PREPRINT Towards Automated System and Experiment Reproduction in Robotics

Florian Lier[1], Marc Hanheide[2], Lorenzo Natale[3], Simon Schulz[1], Jonathan Weisz[4], Sven Wachsmuth[1] and Sebastian Wrede[1]

Abstract— **Even though research on autonomous robots and human-robot interaction accomplished great progress in recent years, and reusable soft- and hardware components are available, many of the reported findings are only hardly reproducible by fellow scientists. Usually, reproducibility is impeded because required information, such as the specification of software versions and their configuration, required data sets, and experiment protocols are not mentioned or referenced in most publications. In order to address these issues, we recently introduced an integrated tool chain and its underlying development process to facilitate reproducibility in robotics. In this contribution we instantiate the complete tool chain in a unique user study in order to assess its applicability and usability. To this end, we chose three different robotic systems from independent institutions and modeled them in our tool chain, including three exemplary experiments. Subsequently, we asked twelve researchers to reproduce one of the formerly unknown systems and the associated experiment. We show that all twelve scientists were able to replicate a formerly unknown robotics experiment using our tool chain.**

I. INTRODUCTION

Research on autonomous robots and human-robot interaction achieved great progress over recent years. In order to conduct research in these fields, the development of highly integrated robotics systems that incorporate a broad set of skills into a single architecture or demonstrate exceptional performance in a single domain is a prerequisite. Furthermore, established "off the shelf" robot hardware, simulators and programming frameworks already provide a sound basis for development and reuse across different system setups [7]. On the downside, given the inherent complexity of these systems, consisting of soft- and hardware components and capable of solving nontrivial tasks — many of the reported results and experiments cannot *easily* be reproduced by interested researchers and reviewers in order to confirm reported findings [2]. In addition, most current robotics systems are realized by implementing a component-based architecture [5] that integrates not only in-house, but also third party components from diverse publicly available software repositories such as ROS [15] or OROCOS [6]. Therefore, these components do not necessarily share the same ecosystem, build system, integration model, deploy and execution environment. This "zoo of tools" introduces additional challenges to (robotics) software development and especially to the reproduction process of these systems. In order to pinpoint these challenges, we introduced four key problem statements with respect to reproducibility: a) information retrieval and aggregation, b) semantic relationships, c) software deployment, and d) experiment testing, execution and evaluation in [10]. These challenges will be further explained in a representative example in section II.

In [10] we also introduced an *integrated software tool chain and its underlying development process* – **The Cognitive Interaction Toolkit (CITK)**. It supports system developers and experiment designers already in an *early stage* of system development, in order to automate tasks and take on the above challenges. Hence, in this contribution we validate the complete tool chain in a *practical user study* testing three hypotheses: *(i)* the tool chain is applicable for different ecosystems and facilitates the reproduction process of robotic systems as well as associated experiments, *(ii)* the tool chain is usable for non-expert users, and *(iii)* the tool chain provides enough information about relevant artifacts of reproduced experiments. To this end we chose three different robotics systems, that are available in simulation, from three **independent** and spatially distributed institutions. We modeled all three systems, including an exemplary experiment, using the CITK.

In the user study, we asked researchers at Bielefeld University to replicate a formerly unknown system — using the CITK process and tools. After having potentially replicated the assigned experiment, we asked the researchers to assess the CITK based on a structured set of questions with regard to reproducibility issues identified in the current literature and the tool chain's usability. To the best of our knowledge this is the first user study in this matter: the investigation of a reproduction methodology for robotics incorporating usability metrics and based on actual "production" systems in a practical user study.

The remainder of this paper is structured as follows: In section II we will introduce the problem statements. In section III we will present related work from different fields of study. In section IV we will shortly recapitulate the concept and tools constituting the CITK. Section V will give an overview of the reproduced systems and example experiments, sections VI and VII will introduce the study design and results. Finally, we will discuss and conclude this contribution in VIII.

[1]Bielefeld University (CITEC/CoR-Lab), 33615 Bielefeld, Germany [flier,sschulz,swachsmu,swrede]@techfak.uni-bielefeld.de

[2]University of Lincoln (L-CAS), Lincoln, LN6 7TS, United Kingdom mhanheide@lincoln.ac.uk

[3]Italian Institute of Technology (IIT), 16163 Genova, Italy lorenzo.natale@iit.it

[4]Columbia University, NY 10027, United States jweisz@cs.columbia.edu

II. Problem Statements

A publication or article and the therein presented results are typically *the only* starting basis in order to technically reproduce and thus *practically* verify results. On the one hand, due to the commonly established peer-review process of journals and conferences, the theoretical (formal or mathematical) approval of results can be assumed as assured. On the other hand, the software artifacts which are the true **embodiments** of the ideas and contributions cannot easily be verified and reproduced. Why is that?

Given the fact, the paper of interest includes links or references to source code repositories or other relevant artifacts it is difficult and time consuming to identify and aggregate **all** relevant artifacts that are required for reproduction of a software intensive robotics system. These are for instance: **all** required software components, data sets, and their documentation [problem statement **a)** information retrieval and aggregation]. This is not surprising since regular publications are not (yet) intended to precisely list and describe this kind of information. However, a few journals and conferences already offer the opportunity to submit code alongside the corresponding publication, but this does not scale for entire robotics systems. Moreover, in order to practically replicate the system that produced the presented results it is indispensable to also provide information about the relationships between significant artifacts. These are, e.g., the *exact* versions (i.e. commit hashes) of software components in combination with an experiment-specific data set (recorded sensory input and actuator output for instance), hardware or setup variant utilized in the presented study [problem statement **b)** semantic relationships]. Thus, simply referencing source code repositories or binary downloads is not sufficient. Moreover, the entire software stack must be deployed, which is often non-trivial and time consuming for diverse ecosystems (problem statement **c)** software deployment). Linux containers, such as Docker[1], are a possible solution in order to mitigate versioning and system deployment issues. However, there are still more obstacles to overcome. One of them is system and experiment execution. If a download link to a docker container that contains the system of interest is referenced in an publication, it is still unclear how and what to execute inside the container in order to obtain and verify the published results [problem statement **d)** experiment testing, execution and evaluation].

Lastly, it is well known that a massive amount of time is spent on robotics experiments because they require significant effort with respect to initial prototyping, system development, integration testing, evaluation and conservation of results. These tasks are usually labour intensive — especially when performed manually. Thus in our opinion, after completing all the required work that is necessary to publish results, it is often hard to find the time and to motivate researchers to "make their work fully replicable", which implies gathering and provide all the required information as described above in a way that it is "ready to use". Tools are required in order to support researchers automating these tasks [problem statement **d)** experiment testing, execution and evaluation].

[1]https://www.docker.com

III. Computational Experiments & Engineering

In this section we present related work from different fields of study with respect to reproducibility of computational experiments — which include experiments in the robotics domain as well. As a first case in point, Schiaffonati et al. [16] point out that computer science, in contrast to fields like physics or biology, focuses on the abstract concept of computation and therefore on the *creation* of technological artifacts that implement these concepts. Thus, before *reproducing* an experiment the subject of study, software in this case, must often be *produced* beforehand. Furthermore, Schiaffonati et al. emphasize the inherently different disciplinary nature of computing that incorporates scientific and, at the same time, engineering efforts.

Not surprisingly, this circumstance opens up additional issues when reflecting on the role of reproducible results in the robotics domain and the way findings are published. Unfortunately, the impact of the engineering aspect is mostly neglected in current literature. Thus, the provision of "technical artifacts" as a useful instrument for reproduction is to be considered as crucial, e.g., a description of the software used, its parameters and configuration. This demand is also discussed by Jill Mesirov [13]. In her work, she discusses experiments where results were derived from simulation in chemistry, materials science and climate modeling. She claims that the scientists are often skilled and innovative programmers who develop and release sophisticated software packages (which sounds familiar to scientists in the robotics domain). Fellow scientists, who may not be software experts themselves, conduct studies utilizing these tools. Usually, the "standard usage" of these tools varies in a way that researchers combine them in novel ways. According to Mesirov this provides an enormous advantage. On the downside, the lack of implementation details and parametrization of follow-up experiments poses new challenges for scientific publications, replication and especially for traceability of intermediate and resulting artifacts. In this context, Schwab et al. [17] assert that publications, lacking those technical details are:

> *"[...] merely the advertisement of scholarship whereas the computer programs, input data, parameter values, etc. embody the scholarship itself."*

Unfortunately, references to the actual source code, parametrization, configurations and related artifacts of the "mashup" that produced these scholarships are usually omitted or unknown and therefore lost and thus — not reproducible. Fortunately, a few approaches can be found in current literature tackling these issues. Most recently, in "The Real Software Crisis: Repeatability as a Core Value" Krishnamurthi et al. [8] stated that:

> *"[...] software artifacts play a central role: they are the embodiments of our ideas and contributions."*

Subsequently, they describe a newly initiated artifact evaluation process that allows publishing authors (of accepted papers) to submit software alongside different kinds of non-software artifacts, e.g., data sets, test suites, and models that back their findings.

In "Reproducible Research — Addressing the need for data and code sharing in computational science", Stodden [18] proposed the following (and many others) goals for reproducible research: foster the development of web-based tools to facilitate the ease of use of software repositories and related artifacts. Publish code accompanied by software routines that permit testing of the software test suites, including unit testing and/or regression tests. Develop tools to facilitate both routine and standardized citation of code and data. Develop deeper communities that maintain code and data, ensure ongoing reproducibility, and perhaps offer tech support to users. Without maintenance, changes beyond individual's control (computer hardware, operating systems, libraries, programming languages, and so on) will break reproducibility.

Besides these (more general) approaches, researchers in the robotics domain also investigate current challenges with regard to reproducibility. In *Defining the Requisites of a Replicable Robotics Experiment* Bonsignorio et al. [3] list requisites based on an *exemplary experiment* in visual servoing. These requisites are, amongst others, provision of a) technical specifications of the robot used, b) technical specifications of the camera with respect to frame rate, resolution, [...] c) description of computers' used memory, computation power, operating system, [...] and d) used software libraries and configuration. Moreover, Bonsignorio et al. investigated how "Good Experimental Methodology" in diverse fields of robotics, towards publication of findings, must be carried out in order to help reviewers recognize, and authors write high quality publications of replicable experimental and theoretical work. These methods include, for example, that experimental results on real datasets must be publicly available, experiments in simulation should be made available including the simulator alongside with the setup used for the experiments and what set of objects or items were tested.

Another approach has been realized by the ROS community. Here, scientific reports are linked to their associated engineering component — software packages in this case — thus, the ROS wiki provides dedicated web sites[2] for a few publications. These sites aggregate source code, data sets and usage examples to improve the potential of reproducing results, published in related papers. Lastly, in order to provide a repository of aggregated artifacts the ROS community has recently released a newly created package index for software incorporated in the ROS ecosystem. This approach was mainly realized based on the question *"how do you know what ROS software is available, where to find it, and how to use it?"*. To solve this issue, a community site [3] was created that aims at helping users to search across all ROS repositories at once.

[2]http://wiki.ros.org/Papers
[3]http://rosindex.github.io/about/

To summarize: the examination of related work from diverse fields clearly indicates that reproducibility of computational experiments is an important and "hot" topic — not only in robotics research. Nevertheless, the increasing notion of software, input data, parameter values, etc. as the embodiment of scholarship itself is mutually shared across disciplines. However, robotics systems additionally introduce complexity due to their diversity and amount of required (often non standardized) artifacts.

IV. CITK in a Nutshell

Since reproducibility is considered a critical challenge in robotics research, we developed an integrated tool chain that incorporates the complete development, reproduction, and refinement process of robotics systems: *The Cognitive Interaction Toolkit (CITK)*. The CITK consists of a software tool set that is tied to an underlying artifact model, software development and reproduction process. This process defines how users, system engineers and empirical scientists interact with our tool chain. It has been inspired by current best practices in research and software development — especially in robotics. While the CITK tool chain and its associated development process is explained in greater detail in [10], the following paragraphs will introduce the reader to the basic concepts required in order to assess the study presented in Section VI.

The basis of the CITK is a domain model for reproducible robotics systems. This model is based on the concept of a *system version*. A system version consists of multiple software *component versions* (mandatory) — the functional building blocks. Versioning is enforced by, e.g., referencing specific source code repository tags, branches or binary release identifiers. Moreover, *documentation* such as wiki pages, API docs, etc. and *publications* as well as obtained *data sets* are also associated with the system version. An *experiment protocol* that was adhered in order to acquire the published results must also be provided along with a system version. Additionally, the system's software components and their configuration are frequently and automatically built, deployed and tested on a continuous integration (CI) server to provide functional monitoring. If feasible, a simulated experiment must also regularly be executed on a CI server, based on the experiment's protocol and data sets to prevent regressions over time.

From a *researcher's* point of view (illustrated by the orange path in Figure 1) the starting point for the reproduction and development process of a robotics system is the source code of her/his software components (Figure 1 (1)). These components are often written in different programming languages and thus make use of diverse build environments: CMake, Maven, Catkin to only name a few. Managing many different environments either requires convention over configuration and the use of ecosystem-specific tooling (cf. ROS catkin) or is a rather **cumbersome and time-consuming**

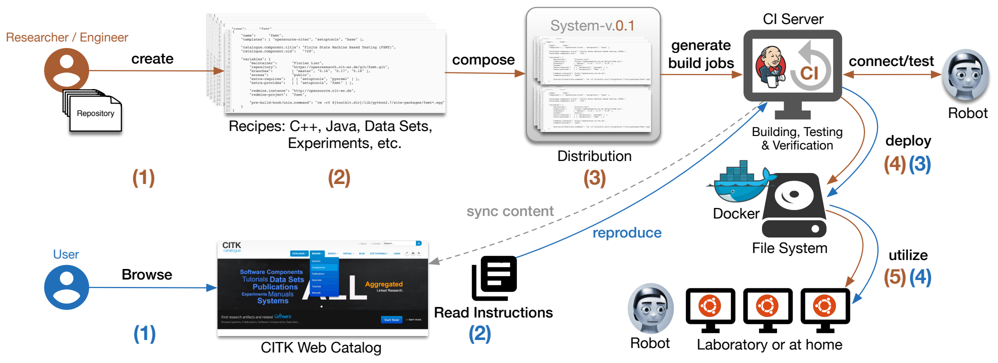

Fig. 1: Cognitive Interaction Toolkit: tool chain and work flow

task. We address this issue by applying a generator-based solution that utilizes minimalistic template-based descriptions — we call them recipes — of the different components that belong to a system distribution (Figure 1 (2)). Recipes are written by researchers and describe a software component by inheriting a template for, e.g., CMake-based software, or for downloading data set archives, an experiment description, or any other publicly available artifact that is required in order to replicate a system. Thus, the composition of different recipe types comprises a distribution file (Figure 1 (3)).

Distribution files are interpreted by a newly implemented, so-called "build-generator" tool. Firstly, the build-generator analyzes all included recipes based on the nature of their inherited template and the downloaded source code. Based on this information, the build generator *automatically* derives the following: a) how to build and deploy a component and b) the sequence in which components must be built in order to satisfy inter-component dependencies. Afterwards, the build-generator creates per-component build jobs and uploads them to a running (local or remote) CI server instance (Figure 1 (3)). At this point the researcher is provided with a set of build jobs that reflect her/his system. Additionally, a special build job is created that, if triggered, orchestrates the complete build and deployment process of the system. After all jobs are finished, the system is deployed (Figure 1 (4)) in the file system and is ready to use (Figure 1 (5)). Since setting up a CI server and the required plugins takes time and requires expert knowledge, we provide pre-packaged installations for CITK users. Moreover, we recently introduced deployment of CITK-based systems using Linux containers. For this, Docker [12] was integrated, the leading application for using Linux containers, to create, package, and distribute robotics systems. System descriptions and their meta data, e.g., source code locations, wiki pages, issue tracker, current build status, experiment descriptions, and so forth are frequently synchronized to a **web-based catalog that also implements the CITK data model** — providing a global human readable and searchable platform.

Besides gathering information about a system and automatically deploying it, successful reproduction also includes repeating tests and experiments. To address this in the context of robotics systems, we suggest to transfer the concept of an experiment protocol to the orchestration of software components involved in an experiment and to execute, test and evaluate software intensive experiments in an automated manner. To this end, CITK users are utilizing the FSMT [9] framework. FSMT is a lightweight and event-driven software tool that implements the aforementioned suggestions, formalizing experiment execution with respect to configuration and orchestration of invoked software components. It supports automated bootstrapping, evaluation and shutdown of a software system used in an experiment. An FSMT experiment description includes three mandatory steps a) environment definition, e.g, executable paths, required environment variables and runtime configurations, b) software component descriptions such as, path to executable plus command line arguments (*configuration!*) and c) success criteria. After specifying the environment, components and their success criteria, researchers must provide the execution order of their components. Here, empirical scientists may provide scripts (R, Python) to evaluate data produced in each run or trial, to generate plots for instance. These scripts are defined and executed like regular system components at the time the system is completely bootstrapped by FSMT, hence, is producing data. Here, existing solutions like workflow engines (i.e. Taverna) are not sufficient since they do not provide a deterministic execution behavior and do mostly not support component health checks. Moreover, worflow-engines are ususally data, and not event-driven. However, this formalization of an experiment protocol allows to consistently reproduce and acquire test results in an automated fashion. FSMT experiment descriptions and related material such as evaluation scripts are also deployed using the build-generator and the CI server. Furthermore, required hardware/robots can be connected to the CI server that runs an experiment — software stack respectively. Thus live sensor data can be evaluated directly when running an experiment incorporating hardware-in-the-loop. Alternatively, the CITK tool chain can also be instantiated **directly on a robot itself** — if feasible (cf. Figure 1).

From a *user's perspective* (illustrated by the orange path in Figure 1), who is interested in replicating an experiment, the starting point is the experiment representation in the web catalog that has been referenced in a *corresponding* publication (Figure 1 (1)). Here, she or he is directly pointed to a system version. The first step for the user is to browse the catalog and to load the page that contains a general textual description of the system and meta information. Subsequently, she/he follows the link to the build generator recipes comprising the system and downloads the required recipes[4]. Following the provided installation instructions for a pre-packaged CI server the user bootstraps the CITK environment on her/his computer (Figure 1 (2)). After starting the local CI server instance, the generator is invoked in order to configure the CI server with build jobs for the system. The last step is to start the installation process. Therefore, the user starts the aforementioned orchestration build job. After this job has finished, the system is deployed on the user's computer Figure 1 (3)). Now, the user can focus on executing experiments that have been defined and deployed alongside the system. These experiments are also available as build jobs on the CI server. Simply starting the associated build job is enough. Moreover, the web catalog entry for the system version lists all available experiments. Each experiment also comes with a description of how to execute it. After the experiment has been executed successfully, the resulting output, i.e., in form of a plot or textual report, can be found on the user's computer (as well as all required logs and data files). Besides the FSMT report, the generated plot can be compared to a reference plot from the catalog entry of the experiment Figure 1 (4)).

V. Target Systems and Experiments

After having briefly explained the CITK approach, we will now describe the systems and example experiments that were modeled using the CITK tool chain. It is noteworthy, that the three systems are of different nature when it comes to their deployment strategies and incorporated software parts. Even though the STRANDS system is mostly based on Ubuntu packages, a few components which utilize the ROS-catkin build system, were modeled. The iCub system is consistently built from source using CMake. The Flobi system is probably the most "complex" with regard to its deployment. Here, the different software components are built from source using: CMake, catkin, Python-setuptools, Maven and even some legacy shell scripts. Thus, by instantiating the CITK tool chain, the desired target system can be deployed including all relevant software components, required data sets, and a formalized and executable example experiment.

A. STRANDS

As a system aimed at long-term deployment of intelligent robotic functionalities in application domains of security and care, the STRANDS project has a high demand of robustness and, consequently, high software quality, to achieve its objectives of deploying robots for autonomous operation of 6 months. The scientific aims of STRANDS are to not only achieve long-term autonomous behavior of its deployed robots, but to actually *exploit* the experience these robots gather over such long run-times.

The system realized for replication is based on the continuous patrolling demo, inspired by STRANDS' security deployment system, where a SCITOS G5 robot equipped with two Kinect-type sensors, and a SICK laser scanner, continuously patrols a known environment to a) gather long-term data sets, and b) spot deviations from the normal routine. The system studied for this paper is composed of approximately 30 continuously running ROS components, enabling the robot to localize, plan path utilizing metric and topological representation, recover from navigation failures, and perceive an environment. The overall area patrolled by the robot is approximately $500m^2$ in an environment simulated using the MORSE simulator.

Task and expected result: initially, the system as described above must be *completely and successfully* deployed, and the example experiment must be started by the test subject — using the CITK tool chain. As soon as the system is running, the robot is requested to patrol two predefined waypoints in the simulated environment within 360 seconds. After the given time period, the system's functional components are shutdown and the evaluation phase starts. Thus, in the automated evaluation phase of the FSMT run, the robot's system logs are processed and evaluated due to the predefined success criteria. Only if the waypoints have been reached ("waypoint reached" reported in the waypoint-request component), the experiment is evaluated as successful. Finally, the evaluation results and associated log files are automatically presented to the subject. Based on the available information in the web catalog and the FSMT output, the subject must assess the outcome of the experiment.

B. iCub

In this case, the investigated system for replication is part of the example applications distributed with the iCub software [14]. This application is meant to help new iCub users approach the software. In short, the investigated iCub system replays the complete sensory data of the iCub as it was recorded during a former experiment. In the original experiment the robot was programmed to track a colored object with the gaze. In this example experiment, a tool is executed that replays the dataset with the original timing and format. Furthermore, the GUIs for visualizing the original images from the cameras and a 3D representation of the iCub are utilized. The latter is animated starting from the encoders and it includes a representation of the forces at the arms and the inertial feedback from the head IMU.

Task and expected result: again, at first the system must be installed and brought-up by the test subject. As soon as the system is running, the currently produced joint values of the iCub's right arm are collected (via Yarp middleware) and saved in a log file. After exactly 70 seconds all components are shutdown. Next, in the automated evaluation phase of the FSMT run, the recorded joint values are processed and evaluated. In this case, the recently recorded joint values are compared to a reference run (distributed with the experiment). If the total axis angle offset, between the reference run and the current run, for the right arm is smaller than 5 degrees (for the first and last data point in

[4]https://opensource.cit-ec.de/projects/citk/repository

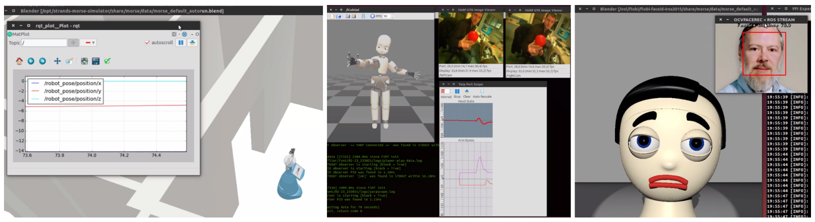

Fig. 2: Systems (ltr): STRANDS navigation, iCub balltracking and Flobi face identification

the trajectory) the experiment is evaluated as successful. Finally, the evaluation results and a plot of the compared joint angles are automatically presented to the subject. Based on the available information in the web catalog and the FSMT output, the subject must assess the outcome of the experiment.

C. Flobi

The robot Flobi was designed to facilitate social interaction by designing the exterior in a way that allows facial expression and emotional feedback, while at the same time avoiding the uncanny valley effect by using a comic style design [11]. The robot's neck features three degrees of freedom (DOF) for pan-, tilt-, and roll motion. In order to facilitate a variety of facial expressions, a total of 14 actuators are responsible for moving the eyes, the eyebrows, the individual eyelids, and the mouth. In this system setup Flobi's emotional feedback, as well as face detection and recognition capabilities are demonstrated using the MORSE-based Flobi simulator. Therefore, a total set of four images are presented to the robot. The images are processed by a person identification component. Based on the classification result, a corresponding emotion is triggered and executed by the robot's high- and low-level control framework and visualized in the simulator.

Task and expected result: As usual, at first the system must be deployed and instantiated by the subject. The example experiment must be started. As soon as the system is running, the virtual Flobi starts to identify the presented persons, a corresponding emotion is triggered. After 25 seconds all components are shutdown. In the evaluation phase the robots emotion states are processed and evaluated. In this case, the system is supposed to report "all four persons found". Finally, the evaluation results and associated log files are presented to the subject. Based on the available information in the web catalog and the FSMT output, the subject must assess the outcome of the experiment.

VI. Study Goals & Design

The main objective of this study was to evaluate the CITK tool chain from a user's perspective (cf. Section IV). In order to test our hypothesis as introduced in section I and to gain insight, we defined three key questions targeting different aspects of our tool chain: *i)* are the users able to technically reproduce an example experiment at all? *ii)* how is the usability of the tool chain perceived? *iii)* does the tool chain provide enough information to confirm and assess the expected outcome of an experiment?

Besides recognizing potentially unidentified disadvantages or benefits of our tool chain, these three key questions also address issues identified in the current literature (cf. Section III). For instance, before actually reproducing an experiment, the target system including all required software component *versions*, as well as their configuration parameters and execution sequence must be identified and deployed on the user's machine. Thus, question *(i)* addresses the "frequently neglected" impact of the engineering aspect on computational experiment replication. Subsequently, in order to support users in the replication process, suitable transparent and open tools are required (cf. Stodden). Hence, the usability and effectivity of these tools will play an important role towards reducing the effort spent on reproducing reported results — for reviewers as a case in point. This issue is addressed in question *(ii)*. Lastly, the outcome and success criteria of an experiment must be clearly stated and traceable (cf. Bosignoro) — this issue is addressed in question *(iii)*. In order to test our hypothesis and to answer the aforementioned questions, we conducted this study as follows.

We recruited twelve subjects from the Faculty of Technology at Bielefeld University. One subject was a postdoctoral researcher, nine were PhD candidates and two subjects were in their final year of a master's degree program. The subject's academic background was: 9 participants had a computer science background, while mechatronics, interaction design and technologies, as well as chemistry was reported by 3 subjects. Two participants were women, ten men. All subjects confirmed that they were neither involved in the development of the CITK tool chain, nor that they were familiar with the assigned system or experiment. The subjects were instructed and supervised by a student worker who was also not part of the CITK development team.

The subjects were asked to reproduce one of the experiments presented in section V by utilizing the CITK tool chain. Before actively starting the replication process, the subjects were instructed to read a documentation manual (PDF file) that specified all necessary steps, e.g., required shell commands in order to deploy the assigned system (cf. Section IV). The replication process incorporated the following tasks: **1)** bootstrap the tool chain and deploy the entire target system. **2)** navigate to the CITK web catalog, a dedicated link was provided in the instruction set, and gather information about the referenced experiment, its expected outcome and how to execute the experiment. **3)** execute the experiment and fill in a questionnaire. In the interest of maintaining consistency across all experiments, the instruction manuals, as well as the web catalog sites, were structured alike. All subjects utilized the same computer to provide a uniform technical basis. The manual PDF was accessible throughout the whole process and the subjects were allowed to copy&paste required shell commands. Eventually, every experiment was expected to be reproduced four times by three different subjects.

VII. Evaluation Methods & Obtained Results

In this section we will present the applied evaluation methods and the study results. The discussion of these results and their potential implications will be presented in the final section of this contribution. In the questionnaire, we used a 5-point Likert scale ranging from **strongly disagree (1)** to **strongly agree (5)** where applicable. The remaining categories were either "Yes/No" answers, or statements such as "one — more than one — more than five".

All participants had no or little knowledge about the topic of computational reproducibility. However, surprisingly 84% of the subjects strongly agree that reproducible and traceable research is an important topic. One subject neither agrees nor disagrees (3) and one subject disagrees (2) with this statement. Thus, altogether the subjects had no or at most little theoretical and practical experience in reproducing third party systems and experiments.

The first notable result, based on the subject's self assessment and the supervisor's evaluation, is that **all 12 subjects** were able to reproduce their assigned experiment (key question *i*).

The second set of questions in the questionnaire was targeted at the tool chain's usability (key question *ii*). Therefore, we applied the System Usability Scale (SUS) as proposed by John Brooke [4]. The SUS is an elementary, standardized ten-item attitude scale often used in systems engineering in order to provide an overview of the system's usability. The resulting average score of **77.5**, as well as the average score of each item, is depicted in Table I

TABLE I: CITK SUS Evaluation

Subject	SU Questions 1 - 10										SUS
1 - 12	4,0	1,8	4,1	2,1	4,0	1,3	3,9	2,1	4,0	1,7	**77,5**

Individual and overall scores are average values

According to Bangor et al. [1] an average SUS score above 70 can be considered "good". Thus, based on the subjects rating, the usability of the CITK tool chain is more than acceptable. However, the predicate "excellent" would have required a score equal to or above **85** (cf. Figure 3).

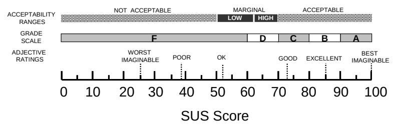

Fig. 3: Acceptability scores, school grading scales and adjective ratings in relation to the average SUS score [1]

In the third and last section of the questionnaire we evaluated key question *iii*. The results are depicted in Figure 4 and 5. First of all, it is noticeable that the majority of the participants assessed that it was clearly stated what was measured in the experiment. The median value for this statement is 4 (agree) and the majority of all answers reside within a range of 3.5 and 5 (first and third quartiles). Furthermore, is was also clear to the subjects how the results were measured (median is value is 4). However, it is also noteworthy that the results for this statement are inferior to the previous statement. Most of the participants also perceived that the obtained results in the experiment addressed the system's capabilities in a reasonable manner. Lastly, the participants strongly agreed that it was easy to assess whether the experiment was successfully replicated, which is a promising result. Here, the mean value is 5, the first and third quartiles reside between 4.5 and 5. There is, however, one outlier assessing this statement with neither agree nor disagree (3). However, all subjects approved that automatable execution of experiments is essential for replication (strongly agree 75% / agree 25%).

Furthermore, we asked the subjects about their notion of included software components and their configuration, as well as information about the hardware (computer) used in the original experiment. With regard to included software components the median value is 4. The median value for included software configurations and parameters is comparatively low with 3.5 (first and third quartiles between 3 and 4). Surprisingly, the subjects also couldn't neither agree nor disagree (median 3) to the statement that sufficient information about the hardware used in the original experiment was available. It is noteworthy, that the first and third quartiles range from 2 to 5 which depicts a wide distribution of ratings. Lastly, the participants disagreed to the statement (median 2) that they could have achieved the same results without using the tool chain.

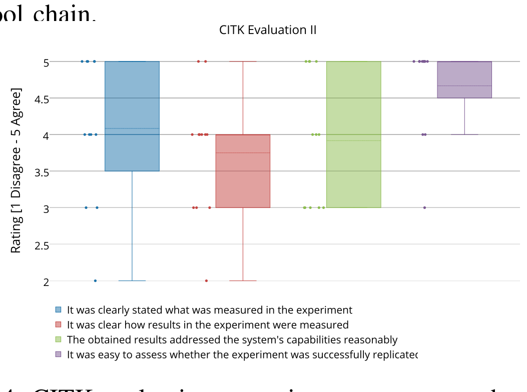

Fig. 4: CITK evaluation: experiment outcome and metrics

In order to also make this study *traceable*, the instruction manuals and the questionnaire results, can be found on the CITK catalog website [5]

VIII. Discussion & Conclusion

The goal of this study was to test the following hypotheses: *(i)* the tool chain is applicable for different ecosystems and

[5]https://toolkit.cit-ec.de/citk-evaluation-study-i

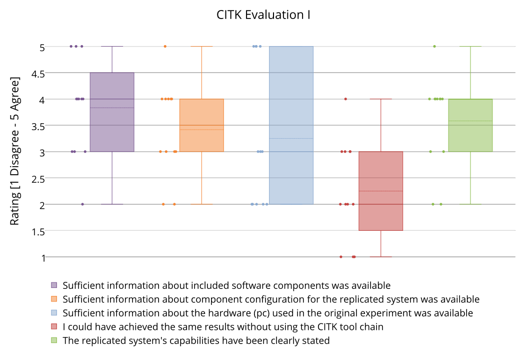

Fig. 5: CITK evaluation: information about included software components and configuration

facilitates the reproduction process of robotic systems as well as associated experiments, *(ii)* the tool chain is usable for non-expert users, and *(iii)* the tool chain provides enough information about relevant artifacts of reproduced example experiments. Concerning *(i)*: due to the fact that all systems and experiments were successfully reproduced by all 12 subjects, using the CITK tool chain, we assert this hypothesis is verified. Since all subjects had no or little experience in reproducing third party systems, and an average SUS score of 77.5 was reached, we also assert *(ii)* is verified. Verification for *(i)* and *(ii)* is additionally backed by the facts that is was easy for the subjects to assess whether the experiment was successfully replicated and that it was clear how and what was measured. Moreover, the subjects could associate their experiment result with the system's capabilities. Lastly, the subjects stated that they could not have achieved the same results *without* the tool chain. Unfortunately, hypothesis *(iii)* is not as well supported by our results as the others hypotheses. While the subjects approved there was sufficient information about included software components, it seems that roughly one half of the subjects could not determine whether there was enough information about component *configuration*. However, since the component configuration is explicitly modeled in the tool chain (cf. FSMT), we assume that the available information source, the web catalog in this case, needs refinement of the visual representation for this kind of artifact. A first assumption is that, due to the fact that FSMT configurations are "only" represented as hyperlinks on the experiment web sites, only the minority of the subjects did actually follow that link. Thus, although there are positive tendencies, we assume this hypothesis can not be clearly verified.

Current limitations: since our approach heavily relies on the provision of previously recorded, or otherwise accessible sensor data (e.g. live via network/middleware), highly interactive experiments, such as in HRI are difficult (and sometimes impossible) to capture and thus difficult to replay/reproduce. Despite the fact that hardware is explicitly modeled in the CITK, an (open) standardized hardware description format is still lacking integration. The CITK tool chain is directly runnable or at least connectable to robots that are based on "consumer hardware" such as the PR2 or iCub control PCs. However, specialized embedded systems for instance are currently locked out — if not available in simulation.

To conclude: in this contribution we assessed the applicability and usability of a novel tool chain that addresses current issues with respect to reproducibility in the robotics domain in an applied and — so far — unique study. We have shown that scientists were able to replicate a formerly unknown exemplary experiment using our approach. While we were not able to clearly verify all of our hypotheses, we consider the results promising. We can hopefully expose our approach to the general community in order to gather additional data and use cases, as well as fields of application.

We invite the reviewers to watch accompanying video submission and to browse the catalog:
https://toolkit.cit-ec.uni-bielefeld.de/
In order to get access to all features we provide a temporary guest account.
user: guest password: AXJc9rZgW

References

[1] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[2] F. Bonsignorio, J. Hallam, and A. del Pobil. Good experimental methodology-gem guidelines, 2007.

[3] F. Bonsignorio, J. Hallam, and A. del Pobil. Defining the requisites of a replicable robotics experiment. In *RSS2009 Workshop on Good Experimental Methodologies in Robotics*, 2009.

[4] J. Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[5] D. Brugali and P. Scandurra. Component-based robotic engineering (part i) [tutorial]. *Robotics Automation Magazine, IEEE*, 16(4):84–96, December 2009.

[6] H. Bruyninckx. Open robot control software: the orocos project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2523–2528. IEEE, 2001.

[7] S. Cousins, B. Gerkey, and K. Conley. Sharing software with ros [ROS Topics]. *Robotics & Automation Magazine*, 17(2):12–14, 2010.

[8] S. Krishnamurthi and J. Vitek. The real software crisis: Repeatability as a core value. *Commun. ACM*, 58(3):34–36, Feb. 2015.

[9] F. Lier, I. Lütkebohle, and S. Wachsmuth. Towards automated execution and evaluation of simulated prototype HRI experiments. Proc. 2014 ACM/IEEE Int. Conf. on Human-robot interaction, pages 230–231. ACM, 2014.

[10] F. Lier, J. Wienke, A. Nordmann, S. Wachsmuth, and S. Wrede. The cognitive interaction toolkit – improving reproducibility of robotic systems experiments. SIMPAR 2014, LNAI, pages 400–411. Springer International Publishing Switzerland, 2014.

[11] I. Lütkebohle, F. Hegel, S. Schulz, M. Hackel, B. Wrede, S. Wachsmuth, and G. Sagerer. The bielefeld anthropomorphic robot head. 2010 IEEE International Conference on Robotics and Automation, pages 3384–3391. IEEE, 2010.

[12] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014.

[13] J. P. Mesirov. Computer science. accessible reproducible research. *Science (New York, NY)*, 327(5964), 2010.

[14] L. Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, and G. Sandini. The icub platform: A tool for studying intrinsically motivated learning, 2013.

[15] M. Quigley et al. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[16] V. Schiaffonati and M. Verdicchio. Computing and experiments. *Philosophy & Technology*, pages 1–18, 2013.

[17] M. Schwab, M. Karrenbach, and J. Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering*, 2(6):61–67, 2000.

[18] V. C. Stodden. Reproducible research: Addressing the need for data and code sharing in computational science. *Computing in Science & Engineering*, 12(5):8–12, 2010.