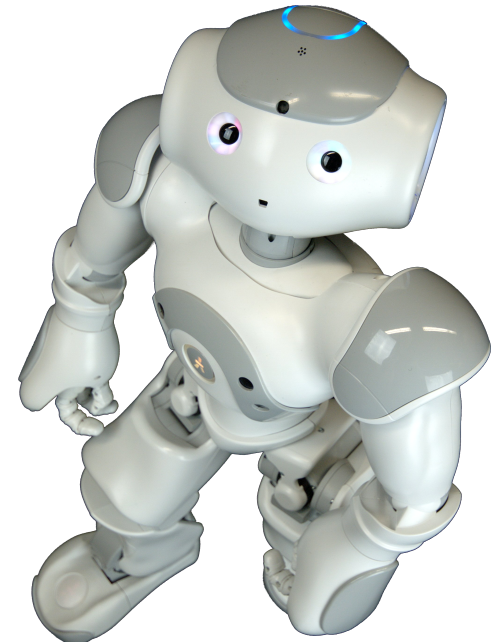
- Distributed development team
 - Increased communication overhead
 - Partially varying interest and requirements on the developed methods and software
- Each partner requires a replicated system
 - System setup needs to be as fast as possible
- NAO robot requires cross-compilation
 - Increased complexity for development
- Different OSes and Linux Distros

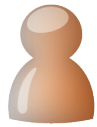




HUMAVIPS: Achievements with Jenkins

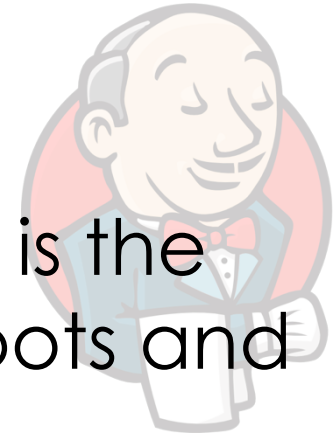
- Cross-compiled NAO-setup built by Jenkins:
 - Whole NAO SDK by ALD and HUMAVIPS extensions
 - Tutorial with build instructions for this setup spans *several pages*
 - Installation of binary artifacts from Jenkins spans *10 lines* description text
- Remote setup:
 - Significantly reduced setup time by binary artifacts
 - Communication for compilation issues reduced
- CI ensures compilation on OSes

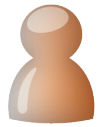




RoboCup

- The RoboCup world championship is the most important competition for robots and their programmers.
- The @HOME League, in which we are participating since 2009, involves challenges for service robots with tasks drawn from basic household activities.
- Tasks: Who is who, Go get it, Follow me

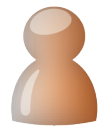




RoboCup

@Home league arena

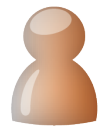




RoboCup

Who is Who





RoboCup

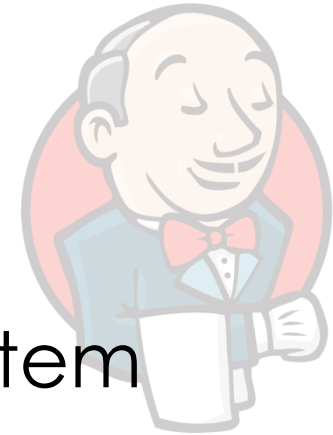
Go get it





RoboCup Challenges

- Rotating team members (students)
- Constantly evolving, distributed system with complex dependencies
- Compile time is critical (> 1h, 2 cores)
- Domain experts for Speech recog., navigation, manipulation, ...
- Multiple build tools: ant, CMake and various shell scripts
- Agile programming at the actual event

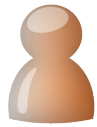




RoboCup Achievements

- Continuous deployment on the robot saves time and ensures functionality.
- System state overview for all developers, visual feedback after the build process.
- Comfortable build environment, just commit your code, Jenkins does the rest.
- 5th place out of 25 teams in 2011- partly, as we believe, because we used Jenkins





CI vs. Guidelines, Style Guides etc.

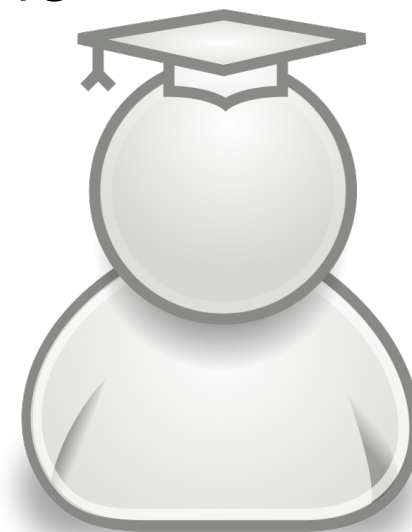
- Diverse and interdisciplinary research context with multiple overlapping organizational units prevents common guidelines, style guides etc.
- Especially, quality assurance processes are impossible to enforce
- Sometimes, even if these processes or guidelines are research topics
- → Continuous integration is the **only** affordable QA technique

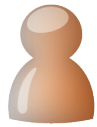




CI vs. Guidelines, Style Guides etc.

- When it comes to agile programming, especially with unexperienced developers, Jenkins is a brilliant tool for instant visual feedback not only for software builds
- Developers can actually focus on their code which makes development more comfortable. We will conduct further research on that topic in relation to the RoboCup
- The utilization of Jenkins drastically decreased system setup time in our research projects





Outlook/Discussion

CITEC software toolkit for cognitive systems,
based on Jenkins artifacts
<https://toolkit.cit-ec.uni-bielefeld.de>



Cognitive Interaction Toolkit
A Collection of Linked Research Data to Facilitate Interactive Cognitive Systems

The Toolkit Browse Tags Advanced Search Contact & Feedback FAQ My Contribution Login

Cognitive Interaction Toolkit

The Cognitive Interaction Toolkit provides. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus.

Latest Components

BielefeldTypeLibrary	iceWing	BonSAI	XSRAD
The Bielefeld Type Library (BTL) is a collection of XML-based data	iceWing, an Integrated Communication Environment Which Is Not Gesten (this is)	BonSAI is a robot behavior abstraction layer written in Java.	A Unifying Model-Centric Analysis Approach for Robotic

ToolKit Contexts

Context	Count
Communication	0
Dialog	0
Generic	1
Manipulation	0
Tools	2
Vision	1

CITKit Features

- Systems (uncovered yet)
- Components
- Publications
- Open Data
- Data types

Search

type a keyword

Interactive Intelligent Systems

iceWing-SF-trunk

Artifacts:

icewing

Build Job Url:

<https://ci.cit-ec.de/job/SourceForge-iceWing-trunk/11/>

SCM Locations:

<https://icewing.svn.sourceforge.net/svnroot/icewing>

SCM Version:

2018

Build Number:

11

Built On:

Ubuntu32-Lucid

Hardware in the loop, running Jenkins on a robot –
metrics for cognitive systems – worth a plug-in?

Linked Co
iceWing








ToolKit C
Communic
Dialog
Generic
Manipulat
Tools
Vision

CITKit Fe
Systems
Compon
Publicat
Open Da
Data typ



Thank You To Our Sponsors



Platinum Sponsor	 CloudBees
Gold Sponsor	 LIFERAY
Silver Sponsor	 redhat.
Bronze Sponsors	<div> New Relic</div> <div> sauce LABS</div> <div> CHARIOT SOLUTIONS Practical, Smart Software Development</div> <div> exo</div>

Coming Soon: The CloudBees Newsletter for Jenkins

✓ Please complete the Jenkins survey to help us better serve the community
(bonus: a chance to win an Apple TV!)