How to integrate Webservices in Embedded System Design?

Achim Rettberg, Wolfgang Thronicke
University of Paderborn & Siemens Business Services / C-LAB,
Fuerstenallee 11, D-33102 Paderborn,Germany
Tel:. +49 5251 606110, Fax: + 49 5251 606065,
Email: achim.rettberg@c-lab.de, wolfgang.thronicke@c-lab.de

Abstract: The structure of Internet applications and scenarios is evolving rapidly. New software architectures and formats for transfer of data and site-spanning interoperability are emerging and reshaping the realm of web-centered computing . These changes are having repercussions that will change the established methodologies of design processes and business-to-business applications. Therefore, these effects on the domain of the electronic design automation (EDA) have to be considered and their validity shown.. In this paper we present an approach to exploit webservices technology in the field of embedded system design.

Key words: webservices, embedded system design, collaborative design

1. INTRODUCTION

In general the structure of Internet applications is changing rapidly. New information technologies and standards are emerging and - together with new infrastructures (high speed internet, wireless applications, UMTS) - design processes and business-to-business transactions are reshaping.

In addition to Internet technology that has been a true enabler for distributed technologies and applications the development in this area is shifting towards service-oriented structures. This falls into line with the evolution of programming paradigms: Object-orientation denotes the view of a program during design and execution as a collection of objects that send messages to each other invoking certain qualified operations. Deliberate

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35599-3_29

B. Kleinjohann et al. (eds.), *Design and Analysis of Distributed Embedded Systems*

design of software-systems seemed to become feasible and several methodologies have proven their value in this field, like UML-based techniques [12], [13]. However, the notation had to be augmented to cope with describing distributed systems effectively. Using *distributed objects* complex networked applications with high interoperability could be specified. On the one side, definition and implementation on object level is still the adequate design style for tightly coupled components, on the other side with respect to Internet scenarios the concept of interacting *services* is now the state-of-the-art specification method. This *service-based* programming paradigm is backed by new Internet protocols and languages like SOAP [14] and WSDL [15] which serve exactly the purpose of defining and describing services and their intercommunications. New applications or services can rely on services from other service-providers. Since the client of a service is not defined at the time the service is provided the deployment and publication are a most important part during its life cycle. With UDDI [16] a "dictionary" with a standardized access mechanism has been defined to alleviate this problem.

Since issues like time-to-market and distributed development and design are common factors in the affected processes, this progress affects the traditional tool-centered engineering domains as well. Moreover the sophistication of tools reaches a new level as the design technology for new products evolves rapidly. So they present valuable assets for a company by forming an important part of their intellectual property (IP).

Using the *service-centered* approach such companies have the chance to offer their knowledge without the need to transfer programs or algorithms. For the user of such services one important question is how to integrate such a service into their work environment. Usually integration focuses on the principle of coupling existing applications or components tightly together to ensure smooth and reliable operation. The resulting (and available) integration environments use therefore proprietary integration mechanisms on top of existing base-technologies like CORBA [18], JAVA [19], and JAVABEANS [20] or similar middleware components. In fact, CORBA-Services for instance realize conceptually the same idea as webservices with WSDL. They define a common interface that can be accessed from different applications. The important difference of this approach is that webservices are build on top of a foundation that is centered on the Internet platform. Which means the common denominator for running such services is a standard webserver technology (or any kind of server implementation supporting Internet protocols), TCP/IP networks and the HTTP protocol accompanied by the flexible XML metaformat. Thus integration technology will change and hopefully become as the usual way of accessing content in the Internet [17]. The usage of this technology for embedded system design

is of high interest, because on the one side it offers more freedom for the user and on the other side new business concepts or licensing models for the tool vendor.

This paper is organized as follows. First we describe the web-based integration scenario. Then, we present a collaborative design environment for embedded system design, called PARADISE [2]. After that, we describe the combination of the web-based integration scenario with the existing PARADISE environment. We conclude with a discussion of the presented approach.

2. THE WEB-BASED INTEGRATION SCENARIO

For integration aspects, webservices can provide a suitable solution to overcome certain critical issues in exploiting remote facilities: Description and protocols. With XML-based formats like WSDL and SOAP, that additionally defines the messages which are used to access and control remote services, a common infrastructure is provided.

Popular pre-web integration methodologies have been focusing on combining tools and software components by using enabling technologies. These have supplied the inter-tool "glue" which allowed assembling a new solution from these parts and supports reuse in different scenarios. The new scenario using a web-service is shown in Figure 1: The main difference concerning the integration method is that there is no tight coupling of the integrated service in the client's process. The integration efforts are usually taken at the site of the provider of the webservice. The provider registers his service at a webservice directory where it can be queried using a standardized format. As a result the client retrieves the description and address of the service and can locally integrate the webservice interface. Using this two-step approach of dynamically integration has different advantages:

- The provider can move the implementation of the webservice to another server, which will be reflected in the directory. So the client process adjusts automatically to the new location (address).
- For reliable services the provider can supply different instances of the webservice at different sites, so that the others will compensate the failure of one server.
- Different providers may offer the same service at different costs. So the client can select dynamically the most cost-effective one.

There are certain advantages that appear during the design phase of an application: The designer of a new client process can search for needed services conveniently in the directory and create highly distributed scenarios without configuring network structures or determining concrete hosts. Since WSDL is standardized, effective user-friendly integration assistants will simplify the whole integration process on the implementation side so that more weight can be laid on conceptual issues. The XML-based webservice technology is well supported from major IT-providers with sets of tools, so it can be assumed that this standard will have a lasting life cycle in the fast changing world of applications and technology trends.

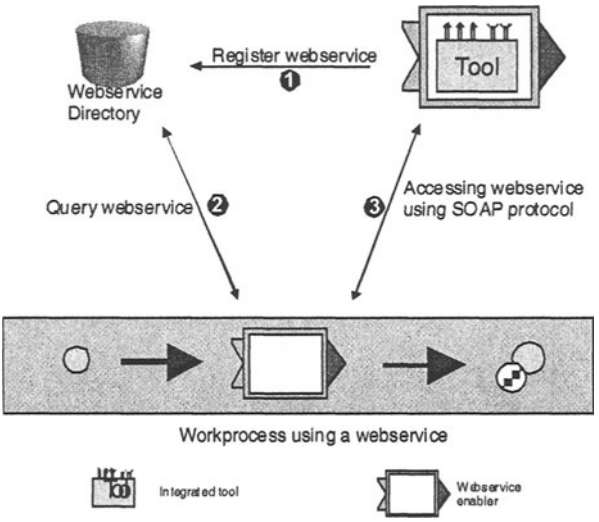

Figure 1. Integrating a webservice

However, ordinary integration technologies still have their merits. Because webservices represent a weak coupling and require a more dynamic processing for the protocols a certain overhead is generated which slows down the interaction. Especially "popular" webservices may have long latencies that reflect the same behavior as visiting a heavy-loaded webservers. In the intranet the common integration techniques are to be preferred because the location of tools and applications under direct administration. Additionally this technique of integration allows an efficient adaptation to the clients requirements, whereas changing a webservice depends on the cooperation of the service provider. Usually there are different clients with contradicting requirements which are not easy to meet by the provider.

3. THE PARADISE APPROACH

The PARADISE design environment is implemented with the ASTAI(R) tool [11] that has been developed at C-LAB [2], [3], [4]. This design environment focuses on the design of embedded hardware/software systems. The design complexity of embedded systems combined with a very tight time-to-market window has revolutionized today's embedded system design process. Several interacting design dimensions to implement parallelism, distribution over different locations, hard real-time requirements [1] and the collaboration have to be addressed by a design environment. Furthermore the usage of IP components is important for embedded system designs [10].

Consequently, the modern, structured design process has to deal with heterogeneous requirements and restrictions. Therefore, the collaborative work is very important for efficient embedded system design. That mean, hardware and software designers and system architects must synchronize their work progress to optimize and debug a system in joint effort. The PARADISE environment distinguishes between these different design domains (see Figure 2). Each design domain reflects a core competency in the design of an embedded system. Each design domain is structured by levels and views. For example the HW-Synthesis design domain corresponds to the structuring of Gajski Y-Chart [7], [8]. PARADISE integrates this databased design methodology with a variety of tools that are suitable for certain design steps. For the first time, the distribution of tools, data and developers over different network domains is possible. This means that any developer at any location can launch a tool from another location. The ASTAI(R) software ensures consistency and the protection of private data. All design steps can be monitored and controlled by means of a design flow. Developers with access to local or remote locations can incorporate existing design data in the design flow. Therefore, system designs can then be carried out with the aid of the bottom-up or top-down method. Besides the integration of tools in a workflow, ASTAI(R) supports the distribution and generation of a design data. Also the update of design data is well supported. Tools within a workflow for which input data does not exists or is invalid yet are automatically blocked until the data is available and valid. Newly generated data is initially classified as local data until developers release it explicitly. Releasing the data effects all levels within the hierarchical design flow to be notified without delay. Developers on these other levels can then accept or reject this data. This protocol is managed by ASTAI(R) by using the Internet. Therefore, the location at which a design data, tool or design flow is stored or installed is irrelevant. A complete overview of the installed tools within PARADISE can be found in [2], [5] and [6].

To show the practicability of the PARADISE design environment two design scenarios are shortly described. The first example is the design of a traffic light controller and the second one is the extension of mobile robot.

The traffic light controller is specified on a very high level of abstraction using Predicate/Transition-Nets (Pr/T-Nets). This formalism is supported within the PARADISE design environment by the 'System Engineering and Animation' tool SEA. The tool provides a homogeneous model based on Pr/T-Nets. To analyze the worst-case execution time (WCET) of a single task, we use the C-LAB Hard Real-Time System (CHaRy) [1]. This is a software synthesis system for distributed (parallel) periodic hard real-time applications. The application is described on a high level (e.g. using SEA), whereas the implementation is left to CHaRy. CHaRy analyzed the WCET for each task of the task-graph. The annotated task-graph is read into the tool SSEA which was developed jointly at the ETH Zurich and the University of Paderborn for the purpose of solving the generalized hardware/software-partitioning problem. This includes also the design space exploration of multi-objective cost functions for the system-level synthesis of embedded systems. For a detailed description of the traffic light controller design see [3]. This design scenario shows the modeling of an embedded system on a high-abstraction level by using PrT/Nets and how to analyze the model with the respect of mapping model tasks to functional units of a given target architecture.

The second design scenario is the extension of a small mobile robot called Pathfinder. This mobile robot is an experimental platform and is as simple as possible in order to keep the focus on the methodological issues and not on the vehicle itself. The *Pathfinder* until now is fully operational, see [21]. The software has been manually coded and implemented on the *Pathfinder*. The architecture is component aware, i.e. is prepared to integrate new modules that add functionality. For demonstration of our IP-based design approach we extend our example in order to establish an advanced interaction facility of the vehicle with it's environment, especially persons being around there. Thus we would like to communicate with people around the *Pathfinder*, i.e. the robot is equipped with a microphone and speakers. Any solution that will fit the needs has to be aware of the existing system architecture and it's real-time characteristics. The time-critical parts of this new functionality are speech compression and decompression tasks. Therefore, the tools of the PARADISE design environment are used to develop or integrate existing solutions in the *Pathfinder* architecture. For the extension of the architecture we use the modeling tool SEA. CHaRy analyzed the WCET for the different software tasks. Within PARADISE we use a generalized approach to HW/SW-partitioning problem which has been implemented by the previously mentioned tool SSEA. The tool TEReCS

decides which of the developed or existing solutions really can be combined, i.e. whether their interfaces are compatible. For a detailed description of the scenario see [10].

At this time four partners participate in the PARADISE design environment. These are namely the C-LAB, the University of Paderborn with two different departments and the University of California at Irvine. The main server is located in the C-LAB. All other partners run clients and small servers to distribute the appropriated tool in the environment over the Internet.

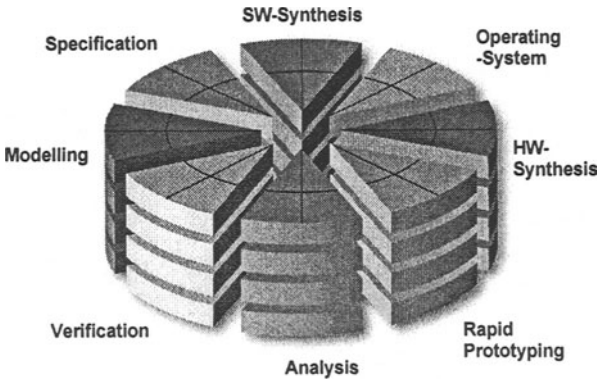

*Figure 2.*PARADISE design domains

4. SYSTEM DESIGN WITH WEBSERVICES

Surely, the presented realization of the PARADISE design environment is a good structuring of today's embedded system design but it lacks by using new approaches like webservices. Therefore, the extension of PARADISE with webservices is a solution to integrate this new trend in the electronic system design area. Clearly, it is not necessary to use webservices for in-house realization of a design environment, because ASTAIR(R) offers all necessary services for integration and workflow implementation for in-house solutions. Additionally ASTAI(R) takes full advantage of the Intranet infrastructure leading to a very efficient integration of tools. Webservices are only necessary for collaborative work between different departments on different locations.

In our approach we interpreted a tool as a webservice. Consequently in the area of embedded system design, especially for our PARADISE scenario, each tool from an EDA vendor could be registered in the webservice directory (see Figure 1). Through this directory any client can transparently select and access the tools required for a certain design-task.

Furthermore, the designs or workflows can also be interpreted as webservices and be registered in the webservice directory. This mechanism has the following advantages for the user of such a system:

- Webservices are in fact a standardized integration approach. They are not limited to one environment, but can be integrated into every software-system that is webservice-aware.
- Webservices can be built from other webservices. This means the integration of tools as webservices into complex design flows which can be republished as a new high-level webservice. Through this approach a client could easily access a complete design methodology.
- The directory can act like a marketplace enabling a client to choose from different solutions the most suitable one for his design problems.

Consequently, the combination of PARADISE by the webservice scenario, described in section 2 offers a really new design method for embedded systems.

Currently ASTAI(R) is expanded by a webservice integration module. This module will provide the following functionality:

- Any "regular" tool can be encapsulated by the server-side module and thus becomes accessible as a webservice. Therefore a standard set of tool-control functions is provided by the tool-webservice module. Using this part conceptually any web service-aware program or tool could use the integrated tool. On the tool site, there is not necessary to install any ASTAI(R) specific software components.
- The integration module on the ASTAI(R) side is embedded into ASTAI(R)´s tool encapsulation specification and therefore virtually undistinguishable from other tools from the ASTAI(R) point of view.
- The actual configuration for a tool is stored in a tool description file using XML. This definition serves two purposes: First it is used by the tool-webservice module to configure the actual calling parameters of the webservice. Additionally a query of these tool properties is possible through the webservice. Second the tool description is used to create an appropriate webservice stub for ASTAI(R) in order to make the tools properties visible in workflows, especially in the workflow editor.

On the client side within the PARADISE environment it is not necessary to deploy ASTAI(R) itself. Only the integration module for ASTAI(R) has to be installed which can be interpreted as a webservice adapter.

5. CONCLUSION AND FUTURE WORK

The notion of "webservices" has introduced a very promising new integration approach, fully based on open Internet standards. This alliance including structured portable data formats addresses both data and service integration issues. In combination with already proven environments, this technology can leverage the power of distributed scenarios in combination with a standardized integration approach. More important "providing a service" may offer a greater market-potential than "supplying a tool" in highly cooperative design environments.

This paper introduces webservices as means to integrate remote tools in a workflow-driven design process for embedded systems. Thus alleviating certain aspects of tool access and extending the area of integration beyond a single environment.

However, the implementation and toolkit support of webservices is yet under development and is estimated to become commercially usable in 2002. Our solutions will provide us with basic components start using this technology in the design domain.

REFERENCES

[1] Peter Altenbernd: "Timing Analysis, Scheduling, and Allocation of Periodic Hard Real-Time Tasks" Dissertation, Paderborn, 1996

[2] W. Hardt, A. Rettberg, B. Kleinjohann. *"The PARADISE design environment"*, 1st Embedded System Conference, Auckland (New Zealand), 1999

[3] A. Rettberg, W. Hardt, J. Teich, M. Bednara. *"Automated Design Space Exploration on System Level for Embedded Systems"*, Ninth Annual International HDL Conference and Exhibition, San Jose (USA), March 2000

[4] A. Rettberg, F. Rammig, A. Gerstlauer, D.D. Gajski, W. Hardt, B. Kleinjohann. *"The Specification Language SpecC within the PARADISE Design Environment"*, in Proceedings of the Distributed and Parallel Embedded Systems Workshop (DIPES 2000), Paderborn, October 2000

[5] A. Rettberg, W. Thronicke. " "Collaborative Design for Embedded Hardware/Software Components with the Distributed PARADISE Environment", Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, FL (USA), July 2001

[6] A. Rettberg, W. Thronicke, "The Distributed PARADISE Environment for Collaborative Design of Embedded Hardware Components", Proceedings of the 8th European Concurrent Engineering Conference (ECEC 2001), Valencia (Spain), April 2001

[7] Franz J. Rammig. "*Systematischer Entwurf digitaler Systeme*". B. G. Teubner, Stuttgart, 1989

[8] D.D. Gajski, "*Silicon Compilation*", Addison Wesley Publishing Company, 1988

[9] R. Ernst, "*Codesign of Embedded Systems: Status and Trends*", Journal of IEEE Design & Test of Computers, pp. 45-54, April-June 1998

[10] W. Hardt, F. J. Rammig, C. Böke, C. Ditze, J. Stroop, B. Kleinjohann, A. Rettberg, and J. Teich, "*IP-based System Design with the PARADISE Design Environment*", accepted for Journal of Systems Architecture, The Euromicro Journal

[11] http://www.c-lab.de/astair

[12] Ivar Jacobson, Grady Booch, James Rumbaugh. *The Unified Software Development Process*. Addison Wesley. 1998. ISBN 0-201-57169-2

[13] Sinan Si Alhir. *UML in a Nutshell*. O'Reilly. 1998. ISBN 1-56592-448-7

[14] Technical Report: *SOAP Version 1.2 Working Draft*. http://www.w3.org/TR/soap12

[15] Technical Report: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl

[16] Universal Description, Discovery and Integration. http://www.uddi.org

[17] Heinz-Josef Eikerling, Wolfgang Thronicke, Siegfried Bublitz. *Provision and Integration of EDA Web-Services using WSDL-based Markup*. FDL 2001. Lyon.

[18] Robert Orfali, Dan Harkey, Jeri Edwards, Robert Crfali. *Instant CORBA*. John Wiley & Sons. 1997. ISBN 0471183334.

[19] Java 2 Platform Enterprise Edition. http://java.sun.com/j2ee

[20] Mark Wutka. *Special Edition Using Java 2: Enterprise Edition (J2EE)*. Que. May 2001. ISBN 0789725037.

[21] http://www.c-lab.de/~pathfinder

This work has been partially funded under grant number 01 M 3048 E (German BMBF project "IP-Qualifikation für effizientes Systemdesign (IPQ)").