

DESIRE WEB 2.0 - Integration Management and Distributed Software Development for Service Robots

Ulrich Reiser, Regina Klauser, Christopher Parlitz and Alexander Verl

Abstract—In the last decades many robotic demonstrators were built, the vast majority of which are constrained to a limited set of operations, often focusing on the development of single robotic components like navigation, manipulation, human-machine interaction, etc.

These specialized robots thus are not intended and commonly also not flexible enough to solve many complex tasks, in particular concerning dynamic and unstructured environments, requiring the combination of different components.

More recently there have been efforts to create robots that are capable of performing various tasks in households or public domain and are dependable enough for every-day use (e.g. DESIRE [1]). This requires the integration of many expert technologies from the different robotic domains and different developers that are located at different places. A capable integration management is needed, including not only software engineering tools but support for distributed development and remote testing on robot hardware in particular.

This paper proposes software engineering methods and tools that support integration and a new webportal for the distributed development and remote testing of service robots. The webportal permits centrally managed configuration, control and monitoring of software component processes through a user interface, that is available at any time from any place, through arbitrary operating systems. It is shown in the results, that the effort for organisation and coordination of hardware tests could be substantially reduced in the DESIRE project by the usage of the webportal.

I. INTRODUCTION

The development of robotic systems has always been a great challenge, all the more if different partners at different locations are involved. It is widely agreed that a high level of integration is necessary, if robotics wants to overcome the current state. At present, robotic applications are mostly designed for special classes of problems. To get robots that are capable, dependable and flexible enough to solve many different tasks in different environments, the contribution of experts from all robotic domains (navigation, mobile manipulation, perception, planning, etc.) with many different backgrounds (mechanical and electrical engineering, computer science, etc.) is required. In joint projects that aim to integrate on a common hardware several distinct challenges are encountered:

- In order to specify the target system, first of all a common language in the sense of shared notations and ideas has to be established throughout the different domains, reaching from low level control to symbolic planning.

This work was funded as part of the research project DESIRE by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IME01A

- A capable and flexible software architecture is necessary that allows the integration of many heterogeneous components.
- An appropriate order for the integration of single components needs to be determined by comprehensive methodologies for integration planning.
- Most commonly, the hardware is only available at one location while component suppliers are distributed. This makes an extensive support for distributed development and remote testing necessary.

There have been many efforts recently concerning software and integration frameworks, more or less dedicated to a special domain (e.g.[2],[3],[4]). The player/stage project has put much effort on supporting different robotics hardware to relieve developers from writing hardware drivers [5]. In the joint project DESIRE methodologies for distributed development in robotic projects have been explored [6]. There has also been great interest on web-based control and networked robots [7]. Belousov, e.g. introduces a framework based on java3D to teleoperate robots in fast changing environments [8],[9].

Most of these frameworks are confined to the integration of software components. A framework supporting the integration of components on hardware, i.e. the easy deployment of components from distributed suppliers while sustaining the operability and testability of the system, is still missing.

This paper proposes a web-based application that integrates *both* distributed development *and* remote testing on common robotic hardware. It was developed within the DESIRE project and supports integration and web-based control on a quite abstract level, such that it can be used for many robotic domains and many different robot platforms. It is based on the collaboration platform Trac [10], that already contributes a lot of standard integration tools like a version control system.

The paper is organized as follows: first the service robotics initiative DESIRE is presented as a case study of distributed development. It is shown, which challenges and problems arised during the integration on one common hardware platform. The next chapter introduces already existing software engineering tools appropriate for large integration projects. Furthermore the webportal developed in DESIRE is presented, which provides special tools to relieve integration and remote testing. Finally an evaluation of the webportal with respect to the DESIRE project is given along with an outlook on future work.



Fig. 1. Technology Platform developed in DESIRE [1]

II. DISTRIBUTED SERVICE ROBOT DEVELOPMENT - A CASE STUDY

In contrast to many joint service robot projects of comparable size, the research project DESIRE [1] (project runtime 10/2005 - 09/2009), funded by the German Federal Ministry of Education and Research (BMBF), aims at integrating all software components on one common hardware platform, shown in fig. 1. The main goal of the project is to create a technology carrier to advance current service robot components towards a robust and dependable state, suitable for every-day use.

More than 14 partners from universities, research institutes and industry supply components to that platform. In the beginning of the project, a common knowledge base as well as common notions and ideas had to be established. A long phase of specification was undergone in order to be sure about *what* exactly to do. end of the project. An integration plan was generated which scheduled the single integration steps and defined certain milestones associated with fixed dates.

From the first integration meeting it became clear that this integration plan could not be adhered to. The main challenges and problems that have been encountered were quite trivial, although the meetings had been prepared very carefully:

- Although all interfaces had been defined in advance, adaptations of component interfaces during the meetings took a great amount of time.
- Often only one developer was working on a task, while the others had to wait on his completion. Thus the

meetings were often not very efficient.

- For the debugging of one single component in the system, many partners were needed.
- Many components could only be operated by individual developers. Basically all component developers had to participate at the workshops, even if merely to start and stop their components on demand.
- The absence of only one partner due to vacation or illness retarded the testing of the complete system.
- The time, cost and effort of every testing and integration participant exceeded the budget of all partners by far.
- The components were distributed on the computers of the developers, such that the robot was not operable unless component suppliers were on-site.
- The project advanced very slowly between the integration meetings: necessary tests (if only little interface tests) often required other partners and so the development was stuck until the next meeting.

Generally, progress could only be achieved by many integration meetings with many attendants. The number of possible integration meetings is, however, limited by finite travelling budget, timing constraints and researcher availability. Conference calls between the meetings and a virtual private network improved the situation a little bit, but progress was still mainly accomplished during the integration workshops. The main problems could thus be summarized with the following: time needed for scientific progress was mainly spent on integration issues.

Fig. 2 shows the relative amounts of time spent on specification, functional development of single components and integration of components on the technology platform. The first year was more or less dedicated to a thorough specification of the system, covering both component and system level (in the form of target scenarios). Derived from this specification, interface descriptions were formally defined, such that integration of the single components could be started very early in the development process. The effort of integration, however, was harshly underestimated and lasted for more than one year, substantially due to the points mentioned above.

Nevertheless, the consortium was still absolutely convinced of the necessity for integration, such that methods and tools were developed to reduce integration overhead. Gradually more and more tools to support integration were introduced, e.g. a virtual private network and at the end of the third year the webportal proposed in this paper. The webportal will be developed and evaluated further in the remaining 6 months of the DESIRE project.

The authors of this paper believe that similar problems mentioned above are encountered in many projects with a comparable amount of component suppliers that are developing on a common hardware.

III. GENERAL SOFTWARE-ENGINEERING METHODS AND TOOLS

The software development process can exploit many development tools and methods. Following described technologies

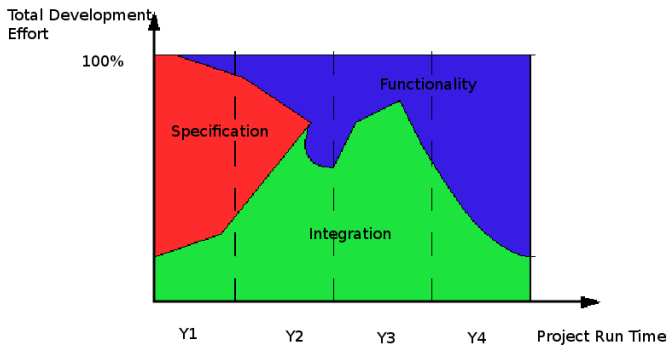


Fig. 2. Proportion of specification, functional development of components and integration with respect to the total amount of development time in the DESIRE [1] project.

are most fundamental and essential instruments used in service robotics projects.

- Version Control Systems record versions and releases of each part of an evolving software system. It is primarily used to maintain source code revisions, but it is also useful to manage multiple revisions of other kinds of artifacts such as plans, specifications, documentations, test cases, logs, etc. The historical information kept in version control systems allows tracing faults across versions and collecting data for improving process.
- Bug Tracking Systems help testers to document and monitor system faults and enhancements. Developers obtain a structured overview of software faults, enhancements, tasks and their history of development. Bug Tracking Systems can also be used to generate workflows.
- Continuous Integration (CI) is a software development concept that involves regular automatic compilations and integration tests of all software units. It is often associated with Agile Software Development approach such as Extreme Programming. On the other hand this concept is very common and can be considered as best practice in software development [11], [12]. The main purpose of CI is to discover and eliminate integration bugs as early as possible. Compared to the conventional CI approach the usage of an integration tool provides a better overview of compilations, tests and related bugs, as well as features for build statistics.
- Most common software development processes schedule the test and integration phase as one bounded block. Alternatively testing and integration can be scheduled successively after each process step. Every accomplishment of a part of a software system triggers a run of all tests. The developers execute software tests on a daily basis. A compromise between these extremes is the successive testing after a module is completed. In the early stage of implementation the so called unit tests deal in conventional software architectures with individual modules, in object oriented architectures with classes or class packages.
- Collaborative Software provides relevant objective in-

formation or facts about the project, synchronous and asynchronous communication instruments such as instant messaging, e-mail or video-conferencing and serves with tools for sharing of documents and data.

IV. DESIRE WEB PORTAL

The web interface developed in DESIRE is based on the open source collaboration platform Trac, which is composed of different plugins. In the following, already existing Trac plugins are presented, that are used by the webportal. Subsequently new plugins for integration and remote testing support are introduced.

A. Trac Collaboration Platform

Web-based project management and collaboration platforms are provided as a free service for uncommercial purposes. The providers are big software hosting platforms such as Sourceforge [13], JavaForge [14] or BerliOS Developer [15].

Collaborative software development platforms are essential instruments and very common in Open Source development. Most popular examples of available open source solutions are GForge [16] and Savane [17], which are free software forks of the SourceForge software. Increasingly widespread is the platform Trac [10].

All software project management platforms offer version control, bug tracking, project and document management, e-mail integration, discussion forums, survey functions and real time communication. The extensive features of such platforms are focused on collaboration in development of a software system.

Trac stands out as a lightweight and transparent solution. The component based and quite simple architecture of Trac simplifies project-specific enhancements.

Beside the standard platform modules such as *subversion browser*, *bug tracking system* and *wiki pages*, Trac offers the additional project management modules *Roadmap* and *Timeline*. Roadmap is a special view on the ticket system of Trac. Project milestones are specified in this module. The represented project progress depends on the relation between open and closed tickets which may be tasks or work packages. Timeline is a chronological view of all project events such as repository commits, changes in tickets or wiki pages, builds or achieved milestones.

For the realisation of continuous integration a built-in integration server module is used [18]. The integration server eases the monitoring and analysis of automatic builds and tests.

The most important advantage of Trac is the integrity and the consistent linkage between Trac modules. A change in the source code that fixes a bug can be linked to a bug ticket through a keyword entry in the commit message. The link to the revision number will show up in the corresponding ticket. The related build in the continuous integration module is bound through the revision number. This change can be documented in the wiki pages in detail. Distributed teams

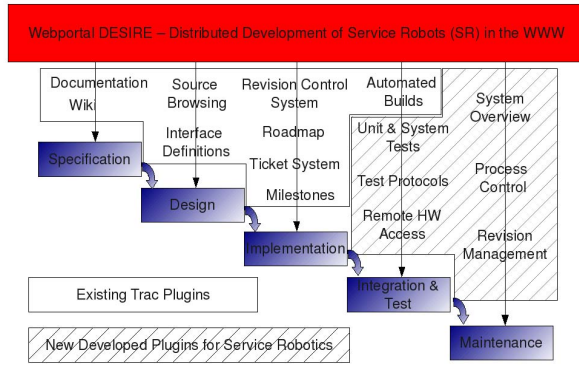


Fig. 3. Supportive tools comprised in the webportal with respect to the different development phases.

gain more clarity and continuity in the software development process.

In addition to the plugins that are already available, the DESIRE webportal offers enhancements supporting special features for service robotics. As pointed out in section II, in particular remote testing is important for projects integrating on common hardware. To be able to support remote tests, access to the computers on the remote platform must be provided via world wide web in an appropriate manner. Furthermore it must be possible to operate components on the robot, preferably without knowledge of the exact location of the components on the remote platform computers.

Fig. 3 gives an overview of the features of the webportal. The functions in left box already existed as Trac modules (see section IV-A) while the hatched functions mark plugins developed in DESIRE. The new features are described in the following subsections.

B. Remote Hardware Access

The DESIRE webportal runs on a dedicated server, which is connected both to the internet and to the platform computers (see fig. 4). This allows various scenarios of distributed development and testing: if, e.g. a couple of developers (C) are testing their components locally connected to the platform, developers A and B located at different sites can participate transparently of their location at the integration and testing session via the webportal. As the remote operation of hardware components like robot manipulators or mobile platform is commonly very critical, be assured by the Trac authentication module, which grants access to safety critical components only to authorized persons.

C. System Overview

The remote developers, however, need the same information about the current state of the platform as the local developers do. Therefore a new plugin was created that permits centrally managed configurations, control and monitoring of

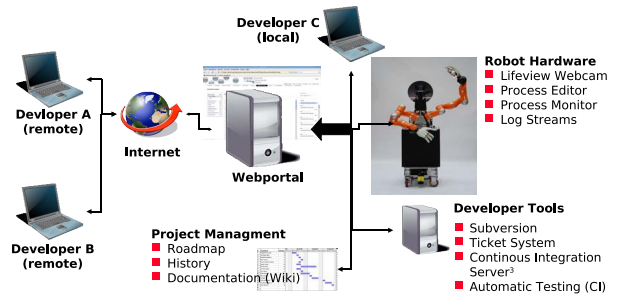


Fig. 4. Typical distributed development set-up in a service project with centralized hardware.

software component processes. The developer is able to perceive at one glance, which components are online and their current states via web log streams. Graphical applications such as simulations or visualisations are supplied through a graphical desktop sharing system (e.g., tightvnc [19]). For the video surveillance two network cameras are installed in the test area where the robot hardware is placed. The video recordings are also available through the web interface.

D. Process Control

In addition to monitoring of the current running components, developers are able to start and stop components. Note that this is possible without knowledge of the physical location on the computing hardware. New components can be integrated easily into the web interface: the administrator of a component merely needs to provide the following configuration information once:

- deployment of the component (host address, operating system, authentication information)
- scripts to start and stop the component
- log file name

As component control is residing on process level of the operation system, it is abstract from the specific implementation of the component and thus very flexible. Currently, Linux and Windows operating systems are supported. Once this information is given, the component can be operated without knowledge of the insides and the deployment.

V. RESULTS

In combination with the standard plugins of the collaboration platform Trac, the newly created plugins of the webportal presented in section IV provide the following improvements of the integration and testing phase in the DESIRE project:

Reduction of Personal Dependencies

Developers don't need to be on-site any more for integration tests concerning their components. Through the

webportal components can be operated, and even changes to components can be carried out easily.

Transparency of Software Updates

As soon as changes to components are uploaded to the version control system, the continuous integration plugin automatically builds the component and, provided that there are no errors, deploys it according to the configuration set in the process control plugin. This prevents on the one hand corrupt software revisions and ensures on the other hand that always the most current versions of the components are used. Note that software updates of a component are thus made transparent to other developers using that component.

Unit and System Tests

Based on the continuous integration plugin that performs automatic builds, additionally unit tests can be executed. If the developers responsible for a component create these unit tests thoroughly, the integrity of components can be assured at any time. To prevent side effects due to changes of dependent components (e.g. interface adaptations), integration test spanning several components are advisable. Even system tests might be executed automatically.

Increase of System Operability

As components can easily be started and stopped via a web browser, it is not necessary any more to have all component responsables on-site to operate the platform. The operator needs no deep knowledge of the involved software components, such that only hardware components like robot manipulators have to be supervised appropriately.

Reduction of Organisation Effort and Costs

The less persons are necessary to integrate and test a certain functionality, the easier is the integration planning and coordination of activities. Moreover, travelling expenses can be reduced.

Fig. 5 depicts the number of integration meetings and the number of attendants during the project run time. Obviously the number of meetings and the number of attendants increased before the project milestones MS2 - MS4. The number of attendants, however, could be significantly reduced by remote testing. The introduction of a virtual private network after the third milestone decreased the number of necessary developers on-site, but still required them to be online remotely. Only the webportal, that was developed after milestone 4, substantially reduced the integration and testing effort.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The experiences made in the DESIRE project, representative for service robotic projects with many distributed component developers, made evident that extensive support of distributed development and remote testing is crucial for an efficient project progress. The webportal that was developed within DESIRE supports project members in all phases of development from specification over design to implementation and test. Furthermore it provides a quick overview about both the implementation state and the current robot state and sustains the operability of the platform.

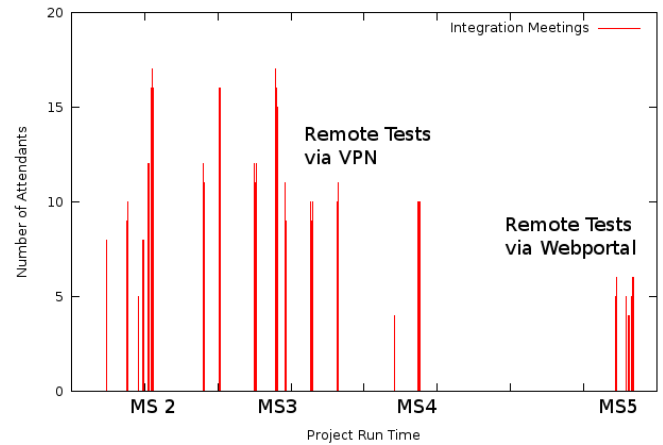


Fig. 5. Integration Meetings during the project run time of DESIRE. MS2-MS5 indicate the project milestones.

B. Future Work

The webportal as presented in this paper still offers a lot of potentials, in particular for automatic testing and ease of operation of hardware platforms. The next enhancement steps will be plugins for automatic unit, integration and system tests, the results of which will be documented in order to both detect errors and report them via the ticket systems and also to capture the current status of the performance of the system. This premises that the tests are designed not only to detect errors but also to provide measurements.

The current state of the system can be associated with the corresponding software revision, such that development progress can be expressed in numbers and different implementations of algorithms can be compared on a defined bases. On top of that a release management could be installed, that provides the easy and fast switching from one software state to another consistently.

Furthermore, it is intended to provide an interface for component configuration, e.g. via xml-Files, such that component parameters can be changed via the webportal.

The allover goal of the future webportal development is to make the platform available for a larger community to implement new algorithms and components on capable hardware in an easy manner with as little overhead as possible.

It is envisaged to make the new developed plugins available to the trac and robotics community.

REFERENCES

- [1] "Deutsche Service-Robotik Initiative (DESIRE)." [Online]. Available: www.service-robotik-initiative.de
- [2] S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer, "An XML based framework for cognitive vision architectures," *International Conference on Pattern Recognition*, vol. 1, pp. 757-760, 2004.
- [3] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06) Workshop on Robotic Standardization*, 2006.
- [4] C. Côté, D. Létourneau, F. Michaud, and Y. Brosseau, *Robotics System Integration Frameworks : MARIE's Approach to Software Development and Integration*. Heidelberg: Springer-Verlag, March 2007, vol. 30.

- [5] B. Gerkey, R. Vaughan, K. Sty, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: A robot device server for distributed control," in *In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Wailea, Hawaii*, 2001.
- [6] U. Reiser, C. Mies, and C. Plagemann, "Verteilte Software-Entwicklung in der Robotik - ein Integrations- und Testframework," *Robotik*, May 2008.
- [7] N. Hagita, K. Schilling, and D. Song, "IEEE Society of Robotics and Automation's Technical Committee on: Networked Robots," 2007. [Online]. Available: <http://faculty.cs.tamu.edu/dzsong/tc/index.html>
- [8] I. R. Belousov, S. Chebukov, and V. V. Sazonov, "Web-based teleoperation of the robot interacting with fast moving objects." in *ICRA. IEEE*, 2005, pp. 673–678.
- [9] I. R. Belousov, J. Tan, and G. J. Clapworthy, "Teleoperation and java3d visualization of a robot manipulator over the world wide web," *Information Visualisation, International Conference on*, vol. 0, p. 543, 1999.
- [10] Edgewall Software, "The trac integrated scm and project management," June 2008. [Online]. Available: <http://trac.edgewall.org>
- [11] K. A. Lee, "Realizing continuous integration," September 2005. [Online]. Available: <http://www.ibm.com/developerworks/rational/library/sep05/lee/>
- [12] M. Fowler, "Continuous integration," Mai 2006. [Online]. Available: <http://martinfowler.com/articles/continuousIntegration.html>
- [13] SourceForge, Inc, "What is sourceforge.net?" 2008. [Online]. Available: {<http://apps.sourceforge.net/trac/sourceforge/wiki/What%20is%20SourceForge.net?>}
- [14] Inland Software, "Javaforge scalable project hosting," March 2009. [Online]. Available: <http://www.javaforge.com/project/11>
- [15] FOKUS, "Berlios the open source mediator," 2009. [Online]. Available: <http://www.berlios.de/index.php.en>
- [16] The GForge Group, "Gforge advanced server," March 2009. [Online]. Available: <http://gforge.org>
- [17] The Gna! people, "Savane - Zusammenfassung." [Online]. Available: <https://gna.org/projects/savane>
- [18] Edgewall Software, "Bitten continuous integration rethought," June 2008. [Online]. Available: <http://bitten.edgewall.org>
- [19] "TightVNC Software." [Online]. Available: <http://www.tightvnc.com/>