UPGRADE

The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

Vol. IX, issue No. 2, April 2008

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- **Informatica**, journal from the Slovenian CEPIS society SDI
- **Informatik-Spektrum**, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
- **Novática**, journal from the Spanish CEPIS society ATI
- **OCG Journal**, journal from the Austrian CEPIS society OCG
- **Pliroforiki**, journal from the Cyprus CEPIS society CCS
- **Pro Dialog**, journal from the Polish CEPIS society PTI-PIPS
- **Tölvumál**, journal from the Icelandic CEPIS society ISIP

Editorial correspondence: Llorenç Pagés-Casas <pages@ati.es>
Advertising correspondence: <novatica@ati.es>

UPGRADE Newslist available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Monograph of next issue (June 2008)

"Next Generation Technology-Enhanced Learning"

(The full schedule of **UP**GRADE is available at our website)

Editorial

New UPENET Partners

As **UP**GRADE readers very likely know, our journal is the anchor point for **UP**ENET (**UP**GRADE European Network). This network of CEPIS member societies' publications, created in 2004, has as its main purpose to make available to the Information and Communication Technology communities (professional, academic, business, government), mainly in Europe but also worldwide, the wealth of knowledge and experience accumulated by these publications, as well as to foster cooperation among them.

To this end **UP**GRADE republishes, in English, in its section **UP**ENET, articles from the syndicated publications while these can republish, in their own languages, articles published in **UP**GRADE or in any of the syndicated publications.

Through these years several CEPIS member societies' publications have joined the network and now we are proud to announce that two new journals have decided to join in, so augmenting the **UP**ENET membership to nine.

The two new **UP**ENET partners are:

■ **Informatica**, quarterly journal published, in English, by the Slovenian CEPIS member society *Slovensko društvo Informatika* (Slovenian Society Informatika, SDI, http://www.drustvo-informatika.si). Informatica started in 1977, one year after SDI was established.

■ **Tölvumál**, biannual journal published, in Icelandic, by the Icelandic CEPIS society *Skýrslutæknifélagið* (Icelandic Society for Information Processing, ISIP, http://www.sky.is). It was created in 1976. By the way, ***Tölvumál*** means "computer issues" in Icelandic.

Accordingly, **Informatica**'s Managing Editor, ***Matjaz Gams***, and ***Thorvardur Kári Ólafsson***, Chief Editor of **Tölvumál,** become members of the **UP**ENET Advisory Board, the former on behalf of the Chief Editor, ***Anton P. Zeleznikar***.

Let me express, on behalf of the Executive Committee of CEPIS and in my name, my satisfaction on this eventful fact which shows that our networking initiative is moving in the right direction even though, of course, it still has room for growth and improvement. I therefore sincerely hope that this announcement will motivate further CEPIS member societies to reflect upon what they have to offer to increase visibility of their work, to inform their constituencies about the work of others, to broaden **UP**ENET, and to enrich the **UP**GRADE content. I am sure that our new partners will contribute to enhance the scope and quality of materials as well as the relevance and usefulness of the **UP**GRADE contents for the ICT community in Europe and worldwide.

Dobrodošli, *velkomin*, welcome!

Niko Schlamberger
President of CEPIS
<president AT cepis DOT org>

From the Chief Editor´s Desk

Welcome to our Deputy Chief Editor

It is my pleasure to announce that the Executive Committee of CEPIS, the governing body of our publisher, has appointed ***Francisco-Javier Cantais-Sánchez*** for the new position of Deputy Chief Editor of **UP**GRADE.

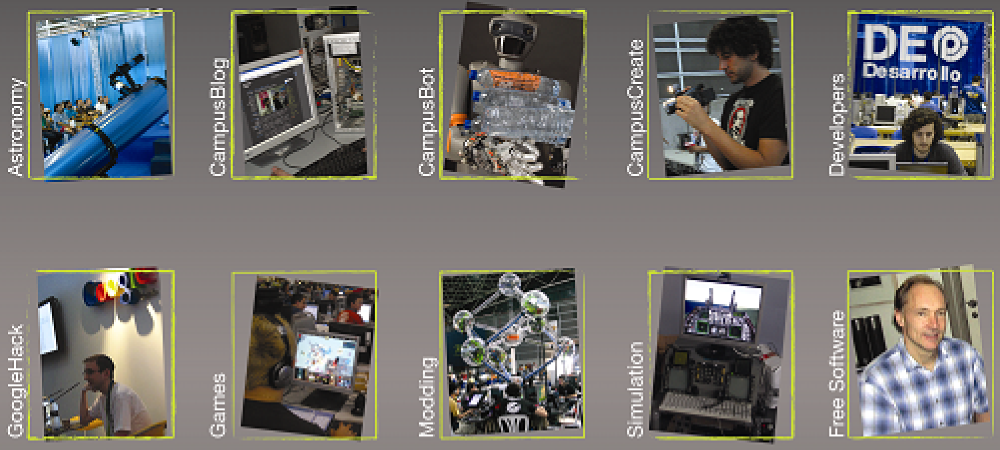
Javier, a young Spanish lawyer and economist who has worked as an IT consultant for important firms for several years, will cooperate closely with the Chief Editor in the diverse tasks required to publish a new issue of our digital journal every other month.

Welcome on board, Javier!

Llorenç Pagés-Casas
Chief Editor of ***UP****GRADE*
<pages AT ati DOT es>

Campus Party™

Six thousand computers connected in the biggest party you can imagine of information technology, art and digital leisure. Come to Campus Party and enjoy a week of strong emotions.



Come and share your passion for technology

28th july - 3th august | Feria Valencia

www.campus-party.es

GENERALITAT VALENCIANA

E3 Futura
www.e3futura.org

Telefónica

CEPIS **is partner of** 12 Campus Party™

Presentation

MDA® at the Age of Seven: Past, Present and Future

Jean Bézivin, Antonio Vallecillo-Moreno, Jesús García-Molina, and Gustavo Rossi

The Model-Driven Architecture (MDA) initiative was launched by the OMG (*Object Management Group*) in late 2000 to propose a new way to consider the development and maintenance of information systems, using models as the essential artefacts of the software development process.

In the MDA approach, models are the key elements used to direct the course of understanding, design, construction, testing, deployment, operation, administration, maintenance, and modification of systems. MDA also raises the level of abstraction by enabling specifications that use different models to focus on different concerns, and by automating the production of such specifications and the software that meets them. In particular, MDA differentiates between platform-independent models and platform-specific models. Thus, the basic functionality of the system can be separated from its final implementation; the business logic can be separated from the underlying platform technology, etc. The transformations between models enable the automated implementation of a system from the different models defined for it or alternatively allow abstract models to be reconstructed from legacy code for the purpose of software modernization or migration. In addition, MDA allows further models of the system to be defined, each one focusing on a specific concern, at the right level of abstraction. These specific models are described using Domain Specific Languages (DSLs) and are related by Model Transformation (MT) specifications. They can also drive tools that automate the model transformations into the final implementations.

Since the emergence of MDA much has happened in the field of modern system and software model engineering. A variety of new acronyms (MDD, MDE, MIC, ADM, MBA, etc.) are appearing to delimit the constantly extending scope of application of core modelling techniques. In addition, the evolution towards modelling practices has combined with the Open Source Software movement in environments like Eclipse to reinforce this important paradigm shift.

The Guest Editors

Jean Bézivin is a professor of Computer Science at the *Université de Nantes*, France, and member of the ATLAS research group recently created in Nantes (INRIA & LINA) by Patrick Valduriez. He has been very active in Europe in the Object-Oriented community and initiated the ECOOP (*European Conference on Object-Oriented Programming*) series of conferences (with Pierre Cointe), the TOOLS series of conferences (with Bertrand Meyer), and the <<UML>>/ MoDELS (*Model Driven Engineering Languages and Systems*) series of conferences (with Pierre-Alain Muller). He has also organized several workshops at OOPSLA (*Object-Oriented Programming, Systems, Languages, and Applications*), such as those in 1995 on "Use Case Technology", in 1998 on "MDD with CDIF", at ECOOP in 2000 on "Model Driven Engineering", etc. His present research interests include legacy reverse engineering, general model engineering and, in particular, model-transformation languages and frameworks, and the building of model-engineering platforms. <jean.bezivin@univ-nantes.fr>.

Antonio Vallecillo-Moreno is an associate professor at the Department of Computer Science of the *Universidad de Málaga*, Spain. His research interests include model-driven software development, componentware, open distributed processing, and the industrial use of formal methods. He holds BSc and MSc degrees in mathematics, and a PhD degree in Computer Science from the *Universidad de Málaga*. He is the *Universidad de Málaga* representative at ISO and the OMG, and a member of ACM, IEEE, IEEE Standards Associations, and the IEEE Computer Society. <av@lcc.uma.es>.

Jesús García-Molina is a full professor at the Faculty of Computer Science at the *Universidad de Murcia*, Spain where he leads the Software Technology Research Group. His research is focused on model-driven software development, in particular model transformation languages, embedded DSL, frameworks, and model-driven modernization. He received his PhD in Science from the *Universidad de Murcia*. <jmolina@um.es>.

Gustavo Rossi is a full professor at the Faculty of Computer Science at the *Universidad Nacional de La Plata*, Argentina where he heads LIFIA, a research Laboratory in Computer Science. He has a PhD from PUC-Rio, Brazil. His research interests are: web design methods, separation of concerns in web engineering and in mobile computing. He is one of the developers of the Object-Oriented Hypermedia Design Model (OOHDM), a mature model-driven design method for Web applications. He has published many papers on these issues in specialized journals and conferences. <gustavo@lifia.info.unlp.edu.ar>.

Seven years after MDA was originally proposed the time is now ripe to look at the old and new objectives, the achievements so far, the incomplete realizations, the difficulties encountered, the ongoing efforts, the research roadmap, and the work still to be done in order to fulfil the initial promise. With this special issue we hope to contribute to such an assessment.

For this task we have been fortunate enough to receive a set of contributions from some of the most relevant and influential people in MDA and Model-Driven Engineering. The papers of this special issue reflect the personal views and insights of some of the creators of MDA on how things have progressed since the initial MDA proposal, and what might be the way ahead; some of the new practices and tools for performing software model engineering; and some of the projects and domains in which MDA is being successfully applied.

In the first group of papers we have a contribution from ***Andrew Watson***, from OMG, who presents the official view on MDA together with a brief history which charts the influences that led to its creation, shows how it has evolved, and outlines the contributions it can make in the future. Then, ***Bran Selic*** revisits his influential paper "*An MDA Manifesto*" in the light of all that has happened in recent years. The original article identified the key elements that characterized the MDA approach and its value proposition. His present article contains an assessment of the progress made since then towards fulfilling that vision, identifies the key obstacles that are hindering a more extensive realization of that vision, and outlines a long-term strategy for overcoming these hurdles.

The second group of papers describes some of the new practices and tools for software modelling. ***Steve Cook*** and ***Stuart Kent*** describe the Microsoft approach to model engineering using *Domain Specific Languages*, and explain how powerful these languages can be when used within the appropriate software development processes and supported by the right kind of tools. Then, ***Jules White***, ***Douglas C. Schmidt***, ***Andrey Nechypurenko***, and ***Egon Wuchner*** introduce the new concept of *Model Intelligence,* which uses domain constraints to guide modellers in the writing of correct models, something which is especially important in the case of models of industrial software systems comprising of tens of thousands of elements. The third paper of this group is written by ***Cédric Brun*** and ***Alfonso Pierantonio***, who analyse one of the key operations in the embryonic and fundamental fields of model management and model evolution: *Model Comparison*.

Finally, in the third group of papers we have the article by ***Rich Gronback*** and ***Ed Merks***, who present one of the most successful contributions to the development and widespread acceptance of MDA: The *Eclipse Modelling Project*. Then, ***Nora Koch***, ***Santiago Meliá-Beigbeder***, ***Nathalie Moreno-Vergara***, ***Vicente Pelechano-Barberá***, ***Fernando Sánchez-Figueroa*** and ***Juan Manuel Vara-Mesa*** show how MDA principles can be successfully applied in the *Web Engineering* domain, not only to build complex Web applications, but also to achieve a smooth interoperability between existing Web Engineering methods and tools.

We sincerely hope that the readers will enjoy this special issue as much as we, the editors, have enjoyed talking with the authors of the papers and merging their excellent contributions into a comprehensive assessment of the current state of MDA and the future opportunities (and challenges) facing MDA if it is to fulfil its original vision in the realm of Model-Driven Engineering.

Useful References on Model-Driven Software Development

The following references, along with those included in the articles in this monograph, will help our readers to dig deeper into this field.

Books

■ A. Kleppe, J. Warmer, W. Bast. "MDA Explained. The Model Driven Architecture: Practice and Promise". Addison-Wesley, 2003. ISBN: 032119442X. A very clear and rigorous introduction to MDA, approached from a practical point of view.

■ David Frankel. "Model Driven Architecture. Applying MDA to Enterprise Computing". OMG Press, 2003. Analysing the application of MDA within the context of enterprise software systems.

■ Thomas Stahl, Markus Völter. "Model-Driven Software Development". John Wiley, 2006. ISBN: 0470025700. An excellent introduction to model-driven development. The most comprehensive text written about Dynamic Systems Development Method (DSDM).

■ Tony Clark, Andy Evans, Paul Sammut, James Willans. "Applied Metamodelling. A Foundation for Language Driven Development". Provides a comprehensive vision of metamodelling and DSL creation. Downloadable from <http://www.ceteva.com/book.html>.

■ Jack Greenfield, Keith Short, Steve Cook, Stuart Kent. "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools". John Wiley, 2004. ISBN: 0471202843. Describes the approach of software factories which combine software product lines with model-driven development. Also includes an assessment of MDA and its standards. An excellent book on software development.

■ Jos Warmer, Anneke Kleppe. "The Object Constraint Language: Getting Your Models Ready for MDA", 2nd edition, Addison Wesley, 2003. ISBN: 0321179366. Considered to be the foremost guide to OCL.

Events

Every year a number of MDA and DSDM related conferences and workshops are held, among the most important of which are the *European Conference on Model Driven Architecture* (EC-MDA), *Model Driven Engineering Languages and Systems* (MoDELS), and the *International Conference on Model Transformation* (ICMT). For the last four years Spain has hosted the *Taller sobre Desarrollo de Software Dirigido por Modelos* (Model-Driven Software Development Workshop) as part of the *Jornadas de Investigación en Ingeniería de Software y Bases de Datos* (JISBD or Conference on Software Engineering and Databases).

OMG

At OMG's web site (*Object Management Group*, <http://www.omg.org>) you can find standards specifications for MDA (MOF, XMI, etc.) and MDA's official web page <http://www.omg.org/mda/>.

Eclipse

The Eclipse Modeling project aims to promote model-directed development within the Eclipse community. From <http://www.eclipse.org/modeling/> you can access all related subprojects (EMF, GMF, GMT, etc.).

Web Sites

■ <http://planet-mde.org/> Portal for the DSDM scientific and educational community.

■ <http://www.model-transformation.org/> Includes links to conferences and research groups and a very interesting collection of scientific papers.

■ <http://www.metamodel.com/> Contains information on metamodelling and references to conferences and news about DSDM.

■ <http://www.dsmforum.org/> Dedicated to domain-specific language development. Includes information about examples of its application in industry, tools and events.

■ <http://www.lcc.uma.es/~av/MDD-MDA/> Site maintained by the *Universidad de Málaga* research group which leads the *Red Española de Desarrollo Dirigido por Modelos* (Spanish Network on Model-Driven Development) and organizes the DSDM Workshop. Contains information about MDA, publications on MDA and DSDM in general, presentations, tools, and mailing lists, and a list of the Spanish groups researching into DSDM.

List of Acronyms

ATL	ATLAS Transformation Language.
CMOF	Complete MOF.
DSL	Domain Specific Language.
EMF	Eclipse Modelling Framework, Eclipse project.
EMOF	Essential MOF.
EMP	Eclipse Modelling Project, Eclipse project.
GMF	Graphical Modelling Framework, Eclipse project.
GMT	Generative Modelling Technologies, Eclipse project.
MDA	Model-Driven Architecture.
MDD	Model Driven Development.
MDE	Model Driven Engineering.
MIC	Model Integration Computing.
MOF	Meta-Object Facility. OMG's metamodelling language.
OCL	Object Constraint Language.
OMG	Object Management Group.
PIM	Platform Independent Model. MDA related.
PSM	Platform Specific Model. MDA related.
QVT	Query View Transformation. OMG's model transformation language.
SOA	Service Oriented Architecture.
UML	Unified Modelling Language.
XMI	XML Metadata Interchange. OMG's model interchange language.

A Brief History of MDA

Andrew Watson

On 8th March 2000 Object Management Group (OMG) announced that its Architecture Board had voted to adopt the Model-Driven Architecture (MDA) as both the strategic approach to developing OMG's own integration standards, and as its recommended application development technique. MDA was devised before the term "Service-Oriented Architecture" (SOA) became fashionable, and when many Business Process Management (BPM) techniques and languages were in their infancy. However, through a combination of foresight and good fortune MDA techniques are, if anything, more relevant today in the world of SOA and BPM than they were in 2000. This short history of MDA charts the influences that led to its creation, shows how its has evolved, and outlines the contributions it can make in the future.

Keywords: Business Process Management (BPM), Model-Driven, Model-Driven Architecture (MDA), Object Management Group (OMG), Service-Oriented Architecture (SOA), Unified Modelling Language (UML).

Author

Andrew Watson is Vice President and Technical Director at OMG. Andrew has overall responsibility for OMG's technology adoption process, and also chairs the Architecture Board, the group of distinguished technical contributors from OMG member organisations which oversees the technical consistency of OMG's specifications. Previously Andrew researched service oriented architectures and their type systems with the ANSA core team in Cambridge, wrote Lisp compilers at Harlequin, and worked on distributed systems and software engineering at HP Laboratories. <andrew@omg.org>.

1 Origins

OMG came into being in the late 1980s as an independent, not-for-profit industry organisation to specify object-based middleware that could help solve the growing problem of integrating IT systems that spanned multiple platforms. The resulting Common Object Request Broker Architecture (CORBA®) middleware and its related specifications became very widely used, and by 1999 an analyst survey [1] found that "*70 percent of respondents cited CORBA compliance as 'important' or 'very important' to integration, outpacing every other factor in the survey*".

From the mid-1990s OMG also began developing specialised middleware-based interoperability standards for application domains ranging from finance through telecoms to healthcare. In each of these areas, groups of highly-qualified domain experts devoted several man-years of effort to specifying standards for domain application components, using CORBA's Interface Definition Language (IDL) to specify the service interfaces that these components would provide and use. These standard services and the CORBA middleware they used to communicate formed the basis of OMG's Service-Oriented Architecture, known as the Object Management Architecture (OMA).

By the late 1990s OMG had used CORBA and IDL to specify several families of domain-specific services for different industries, and in the process had identified two limitations with this purely middleware-based approach to creating integration standards:

- IDL provides a precise way to specify the structure of the data that application components exchange with each other. However, since the CORBA middleware doesn't need to know or constrain the order in which the data are exchanged or the semantic relationships between the data fields, IDL doesn't provide any way of specifying these parts of the design. These important application-level constraints could only be captured using imprecise natural language, which was becoming more and more of a difficulty as the domain specifications became more sophisticated.
- The application component designs created by OMG's Domain groups were often equally usable with other middleware architectures; for instance, the Java Transaction Service (JTS)[2] is a translation into pure Java interfaces of the functions defined by the CORBA Object Transaction Service (OTS)[3]. However, converting OMG's specifications from IDL to another platform involves knowledge of both the source CORBA environment and the chosen target, and not all designers have this detailed knowledge. Furthermore, where there are multiple options for translating an interface element, multiple mappings are possible; hence different designers would likely generate different (and incompatible) translations.

It became clear that each of OMG's domain groups incorporated a large pool of priceless domain expertise, and in the process of creating domain interoperability specifications were actually creating valuable models for standard subsystems. However, the difficulty of precisely capturing non-structural aspects of the interfaces or translating the interfaces into other notations were preventing this valuable work being used to its full potential.

2 MDA is Born

To address these concerns, OMG decided to switch from a middleware-based approach to specifying SOA services to a platform-independent approach which could capture behavioral as well as structural aspects of interoperability. These Platform-Independent Models (PIMs) could then be translated via standardised transformation rules into interface specifications for any particular application platform, such as CORBA, Java, or one of the emerging families of "Internet Middleware" based on eXtensible Markup Language (XML), such as Simple Object Access Protocol / Web Services Description Language (SOAP/WSDL).

During the mid-1990s OMG had also helped broker agreement within the fledgling Object-Oriented (OO) visual modelling community, creating the Unified Modelling Language (UML®), a family of 13 diagram types for the visual representation of different static and dynamic aspects of application software design. Applying UML and the Meta-Object Framework (MOF™), the standardised metadata framework on which it's based, to the problem of creating platform-independent service specifications led to the creation of MDA. Use of formal, rigorously-defined modelling languages is the key; only with a precise definition of the meaning of every construct in the language is it possible to mechanise translating the PIM into the implementation artefacts for the target platforms (such as IDL or Java interfaces), and thereby achieve the goal of platform independence.

MDA was thus first mooted as a way of creating standards. However, it was immediately obvious that the same tools and techniques could be used to build applications; transforming a precise but abstract design into the framework of an application is a very similar problem to translating into a platform-specific standard. Depending on the modelling language being used, it might not be possible to completely specify a whole application as a PIM, but at the very least a large part of the application's static structure and interface design could be captured and then translated into code or other platform-specific artefacts. Many applications use multiple platforms and programming languages simultaneously; transforming different parts of a common PIM into complementary Platform Specific Models (PSMs) for the different platforms used helps address the problem of maintaining common interface definitions across a variety of implementation technologies. By creating application outlines directly from models, and helping to automatically write the "glue code" between different platforms within one application, it was initially estimated that even the early modelling technology available at the time could be used to create 30-40% of the application code directly from an MDA PIM, yielding useful increases in software quality and productivity.

It's important to note that the PIM is one of the main products of the MDA design process, not just a transient stage in the process. If changes are later needed as the specification or application evolves, it is the PIM, not the generated artefacts, that are modified. In short, MDA treats design as a *product* not a *process*.

3 OMG Specification Developments to Support MDA

Once the MDA vision was in place, OMG began work to evolve its modelling specifications to better support it. The main results were the UML 2 revision and ongoing work on the MOF 2 metamodelling specification.

Although the basic structure and specification of UML 1 and UML 2 are much the same, the detailed design and underpinnings of UML have been shaped by MDA over the past 7 years. Even today, many engineers use UML merely as a way of sketching software designs, as an aide memoire or a way of documenting or communicating design. Since sketches are meant to be read by people, not tools, some imprecision, while undesirable, can be tolerated, or even go completely unnoticed. When UML is used for only for sketching, the appearance and readability of the diagrams matters much more than the underlying representation of the model itself within the modelling tool. Although UML 1 provided a formal, standard metamodel for each diagram type, common features of these metamodels had not been factored out, and there were also some inconsistencies between the metamodels for different diagrams.

With the advent of MDA, that began to change. OMG began a major revision of the UML specification to UML 2; one of the aims of this revision was to improve the quality of the UML metamodel to make it easier to extract information from them as part of the MDA process. At the same time, UML tool vendors started to devote more effort to producing compliant models corresponding to the diagrams that their tools were used to create. As a result both today's UML and the tools that implement it are much better suited to model-driven development techniques.

MOF, the metamodelling foundation on which all modelling languages used for MDA are based, has also evolved over the last seven years. A new version of the core MOF specification was released at the same time as UML 2, building on the experience of MOF 1 and UML 1 to make MOF into a truly versatile foundation for models and model transformation. The MOF2 Core specification contains the basic metamodelling framework, and has two compliance points: EMOF (Essential MOF) and CMOF (Complete MOF). Further specifications provide extra MOF-related features. Perhaps the most important is the MOF Query, View & Transformation (QVT) specification, which provides standardised mechanisms for making model-to-model transformations. Such transformations, for example from PIM to PSM, lie at the heart of MDA, and providing a standard language for executing them allows libraries of standard transformations to be created. Other MOF-related standards include Versioning and Lifecycle, which provides standard ways to support version control of MOF models. The work on MOF standardisation continues within OMG, building on this core set of MOF standards, and providing the essential tools for the metadata manipulation that underpins MDA.

4 Which Modelling Language?

At the time MDA was first mooted, and even more so today, most software modelling uses UML. By 2004 it was estimated that more than 2/3 of all industrial applications used at least some UML during their specification phase, with 82% of developers saying that they planned to use UML in future [4]. Because of its ubiquity, it was clear from the start that UML would be the language predominately used for MDA. However, to help apply UML and MDA to the widest-possible range of application areas, OMG is also publishing a rapidly-expanding family of UML profiles which extend and adapt UML to allow it to represent concepts in specific application domains. Examples include:

- MARTE – A UML Profile for Modelling and Analysis of Real-Time and Embedded systems.
- SysML – This extends UML to support modelling of complex systems with human and hardware as well as software components.
- EAI – A UML profile for Enterprise Application Integration.
- Testing – A UML profile defining a language for designing, visualizing, specifying, analyzing, constructing and documenting software test systems.
- Voice – A UML profile for modelling voice dialogs in telecom applications.

In effect, each UML profile creates a customised Domain-Specific Language (DSL) for modelling concepts in that domain. However, because each language is strongly tied to the well-understood UML syntax and semantics, and defined using UML's standard extension mechanism, it's easier to learn than a language designed from scratch, and can be used with existing and well-supported UML and MDA tools.

Although UML and its profiles are the most widely-used language for MDA, using UML is not actually an MDA requirement; completely un-UML-like MOF-based modelling languages can be defined and used with MDA. One recent example is Semantics of Business Vocabulary and Business Rules (SBVR), a text-based language for representing business rules. Work is also underway to provide a MOF foundation for Business Process Modelling Notation (BPMN™), a popular flowchart-like syntax for creating business process diagrams that represent the activities of a business process and the flow of control that defines the order in which they are performed.

5 Specifications Which Have Been Developed Using MDA

Over the last seven years OMG has created numerous specifications in a number of vertical domains using the MDA approach. A few representative examples will give the flavour of the variety of problems addressed:

- *Microarray and Gene Experiment Object Model*. A Platform-Independent Model for the representation of life-science gene expression data and relevant annotations, along with a standard mapping onto an XML Document Type Definition (DTD) for representing and exchanging this data using XML [5].
- *Product Lifecycle Management (PLM) Services*. This specification defines a PIM for standardised services for use in managing and representing the different configurations and versions a product may be sold under over its lifetime. The specification includes a PIM and a PSM for WSDL/SOAP [6].
- *PIM and PSM for Software Radio Components*. "Software Radio" is a generic term for radio receivers and transmitters where some or all of the signal processing is performed by software running on general-purpose processors, specialised Digital Signal Processors (DSP), or exotic devices like Field Programmable Gate Arrays (FPGAs). This five-volume specification provides both Domain-Specific Languages (defined as UML profiles) for designing Software Radios, and PIMs standardising parts of software radio designs. Mappings of these PIMs onto the PSMs for the industry-standard CORBA-based Software Communication Architecture (SCA) are also provided [7].
- *Application Management and System Monitoring for Combat Management Systems.* This specification addresses the problem of centralised management of CMS applications running on the wide variety of hardware and software platforms found on modern warships. It includes a PIM and PSMs for several widely-used platforms including CORBA, DMTF CIM Managed Object Format and Data Distribution Service (DDS™) middleware, as well as defining a PSM for exchange of management data using XML [8].

OMG's web site has a full list of specifications developed using MDA [9].

6 MDA and Software Development

One important difference between creating interoperability specifications and designing software is that the latter almost always involves modifying and interfacing with existing application code. Although this is often dismissively termed "legacy integration", as though it involved working with a few quaint leftovers from a former age, studies over the last 20 years have repeatedly shown that IT users spend far more effort on modifying existing software than on deploying "new" applications [10] [11] [12]. The original cost of acquiring an application (whether purchased or developed in-house) is often only 10-20% of its Total Cost of Ownership (TCO) when software lifetimes are measured in decades.

If using MDA for software development is going to help achieve a meaningful reduction in TCO, it clearly has to address the issue of maintaining and updating existing software, since this can account for up to 90% of software's true cost.

One way that MDA reduces TCO is by creating new application code with fewer bugs. Higher-quality code results in less effort later being spent on problem diagnosis and remedies, making it easier to adapt the code to new business needs, and lowering the costs to train and support users of the system. An informal side-by-side study in 2006 comparing MDA with traditional hand-coding for a new commercial billing application showed that MDA techniques produced almost three times as many lines of code per dollar spent, but with less than one third the defect rate discovered during testing (1.1 defects per thousand lines of code for MDA, 4.1 for traditional coding) [13]. Although the lowered coding costs are impressive, the higher code quality will have a much greater impact on the system's TCO over its life.

Having models as a product of the development process also helps lower the costs of making subsequent modifications to the system. Maintainers working on existing applications typically spend more than half of their time simply trying to understand how the code works before they can begin to modify it [14]. With MDA, the design models are one of the products of application development, along with the code itself, so maintenance involves modifying the models and regenerating the corresponding parts of the code. The savings in time spent understanding and then modifying the code can be substantial. In early 2003 a side-by-side laboratory study of maintenance of MDA-based and non-MDA-based J2EE applications found that MDA increased maintainers' productivity by 37% compared to traditional code-based maintenance [15].

MDA's architects have also recognised that not all applications will have been developed this way, so there will be times when a team equipped with MDA tools and skills is faced with modifying an application for which no MDA model exists. It is therefore essential that there's some way of recovering design information from existing software, even where original designs have been lost (or even never existed in the first place). Acknowledging this, OMG started work on "Architecture Driven Modernisation" (ADM) standards in 2003, two years after the MDA initiative began.

The first product of the ADM effort is the Knowledge Discovery Metamodel (KDM) specification, which provides standard metamodels for documenting and formally representing existing software assets and their operational environments with MOF. KDM defines a common vocabulary of knowledge related to software engineering artefacts, regardless of the implementation programming language and runtime platform (a checklist of items that a software mining tool should discover and a software analysis tool can use. KDM's common MOF models and interchange format provide an integration layer between the syntax-specific parsers used to extract information from raw source code and the analysis and transformation tools which process abstract program structure information, thus allowing export and import of data currently contained within individual software modernisation tools). In this way KDM can provide both a common platform to help integrate diverse software modernisation tools, and also provide the basis for bringing knowledge about existing software assets into the MDA software creation process [16].

Independently of MDA, the KDM specification is also finding application in the Software Assurance field, to help analyse existing software to detect security vulnerabilities and other ways in which it might behave outside its required specification. As with software modernisation, many tools are likely to be involved, each producing a portion of the required knowledge about the software assets. KDM is also being used in the Software Assurance field as a standard way of representing knowledge about software collected via a variety of different tools.

Creating tools for recovering design knowledge from existing software is a challenging problem, and OMG's work on standards in this area will continue for some time; for instance, a forthcoming specification will standardise a metamodel for Abstract Syntax Trees, to facilitate the analysis, visualization and refactoring of application code below the procedural level supported by KDM. However, the work in this area is already yielding benefits in helping modify and update existing application code, rather than merely encapsulating it unchanged, and have new applications communicate with it at arms' length, as too often happens today.

7 The Future

As business users become increasingly dependent on Information Technology to deliver products and services, so problems caused by the inflexibility of IT systems becomes ever more pressing. Rather than adapting to the changing needs of the businesses they supposedly serve, IT systems' capabilities increasingly dictate business policy. Traditional software engineering techniques demand stable, well-defined requirements and long timescales for creating systems tailored to users' needs, yet the business environment is changing with increasing speed. All the while, an ever-growing deployed software base is accumulating post-design modifications that distort its structure to the point of "software death", where any further modification (whether to fix a bug or introduce a new feature) in turn introduces a new bug.

The convergence of MDA with Business Process Management (BPM) and Service Oriented Architecture (SOA) offers a road-map for organisations seeking to escape the straightjacket of software inflexibility.

BPM is an umbrella term for the techniques of identifying, documenting and managing the complete end-to-end processes an organisation uses to perform an individual task, especially where this involves the cooperation of multiple individuals, departments or separate organisations. The dozens or hundreds of processes within an organisation typically have both human and IT components, and many of the individual activities within any one process are also used in conjunction with other activities in other processes. Just as different organisations have different levels of maturity in their software production processes, so business organisations have different levels of business process maturity. OMG's Business Process Maturity Model (BPMM) [17] helps organisations discover and improve the level of precision with which they understand their own processes. Given well-understood processes, precise yet easily-learned visual notations like SBVR and BPMN can be used to document and communicate them between business stakeholders, process participants and the engineers building software to support them.

Service-Oriented Architecture is one of the foundations on which OMG's technical architecture was built almost 20 years ago, but it has now gained new prominence as a method of designing, deploying and managing the individual activities that make up a business process. At the software level, use of SOA entails building components with well-defined meta-data that defines both the information they require from each other and the services that they provide, allowing tools to orchestrate the late binding of SOA services into new processes as the needs of the organisation change. MDA provides the vital bridge between BPM design and SOA infrastructure, allowing process models captured via MOF-based languages like SBVR, BPMN or UML activity diagrams to be translated into that orchestration code.

The convergence of model-driven software development, service orientation and better techniques for documenting and improving business processes holds out the promise of rapid, accurate development of software that serves, rather than dictates, software users' goals.

8 Conclusion

Put in a historical context, MDA can be seen as the most recent step in the progressive development of better and more powerful tools for writing software over the 60 year history of electronic data processing. The first programmers write their applications directly in machine code, entering the bit patterns for instructions from memory and calculating branch offsets and index register settings by hand. Assemblers moved programmers one level of abstraction away from the raw machine, then 3rd Generation Languages (3GLs) and 4th Generation Languages (4GLs) each added another level of tooling between the user and the raw machine, providing abstractions that are progressively closer to the concepts used by the ultimate user of the IT system. MDA continues this trend to better tools and increasing abstraction. There has always been a fierce rearguard action (clinging to the efficiency argument) to any increase in the level of abstraction, but the flexibility, reduced complexity and increased productivity of the abstractions have always won through in the end [18]. MDA is proving to be no exception.

References

[1] Gartner Group. "Middleware: what end users are buying and why", February 1999.
[2] Sun Microsystems. "Java Transaction Service 1.0 Specification", <http://java.sun.com/products/jts/>, 1999.
[3] OMG. "Object Transaction Service 1.2.1", <http://doc.omg.org/formal/01-11-03>, 2001.
[4] A. Zeichick. " UML Adoption Making Strong Progress", SD Times, 15th August 2004.
[5] OMG. " Gene Expression Specification 1.1", <http://doc.omg.org/formal/03-10-01>, October 2003.
[6] OMG. "Product Lifecycle Management Services 1.0.1", <http://doc.omg.org/formal/06-04-03>, April 2006.
[7] OMG. "PIM and PSM for Software Radio Components Specification 1.0", <http://doc.omg.org/formal/07-03-01>, March 2007.
[8] OMG. "Application Management and System Monitoring for CMS Systems Beta 2 specification", <http://doc.omg.org/dtc/07-05-02>, May 2007.
[9] OMG. Specification Catalogue, <http://www.omg.org/technology/documents/spec_catalog.htm>.
[10] L. Erlikh. "Leveraging Legacy System dollars for E-business", IEEE IT Pro, May/June 2000.
[11] A. Eastwood. "Firm Fires Shots at Legacy Systems", The Standish Group, 1993.
[12] J. Moad. "Maintaining the competitive edge", Datamation 61-62, 64, 66., 1990.
[13] Steve Hudson. Private communication, 2006.
[14] B.P. Lientz, E. Swanson. "Problems in application software maintenance", Communications of the ACM 24 (11), 763-769, 1981.
[15] The Middleware Company. "Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach - Maintainability Analysis", January 2004.
[16] OMG. " Architecture-Driven Modernization: Knowledge Discovery Meta-Model 1.0 beta3", <http://doc.omg.org/ptc/2007-03-15>, March 2007.
[17] OMG. "Business Process Maturity Model 1.0 beta1", <http://doc.omg.org/dtc/2007-07-02>, July 2007.
[18] D. Otway. "Abstract & Automate", Architecture Projects Management Ltd, <http://www.ansa.co.uk/ANSATech/94/Primary/102001.pdf>, May 1994.

MDA Manifestations

Bran Selic

In 2004 the author, along with several colleagues, published an article titled "An MDA Manifesto", which outlined a strategic vision for Model-Driven Development (MDD). That article identified the key elements that characterized this approach to software development and its value proposition. The present article contains an assessment of the progress made since then towards fulfilling that vision, based on practical experience in applying MDD in industry. The key impediments that are hindering a more extensive realization of that vision are identified and categorized. Finally, a long-term strategy is outlined for overcoming these hurdles.

Keywords: Model-Driven Architecture (MDA), Model-Driven Development (MDD), Object Management Group (OMG).

Author

Bran Selic is currently President of Malina Software Corp. In 2007, Bran retired from IBM Canada, where he was an IBM Distinguished Engineer responsible for defining the strategic direction for software modeling tools for the Rational brand. He is currently the chair of the OMG task force responsible for the UML standard. Bran is also an adjunct professor of computer science at Carleton University in Ottawa, Canada. <bselic@ca.ibm.com>.

1 Introduction

Almost four years ago, some of my colleagues at IBM Rational and I co-authored an article entitled "An MDA Manifesto", which was first published in the *MDA Journal* and then again in the eponymous book by Frankel and Parodi [1]. The primary intent was to articulate our shared vision of model-driven development (MDD). IBM and its Rational business unit in particular were pioneers in the application of modeling methods to software development.

Jim Rumbaugh and Grady Booch, both of Rational (and both of whom were authors of the Manifesto article), were the primary designers of the Unified Modeling Language (UML), much of it based on their industry-leading earlier work in model-based object-oriented methodologies. Rational's modeling tools were and still are market leading MDD tools.

It is both interesting and instructive to reflect on that vision in the light of subsequent practical experience with MDD since the article was written. Has anything fundamental changed in the vision? What are the current states of practice and adoption of MDD? What stands in the way and how serious is it? The purpose of this article is to examine some of these issues and also to investigate potential strategies that would enable broader application of MDD in industry and a more comprehensive realization of the vision behind it.

It would have been ideal if all of the original authors were involved in this assessment, but, due to a number of operational reasons this was not feasible (for one, I have since retired and have a bit more time at my disposal than my co-authors). Nevertheless, I have maintained close contact with all of them and, although I certainly cannot claim to represent their views, I am confident that we share pretty much the same vision and objectives outlined in the original article.

2 The MDA Manifesto Revisited

A "manifesto" is an explicit declaration of set of principles and a plan of action for reaching some objectives. The original article identified three keystones of the Model Driven Architecture (MDA) initiative from the Object Management Group (OMG) [2], as interpreted by IBM's technical team responsible for its MDD strategy. These were:

- Use of higher levels of *abstraction* in specifying both the problem to be solved and the corresponding solution, relative to traditional software development methods (NB: in the original article, this was referred to as "direct representation").
- Increased reliance on computer-based *automation* to support analysis, design, and implementation.
- Use of *industry standards* as a means facilitating communications, product interworking, and technological specialization.

The following is a brief summary of the nature and rationale of each of these key elements. Readers interested in a more in-depth description should refer to the Manifesto article itself.

2.1 The Issue

In essence, the primary problem that MDD is intended to address is the often overwhelming complexity involved in designing and implementing modern software. The magnitude of this problem just keeps growing, as our demands for more sophisticated functionality and more dependable software increase (as Grady Booch notes, in some ways "software runs the world" [3]). It is, therefore, critical for us to understand the sources of this complexity lies and what can be done about them.

In his seminal work on software development, "The Mythical Man-Month" [4], Fred Brooks Jr. identifies two kinds of complexity: *essential complexity*, which is inherent to a particular problem and, consequently, unavoidable, and *arbitrary complexity*, which is due to the methods and tools used to address the problem. Brooks points out that software designers face more than their share of arbitrary complexity.

For example, they often have to cope with the idiosyncrasies of traditional programming languages, in which a single uninitialized variable or misaligned pointer can have disastrous consequences, whose impact can extend far beyond the localized context in which the error was made. Similarly, many crucial and difficult to detect errors can be introduced in the process of translating complex domain-specific concepts into corresponding computer-program implementations.

The basic motivation behind MDD can be reduced to the elimination of arbitrary complexity, through the definition of improved methods and tools.

2.2 Abstraction

Abstraction is a primary technique by which human minds cope with complexity. By hiding from view what is irrelevant or of little consequence, a complex system or situation can be reduced to something that is comprehensible and manageable. When it comes to software, it is extremely useful to abstract away technological implementation detail and deal with the domain concepts in the most direct way possible. For instance, it is typically easier to view and comprehend a state machine as a graph, rather than to see it in the form of nested "case" statements in some programming language rife with distracting low-level syntactical details.

The MDD approach to increasing levels of abstraction is to define domain-specific modeling languages whose concepts closely reflect the concepts of the problem domain whilst minimizing or obscuring aspects that relate to the underlying implementation technologies.

To facilitate communications and understanding, such languages use corresponding domain-specific syntactical forms. This often means using non-textual representations such as graphics and tables, which more readily convey the essence of domain concepts than text.

2.3 Automation

Automation is the most effective method for boosting productivity and quality. Software, of course, is an excellent medium for exploiting automation, since the computer is in many ways the ideal machine for constructing complex machines. In case of MDD, the idea is to utilize computers to automate any repetitive tasks that can be mechanized, tasks which humans do not perform particularly effectively. This includes, but is not limited to, the ability to transform models expressed high-level domain-specific concepts into equivalent computer programs, as well as into different models suitable for design analyses (e.g., performance analyses, timing analyses). In case of executable modeling languages computer-based automation can also be used to simulate high-level models to help evaluate the suitability of a proposed design in all stages of development.

2.4 Standards

MDA is OMG's initiative to support MDD with a set of open industry standards. Such standards provide multiple benefits, including the ability to exchange specifications between complementary tools as well as between equivalent tools from different vendors (thereby avoiding vendor lock-in). Standards allow tool builders to focus on their principle area of expertise, without having to recreate and compete capabilities provided by other vendors. Thus, a performance analysis tools need not include a model editing capability. Instead, it can interact with a model editing tool using a shared standard .

As part of MDA, the OMG has defined a basic set of MDD standards for modeling languages (e.g., UML, MOF), model transform definition (MOF QVT), MDD process definition (SPEM), and a number of other areas. It is somewhat ironic that MDA is sometimes viewed as an approach to MDD that is contrary to the domain-specific languages approach, this is not the case, since many of the MDA standards are specifically designed to support specialization for domain-specific needs. The MOF language, for instance, is a language for defining domain-specific languages. Furthermore, UML can also be used to define different domain-specific languages by taking advantage of its profile mechanism. This not only allows reuse of the effort and ideas that went into the design of UML but also enables the reuse of existing UML tools. In many ways, this approach to domain-specific language design overcomes one of the major barriers that has impeded such custom approaches in the past: the lack of adequate tooling as well as the cost of maintaining it and evolving it. Thus, it is possible to reap the benefits of both standardization and customization.

3 The State of the Practice in MDA

While one can argue against concrete realizations of the MDA idea, such as the technical features of UML or MOF, it is hard to argue with any of its basic premises. Increasing the levels of abstraction and automation and the use of standards, executed properly, are all undeniably useful. Furthermore, there have been numerous examples of successful applications of MDA in large-scale industrial projects (cf. [5] [6])[1]. Yet, there is still a significant amount of controversy about whether or not MDA is useful. It is fair to say that the dominant perception among today's software practitioners is that MDA has yet to prove itself, or, at the ex-

[1] On the other hand, many enterprises that have achieved successes with MDD are inclined to keep them confidential in the belief that their use of MDD is an important advantage they hold over competitors.

treme end of the opinions scale, that it is a distracting academic fairy tale, concocted by software theologians who are disconnected from any practical reality[2].

I am unaware of any published results, but my personal estimate from numerous discussions with software development teams in industry is that the penetration of model-based methods hovers around 10%. If this stuff is really as good as claimed, why isn't everyone using it?

It turns out that there are numerous and varied reasons for this glacial pace of adoption. They can be roughly classified into technical, economic, and cultural.

3.1 Technical Hurdles

A major problem that plagues many software products these days is *usability*. In the case of MDA, it is mostly manifested in MDA tools. Although often endowed with diverse and very powerful functionality, such tools almost invariably tend to be extremely difficult to learn and to use. Users are typically faced with a bewildering spectrum of menu items arranged in seemingly arbitrary groupings. Common operations that should be easy to use often require complex and counterintuitive tool manipulations. One of the reasons for this is that, ironically, many of the tool designers and implementers are not themselves practitioners of MDD and, therefore, do not have a sense for how the tools should look and behave.

Consequently, poor tool usability is one of the biggest current impediments to greater penetration of MDD methods in practice.

A second major technical problem is that there is still very little theoretical underpinning for MDD. Much of the MDD technology that is available today was developed in ad hoc ways by industrial teams who were trying to solve specific problems in circumstances that do not afford the luxuries of reflection and generalization. As a result, when it comes to supporting MDD, we do not yet know precisely what works, what does not, and why. The result is not only gratuitous diversity but also substandard and inadequate technologies. In contrast, traditional programming-oriented methods and technologies have been studied amply and one can rely on a sound body of theory to ensure that common problems are avoided and solid and proven solutions are chosen.

The lack of a sound theory of MDD is also manifested in interoperability problems between MDD tools that often result in highly undesirable vendor lock-in for users. This is true even in the presence of standards.

3.2 Cultural Hurdles

Despite the availability of hard evidence of the success of MDA in practice, there is still insufficient awareness of its potential and its capabilities among practitioners who could be exploiting it. However, even in cases where a project team might be fully aware of the potential benefits of applying MDD, there is still a problem in adopting it due to the inevitable overheads whenever new methods and tools are introduced into a running production environment. It takes time to learn and adjust to new ways of working (not to mention that it may be necessary to support the old and the new methods and tools during phased cutovers). In today's highly competitive environment, where time-to-market is a fundamental driver of development, this overhead is difficult to accept, since the investment payback is generally deferred to subsequent projects.

However, perhaps the most difficult issue to overcome of all is the conservative mindset of many software practitioners. Because they tend to invest vast amounts of time and effort in mastering specific implementation technologies (which, due to their often arbitrary complexity, do require significant investment), many practitioners define their expertise in terms of computing technologies they have mastered rather than the domain in which they are working. For instance, they are much more likely to view themselves as, say, J2EE experts rather than as financial software experts. Consequently, there is often major resistance to technological change, even if the new technology could lead to better solutions for the specific domain problem on which they are working. This same technology-centered culture means that many software developers have a very superficial interest in and understanding of how the products they implement are to be used, which, in turn, leads to poor product usability discussed above.

One major barrier in overcoming all such cultural issues is the sheer number of software practitioners, which is estimated between 10 and 15 million [7]. This is, of course, a huge inertial mass that is very difficult to shift from its present cultural base.

3.3 Economic Hurdles

Today's prevailing business environment is focused on relatively short-term return on investment (ROI). Public companies report their results on a quarterly basis and a failure to meet profit expectations in a given quarter is likely to result in a falling stock prices and a shift of stockholders to other apparently more immediately profitable businesses. This has the unfortunate effect that most investment in technological development tends to be short-term. Consequently, it is hard to justify longer-term investments in new software development methods and tools, particularly if the payback is not guaranteed. And, to be fair, switching to MDD does not guarantee success, partly because of the other issues discussed above. For example, the risks of introducing MDD into a software development organization can be greatly mitigated if it is led by individuals with prior experience. Unfortunately, such expertise is still quite difficult to find and secure. And, with the aforementioned absence of a systematic foundation for MDD, organizations are of-

[2] Steve Mellor tells the following anecdote that epitomizes the current state of affairs in MDA: When he was asked over ten years ago about when he expected MDA to become mainstream, he suggested that it would likely happen within the following year and a half to two. And he has been giving the same answer to that question ever since.

ten left to fend for themselves through a risky process of trial and error.

Clearly, these are all substantial barriers to overcome and it seems likely that the pace of introduction of MDD in industrial practice will remain a slow for several years to come. In the next section, we describe some initiatives that could help accelerate this trend.

4 The Way Forward

There are at least three possible areas in which to address the problem of increasing the penetration of MDD in practice:

- Education
- Research
- Standardization

4.1 Education

Given the difficulties of changing the dominant technology-centric culture noted above, it is necessary to initiate such change through education, starting at the undergraduate level. This means instilling an understanding and respect for users among software engineering students. A primary need is comprehending the value that the product to be developed has for its customers and users. That, in turn, typically requires an understanding of the economic and business context of the product. In other words, what is needed is insight that extends beyond the immediate technological issues. Software engineering graduates must have an understanding and working knowledge of economic and business factors that influence what they design and build[3]. As Charles Babbage put it: "It is doubly important for the man of science to mix with the world".

In addition, designing software products that are used by people requires an understanding of human psychology. At present, the prevailing attitude among software developers seems to be that human factors constitute a second-order concern, to be addressed by user-interface design specialists once the main system architecture has been finalized. Often this is viewed as a mere matter of designing suitable graphical interfaces and menu items. The understanding that usability requirements might have a fundamental impact on the architecture of a software system is still rare among software professionals.

Last but not least, it is necessary to increase the introduction of MDD methods into software engineering education. Most current undergraduate curricula already include some basic elements of model-based engineering, such as courses on UML. But, with no systematic theoretical foundation on which to base this, the results are often haphazard and inadequate. To address that, more research is needed into the theory behind MDD.

[3] One additional benefit of a broader education is an understanding when technological solutions are appropriate and when they are not. There are many examples when technological solutions have created more problems than they solved.

4.2 Research

The research community has embraced the notion of MDD, partly because they see it is an opportunity to effect a dramatic sea change in software technology. For example, modeling languages can be designed to avoid the arbitrary complexity of current programming languages. This complexity is sometimes a barrier to the application of highly-effective engineering methods, such as the use of mathematical analyses to predict key system characteristics before the system is constructed. All too often in software, such characteristics remain unknown until the complete system is designed and built, at which point the cost of redesign can be prohibitive. New modeling languages can be designed that are specifically designed to support such analyses.

Yet, despite the eagerness with which researchers have accepted MDD, there are some issues with the current research efforts. One of them is that much of the research focuses on particular point problems, in large part because research funding is provided by industrial partners who are primarily interested in solving their immediate problems. Consequently, there is insufficient exploration of the theoretical foundations of MDD. What is needed, therefore, is an overall map of the MDD research space in which the various areas of exploration are clearly identified as are the relationships between them. Only when this is properly understood will we be able to talk of a comprehensive and systematic *theory of MDD*.

One recent initiative is intended to deal with this issue: the newly formed Centre of Excellence for Research in Adaptive Systems (CERAS) [8]. This is a multi-year research effort organized and funded by the Ontario Centres of Excellence (specifically, its Centre for Communications and Information Technology) and the IBM Center for Advanced Studies, with the objective of exploring the foundational issues behind a number of related emerging computing technologies, including MDD. One of its primary objectives in the domain of MDD is to define a comprehensive framework for MDD research (the research map mentioned above). Another key objective is to provide a communal focal point and a kind of clearinghouse for MDD research worldwide. In addition to exploring the foundations of MDD, CERAS will be doing research in the following areas (and, likely, others):

- Modeling language semantics and design (including domain-specific languages).
- Model transformations (including model-to-model and model-to-code).
- Model analysis (safety and liveness property checking).
- Model-based verification.
- Model management.
- MDD methods and processes.
- MDD tooling.

4.3 Standards

The role of standards, *de facto* or *de iure*, is key to the success of any widely used technology. Standardization will not only allow the distilling of proven results in a vendor-neutral manner, it will also facilitate specialization by providing a common foundation for interworking between specialties. Standardization bodies, such as the OMG are not only useful, but necessary. However, given the highly competitive nature of the industry and the unprecedented flexibility of software, it is difficult to expect software vendors to voluntarily conform to standards. Therefore, users of MDD tooling and software professionals in general should do their utmost to pressure vendors to contribute and adhere to software standards

Clearly, there should be a close link between research, industry, and standards bodies. It is critical that only things that are both well understood and proven in practice be standardized. Premature standardization can be counterproductive.

5 Summary and Conclusions

MDD has the potential to provide significant improvements in the development of software. It is based on sound and time-proven proven principles: higher levels of abstraction, higher levels of automation, and standardization. Furthermore, there are numerous verifiable examples of successful applications of MDD in industrial practice, that are existence proofs of its viability and value. Yet, the use of MDD is still an exception rather than the norm. This is due to the not insignificant hurdles that need to be overcome. Although many of these are technical in nature, what may be surprising to some is that the most difficult hurdles to overcome are the ones stemming from the current idiosyncratic culture of software development. This culture places far too much emphasis on technology and not enough on technology users and their needs. It is a very pervasive culture that is sustained in part by the current business climate that is heavily focused on short-term gain and, thus, discourages investment in new methods and tools.

In such circumstances the best that can be expected is a gradual introduction of MDD, facilitated primarily through changes in educational curricula and investment in foundational research. Software engineers must be much better educated in human factors and the workings of the marketplace; they should view technology more as a tool rather than as an end unto itself. At the same time, we need to research and develop a systematic theory of MDD, to ensure that the corresponding technology and methods are well understood, useful, and dependable.

References

[1] G.. Booch et al. "An MDA Manifesto", en Frankel, D. and Parodi, J. (eds.) The MDA Journal: Model Driven Architecture Straight from the Masters. Meghan-Kiffer Press, Tampa, Florida, 2004 (pp. 133-143).

[2] OMG. MDA Guide (version 1.0.1), OMG document number omg/03-06-01, <http://www.omg.org/docs/omg/03-06-01.pdf>, 2003.

[3] G. Booch. Saving Myself. <http://booch.com/architecture/blog.jsp?archive=2004-00.html>, July 22, 2004.

[4] F. Brooks. The Mythical Man – Essays on Software Engineering (Anniversary edition). Addison-Wesley, 1995. ISBN: 0201835959.

[5] OMG. MDA Success Stories (web page). <http://www.omg.org/mda/products_success.htm>.

[6] N.J. Nunes et al. Industry Track papers in UML Modeling Languages, and Applications – 2004 Satellite Activities (Revised Selected Papers), Lisbon, Portugal, October 2004, Lecture Notes in Computer Science, vol. 3297, Springer-Verlag, 2005 (pp. 94-233). ISBN: 3540250816.

[7] IDC. The 2007 Worldwide Professional Developer Model. IDC document number #207143. <http://www.idc.com/getdoc.jsp?containerId=207143>, 2007.

[8] Ontario Centres of Excellence (OCE). Centre of Excellence for Research in Adaptive Systems (CERAS). <https://www.cs.uwaterloo.ca/twiki/view/CERAS/CerasOverview>.

The Domain-Specific IDE

Steve Cook and Stuart Kent

Years of pursuing efficiencies in software development through model-driven development techniques have led to the recognition that domain-specific languages can be an effective weapon in the developer's armoury. But these techniques by themselves are necessarily limited; only by assimilating them into the overall context of a domain-specific development process and tools can their real power be harnessed.

Keywords: Domain-Specific Languages (DSL), Domain-Specific Tools (DST), Model-Driven Architecture (MDA), Model-Driven Development (MDD).

Authors

Steve Cook works at Microsoft, and is the software architect of the Domain-Specific Language Tools which are part of Microsoft Visual Studio. He is currently working on future versions of these tools. Previously he was a Distinguished Engineer at IBM, which he represented in the UML 2.0 specification process at the OMG. He has worked in the IT industry for more than 30 years, as architect, programmer, author, consultant and teacher. He is a member of the Editorial Board of the Software and Systems Modeling Journal, a Fellow of the British Computer Society, and holds an Honorary Doctor of Science degree from De Montford University (United Kingdom). <steve.cook@microsoft.com>.

Stuart Kent is a Senior Program Manager on the Visual Studio team in Microsoft. Stuart joined Microsoft in 2003 to work on tools and technologies for visual modelling. This culminated in the Domain-Specific Language Tools, which are now part of the Visual Studio core tooling platform and are described in a recent book (Domain-Specific Development with Visual Studio DSL Tools) that he co-authored. Before joining Microsoft, Stuart was an academic and consultant, with a reputation in modelling and model driven development. He has over 50 publications to his name and made significant contributions to the UML 2.0 and MOF 2.0 specifications. He is a member of the editorial board of the Software and Systems Modeling journal, and on the steering committee for the MoDELS series of conferences. He has a PhD in Computing from Imperial College, London. <stukent@microsoft.com>.

1 Introduction

The development of information systems is getting increasingly complex as they become more and more distributed and pervasive. Today's advanced software developer must be familiar with a wide range of technologies for describing software, including modern object-oriented programming languages, eXtensible Markup Language (XML) and its accessories (schemas, queries, transformations), scripting languages, interface definition languages, process description languages, database definition and query languages, and more. Translating from the requirements of a business problem to a solution using these technologies requires a deep understanding of the many architectures and protocols that comprise a distributed solution. Furthermore, end-users expect the result to be fast, available, scaleable and secure even in the face of unpredictable demand and unreliable network connections. It can be a daunting task.

In areas other than software development, such as electronic consumer products (TVs and HiFis), cameras, cars and so on, we have come to expect a high degree of reliability at low cost, coupled increasingly in many cases with the ability to have items customized to satisfy individual needs. These expectations are met because of advances in industrial manufacturing processes made over many decades. Building a car or a television involves the coordination of a complex chain of manufacturing steps, many of which are wholly or partially automated.

We would like to apply similar principles to the construction of software. The main difficulty in doing so is that we have not yet developed techniques for software description that allow different concerns within the software development process to be effectively separated and effectively integrated. Although we increasingly use different languages for different tasks (programming languages for writing application logic, XML for transmission of data between application components, Structured Query Language (SQL) for storing and retrieving data in databases, Web Services Description Language (WSDL) for describing the interfaces to web-facing components) there are many complexities involved in getting these languages to work effectively together.

2 Domain Specific Modelling Languages

2.1 Model Driven Development (MDD)

Model driven development is an approach to software development where the main focus of attention shifts from writing code by hand to dealing with higher level abstractions (models). The approach aims to increase productivity, improve reliability and be more predictable.

Typically, a model driven development solution develops incrementally in stages, as follows.

In the first stage, a solution starts out as a way of getting an initial boost in productivity by generating code that is duplicated within and between software applications, instead of writing it by hand. In this situation, the model provides the information that is variable in amongst the duplicated code, and the code generators merge this with boilerplate code to produce the final result. As it is unlikely that all the required code can be generated from the model, the architecture of the software application may need to be adjusted to ensure that the generated code is kept separate from any hand written code.

In the next stage, as the code generators become more complex, it is realized that much of the duplication can be removed by creating a framework using constructs, where available, in the underlying programming language. This generally won't remove all the duplication, but it will remove bloat from the code generators and make them easier to maintain.

In a third stage, it may be possible to remove the need for code generation altogether, and write the framework so that it directly interprets the model.

In subsequent stages, once models have become a first class citizen in the software development process, they can then be treated as the target of transformations from yet more abstract models.

However, there is a lot to consider, even in the first stage, including:

- What language should the model be expressed in?
- What's the best way to write the code generators?
- How do we expose the code generators to the users of the tools, for example when a ship-blocking bug needs to be fixed?
- How do we ensure that generated code can be mixed with non-generated code so that regeneration does not overwrite the non-generated code?
- How do we ensure that generated code builds and exhibits the correct behaviour?
- How is the generated code tested?
- How does a developer debug through the generated code? Should he need to?

There aren't easy answers for all these questions. Indeed, in the next section we argue that questions such as these require us to think in terms of making the whole tooling environment domain specific, with models and code generators being only a part of a more holistic, integrated environment. Nevertheless, when models are an important part of the overall solution, the most burning question is the first: what language is used to express the models? That's the focus of the remainder of this section, and, when considering an answer, it's worth noting that answers to the other questions involve writing tools, including code generators, which must be able to access the models programmatically. The range of situations in which model driven development provides a productive approach depends directly on how easy it is to build and test those tools.

[1] The OMG also uses the trademarked term *Model Driven Architecture (MDA)* for its particular take on MDD.

2.2 UML

A language often associated with MDD is the Unified Modeling Language (UML), which is a standard of the Object Management Group (OMG)[1].The development of the UML started during the early 1990s, when it emerged as a unification of the diagramming approaches for object-oriented systems developed by Grady Booch, James Rumbaugh and Ivar Jacobson. First standardized in 1997, it has been through a number of revisions, most recently the development of version 2.

UML is large and complicated, version 2 especially so. To understand UML in any depth it is important to understand how it is used. We follow the lead of Martin Fowler, author of "UML Distilled," one of the most popular introductory books on UML. Martin divides the use into UMLAsSketch, UMLAsBlueprint, and UMLAs ProgrammingLanguage (for more details see <http://martinfowler.com/bliki>).

UMLAsSketch is very popular. Sketches using UML can be found on vast numbers of whiteboards in software development projects. To use anything other than UMLAs Sketch as a means of creating informal documentation for the structure of an object-oriented design would today be seen as perverse. In this sense, UML has been extremely successful, and entirely fulfilled the aspirations of its creators who wanted to eliminate the gratuitous differences between different ways of diagrammatically depicting an object-oriented design.

UMLAsProgrammingLanguage is an initiative supported by a rather small community, which is unlikely to gain much headway commercially, and which we will not dwell upon.

UMLAsBlueprint characterizes the use of UML in MDD. In this role, UML suffers from two problems:

- *Bloat*. In most MDD solutions, it is likely that only a small subset of the language will be applicable. So, although tools implementing UML do generally provide rich programmatic access to models created using them, the surface area of the API against which tools - such as code generators - are written is large, which just makes it that much harder to code against. Also, unless the chosen graphical UML editor allows significant parts of the language to be 'switched off', the user of the MDD solution needs to have external guidance on how to use the UML tool within the context of that solution.

- *Not domain specific*. In order to drive code generators in MDD, it's necessary to develop models that fit the domain, that can supply the exact information required by the code generators to fill the gaps in the boilerplate and produce fully executable code, and which conform to the necessary validation rules that ensure the code generated is well-formed and builds correctly. UML does support an extension mechanism, called UML Profiles, which allows additional data and validation rules to be added to a model. But this cannot do anything about fundamental conceptual mismatches between a domain and UML: at best, profiles allow UML to be used for MDD where there is a reasonable conceptual match between the domain and UML, and all that is required is some additional data and validation rules to make the models precise enough to drive code generators.

2.3 Domain-Specific Modelling Languages

An alternative to using the UML is to define a domain-specific language specially designed for the MDD solution.

At first sight, this seems like a significant undertaking, especially when you consider that in an MDD context the language needs to be supported by an editor, often graphical, which validates models and delivers them in a machine-readable form, as well as providing rich API access to those models. Indeed, a reason that implementers of MDD solutions may have opted for the sub-optimal UML-based approach is that it means they don't have to build their own graphical editor!

However, this is changing. There are now environments available such as Microsoft's Domain Specific Language Tools (DSL Tools) [1], Eclipse Graphical Modeling Framework (GMF) [2] and MetaCase's MetaEdit+ [3] which significantly reduce the cost of creating your own graphical, domain specific modelling language and editor. These tools generate a clean API for accessing models, and also include support for writing code generators. The editors that are created support graphical modelling using custom graphical notations, and allow rich model validation constraints to be included.

One criticism aimed at using DSLs is that you can end up creating a range of slightly different languages, one for each MDD solution. This can be confusing to users, leading to a lot of very similar looking but subtly different languages and editors, and also confusing to tool builders who might end up writing against similar but subtly different APIs. In contrast, in the UML approach you have a base language which can be shared between different solutions, and then have an extensibility mechanism which allows the base language to be customized for each MDD solution. The problem with the UML approach is not the principle of having a base language that is then extended and customized, but the fact that the UML has not been architected well to support this. For this approach to be effective, the UML should have been defined as a collection of small, loosely coupled, unconstrained base languages capturing specific modelling styles (class diagram style, component style, state diagram style, sequence style etc.), where any detailed content was defined through extensions (e.g. the Java, .Net and Object-Oriented analysis class diagram extensions).

3. The Domain-Specific IDE

We've suggested that model-driven development can be made more efficient by designing and implementing domain specific modelling languages aimed at specific software development problems. But there's much more to software development than modelling, and we'd like to make the entire software development process more efficient, not just the modelling part of it.

Models form one aspect of the software development experience. We must integrate this aspect with others across the entire lifecycle: envisioning, architecture, design, coding, debugging, testing, deploying and managing. We are increasingly discovering that this entire lifecycle is domain-specific, and that implementing domain-specific software development languages goes hand-in-hand with implementing domain-specific processes.

Narrowing the domain means building more tools. These are not simply modelling tools, or domain-specific extensions to programming languages. The domain-specific lifecycle also requires domain-specific commands, user interfaces, processes and guidance. Elsewhere we've described these requirements as "software factories" [4]. We've found that this can be an overloaded term, referring as it does to several distinct concepts:

- an organization designed to develop a particular kind of software;
- a set of processes that execute to deliver a particular kind of software;
- an integrated set of interactive software tools designed to support such processes.

In this article we are mainly interested in the third of these, which we'll call here the Domain-Specific Interaction Development Environment, or Domain-Specific IDE. We'll look now at some of the requirements for this.

3.1 Agility

Agile programming embraces evolutionary change throughout the software development lifecycle and is increasingly recognized as best practice for software development. The Domain-Specific IDE must support agile software development practices. The IDE must avoid processes or commands that force the developer into premature commitments that are expensive and time-consuming to reverse. Examples of these are code generation steps or "wizards" that cannot be repeated at a later time. The use of models within the IDE can help with this, as illustrated by Microsoft's Web Services Software Factory: Modeling Edition [5].

3.2 Integrating Multiple Languages

The definition of software inevitably involves a combination of languages. Even in the most traditional environment there will be distinct languages for programming and for defining and manipulating data. Today's popular software stacks involve multiple languages: for .NET they in-

[2] See <http://www.martinfowler.com/bliki/InternalDslStyle.html>.

clude C#, Visual Basic, SQL, and many dialects of XML e.g. XAML, with domain-specific languages added to the mix.

As soon as the artefacts implemented by these languages cross-reference each other, as they must, we find breakdowns in agility. For example, renaming a class or a namespace in a code file can cause compilation errors in a XAML file which refers to the class or namespace. Finding and correcting such errors can be costly, and a better solution would be a refactoring engine that spans multiple languages. This means that new domain-specific languages should be able to participate in such a refactoring engine, in order to be first-class citizens in the IDE.

One approach to alleviating such breakdowns is to expand the scope of one language to cover more of the lifecycle. Good examples of this approach are the latest versions of C# and VB .NET, which incorporate LINQ (Language-Integrated Query) giving strongly-typed integrated capabilities for accessing data stores. Further down this route would be more facilities in these languages for enabling embedded DSLs[2] . Few truly multi-paradigm languages exist today, though. LISP was the first example of a multi-paradigm language, and has fallen out of widespread use in favour of the strongly-typed object oriented programming languages which are commonly in use today. We're sure that these languages do not represent the ultimate destiny of programming: new developments in programming languages such as F# [6] enable considerable increases in expressive power and the design of rich abstractions. F# is complicated, though, and truly mastering it may be out of the reach of many software developers.

But however powerful the language, it is unlikely ever to be the case that a single language can successfully span the entire software lifecycle. Whether through limitations in languages or through limitations in their users, techniques for integrating multiple languages will inevitably be required.

3.3 Code Generation and "Reverse Engineering"

Code generation is an increasingly prevalent feature of modern IDEs that allows designs to be expressed in domain-specific terms. Code is often generated from models, defined in a domain-specific modelling language or in UML. Code can also be generated from other artefacts such as XML dialects, domain-specific textual languages, data held in databases, data captured from a What You See Is What You Get (WYSIWYG) editor (*e.g.* Windows Forms) or via User Interface (UI) automation (wizards, snippets etc). The advantages of code-generation are a substantial reduction in cost and errors for the creation of repetitive boilerplate code. Code generation does, however, have several potential disadvantages:

- *Non-repeatability*. A badly-designed code generation system can lose the developer's input after generation, forcing the developer to commit to a set of values and making it difficult to change their mind.
- *Hand-modification*. A system that requires code to be modified by hand after it has been generated may prevent a re-generation without erasing the hand-modifications, unless markers are placed in the generated code to indicate the hand-written parts, which can make the code unpleasant to read.
- *Difficulty of interfacing generated and hand-written code*. Languages features such as C#'s partial classes, in which a single class can be partially defined in multiple files, are essential to enable the combination of generated and hand-written code.
- *Difficulty of modification*. If the generated code is not what is required, for example it needs to be modified to introduce additional aspects such as logging or events, then it may be necessary to change the code generator itself to make such modifications. Such changes can be error-prone and incur a considerable test burden.

In the early days of model-driven development there was a strong tendency amongst the vendors of model-driven tools to claim the capability to do "reverse engineering", i.e. to create a model by reading and parsing a codebase. It is wrong to think of such a procedure as the inverse of code generation. Useful tools can be created for visualizing a large codebase, for example showing namespaces, dependencies, class hierarchies, call graphs etc. These tools operate at the same level of abstraction as the code and are used to help understand it at that level. By contrast, code generation schemes transform representations from a domain to another, more general domain. It is possible to reverse the generation process to visualize generated code in terms of the source artefacts from which it is generated, but it is not generally feasible to extract domain-specific representations from an arbitrary hand-written codebase.

3.4 Validation, Debugging and Testing

One of the most compelling advantages of domain-specific representations is the ability to validate software artefacts at an appropriate level of abstraction. It is much easier, for example, to establish that communication paths in a layered architecture conform to the architecture's rules in a model of layers, components and connections, than it is by trying to deduce the communication paths from the eventual code. Simple constraint violations, such as non-uniqueness of property names or cycles in supposedly acyclic graphs, can be detected and fixed as early as possible in the lifecycle.

Validation can be "hard", enforced by a user experience that simply does not allow invalid configurations to be created, or "soft", i.e. evaluated on demand with error and warning messages referring the developer to the source of the problem. Hard validations are usually more expensive to implement, but offer a more habitable user experience; soft validations rely on carefully written error messages that effectively direct the user to the root problem. It is important to be able to save invalid representations: the IDE should never force the developer to re-organize their life in order simply to save their work.

Debugging should also offer a domain-specific experience. If the software is described in terms of domain-specific concepts then its execution may be observed and stepped through in terms of the same concepts. Implementing a domain-specific modelling language, for completeness, should include implementing debugger plugins that enable program execution to be visualized and controlled in domain-specific terms.

Domain-specific development also impacts testing. The designer of a domain-specific language has two kinds of testing to think about: testing that the DSL does what it is supposed to do, and providing an environmen, e.g. a generated test harness[3], for users of the DSL to test the systems that they build using it.

3.5 Building the Domain Specific IDE

We've observed that narrowing the domain can improve productivity, but involves building more tools, which in turn means that it must be relatively cheap to do so. We tackle this by applying domain-specific techniques to the tool-building problem. We analyze the domains involved in tool-building, and build tools that support development in those domains.

Work on model-driven development over the past decade or so has involved a lot of thought about "meta-modelling". This subject is fraught with misunderstanding and confusion. Popular definitions are simply wrong: for example, the oft-heard definition of a meta-model as "a model of a model" is both unhelpful and incorrect. If a meta-model is anything, it is a model of the concepts expressed by a modelling language. Discussions about meta-modelling habitually degenerate into a kind of cultish mysticism almost entirely unconnected with the business of making software.

In practice, a meta-model is a model which is used (usually through code-generation) to build some aspects of a modelling tool. The result is that the tool can be built more cheaply, and thus bring extra efficiencies to solving the actual problem at hand. We'd like to generalize this principle to the entire development environment. We don't just model modelling concepts, we model all aspects of how the IDE is constructed, extended and deployed. We're especially interested in how all of the pieces that constitute a domain-specific IDE extension can be modelled: these include languages, commands, extra data carried by the project system, forms and toolwindows to interact with that data, and so on.

4 Conclusion

This article has proposed that an important step forward in software development tools is the development of domain-specific interactive development environments. Such tools recognise the advantages of model-driven development and domain-specific languages, while assimilating these techniques into an overall development experience tuned for the specific problem at hand.

[3] In software testing, a test harness or automated test framework is a collection of software and test data configured to test a program unit by running it under varying conditions and monitor its behavior and outputs. <http://en.wikipedia.org/wiki/Test_harness>.

References

[1] Steve Cook, Gareth Jones, Stuart Kent, Alan Wills. "Domain-Specific Development with Visual Studio DSL Tools", Addison-Wesley 2007. Also see <http://msdn.com/vsx>.

[2] Eclipse Foundation. Graphical Modeling Framework (GMF). <http://www.eclipse.org/gmf/>

[3] MetaCase. MetaEdit+. <http://www.metacase.com/>.

[4] Jack Greenfield, Keith Short, Steve Cook, Stuart Kent. "Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools", John Wiley 2004.

[5] Microsoft. Web Service Software Factory: Modeling Edition. MSDN Library. <http://msdn2.microsoft.com/en-gb/library/bb931187.aspx>.

[6] Microsoft Research. <http://research.microsoft.com/fsharp/fsharp.aspx>.

Model Intelligence: an Approach to Modeling Guidance

Jules White, Douglas C. Schmidt, Andrey Nechypurenko, and Egon Wuchner

Model-Driven Engineering (MDE) facilitates building solutions in many enterprise application domains through its use of domain-specific abstractions and constraints. An important attribute of MDE approaches is their ability to check a solution for domain-specific requirements, such as security constraints, that are hard to evaluate using traditional source-code focused development efforts. The challenge in many enterprise domains, however, is finding a legitimate solution, not merely checking solution correctness. For these domains, model intelligence that uses domain constraints to guide modelers is needed. This paper shows how existing constraint specification and checking practices, such as the Object Constraint Language, can be adapted and leveraged to guide users towards correct solutions using visual cues.

Keywords: Constraint Checking, Constraint Reasoning, Domain-Specific Modeling, Model-Driven Engineering, Modeling Guidance.

1 Introduction

Model-Driven Engineering (MDE) [1] has emerged as a powerful approach to building complex enterprise systems. MDE allows developers to build solutions using abstractions, such as custom diagramming languages, tailored to their solution domain. For example, in the domain of deploying software to servers in a datacenter, developers can manipulate visual diagrams showing how software components are mapped to individual hosts, as shown in Figure 1.

A major benefit that MDE approaches provide is that custom constraints for each domain can be captured and embedded into an MDE tool. These domain constraints are properties, such as the memory demands of a software component on a server, that cannot be easily checked by a compiler or other third-generation programming language tool. The domain constraints serve as a domain solution compiler that can significantly improve the confidence in the correctness of a solution. The most widely used constraint specification language is the Object Constraint Language (OCL) [2].

Although MDE can improve solution correctness and catch previously hard to identify errors, in many domains

Authors

Jules White is a Ph.D. student in the Department of Electrical Engineering and Computer Science (EECS) at Vanderbilt University. His research focuses on using constraint optimization techniques for modeling guidance and optimization, constraint-based automated assembly of component applications, model-driven development, and distributed Java systems. He is currently the lead developer of the Generic Eclipse Modeling System (GEMS) <http://www.eclipse.org/gmt/gems>, a part of the Eclipse GMT project. Before joining the DOC group, he worked for IBM's Cambridge Innovation Center and was involved with constraint modeling and rule-based systems. <jules.white@ gmail.com>.

Douglas C. Schmidt is a Full Professor in the Electrical Engineering and Computer Science (EECS) Department, Associate Chair of the Computer Science and Engineering program, and a Senior Research Scientist at the Institute for Software Integrated Systems (ISIS) at Vanderbilt University, Nashville, TN. For the past two decades, he has led pioneering research on patterns, optimization techniques, and empirical analyses of object-oriented and component-based frameworks and model-driven development tools that facilitate the development of distributed middleware and applications. Dr. Schmidt is an expert on distributed computing patterns and middleware frameworks and has published over 350 technical papers and 9 books that cover a range of topics including high-performance communication software systems, parallel processing for high-speed networking protocols, real-time distributed object computing, object-oriented patterns for concurrent and distributed systems, and model-driven development tools. <d.schmidt@vanderbilt.edu>.

Egon Wuchner works as a researcher and consultant in the Coporate Technology SE2 department of Siemens AG in Munich, Germany. He is an expert in software architecture and distributed systems. His research focuses on concepts, technologies and tools to improve the development of large distributed systems, e.g. their handling of operational requirements, their comprehensibility and maintanability. His recent research has been in Aspect-oriented Software Development and Model-Driven Development. <egon.wuchner@siemens.com>.

Andrey Nechypurenko is a senior software engineer in Siemens AG Corporate Technology (CT SE2). Mr. Nechypurenko provides consulting services for Siemens business units focusing on distributed real-time and embedded systems. Mr. Nechypurenko also participates in research activities on model driven development and parallel computing. Before joining Siemens AG, he worked in the Ukraine on high performance distributed systems in the telecommunications domain. <andrey.nechypurenko@ siemens.com>.

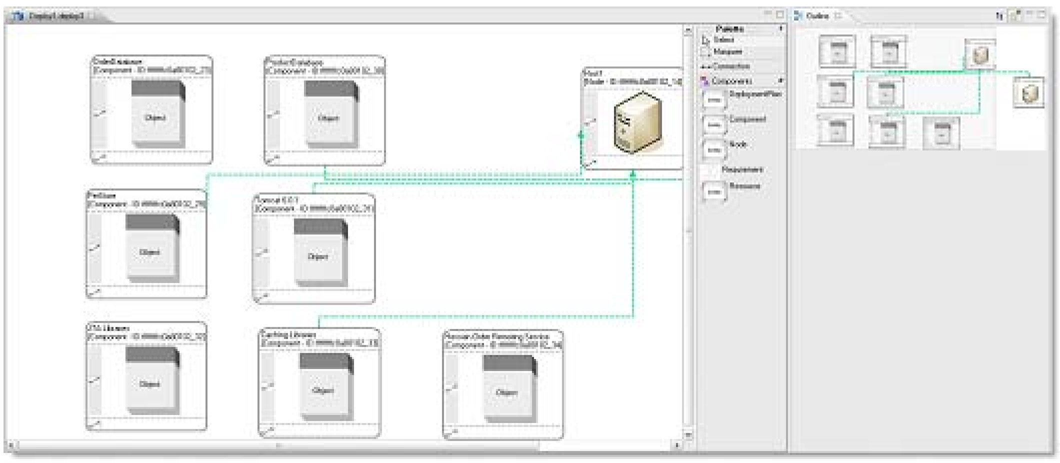


Figure 1: Deployment Model for a Datacenter.

the major challenge is deriving the correct solution, not checking solution correctness. For example, when deploying software components to servers in a datacenter, each component can have numerous functional constraints, such as requiring co-hosting a specific set of other components with it, and non-functional constraints, such as requiring a firewalled host, that make developing a deployment model hard. When faced with large enterprise models with 10s, 100s, or 1,000s of model elements and multiple constraints per element, manual model building and validation approaches do not scale.

Enterprise models can also contain global constraints, such as stipulating that no host's allocation of components exceeds its available RAM, which further complicates modeling. Although languages like OCL can be used to validate a solution, they still do not make finding the correct solution any easier. Developers must still manually construct models and invoke constraint checking to see if a mistake has been made.

The following properties of enterprise models make building models challenging:

- Enterprise models are often large and may contain multiple views, making it hard or infeasible for modelers to see all the information required to make a complex modeling decision.
- Constraints in enterprise systems often involve func-

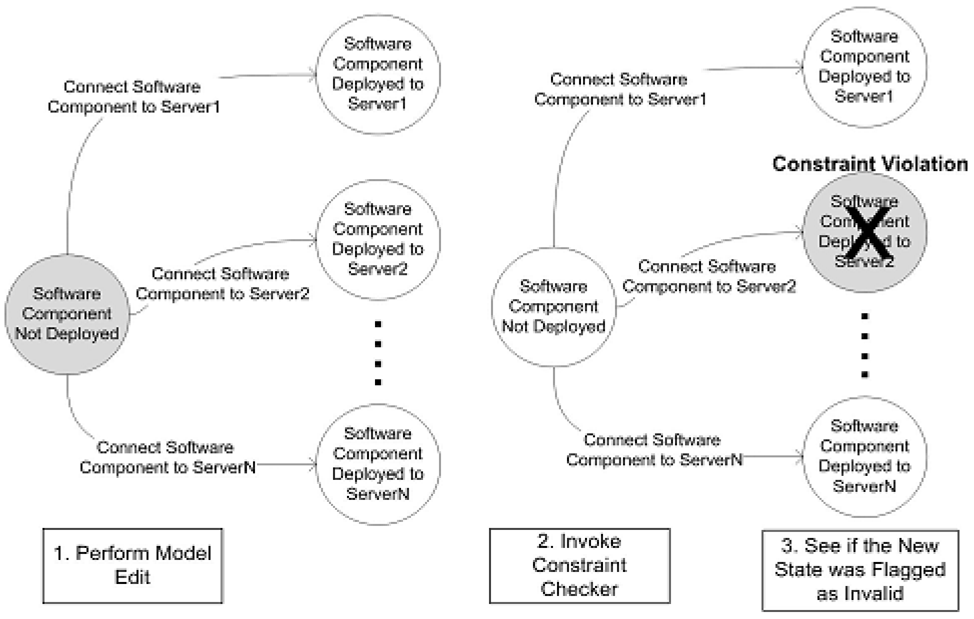

Connect Software Component to Server1
Software Component Deployed to Server1
Connect Software Component to Server2
Software Component Deployed to Server2
Software Component Not Deployed
Connect Software Component to ServerN
Software Component Deployed to ServerN
1. Perform Model Edit
Connect Software Component to Server1
Software Component Deployed to Server1
Constraint Violation
Software Component Deployed to Server2
X
Connect Software Component to Server2
Software Component Not Deployed
Connect Software Component to ServerN
Software Component Deployed to ServerN
2. Invoke Constraint Checker
3. See if the New State was Flagged as Invalid

Figure 2: Model Editing and Constraint Checking.

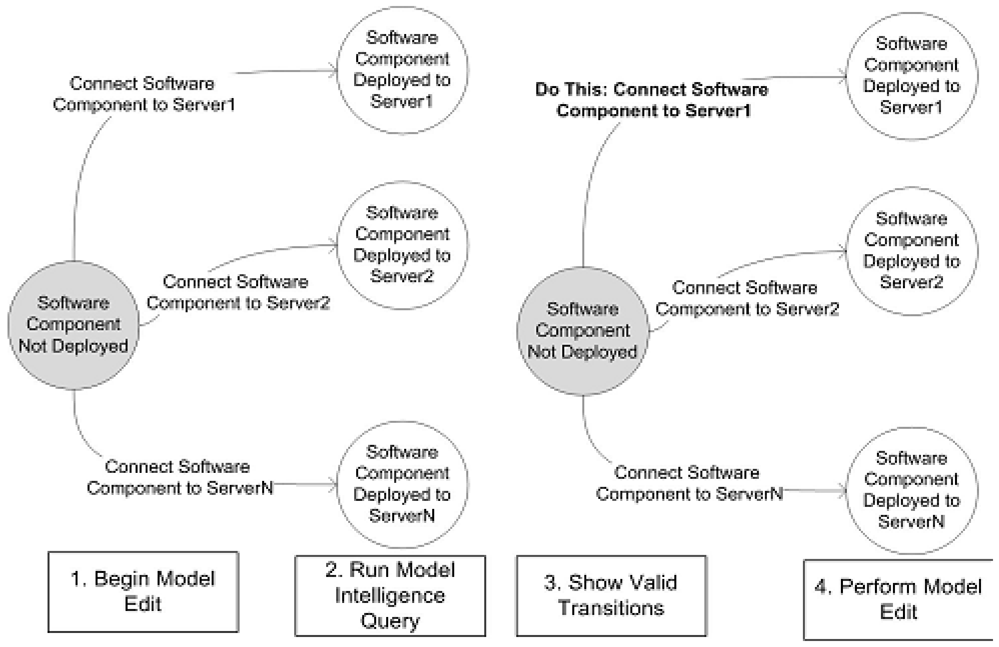

Connect Software Component to Server1
Software Component Deployed to Server1
Software Component Deployed to Server2
Connect Software Component to Server2
Software Component Not Deployed
Connect Software Component to ServerN
Software Component Deployed to ServerN
Do This: Connect Software Component to Server1
1. Begin Model Edit
2. Run Model Intelligence Query
3. Show Valid Transitions
4. Perform Model Edit

Figure 3: Model Editing Sequence for Model Intelligence.

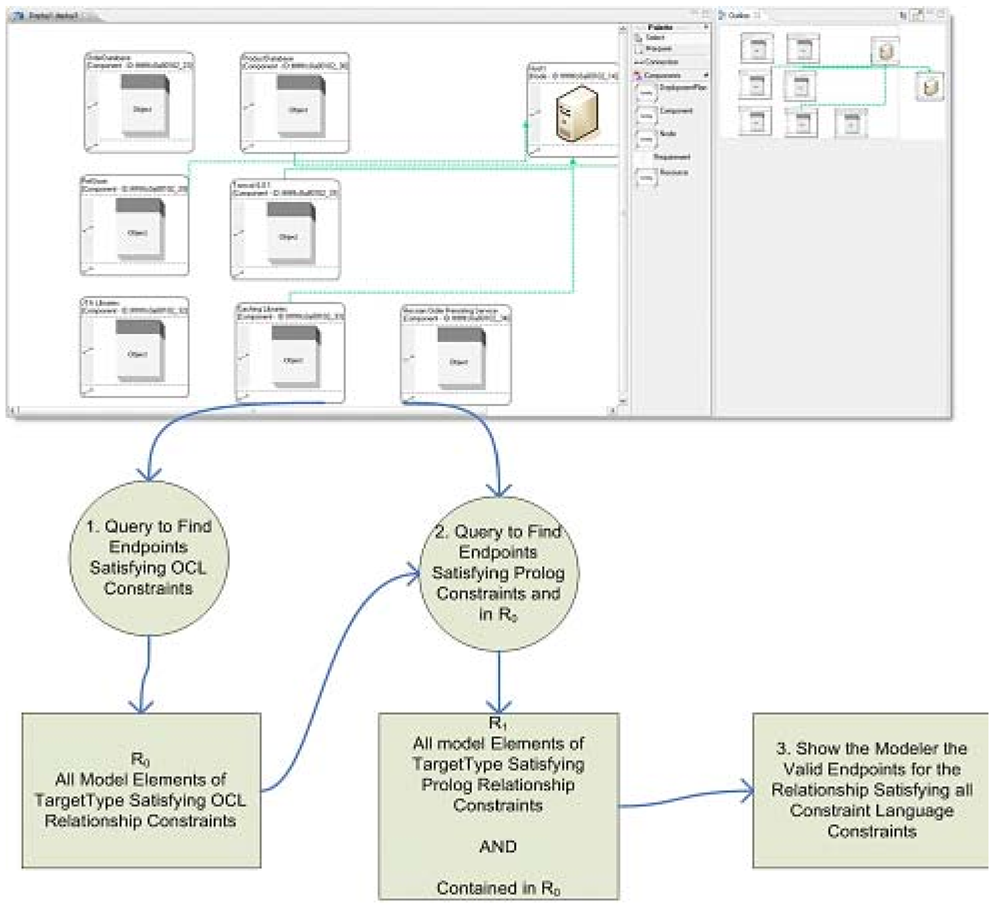

1. Query to Find Endpoints Satisfying OCL Constraints
2. Query to Find Endpoints Satisfying Prolog Constraints and in R_0
R_0 All Model Elements of TargetType Satisfying OCL Relationship Constraints
R_1 All model Elements of TargetType Satisfying Prolog Relationship Constraints AND Contained in R_0
3. Show the Modeler the Valid Endpoints for the Relationship Satisfying all Constraint Language Constraints

Figure 4: Model Intelligence Queries Across Multiple Constraint Languages.

tional and non-functional concerns that are scattered across multiple views or aspects of a model and are hard to solve manually, and

- Enterprise modeling solutions may need to satisfy complex global constraints or provide optimality, both of which require finding and evaluating a large number of potential solution models.

Current model construction techniques are largely manual processes. The difficulty of understanding an entire large enterprise model, coupled with the need to find and evaluate a large number of potential solutions, makes enterprise modeling hard.

To motivate the need for tool support to help modelers deduce solutions to domain constraints, we use an application for modeling the deployment of software components to servers in a datacenter. Ideally, when creating a deployment, as a developer clicked on each individual software component to deploy it, the underlying tool infrastructure could use the domain constraints to derive the viable hosts for the component. We refer to these mechanisms for guiding modelers towards correct solutions as *model intelligence*.

2 Limitations of Current Constraint Checking Approaches

To motivate the challenges of using existing constraint infrastructure, such as OCL, as a guidance mechanism, we will evaluate a simple constraint for deploying a software component to a server. For each component, the host that it is deployed to should have the correct OS for which the component is compiled. This constraint can be captured in OCL as:

```
context:SoftwareComponent;
inv: self.hostingServer.OS = self.requiredOS;
```

After a `SoftwareComponent` has been deployed to a server, the above constraint checks that the host (stored in the `hostingServer` variable) has the OS required by the component. As shown Figure 2, to utilize the constraint, the modeler first makes a change to the model (Step 1), invokes the constraint checker (Step 2), and then sees if an error state has been entered (Step 3). The challenge is that the modeler cannot predict ahead of time if the model is being transitioned to an invalid state. A state is only checked for errors after control has been transitioned to it.

One way around the inability to check the constraint before the host is committed to the `SoftwareComponent` is to use OCL preconditions as guards on transitions. An OCL precondition is an expression that must hold true before an operation is executed. The chief problem of using OCL preconditions as guards, however, is that they are designed to specify the correct behavior of an operation performed by the *implementation* of the model. Using an OCL precondition as a guard during modeling requires defining the constraint in terms of the operation performed by the *modeling tool* and not the model.

For example, the precondition that should be imposed to check for the correct OS is a constraint on an operation (e.g., creating a connection) performed by the modeling tool, not by the model. To define the OCL precondition, therefore, developers must define the OCL constraint in terms of the modeling tool's definition of the operation, which may not use the same terminology as the model. Moreover, defining the constraint as a precondition on an operation performed by the modeling tool requires developers to create a duplicate constraint to check if an existing model state is correct.

Without two constraints (one to check the correctness of the modeling tool action and one to check the correctness of an already constructed model state) it is impossible to identify operation endpoints and ensure model consistency. The OCL precondition approach therefore adds complexity by requiring developers to maintain separate (and not necessarily identical) definitions of the constraint that can potentially drift out of sync. The precondition approach also couples the constraint to a single modeling platform since the precondition is defined in terms of the connection operation exposed by the tool, not the model.

3 Model Intelligence: an Approach to Modeling Guidance

A modeling tool can implement model intelligence, by using constraints to derive valid end states for a model edit *before* committing the change to the model. Traditional mechanisms of specifying constraints associate a constraint with objects (*e.g.*, `SoftwareComponents`) rather than the relationships between the objects (*e.g.*, the deployment relationship between a `SoftwareComponent` and a `Server`).

To determine the validity of a relationship between two objects, therefore, the relationship must be created and committed to the model so that constraints on the two objects associated with the relationship can be checked.

The transitions in the state diagram from Figure 2 correspond to the creation of relationships between objects. To support model intelligence, a tool needs to use domain constraints to check the correctness of the modification of relationships between objects in a model before the modification is committed to the model. If constraints are associated with the relationships rather than the objects, a tool can use the constraints associated with the relationship to deduce valid end states and suggest transitions to a modeler.

3.1 Constraining Relationships

Relationships between objects are edges in the underlying object graph of a model. Each edge has a source and target object. Using this understanding of relationships, constraints can be created that specify the correctness of a relationship in terms of properties of the source and target elements.

For example, the deployment of a `SoftwareComponent` to a `Server` is represented as a *deployment* relationship. A constraint can be applied to a deployment relationship and specified in terms of the properties of the `source` (*e.g.*, a `SoftwareComponent`) and the `target` (*e.g.*, a `Server`):

```
context:Deployment;
inv: source.requiredOS = target.OS;
```

A key property of associating constraints and specifying them in terms of the source and target of the relationship is that a constraint can be used to check the correctness of the creation of a relationship *before* the relationship is committed to a model. Prior to the creation of a relationship, the proposed source and target elements can be substituted into the constraint expression and the constraint expression checked for correctness. If the constraint expression holds true for the proposed source and target elements, the corresponding relationship can be created in the model.

Section 2 showed that using existing OCL approaches to model intelligence requires maintaining separate specifications of each constraint. If constraints are associated with relationships and expressed in terms of the source and targets of a relationship, they can be used to check the validity of a modeling action *before* it is committed to the model. Moreover, the same constraint can be used to check existing relationships between modeling elements, which can not be done with the standard OCL approach.

3.2 Relationship Endpoint Derivation

A model can be viewed as a knowledge base, *i.e.*, the model elements define facts about the solution. The goal of model intelligence is to run queries against the knowledge base to deduce the valid endpoints (e.g., valid hosts for a component) of a relationship that is being created by a modeler. In terms of the state diagram detailing a model editing scenario shown in Figure 3, the queries derive the valid states to which a model can transition.

The creation of a relationship begins by modelers selecting a relationship type (e.g., a deployment relationship) and one endpoint for the new relationship (*e.g.*, a `SoftwareComponent`). Model intelligence uses the relationship type to determine the constraints that must hold for the relationship and then uses the constraints to create queries to search the knowledge base for valid endpoints to create the relationship, as shown in Step 2 of Figure 3. The valid endpoints determine the valid states to which the model can transition. As shown in Step 3 of Figure 3, the transitions that lead to these valid states can then be suggested to modelers as valid ways of completing an in-progress modeling edit.

The creation of a new relationship begins by the modeler selecting a source for the relationship and a type of relationship to create. Each relationship type has a set of constraints associated with it. Once model intelligence knows the source object and the OCL constraints on the relationship being modified, a query can be issued to find valid endpoints to complete the relationship. Using the OS deployment constraint from Section 2 the query to find endpoints for a deployment relationship would be:

```
Server.allInstances()->collect(target |
                    target.OS = source.OS);
```

In this example, model intelligence would specify to the OCL engine that the source variable mapped to the `SoftwareComponent` that had been set as the source of the deployment relationship. The query would then return the list of all Servers that had the correct OS for the component. For an arbitrary relationship, with constraint `Constraint`, between elements `Source` and `Target` of types `SourceType` and `TargetType`, a query can be com-

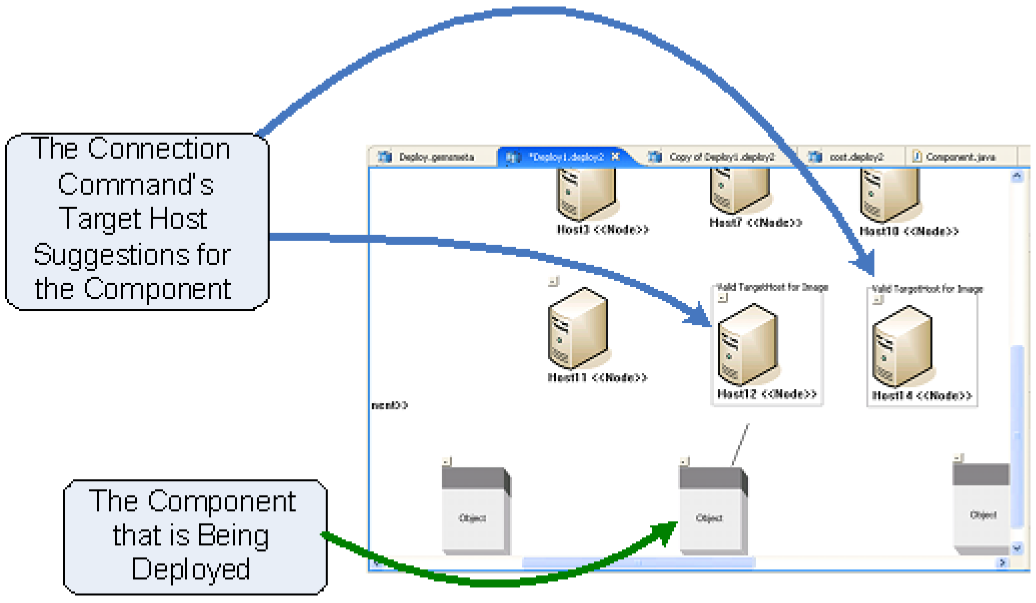


Figure 5: The Deployment Command Showing Valid Endpoints Derived via Model Intelligence.

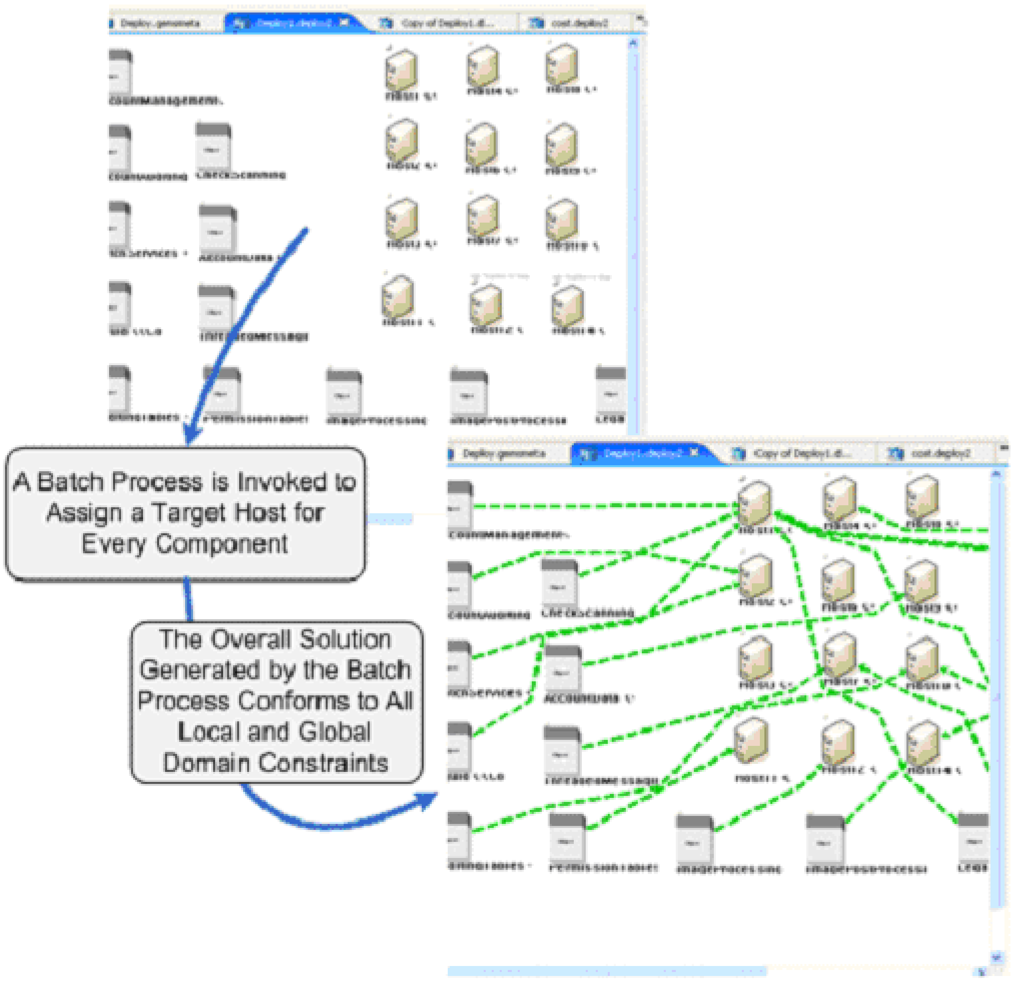

A Batch Process is Invoked to Assign a Target Host for Every Component
The Overall Solution Generated by the Batch Process Conforms to All Local and Global Domain Constraints

Figure 6: A Model Intelligence Batch Process to Assign a Host for Every Component.

posed to derive valid endpoints. Assuming that a relationship has endpoint `Source` set, a query can be issued to find potential values for `Target` as follow:

```
 TargetType->allInstances()->collect(target |
Constraint);
```

where `Constraint` is a boolean expression over the source and target variables. More generally, the query can be expressed as: *Find all elements of type* `TargetType` *where* `Constraint` *holds true if the source is* `Source`.

3.3 Endpoint Derivation Across Multiple Constraint Languages

Although we have only focused on OCL thus far, the generalized query definition from Section 3.2 can be mapped to other constraint or expression languages, as well. In prior work [4], we implemented model intelligence using OCL, Prolog, BeanShell, and Groovy. For example, Prolog naturally defines a knowledge base as a set of facts defined using predicate logic. Queries can be issued over a Prolog knowledge base by specifying constraints that must be adhered to by the facts returned. Model intelligence can also be used to derive solutions that are restricted by a group of constraints defined in multiple heterogeneous languages. An iterative result filtering process can be used to derive endpoints that satisfy constraints specified in multiple languages, as shown in Figure 4.

Initially, model intelligence issues a query to derive potential solutions that respect the constraint set of one constraint language.

The results of the query are stored in the set R_0. For each subsequent query language C_i, the results of the query that satisfy the language's constraint set are stored in R_i. For each constraint language C_i, where $i > 0$, model intelligence issues a query using a modified version of the query format defined in Section 3.2: *Find all elements of type* `TargetType` *where* `Constraint` *holds true if the source is* `Source` *and the element is a member of the set* R_{i-1}.

The modified version of the query introduces a new constraint on the solution returned: all elements returned as a result were a member of the previous result set. A simple mechanism for specifying result sets is to associate a unique ID with each modeling element and to capture query results as lists of these IDs. The modified queries can then be defined by checking to ensure that both the constraint set holds and the ID property of each returned modeling element is contained by the previous result set.

4 Integrating Model Intelligence with the Command Pattern

There are a large number of uses for model intelligence, including automatically performing an autonomous batch process of model edits and providing visual feedback to modelers. In this section, we show how model intelligence can be integrated with the *Command pattern* [3] to provide visual cues to aid modelers in correctly completing modeling actions. The Command pattern uses an object to encapsulate an action and its needed data and is used in many graphical modeling frameworks, such as the Eclipse Graphical Editor Framework [5]. As a modeler edits a model, commands are created and executed on the model to perform the actions of the modeler.

Modeling platforms provide tools, such as a connection tool, that a modeler uses to manipulate a model. Each tool is backed by an individual command object, such as a connection command. When a modeler chooses a tool, an instance of the corresponding command class is created. Subsequent pointing, clicking, and typing by the user, sets the arguments (e.g., connection endpoints) operated on by the command. When the arguments of the command are fully specified (e.g., both endpoints of a connection command are set), the command executes.

Section 3 described the ability to highlight the valid deployment locations for a software component after a modeler clicked on it to initiate a deployment connection. This functionality can be achieved by combining model intelligence with a deployment connection command. After the initial argument to the deployment connection command is set, the command can use model intelligence to query for valid deployment locations. If there is a single server that can host the component, the command can autonomously choose it as the deployment location and execute. If there is more than one potential valid host, each host can be highlighted via a command to help the user select the command's final argument, as shown in Figure 5.

5 Concluding Remarks

Our experience developing models for enterprise application domains indicates that simply determining if a model is correct is not always helpful. We have learned that using constraints to verify the correctness of relationships between objects (rather than just individual object states) allows modeling tools to guide modelers towards correct solutions by suggesting ways of completing edits. Moreover, batch processes can be built atop of suggestion mechanisms to allow tools to autonomously complete sets of modeling actions. For example, a batch process can be created to deploy a large group of software components, by deriving sets of valid hosts for each component and intelligently selecting a host from each set, as shown in Figure 6. In other work [4], we have used model intelligence as the basis for creating batch modeling processes that use constraint solvers to automate large sets of modeling actions and optimally select endpoints for relationships to satisfy global constraints or optimization goals.

Our implementation of model intelligence for the Eclipse Modeling Framework [6], called GEMS EMF Intelligence, is an open-source project available from <www.eclipse.org/gmt/gems>.

References

[1] J. Bézivin. "In Search of a Basic Principle for Model Driven Engineering", Novatica/Upgrade, V(2):21-24, 2004.

[2] J.B. Warmer, A.G. Kleppe. "The Object Constraint Language: Getting Your Models Ready for MDA". Addison-Wesley Professional, New York, NY, USA, 2003. ISBN: 0321179366.

[3] E. Gamma, R. Helm, R. Johnson, J. Vlissides. "Design Patterns: Elements of Reusable Object-oriented Software", Addison-Wesley, Boston, MA, USA, 1995. ISBN: 0201633612.

[4] J. White, A. Nechypurenko, E. Wuchner, D.C. Schmidt. "Reducing the Complexity of Designing and Optimizing Large-scale Systems by Integrating Constraint Solvers with Graphical Modeling Tools", in "Designing Software-Intensive Systems: Methods and Principles", edited by Dr. Pierre F. Tiako. Langston University, Oklahoma, USA, 2008.

[5] Graphical Editor Framework, <www.eclipse.org/gef>.

[6] F. Budinsky, S.A. Brodsky, E. Merks. Eclipse Modeling Framework. Pearson Education, Upper Saddle River, NJ, USA, 2003.

Model Differences in the Eclipse Modelling Framework

Cédric Brun and Alfonso Pierantonio

Increasingly, recording the various kinds of design-level structural evolution that a system undergoes throughout its entire life-cycle is gaining a fundamental importance and cannot be neglected in software modeling and development. In this respect, an interesting and useful operation between the designs of subsequent system versions is the difference management consisting in calculation, representation, and visualization. This work presents EMF Compare, an approach to model difference calculation and representation for the EMF (Eclipse Modelling Framework). Apart from enhancing the rank of model differences to that of first-class artifacts according to the "everything is a model" principle, the approach presents several properties which are discussed according to a conceptual framework.

Keywords: EMF Compare, Model Comparison , Model Differences.

Authors

Cédric Brun is a Research Engineer at Obeo and Project Lead of the EMF compare project in Eclipse. In charge of the Acceleo community, he also works on software evolution, re-engineering and cartography of legacy systems through model driven processes. He is a graduate of the Polytech engineering school and a graduate and research Master at the University of Nantes, and has specialised in software engineering and model driven engineering. Prior to his current jobs, he was an active contributor to Open Source development and worked in Guangzhou on a global video conference solution for the Chinese Education and Research Network (CERNET) <cedric.brun@obeo.fr>.

Alfonso Pierantonio is Associate Professor in the Computer Science Department at the University of L'Aquila, Italy. His present research interests include general model engineering and more specifically model transformation and techniques for model differencing and management in current model-engineering platforms. He has been involved in program and organization committees of conferences and co-edited several special issues on scientific journals about these subjects. <alfonso@di.univaq.it>.

1 Introduction

The last decade witnessed a dramatic growth of software intricacy and different techniques and methodologies have been proposed to ease complex system development. Model Driven Engineering (MDE) [1] shifts the focus of software development from coding to modelling and lets software architects harness the opportunity of dealing with higher-level abstractions. In this respect, models represent descriptions of phenomena of the real (or imaginary) world which are usually complete with regard to the designer's goal, i.e. a specific task which the designer is pursuing such as code generation or software analysis. However, models reach their fundamental effectiveness when they can be manipulated by means of automated transformations in order to obtain different kinds of artifacts ranging from other models to documentation or even implementation code. It is important that designers are able to comprehend the various kinds of design-level structural evolution that a software system undergoes throughout its entire life-cycle. Nurturing the detection of differences between models is essential to model development and management practices, which are traditionally not neglected in high-quality software development processes [2]. Thus, these activities are crucial not only for understanding the system design and its evolution but also for obtaining an accurate picture of the quality requirements of the system so that it can be consistently evolved.

The problem of model differences is intrinsically complex and requires algorithms and notations [3] [4] which permit to benefit fully from its potential in MDE. This paper presents part of the state of the art in calculating model differences and outlines a conceptual framework which prescribes crucial requirements to enhance differences to first-class entities. Accordingly, a solution must necessarily have a high degree of separation between three relevant aspects in model differentiation: *calculation*, *representation*, and *visualization*. In fact, in current proposals the distinctions between the three aspects are often blurred thus compromising the adoption of generic modelling techniques [5]. In this paper, we discuss the problem of model differences and illustrate how EMF Compare [6] addresses this difficult task in the Eclipse generic platform. In particular, the approach is metamodel-independent, i.e. it applies to models which conform to arbitrary metamodels, and is based on similarity techniques (see Sect. 2) which provide enhanced flexibility and interoperability. Moreover, it is model-based in the sense that the outcome of a model comparison is represented by means of a model which enables its manipulations in model-to-model or model-to-text transformations.

The paper is structured as follows: in Section 2 an introduction to the problem of model differences is presented and a number of representation requirements are given. Section 3 presents EMF Compare describing both calculation, representation, and an evaluation with regard to the requirements introduced in Section 2. Finally some conclusions are drawn.

2 Model Difference

As previously mentioned, the problem of model differenciation is intrinsically complex and in order to analyse and/or propose a possible solution, it is important to decompose the problem into its constituent parts. In fact, its complexity is manifold and refers at least to the following aspects:

a) calculation: a procedure, method or algorithm able to compare or contrast two distinct models;

b) representation: the outcome of the calculation must be represented in some form, current notations present deficiencies since they are heavily affected by the calculation method or by the proposed application;

c) visualization: model differences often requires to be visualized in a human-readable notation which let the designer grasp the rationale behind the modification which the models underwent during their lifetime.

In the sequel of the paper we will discuss these aspects according to the available literature and will try to present and characterize EMF Compare according to them.

Calculation. In the context of software evolution, difference calculation has been intensively investigated as witnessed by a number of approaches ranging from text–comparisons to model–differencing techniques. As stated by T. Mens in [7], delta calculation algorithms can be classified by different points of view, each of which is related to the particular application the approach is used for. Specialized differencing methods have been introduced to strictly compare Universal Modelling Language diagrams, such as [8] [9] [10] amongst others. These approaches can be divided in two main categories depending whether they make use of *persistent identifiers* or *similarity* metrics: the former relies heavily on identifiers which are assigned to model elements by the modelling tools. This compromises interoperability and locks the models within a specific platform since identifiers are not universally computable. The latter approach establishes how similar two model elements are by comparing not only the properties local to the elements but also their global properties which makes the method agnostic of the modeling tools being independent from the any identification mechanism thus making the method independent of modelling tools and indentification mechanisms. A generalization of the work by Z. Xing and E. Stroulia [10] is an approach based on structural similarity which is able to compare not only UML (Universal Modelling Language) models but also models conforming to any arbitrary metamodel [11]. This represents an advance towards a wider acceptance of difference and version management in software development and generic modelling platforms (for instance [12] [13]).

Representation. Detecting differences and identifying mappings among distinct versions of a system design is preparatory to represent at least part of such knowledge. Finding a suitable representation for model differences is crucial for its exploitation, as for instance deriving refactoring operations from a delta document[1] describing how a database schema evolved in time. However, the effectiveness of representation of model differences is often compromised by factors such as the calculation method or the scope of the model difference. For instance, in the case of edit scripts the representation is operational since it describes how to modify the initial model in order to obtain the final model. Clearly, such a representation notation suffers from a lack of abstraction and, let alone the capability of reconstructing the final model, does not easily allow any further manipulation or analysis since it requires *ad-hoc* tools. In other cases, the representation may even be model-based (which permits further manipulations of the differences), as in the case of coloring, but the visualization and the representation tend to overlap and the overall method is affected by the way the differences are computed, i.e. in a set-theoretic fashion. In general, a proper representation must contain all the information defining the differences and must make this knowledge available to further analyses and manipulations. Thus, we believe it must be given in terms of *abstract syntax* by introducing suitable metamodels as outlined below.

Visualization. Differences often require to be presented according to a specific need or scope highlighting those pieces of information which are relevant only for the prefixed goal. In other words, a visualization is realized by giving a *concrete syntax* which renders the abstract syntax (representation) and may vary from intuitive diagrammatic notations to textual catalogues as, for instance, spreadsheet data. The same representation may include different visualizations depending on the specific purpose the designer has in mind. In this respect, both edi scripts and colouring represent two different visualizations although they are generated directly by the specific differencing algorithm and letting the representation be rendered by means of internal formats which prevent them from being processed in tool chains. For instance, edit scripts render both representation and visualization with the same notation.

Clearly, the calculations and representations are the central ingredients for any solution. In particular, we are interested in those representations which raise model differences to the rank of first class objects fulfilling the "everything is a model" principle [5]. As a consequence, a number of desirable properties must be imposed on representation techniques as discussed in [14] and described below.

1) model-based, the outcome of a difference calculation must be represented as a model to enable a wide range of possibilities, such as subsequent analysis, conflict detection or manipulations;

2) compactness, the difference model must be compact and contain only the necessary information to represent the modifications, without duplicating parts such as those model

[1] A document which lists the changes in the contents of another document.

elements which are not involved in the change;

3) self-contained, a difference model must not rely on external sources of information, as for instance references to base model elements or base metamodels;

4) transformative, each difference model must induce a transformation, such that whenever it is applied to the initial model it yields the final model. Moreover, the transformation must also be applicable to any other model which is possibly left unchanged in case the elements specified in the difference model are not contained in it;

5) compositionality, the result of subsequent or parallel modifications is a difference model whose definition depends only on difference models being composed and is compatible with the induced transformations;

6) metamodel independence, the representation techniques must be agnostic of the *base* metamodel, i.e., the metamodel which the base models conform to. In other words, it must be not limited to specific metamodels, as for instance happens for certain calculation methods (e.g., [9] [10]) which are given for the UML metamodel.

The above discussion presents a minimal set of requirements which should be taken into account in order to let a generic modeling platform deal with an advanced form of model management. In the next section, we will illustrate EMF Compare showing how our approach fulfills most of the described requirements.

3 EMF Compare

EMF Compare is an Eclipse project which was initiated in 2006 at Eclipse Summit Europe, where the need for a model comparison engine emerged. The Obeo and Intalio companies [15] [16] contributed the first implementation of this component which has had two stable releases since that time. The goals of this component are to provide "out of the box" model comparison and merge support. Even if we think that one unique algorithm is able to provide good results both in term of efficiency and performance, we are aware that there may be several solutions to a problem, at different levels of generality and which depend on the main concerns one wants to address with model comparison (see Sect. 2). That is why this component has been designed with a high degree of extensibility in mind and every part of the entire comparison process is customizable.

The global comparison process is generally admitted as being composed of two main parts: the *matching* and the *differencing* parts. In EMF Compare these parts are explicitly separate and processed by two kinds of data processors, the matching and the differencing engine, respectively. These engines are pluggable components: generic engines are provided to match and analyse any model conforming to an arbitrary meta-model (they will be described in the next section) but one can plug in new ones in order to adapt these operations for a given meta-model or to experiment with new algorithms.

Another strong aspect of this implementation is that we think that models should be implemented as already suggested. That is why EMF Compare is based on model representations of both differencing and matching of two models. That means one can get those models and use them to produce differences reports thanks to model-to-text transformation, or can refactor the differencing model to ignore some differences. In this respect, the method is *model-based* according to the requirements in Section 2.

3.1 Calculation Method

Analysing models to identify the matching information is the fundamental part of the comparison process and inaccuracies in this phase will affect the quality of the overall difference detection mechanism. Consequently this algorithm produces most of the calculation complexity. In essence, we have to consider all the elements of both versions of the model and decide whether an element in the first version is the same as another one in the second version. We do use the "same" word as we do not want to test equality, we are just trying to find out if this element has a common ancestor. Next we will analyse the intrinsic differences of these elements to produce the difference model.

The generic match engine is based on statistics, heuristics, and instances which are compared with four different metrics aggregated in an overall score of matching. These metrics analyse the name of an element, its content, its type and the relations it has with other elements; it returns a value ranging from 0 (nothing in common) to 1 (identity) which will be balanced with additional factors in order to get the overall score. Especially, the "name" metric tries to find an attribute standing for a name of the model element, the "type" metric compares the meta-class features, this is useful if you want to consider the possible types refactoring (an Interface changed in Class for instance). The "relation" metric considers the linked instances both from containment and from non-contained relations, respectively. Finally, the "content" metric analyses the intrinsic content of the instance.

In general, the comparison uses a great deal of information which are not relevant and that can be, therefore, called *information noise*. The metrics gets "false high scores" because most of the data comes from default values which are shared amongst instances. These cases have been processed by means of a filter, which first analyses both models and maintains a record of the features which "always have the same values in both models", then ignores such features while computing the metrics. As a consequence, the metric scores are more realistic as they are not affected by this information noise.

We only described the 2-way comparisons, since the 3-way comparisons can be given in terms of

2-way comparisons as specified by the match model. In particular, a difference can be an instance of either Match2Elements or Match3Elements metaclasses with the latter defined in terms of the several instances of the former. Moreover, model elements which do not have a match are referenced as an UnMatched entity.

In order to evaluate the score of a content match, we first create a string representation of what is contained in the instances, and then we compare both strings using a simple "string pairing" algorithm. Each metric uses the same kind of process; it first gets a string representation of what we call the "relations" of an element, and then compares these strings.

Ideally, for each element of the first version we have to discover the most similar element of the second version. Unfortunately, most of the complexity lies there because it needs to browse the second model for each element of the first model. In the EMF Compare implementation we started from the following assumptions: most of the things do not change and the probability of moving an element outside its "neighborhood" is really low. Thus, the chosen matching strategy analyses both models at the same time, matching the elements available within the limits of a given search window. Upon completion of the analysis, elements which are not matched will be compared with each other to produce new matches. The outcome is a match model which is, in turn, passed to the differencing engine which operates in a quite straightforward fashion. In fact, once elements from both models are put in correspondence, they are compared and eventual differences are evaluated.

With regard to the discussion in Section 2, the computation algorithm is based on a measure of similarity and does not fall within the class of methods which make use of persistent identifiers, which makes the computation quite general and suitable for tool chaining and integration. The decomposition of the algorithm in a matching and differencing module permits the individual reuse of such components and the opportunity for such components to be easily adopted in the realization of additional functionalities, as *model patching*, for instance.

3.2 Representation

Both match and differencing information are represented by means of models which can be reused in model transformations; such models conform to the *match* and *diff* Ecore meta-models [6]. A match model is a specialization of a weaving model [5] which provides associations between elements from the first model and elements from the second model. Another data item which we encode in the model is the overall score evaluated while performing the matching.

In the rest of the section, the representation mechanism of EMF Compare is evaluated with regard to the properties given in Section 2. In particular, the approach satisfies the *model-based* requirement since the calculated differences are represented through models that conform to the provided *diff* metamodel mentioned above. Being more precise, a difference model reflects the changes made on the first model to obtain the second one, representing them by means of meta-classes like AddModelElement or UpdateAttribute and difference containers called DiffGroup. The AddModelElement metaclass has two references, one to the element which has been added in the final model, and another to the corresponding container in the initial one. In this sense, the approach is *metamodel independent*. In fact, the *diff* metamodel provides constructs able to represent differences between arbitrary models and it does not make any assumptions about the metamodel which the models being differenced have to conform to.

With regard to the *compactness*, the approach produces difference models which represent only the elements involved in the changes and the differenced models are not duplicated as in case of colouring. Moreover, EMF Compare provides facilities to reduce the verbosity and the complexity of the difference models. In order to understand them, let us consider a rich metamodel like UML2: this metamodel provides a huge expressiveness since each metaclass has a many attributes and references. For instance, an association between two classes involves many metamodel elements like AssociationEnds, Properties and so on. This means that when we compare two UML models, the user is overwhelmed by too many details and analysing them is quite difficult. For instance, an added property may come from the fact that the developer added an association and that it is one of its end properties.

To cope with these problems, EMF Compare enables the specification of higher level differences. In particular, by means of meta-model extensions one can contribute a new kind of difference, for instance AddNavigableAssociation which will hide the three AddModelElement detected for the association and the two properties. With this new kind of difference a new processor is contributed which will refactor the original diff model in order to create the new AddNavigableAssociation instances. This is useful in order to get different kinds of granularity on the difference and to handle specific merging in which order is important.

The representation of the differences produced by means of EMF Compare are *transformative* but with some limitations. In particular, each difference model induces a transformation which when applied to the first model generates the final one. However, the representation is not context-independent since the induced transformation cannot be applied to arbitrary input models but only to the first one used for the difference calculation. Nevertheless, this aspect does not compromise *compositionality* and difference models of subsequent versions of a model can be composed together.

Even if the difference model is deduced from the match model, we do not want it to depend on the match model. That means that every information item which is relevant to the difference, and as such needed to merge these differences, should be available in the difference model. This confers to the technique the important *self-containment* property.

3.3 Performance

Manipulating realistic scale models and, in particular, calculating differences between models can pose major questions about computational efficiency. In fact, performance has been one of the key concerns with regard to the generic engines provided with EMF Compare. For instance, the latest release compares two UML2 models of approximately five thousand elements in a few seconds. Of course, many parameters affect the performances of the comparison, the first one being the number of differences. The more differences we have (especially added and removed elements), the more we need to iterate through the remaining items at the end of the matching process. Model structure is also an important parameter affecting the approach. In fact, a more structured model allows faster comparisons since the structure eases the task of finding matching elements.

This leads to another issue linked with the way the generic match engine analyses the models. An instance identity can be often regarded as valid within a certain *locality*, as for instance a package containing a class is definitely an important element for the class identity, but for some other kind of models this assumption does not hold and the analysis strategy is consequently inefficient.

Finally, the biggest problem with the current implementation is common to many systems based on threshold values, since these thresholds are based on massive experiments on many real world models and are not based on any formal theory nor able to auto-adapt themselves. Though this pragmatic approach is useful and gives encouraging results, it would probably benefit from techniques that prevent elements from being "just under the metric threshold" leading to an inaccurate comparison.

4 Conclusions and Future Work

Model differencing has been intensively investigated over the last few years. There has been some work (e.g., [8] [9] [10]) that proposed automated UML–aware differencing algorithms which, in contrast with traditional lexical approaches, such as GNU diff-like tools (see [17] [18] [19] among others), are capable of capturing the high-level logical/structural changes of a software system. More recently, another approach [11] based on structural similarity extended differencing to metamodel independency, i.e., to models conformant to an arbitrary metamodel. However, the capability of tools to operate on change documentation which conforms only to their own internal format tends to lock software development into a single tool thus compromising its exploitation as part of a tool chain.

In this paper we have presented EMF Compare, a metamodel-independent approach to model differencing based on similarity techniques and fully implemented on the generic modeling platform provided by Eclipse. The problem of model differences presents several difficulties both in calculation and in representation. As opposed to other approaches, EMF Compare rigorously adheres to the requirements prescribed in [14] which assures that the method may be fully integrated into tool chains where differences can be manipulated or analysed by means of standard model-driven tools. With regard to the work in [4] EMF Compare shares many characteristics and provides a strong distinction among representation and visualization where the dividing line is somewhat blurred in the other approach.

Future work includes the enhancement of the *transformability* property. In essence, difference models can be viewed as model patches with a certain degree of fuzziness or adjustability in their application. To this end, different models as computed by EMF Compare require to be further transformed in another models conforming to the metamodels introduced in [14]. This would essentially need to flatten the weaving model given in the difference model as presented here.

Acknowledgements

We would like to thank Antonio Cicchetti and Davide Di Ruscio for their long and insightful discussions on this project.

References

[1] B. Selic. The Pragmatics of Model-driven Development. IEEE Software, 20(5):19–25, 2003.

[2] R. Conradi, B. Westfechtel. Version models for software configuration management. ACM Computing Surveys, 30(2):232–282, 1998.

[3] D.S. Kolovos, R.F. Paige, F. A. Polack. Model comparison: a foundation for model composition and model transformation testing Proceedings of the Int. Workshop GaMMa '06, ACM Press, 2006, 13-20.

[4] Y. Lin, J. Zhang, J. Gray. Model Comparison: A Key Challenge for Transformation Testing and Version Control in Model Driven Software Development. OOPSLA Workshop on Best Practices for Model-Driven Software Development, 2004.

[5] J. Bézivin. On the Unification Power of Models. Journal on Software and System Modeling, 4(2):171–188, 2005.

[6] EMF Compare. <http://wiki.eclipse.org/index.php/EMF_Compare>.

[7] T. Mens. A state-of-the-art survey on software merging. IEEE Trans. Softw. Eng. 28, 5 (2002), 449–462.

[8] M. Analen, I. Porres. Difference and union of models. In UML 2003 - The Unified Modeling Language (2003), vol. 2863 of LNCS, Springer-Verlag, pp. 2–17.

[9] D. Ohst, M. Welle, U. Kelter. Differences between versions of UML diagrams. In ESEC/FSE-11: Proc. ESEC/FSE (2003), ACM Press, pp. 227–236.

[10] Z. Xing, E. Stroulia. UMLDiff: an algorithm for object-oriented design differencing. In 20th IEEE/ACM ASE (2005), ACM, pp. 54–65.

[11] Y. Lin, J. Gray, F. Jouault. DSMDiff: A Differentiation Tool for Domain-Specific Models, European Journal of Information Systems (2007) 16, pp. 349–361.

[12] J. Bézivin, F. Jouault, P. Rosenthal, P. Valduriez. Modeling in the Large and Modeling in the Small. In Model Driven Architecture, European MDA Workshops: Foundations and Applications (2004), vol. 3599 of LNCS, Springer, pp. 33–46.

[13] A. Ledeczi, M.Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, P. Volgyesi. The Generic Modeling Environment. In Workshop on

Intelligent Signal Processing, 2001.
[14] A. Cicchetti, D. di Ruscio, A. Pierantonio. A Metamodel Independent Approach to Difference Representation. Journal of Object Technology, vol. 6, no. 9, Special Issue: TOOLS EUROPE 2007, Zurich (Switzerland), October 2007, pages 165–185.
[15] Obeo. <http://www.obeo.fr/>
[16] Intalio. <http://www.intalio.com/>
[17] S. G. Eick, T. L. Graves, A. F. Karr, J. S.Marron, A.Mockus. Does code decay? assessing the evidence from change management data. IEEE Trans. Software Eng., 27(1):1–12, 2001.
[18] S. G. Eick, J. L. Steffen, E. E. Sumner Jr. Seesoft-a tool for visualizing line oriented software statistics. IEEE Trans. Software Eng., 18(11):957–968, 1992.
[19] M. Fischer, M. Pinzger, H. Gall. Populating a release history database from version control and bug tracking systems. In Procs. ICSM 2003, pages 23–32. IEEE Computer Society.

Model-Driven Architecture® at Eclipse

Richard C. Gronback and Ed Merks

The Model-Driven Architecture (MDA®) initiative has come a long way in its seven-year history, as has the Eclipse open source community. With the top-level Eclipse Modeling Project (EMP) focused in part on working with standards organizations such as the Object Management Group (OMG), much of the promise of MDA is realized today with the software available to its community. And while much can be done to improve the state of the relationship between the OMG and the Modeling project, it can be argued that Eclipse leads the way in providing open source specification-compliant solutions, and is therefore contributing significantly to the overall success of MDA.

Keywords: Ecore, Eclipse, Eclipse Modeling Framework (EMF), Eclipse Modeling Project (EMP), Graphical Modeling Framework (GMF), Model-Driven Architecture (MDA), Textual Modeling Framework (TMF).

Authors

Richard Gronback is chief scientist for modeling products at Borland Software Corporation. Richard represents Borland on the Eclipse Board of Directors, co-leads the Modeling project Project Management Committee (PMC), and is the project lead for the Graphical Modeling Framework (GMF) project. Previously, Richard has worked for TogetherSoft, Ariba, Andersen Consulting, and was a reactor operator in the U.S. Navy. Richard holds a BSE in Computer Science & Engineering from the University of Connecticut. <richard.gronback@borland.com>.

Ed Merks is the project lead of the Eclipse Modeling Framework project and a co-lead of the top-level Modeling project. He has many years of in-depth experience in the design and implementation of languages, frameworks, and application development environments. He holds a Ph.D. in computing science and is a co-author of the authoritative "Eclipse Modeling Framework, A Developer's Guide" (Addison-Wesley 2003). He works for IBM Rational Software at the Toronto Lab. <merks@ca.ibm.com>.

1 Introduction

As stated in its charter, "*the importance of supporting industry standards is critical to the success of the Modeling project, and to Eclipse in general. The role of the Modeling project in the support of industry standards is to enable their creation and maintenance within the Eclipse community. Furthermore, as standards bodies such as the Object Management Group (OMG) have a strong modeling focus, the Modeling project needs to facilitate communication and outreach through its PMC and project contributors to foster a good working relationship with external organizations.*" When the OMG introduced MDA to the world in 2001, Eclipse was an incipient community. In the past seven years, MDA and Eclipse have experienced success while concurrently undergoing changes in focus, positioning, and applicability to the world of software development. Eclipse is no longer "just a Java IDE" while MDA is now based on a more complete set of specifications, making it much more well defined than seven years ago.

Although there is little mention of MDA proper within the Eclipse Modeling Project, it is nonetheless supported to a large degree, as will be discussed below. In fact, it could be argued that Eclipse has significantly contributed to the success and realization of MDA, providing an open source platform and de facto reference implementations for many of the MDA specifications. Unfortunately, this has been done with minimal collaboration with the OMG. It is likely that improved collaboration will increase the success of both organizations as they strive toward increasing the adoption of model-driven development.

2 Implemented Standards

The Eclipse Modeling Project is a top-level Eclipse project that is logically structured into projects that provide abstract syntax definition, concrete syntax development, model-to-model transformation, and model-to-text transformation. Additionally, the Model Development Tools (MDT) project [1] is focused on providing implementations of industry standard metamodels and exemplary tools for developing models based on those metamodels. This range of functionality provides its community with a full range of model-driven software development (MDSD) capabilities, many of which conform to published MDA specifications.

While the top-level Modeling project is the primary location for MDA-related activity at Eclipse, other projects within Eclipse have modeling-related technology and specification implementations. For example, the Software Process Engineering Model (SPEM) [2] is implemented as part of the Eclipse Process Framework (EPF) [3] project, while BPMN diagramming [4] is provided by the SOA Tools project [5].

It's also worth pointing out that the Eclipse Modeling Project provides alternative technologies for several of the OMG's MDA specifications. These will be discussed below, as they are popular technologies with strong user communities. In most cases, their implementations precede the corresponding OMG specification. Altogether, these projects are able to fulfill most of the MDA vision, while certainly fulfilling general MDSD and domainspecific language (DSL) tooling requirements. What follows is a list of the relevant MDA specifications and their implementation status within Eclipse. This is not an exhaustive list of MDA specifications, but those most relevant and within the current scope of the Eclipse Modeling Project.

2.1 Meta-Object Facility

The importance of a having common underlying metamodel cannot be understated and is provided for by the MOF™ [6] specification. MOF, or more specifically EMOF (Essential MOF), is closely aligned with the Ecore metamodel of the Eclipse Modeling Framework (EMF) [7] and forms the basis of most Modeling project technologies.

The topic of aligning the Ecore metamodel with the EMOF specification has been ongoing for years, and will likely continue, as the implications of alignment are nontrivial. As EMF is such a popular technology used within many Eclipse projects and commercial products, changes to its structure and API are not a viable solution. As has been discussed, updating the EMOF specification to align with Ecore may be a better solution. Although EMF forms the bedrock of nearly every Modeling project, there is still room for improvement. Discussions are ongoing in the areas of large-scale models, metamodel enhancements, alternative persistence mechanisms, and so on. What is important to keep in mind when considering the evolution of EMF and all Eclipse open source projects is that it's contribution-based. EMF itself is comprised of a very small team and has to deal with the maintenance and preservation of its current client base.

2.2 Unified Modeling Language™

UML® [8] is implemented within the UML2 component of the Model Development Tools (MDT) project [1] and currently conforms to the 2.1 version of the specification. This implementation of the UML2 metamodel is based on EMF and has been part of Eclipse for quite some time. Diagramming capabilities for the UML2 metamodel implementation are now provided by the MDT project's UML2 Tools component.

These diagrams are generated using the Eclipse Graphical Modeling Framework (GMF) project, itself an example of model-driven software development using Eclipse technologies. Both the metamodel and diagramming components provide support for the definition of UML Profiles. Profiles play an important role in MDA and in the definition of UML-based DSLs. While no UML Profiles are available at Eclipse today, it is possible they may be implemented and provided to the community in the future. Ideally, a catalog of profiles and other MDA artifacts would be contributed to and maintained by the community for general consumption.

2.3 Object Constraint Language

OCL [9] is an important element of MDA and is used in several Modeling projects. OCL is provided as component of the Model Development Tools project, with a complementary OCL Tools component coming in the near future. The OCL implementation conforms to the 2.0 version of the specification and has bindings to both Ecore and the UML2 metamodel implementations.

2.4 Diagram Interchange

UML Diagram Interchange (DI) [10] is not currently provided at Eclipse, but has prompted many questions from the community regarding its implementation, particularly with the introduction of UML diagramming from the MDT project.

This specification was found to be insufficiently powerful by the team that designed and implemented the notation model for the GMF runtime. It has been suggested that the DI specification should be revised to align with the GMF notation model, as there has been no broad adoption of the original version of the specification, whereas GMF has grown in popularity.

A related topic is the Diagram Definition RFP [11] that itself was inspired by the mapping GMF provides between Ecore models, their notation elements, and their tooling. This RFP will help bridge a gap that currently exists in modeling specifications from the OMG.

2.5 XML Metadata Interchange

XMI™ [12] is supported by EMF and is used by the UML2 project and others. EMF is also able to read serialized EMOF models, in addition to several other format import options, including XML Schema Definition (XSD).

2.6 MOF Query / View / Transformation

QVT [13] is part of the Model-to-Model Transformation (M2M) project [14] and currently provides an implementation of the QVT Operational Mapping Language (OML). The QVT Relations and Core languages are also being implemented within M2M. The M2M project provides another model-to-model transformation technology with its ATL component. ATL (Atlas Transformation Language) was a contender among responses to the QVT RFP, and has fostered a large and successful community of its own.

2.7 MOF Models to Text Transformation Language

MOF2Text is being implemented within the MOFScript [15] component of the Generative Modeling Technologies (GMT) project. This is a recent specification and implementation in an area where there is no shortage of alternative technologies. JET [16] originated as EMF's code generation framework and borrows heavily from Java Server Pages (JSP). JET is undergoing an update to enhance its capabilities, and resides within the Model-to-Text Transformation (M2T) project.

Xpand [17] is an increasingly popular template-based

M2T component that provides an alternative syntax and expression language to JET. Xpand provides additional extension capabilities, and continues to be enhanced via community contributions.

JET and Xpand are well used within the community, and while MOF2Text is relatively new and unproven, it is unlikely that the benefits it may offer will prompt the reimplementation of existing templates. Nonetheless, a MOF2Text contribution exists within the Modeling project for those looking for a specificationcompliant M2T solution.

2.8 Human-Usable Textual Notation

HUTN [18] is not currently implemented, but relates to the proposed Textual Modeling Framework (TMF) project [19] within Modeling. There has been a great deal of interest in tooling for the support of textual concrete syntaxes for modeling languages, particularly as the interest in DSLs and "language workbenches" [20] has grown.

The TMF proposal states that it will allow for the definition of concrete textual syntaxes for abstract syntaxes defined using EMF. A full-featured textual editor will be generated, likely targeting the capabilities of the proposed IDE Meta-tooling Platform (IMP) [21] project. Therefore, TMF will provide complementary concrete syntax to the graphical concrete syntax provided by Graphical Modeling Framework project.

2.9 Business Process Modeling Notation

The SOA Tools Project at Eclipse provides BPMN diagramming, mainly for the purpose of generating BPEL [22]. The diagramming is based on GMF, while the underlying model is based on EMF, thereby making this project compatible with other Modeling technologies.

As BPMN provides no well-defined metamodel, the introduction of the Business Process Definition Metamodel (BPDM) [23] will hopefully lead to a new contribution of this capability at Eclipse. As standards-based model implementations, the implementation of the BPDM metamodel and BPMN diagramming for working with these models would fall within the scope of the MDT project.

2.10 Software Process Engineering Metamodel

As mentioned, SPEM [2] is supported by the Eclipse Process Framework (EPF) project. While SPEM is mentioned in the list of MDA specifications, there is no real requirement for its use in the application of MDA. Within Eclipse, there is currently no connection between EPF and the Modeling project, aside from the fact the SPEM metamodel is implemented using EMF.

3 Working Relationship

To date, very little formal communication has taken place between the OMG and leadership of the Eclipse Modeling Project regarding a working relationship. Of late, the most promising discussions have been with respect to a series of symposia, to be held first at EclipseCon 2008 [24] and later in the year during an OMG technical meeting. The focus of these events will be to discuss individual specification implementations and how the two organizations can strive for more constructive cooperation. The current situation raises a number of questions about the nature of the relationship, which hopefully can be addressed during these meetings. It may turn out that the relationship should remain very informal, with no explicit commitment or expectation that implementations found in the Modeling project represent socalled "reference implementations" of OMG standards, as described in [25].

In the past, specifications such as the UML have suffered from interoperability issues among vendors who had different interpretations or implementation goals. The introduction of XMI, well-defined compliance levels, Diagram Interchange specification, etc. were intended to improve the situation, but have largely failed to deliver and now compound the problem. By developing a reference implementation in parallel with the specification, ambiguities and defects can be identified earlier and serve the larger community through delivery of a platform upon which to implement commercial products.

That said, the UML2 implementation at Eclipse is the de facto reference implementation for the UML2 specification, and its development exemplifies the model we would like to achieve with the OMG for all implemented standards within Eclipse. Only through communication and feedback between implementers and specification authors can our respective communities be best served.

3.1 Membership

Currently, the Eclipse Foundation is a member of the OMG, and the OMG is a member of the Eclipse Foundation. This is a start, but raises a question of what level of interaction and commitment this brings, particularly as corporate members of each are often involved in, and provide contributions to, both of these organizations.

What are the best techniques for aligning standards organization activities with reference implementation project team activities? Should members be required to participate in both contribution areas, where applicable? What does it really mean for the Eclipse Foundation to be a member of the OMG, and vice versa? What role would the Foundation representative have within the context of the OMG, and how would they coordinate with fellow members from the Eclipse community? What if there are competing goals among members? Are there new working models that would be more productive, and perhaps never before explored in this context?

3.2 Specification Delivery

Specifications with defined metadata should be delivered in a serialized format, preferably XMI. This is required by the standard RFP template for new specifications, but has not been mandated or required for all specifications currently published by the OMG.

Graphical notations (concrete syntax) are typically provided by drawings and natural language descriptions. While these are typically sufficient for describing the elements, they are not as precise as they could be and must be manually implemented in order to utilize them in modeling tools.

The delivery of specifications in formats that are machine consumable, particularly if used as inputs to generative tooling frameworks, should be an obvious benefit to those involved in specification, implementation, and consumption of these technologies. This includes metamodel constraints, which should be serialized and interpreted by the underlying tooling. At present, there is no standard way for EMF to define constraints (e.g. OCL), nor interpret constraints on models even if they were provided.

The UML specification contains domain (abstract) syntax and semantics, OCL constraints, and graphical (concrete) syntax, accompanied by natural language description and mapping to the domain. It would seem reasonable for specifications to be delivered in a manner where the abstract model is described separately from the concrete syntax, and where they are only related using a mapping definition. This approach will provide proper separation of concerns, and allow for the generation of graphical editors for various domain models.

As mentioned previously, the RFP for Diagram Definition should address the issue, which leaves us with graphical notation definition issue. Should graphical notations be defined in terms of a graphical definition metamodel, SVG, or another standard? With respect to mapping definitions, for example the myriad of mappings from UML2 Profiles to IMM (among others), should QVT be provided as part of these specifications?

3.3 Specification Compliance

There are generally provided a set of conformance criteria to be met when implementing a specification. With improved collaboration between implementation and specification organizations, it can be expected that some level of minimum compliance level be achieved, so as to provide a proper reference implementation. There are cases today where implementations at Eclipse are well aligned, or nearly aligned with OMG specifications. For example, the Eclipse UML2 project provides a compliant implementation of the UML 2.1 metamodel using the nearly EMOF-compliant Eclipse EMF project.

Should implementations be required to provide the highest level of compliance to defined specification acceptance criteria? Or, is a "best effort" approach adequate? What actions can or should be taken to provide specification alignment and/or conformance?

3.4 Implementations Influencing Specifications

As indicated previously, there are cases where existing implementations are close to a specification, yet not fully compliant. In the case where there is a large existing client base on a high quality, open source, implementation, why not align a specification with the implementation? For example, the previously mentioned case of EMF's Ecore model being not quite aligned with the EMOF specification.

There is a precedent for this type of influence between an open source implementation and OMG specification in the UML. The Diagram Interchange and Diagram Definition RFP are two more areas where this type of cooperation can be mutually beneficial. It is most often the case where specifications are themselves driven from implementations, although typically from a commercial vendor. Wouldn't an open source approach to implementations influencing specifications be a more equitable solution? This leads us directly to the next topic.

3.5 Open and Transparent Nature

In the case of Eclipse, contributions are done in the open, with an emphasis on meritocracy as the basis for achieving more responsibility within the community. Transparency is essential to the open source process at Eclipse, yet is somewhat different from the specification development process at the OMG [26]. Perhaps this is an area where the two organizations can influence one another?

If the development of a reference implementation were to be done in the open, it would follow that the developing version of the specification itself must be available. Otherwise, there would need to be a serial process of first developing the specification, publishing, and then implementing it, which eliminates the benefits of validating the specification while developing an implementation in parallel.

Can the process of developing standards be done in a more open and transparent manner, with an emphasis on addressing the needs of a developing reference implementation? Alternatively, could Eclipse support a model where source is not open until it reaches a required level of alignment with ongoing specification work?

4 Conclusion and Future Outlook

In summary, the promise of MDA can be realized to a large extent today using the capabilities provided by the Eclipse Modeling Project. As MDA encompasses a collection of specifications that align well with the implementation goals of the Eclipse Modeling Project, it seems the future of delivering a solid open source infrastructure for MDA tooling is bright. Practically speaking, there are many challenges remaining before the statement in the Modeling project's charter related to its relationship with standards bodies such as the OMG can be realized. A relationship that is too informal will be unlikely to yield the desired results, while a relationship that is strictly defined and enforced will likely limit the progress of implementation. The right balance will clearly benefit both of these organizations, their members, and ultimately the customers of commercial products that are standards-based.

References

[1] Eclipse Model Development Tools (MDT) Project. <http://www.eclipse.org/mdt>.

[2] Software Process Engineering Metamodel (SPEM) specification. <http://www.omg.org/technology/docu-

ments/modeling_spec_catalog.htm#SPEM>.
[3] Eclipse Process Framework (EPF). <http://www.eclipse.org/epf/>.
[4] Business Process Modeling Notation (BPMN) specification. <http://www.omg.org/technology/documents/bms_spec_catalog.htm#BPMN>.
[5] Eclipse SOA Tools Project (STP). <http://www.eclipse.org/stp>.
[6] Meta-Object Facility (MOF™) specification. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF>.
[7] Eclipse Modeling Framework (EMF) Project. <http://www.eclipse.org/emf>.
[8] Unified Modeling Language™ (UML®) specification. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML>.
[9] Object Constraint Language (OCL) specification. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm#OCL>.
[10] UML Diagram Interchange (DI) specification. <http:/ /www.omg.org/technology/documents/modeling_spec_catalog.htm#UML_DI>.
[11] Diagram Definition RFP. <http://www.omg.org/techprocess/meetings/schedule/Diagram_Definition_RFP.html#RFP_Issued>.
[12] XML Metadata Interchange (XMI) specification. <http://www.omg.org/technology/documents/modeling_ spec_catalog.htm#XMI>.
[13] MOF™ Query / View / Transformation (QVT) specification. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF_QVT>.
[14] Eclipse Model-to-Model Transformation (M2M) Project. <http://www.eclipse.org/proposals/m2m>.
[15] MOFScript component. <http://www.eclipse.org/gmt/mofscript/>.
[16] Java Emitter Templates (JET). <http://www.eclipse.org/modeling/m2t/?project=jet>.
[17] Xpand template engine. <http://www.eclipse.org/modeling/m2t/?project=xpand>.
[18] UML Human-Usable Textual Notation (HUTN) specification. <http://www.omg.org/technology/documents/modeling_spec_catalog.htm#HUTN>.
[19] Textual Modeling Framework (TMF) Proposal. <http:/ /www.eclipse.org/proposals/tmf>.
[20] Language Workbenches. <http://martinfowler.com/articles/languageWorkbench.html>.
[21] IDE Meta-tooling Platform (IMP) Proposal. <http://www.eclipse.rog/proposals/imp>.
[22] Business Process Execution Language. <http://docs.oasis-open.org/wsbpel/2.0/wsbpelv2.0.html>.
[23] Business Modeling Definition Metamodel (BPDM). <http://doc.omg.org/dtc/2007-07-01>.
[24] EclipseCon 2008 <http://www.eclipsecon.org/2008>.
[25] OMG Specification and Products <http://www.omg.org/gettingstarted/specsandprods.htm#SpecProd>.
[26] OMG Technology Adoption Process. <http://www.omg.org/gettingstarted/processintro.htm>.

Other Useful References

Object Management Group (OMG). <http://www.omg.org>.
Eclipse Modeling Project (EMP). <http://www.eclipse.org/modeling>.
Eclipse UML2 Project. <http://www.eclipse.org/uml2>.
EMF Technology OCL Project. <http://www.eclipse.org/emft/projects/ocl#ocl>.
Eclipse Graphical Modeling Framework (GMF) Project. <http://www.eclipse.org/gmf>.
Model to Text Transformation (M2T) Project. <http://www.eclipse.org/modeling/m2t/>.
Architecture-Driven Modernization (ADM). <http://adm.omg.org/>.
Business Process Definition Metamodel. <http://doc.omg.org/dtc/2007-07-01>.
Catalog of OMG Domain Specifications. <http://www.omg.org/technology/documents/domain_spec_catalog.htm>.

Model-Driven Web Engineering

Nora Koch, Santiago Meliá-Beigbeder, Nathalie Moreno-Vergara, Vicente Pelechano-Ferragud, Fernando Sánchez-Figueroa, and Juan-Manuel Vara-Mesa

The continuing emergence of new platforms and technologies make Web applications resemble more and more complex desktop applications. Therefore, the tasks of designing, implementing and adapting the Web applications are becoming a more time-consuming and error-prone issue. In the Web Engineering context, Model-Driven Development (MDD) and Model-Driven Architecture (MDA) principles are being used to successfully address the construction, evolution and adaptation of Web applications. In this article we show how the Model-Driven Web Engineering (MDWE) discipline has arisen and how MDD/MDA principles are applied in the Web Domain to define models and metamodels, to specify model transformations, to manage interoperability issues and to build tools that support the development process.

Keywords: CASE Tools, Interoperability, Metamodels, Model Driven Architecture (MDA), Model Driven Development (MDD), Model Transformations, Web Engineering.

1 Introduction

Since 2001 the Model-Driven Architecture (MDA) initiative has been applied to many application domains showing that, in general, it works better in those domains dominated by functional requirements, with well structured models, precise separation of concerns and standard platforms. Web Engineering, specifically, has proved to be an application domain where MDA has brought significant benefits. In particular, MDA allows successfully addressing interoperability, model evolution and adaptation problems of Web systems, as emerging new platforms and changes in technologies constantly happen in this area.

In this sense, we have seen how many Web Engineering approaches are shifting to becoming "*MDA compliant*", which has resulted in some major changes in their notations, processes and tools. In particular, some of these approaches have had to: (a) redesign their Web modelling languages using meta-modelling techniques, rather focusing on notational aspects of the languages being used; (b) reorganize their original set of models in a modular and platform independent manner; (c) reformulate their development processes in terms of model transformations and model merges; and (d) incorporate and adopt standards that support the realization of the MDA initiative such as UML® (Unified Modelling Language), MOF (Meta-Object Facility), XMI (XML Metadata Interchange), or QVT (Query/View/Transformations)[1] .

However, despite all these challenges, the benefits of the adoption of Model-Driven Development (MDD) and MDA ideas and techniques, to Web Engineering has far outweighed its costs. MDA has provided the opportunity to inject good software engineering practices into the Web applications domain; and has allowed successful bridging

[1] OMG. <http://www.omg.org>

Authors

Nora Koch has been a research assistant at the Ludwig-Maximilians-Universität (LMU) of Munich (Germany) since 1995 and has worked at F.A.S.T. Applied Software Technology GmbH as consultant and project manager since 1998. She received her Ph.D. in Computer Science from the LMU in 2000. Nora is leader of the Web Engineering Group at the LMU, responsible for the development of the UWE methodology. Her main research interests focus on methods for the development of Web applications, personalised Web systems and model-driven engineering. Nora organised the 4th International Conference on Web Engineering (ICWE 2004) in Munich, two editions of the Model-Driven Web Engineering Workshop (MDWE 2005 and MDWE 2007) and is a founder of the network MDWEnet. She is a member of the IEEE and the German Society of Informatics. Nora has published more than 70 publications. <kochn@pst.ifi.lmu.de>.

Santiago Meliá-Beigbeder is Assistant Professor at the Department of Languages and Information Systems at the University of Alicante (Spain), where he is the director of a postgraduate Master in Software Engineering. Also, he is a member of the IWAD research group in the same University. His research interest includes Model-Driven Development, Web Engineering Methodologies, Automatic Code Generation Techniques and Web Software Architecture, all of which were part of his Ph.D. received at the University of Alicante in 2007. He has published in prestigious journals such as the European Journal of Information Systems, Journal of Web Engineering, Revue d'Interaction Homme-Machine and Lecture Notes in Computer Science, and at conferences (OOPSLA, EC-MDA, ER, EC-Web, ICWE, CADUI, etc.). He regularly serves on the Program Committees of several international conferences and workshops. <santi@dlsi.ua.es>.

Nathalie Moreno-Vergara is Assistant Professor in the Department of Computer Science of the University of Málaga (Spain) where she received her MSc. and is completing a Ph.D. in Computer Science. Her research interest is mainly oriented towards model-driven development and, in particular, she is interested in investigating conceptual modeling methodologies, business process modeling, model transformation languages and code-generation techniques in the Web applications context. Her

of the previously existing gap between the high level design models and concepts and the low-level Web implementation code (Preciado et al [1]). This has led to a discipline within the Web Engineering called Model-Driven Web-Engineering (MDWE) that focuses, among others, on the interoperability of the currently existing methodologies for the development of Web applications. Worthy of mention is the MDWEnet[2] initiative started by a group of European researchers working on MDWE, with the objective of improving the interoperability of MDWE approaches and tools in order to widen their scope and provide better tools and methods to the industry.

The adoption of MDD/MDA by the Web Engineering community is not free from problems and limitations. In this article, we shall give a critical overview of the state of the art in MDA-based Web Engineering as currently perceived by some of the groups that actively work on it. Not only the efforts and results already achieved will be described, but also the challenges, threats and weaknesses of this approach will be identified based on our experiences and developed systems, with the aim of helping MDA to evolve towards a more mature and successful development approach for Web systems.

The paper is structured as follows. Section 2 describes the role achieved by models and meta-models in the abstraction and design process of current Web applications as a consequence of the MDA goal of automatically generating implementations from models. In this context, Section 3 illustrates the way in which model transformations are used within the development process. A brief report on tools that support MDD/MDA principles in the Web domain is given in Section 4. Section 5 shows how the Web Engineering community has addressed the difficult problems of interoperability using the MDA concepts and mechanisms. Finally, Section 6 points out current strengths, weaknesses and major challenges that could considerably improve the efficiency of using MDA in the Web context.

2 Models and Metamodels in the Web Domain

MDA is based on the construction and transformation of models which represent a computational independent viewpoint (CIM), a platform independent viewpoint (PIM) or a platform specific viewpoint (PSM). During recent years, the Web engineering community has proposed several methods for modelling Web applications that mainly focus on the construction of PIMs. Web Engineering methods, Rossi et al. [2] like Hera, OOHDM, OOWS, UWE, WebML, WSDM, and methods such as MIDAS [3], OO-H and WebSA (for both see Meliá et al [4]) propose building similar types of models but using different graphical notation for the representation of these models. Most of them use a proprietary notation; some combine the use of the Unified Modeling Language (UML) with their own notation, and only UWE and WebSA use UML 2.0 for all its models. UWE uses plain UML as far as possible and defines for the Web domain features a UML profile following the extension mechanisms provided by UML. As a result, UWE is a UML

most significant publications can be found in relevant journals (the European Journal of Information Systems, the Journal of the American Society for Information Science and Technology or the ET Software), book chapters in Springer Verlag and well-known international conferences such as ICWE, DEXA, MODELs, ICSE, etc. She usually participates as part of program committees and reviewer of conferences and workshops in her research area. <vergara@lcc.uma.es>.

Vicente Pelechado-Ferragud is Associate Professor in the Department of Information Systems and Computation (DISC) at the Valencia University of Technology (Spain). His research interests are web engineering, conceptual modelling, requirements engineering, software patterns, web services, ambient intelligence, business process modeling and model driven development. He received his Ph.D. from the Valencia University of Technology in 2001. He is currently teaching software engineering, design and implementation of web services, design patterns, and model driven development and code generation in the Valencia University of Technology. He is a member of the OO-Method Research Group at the DISC. He has published in several well-known scientific journals (Information Systems, Data & Knowledge Engineering, Information and Software Technology, Internacional Journal of Web Engineering and Technology, etc.), book chapters in Springer, IDEA, M. Sharpe and international conferences (ER, CAiSE, WWW, ICWE, EC-WEB, WISE, AH, ICSOC, DEXA, etc.). He has been a member of Scientific and Organizing Committees of well-known International Conferences and Workshops (CAiSE, ER, ICWE, ICEIS, ACM MT and IADIS). <pele@dsic.upv.es>.

Fernando Sánchez-Figueroa is a professor in the Department of Computer Science at University of Extremadura (Spain), where he teaches Concurrent Programming and Design Patterns. Sánchez has a MSc. in computer science from the University of Sevilla (Spain) and a Ph.D. in computer science from the University of Extremadura. His research focuses on Web Engineering, specifically on Rich Internet Applications, Web Acessibility and Model Driven Web Engineering. He has been Vice-rector of ICT at the University of Extremadura for four years. Currently he holds a “Software Engineering” chair sponsored by INSA-IBM. < fernando@unex.es>.

Juan-Manuel Vara-Mesa obtained his BSc. and MSc. in Computer Science Engineering at the *Universidad Rey Juan Carlos* (Madrid, Spain), where he did the Doctoral Courses on the Computer Science and Mathematical Modeling Program. Currently he works as assistant professor in the department of Informatics Languages and Systems II of the *Universidad Rey Juan Carlos* and he is a member of the Kybele Research Group where he is doing his Ph.D. thesis focused on Model-Driven Engineering for the development of Web Information Systems. He is co-author of several books and book chapters, as well as several publications at national and international events and journals, and he has participated on several regional, national and European research projects. <juanmanuel.vara@urjc.es>.

[2] MDWEnet. <http://www.pst.ifi.lmu.de/projekte/mdwenet/>.

compliant Web domain specific language. There is a clear tendency to the use of UML by other methods as well, mainly due to the advantages provided in the use of CASE tools and meta-modelling for model-driven approaches.

A main characteristic shared by all these Web engineering approaches is the *separation of concerns*. This characteristic allows building different models to address a variety of concerns relevant in the Web domain like content of the Web application, hypertext structure, presentation aspects, adaptivity (in particular personalisation and context-awareness) and Web architectural issues. Models are merged based on integration mechanisms either implemented in proprietary tools or by the application of general rules defined in model transformation languages. In addition, MDWE approaches follow the MDA principles creating CIMs such as the requirements models built by WebRE [5] and OOWS; building PIMs for navigation, presentation and business process specification (almost all methods); building PIMs for architectural models (WEI and WebSA); and obtaining PSMs or transforming into code for specific platforms such as Java EE, Struts, Spring and .NET. The general objective is that in the development process the creation of models that take into account technological aspects is postponed as long as possible. The main advantage is to be able to react efficiently and with low costs to technology changes.

The entities used in all these Web specific models, and the relationships between entities can be also represented as a model, a so-called metamodel, which is needed for the specification of the model transformations. The first metamodel in the Web domain was presented for the UWE approach at the 3rd International Conference on Web Engineering (ICWE 2003) [6]; other methods defined then their metamodel; recently Moreno and Vallecillo [7] proposed a common reference metamodel called Web Engineering Interoperability (WEI), which is described as a MOF metamodel. WEI also defines *lightweight* extensions of UML, i.e., UML profiles, for representing the specific syntax for each of its metamodels.

3 Model Transformations for Generating Web Applications

As the MDA framework suggests, the transformations could be applied to establish a traceable development process from abstract models (CIM or PIM) to the platform dependent models (PSM) or directly to the implementation. Thus, many Web engineering approaches have defined transformations to obtain some parts or the entire implementation of a Web application. As it is well known, a domain-specific strategy such as the Web domain, allows significant parts of the implementation to be generated automatically and to reduce the development effort [8]. Several Web methods have taken advantage of this aspect by developing commercial CASE tools as presented in the next section. These CASE tools use code generation techniques to obtain Web applications from a reduced set of conceptual or design models. Within the model transformations scope we can distinguish between model-to-model transformations and model-to-code transformations. Currently, most Web methods are starting to use model transformations to extend or to implement CASE tools in order to take advantage of the opportunities which transformation languages can provide.

Next, we present how the different Web methods have applied model-to-model and model-to-text transformations to produce Web applications.

3.1 Model-to-Model Transformations

There are two types of model-to-model transformations: (1) vertical transformations that convert models from higher into lower level of abstraction and (2) horizontal transformations which describe mappings between models of the same level of abstraction.

Historically, in the Web domain, most cases of vertical transformations have been developed using using tool-specific proprietary languages. More recently though, Web approaches such as UWE [9], OOWS [10], WebSA [4] and MIDAS [3] have formalized all or part of their development process using model-to-model transformations languages such as QVT, ATL or AGG. In some cases, these model-to-model transformations are defined as merge mechanisms to introduce new concepts like architectural styles (WebSA), user requirements (WebRE [5] and OOWS Requirements Analysis [10]) and quality measures [11]. In other cases, they have been established to formalize the mappings from the original process (UWE [12]).

The horizontal transformations have been applied in the Web domain to maintain the consistency of the model specifications, checking that the different models do not impose contradictory requirements on the elements they share [7]. However, due to the novelty of these models, there is a lack of maturity in their current standards and tools. Therefore, some of the properties of these transformations (reusability, easy of use and a reduced maintainability) have not been tackled yet. Furthermore, there is no Web Engineering commercial CASE tool available that is completely based on model-to-model transformations.

3.2 Model-to-Code Transformations

The Web engineering community has extensive experience in model-to-text transformations or code generation. Approaches such as WebML, OOH, OO-Method/OOWS have been generating Web applications for almost ten years now. In some cases, code generation is realized using general-purpose languages (C++, XSLT, Java or Python) which do not cover some desirable requirements for the model-to-text transformations. Recently, an OMG standard Model-to-Text Request-For-Proposal has established the proper characteristics of the model-to-text languages (e.g. Round-Trip engineering) and some proposals such as Xpand and MOFScript have been adapted to it with mixed success. Recently, OOWS and WebSA have used MOFScript and Xpand in the implementation of their code generators in order to benefit from its properties.

4 CASE Tools supporting MDWE

In this section we give a brief overview of the existing CASE tools supporting MDWE proposals and we make a short review of the main problems and challenges to be tackled in this field. First of all, it should be noticed that we make a distinction between UML based and non-UML based tools.

On the one hand, we consider those CASE tools developed to work with domain specific languages that extend the UML standard (UML profiles). The most relevant CASE tool that falls in this category is ArgoUWE [13]. Initially, ArgoUWE was developed to extend the open-source tool ArgoUML by adding features for modelling content, navigation, and presentation structures. Later on, features were added to model the business logic and the behaviour of workflow-driven Web applications and to detect inconsistencies in UWE models based on the UWE metamodel and its OCL well-formed rules [13]. Currently the main problem of this approach is that ArgoUML and therefore ArgoUWE does not support UML 2.0. WebTE [14] is a UML tool that supports XMI and allows introduction of the OO-H and WebSA models and transformations into a transformation engine which executes them and produces a Java EE Web application. Currently there exists other solutions based on the use of UML, like the AriadneTool [15], the DaVinci framework [16], MIDAS-CASE [17] or VisualWade [18], but they either offer just some limited functionalities or have been closed.

On the other hand, we consider those tools developed to work with languages for Web applications modelling and deployment defined by means of MOF-based metamodels. The tools in this group may be considered not fully "*MDA conformant*", in the sense that they are not UML-based tools. Currently the most representative of these is WebRatio [19]. WebRatio is a commercial tool developed for giving support to the WebML method that is focused on the development of data-intensive applications. With WebRatio, the business objects are modelled using the UML or E/R standards while the front-end is modelled using WebML. Then, the entire application for the Java EE architectures and SQL/XML data sources is automatically generated from those models. There are also more recent non-UML CASE tools, like M2DAT [3] or the OOWS Suite [20], but they are still under development.

When talking about CASE tool support it should be noted that the proliferation of technologies and tools for developing "*your own*" MDD tools is facilitating the adoption and implementation of MDA principles and techniques. Many software companies and research groups are considering the development of their own CASE tool for supporting their own MDWE method (following the MDA, Software Factories, Product Lines, Generative Programming of whatever other more specific model driven proposal). This way, technology is playing a key role in the distinction between UML based and non-UML based tools: the facilities provided in the context of the *Eclipse Modelling Project* (EMP) and other DSL frameworks, like the *Generic Modelling Environment* (GME) or the *DSL Tools*, have shifted the focus from UML-based approaches to MOF-based ones.

Special attention has to be paid to the EMP. The quantity and quality of the MDD facilities provided in the context of this project (a common modelling framework like EMF, meta-editors like GMF, transformation engines like ATL or VIATRA, code generators like MOFScript) has given rise to a new generation of Eclipse tools. As a consequence, more and more MDWE proposals are developing their tools as Eclipse plug-ins, like the OOWS suite and M2DAT, or at least, upgrading or re-defining them to be "Eclipse compliant", like WebRatio [21].

However, there is still a lot of work to do. A very common problem, clearly stated by Moreno and Vallecillo [2], is that the mapping rules are typically hard-coded in the CASE tool (e.g. this is the case of ArgoUWE and WebRatio). This fact results in a gap between the design of the Web application and the final implementation. According to MDA principles, these rules should be defined at a more abstract level, using the QVT standard. Although some proposals have already tackled this task (see [13] for UWE, [4] for WebSA and [7] for WEI), these improvements have still to be integrated in the corresponding CASE tools. The lack of a reference implementation for QVT (which has led to non-complete QVT parsers for a subset of this language [7]) complicates this integration. Another problem is interoperability, in this sense, the use of *weaving* models to automate model migration is becoming widely accepted. Vara et al [22] shows how to apply this approach in a real industrial environment. Such an approach is being studied as a way to automate tools interoperability. Finally, the reader should notice that even though MDD is a widely accepted approach, MDWE is still relatively new: all the tools listed in this section are academic proposals. So, we can conclude that the most outstanding challenge for the developers of MDWE CASE tools is to take their tools from academic to industrial environments.

5 Interoperability Issues in MDWE

As stated in previous sections, current model MDWE approaches provide a set of methods and supporting tools for a systematic design and development of Web applications. However, these proposals have some limitations, especially for exchanging models or representing further modelling concerns, such as architectural styles, technology independence, or distribution. A possible solution to these issues is providing interoperability among MDWE proposals, enabling them to complement each other. Interoperability is at the heart of MDA at different levels: models and metamodels, transformations and tools.

Regarding models and metamodels, there exist three possibilities for achieving this interoperability: a) obtaining a unified method based on the strengths of the different methods; b) obtaining bridges for interoperability between the individual models and tools and c) obtaining a common metamodel. All these possibilities have their own benefits and disadvantages. Currently there are two projects in progress regarding options b) and c). In [23] Wimmer et al., presented a methodology based on MDA for making interoperable three Web modelling approaches (WebML, OO-H and UWE). They used the Ecore implementation of the MOF standard for the definition of the three metamodels. ATL is used as model transformation language to implement the transformation rules and an ATL engine executes the transformation. The next step for them is defining a common metamodel for Web Modeling. Another promising approach is WEI [7], a model-based framework for building Web applications that, among other goals, tries to provide a common framework (and metamodel) in which current proposals could be integrated and formulated in terms of the MDA principles. WEI can be instantiated both to build Web applications from scratch, and to build Web applications based on existing models (including those defined using other methodologies, e.g., UWE, WebML or OO-H).

Regarding transformations, we find a problem due to each approach using its own script language that is incompatible with other languages and tools that users often use. In this sense, QVT is not being used as thought by the OMG as stated in the previous section.

Regarding tools interoperability, despite the efforts of the OMG, the XMI standard has proven to be unsuccessful, especially when working with UML profiles. Until that happens it seems more convenient to take advantage of model-to-model transformations to achieve CASE tools interoperability, whether or not they are UML based tools.

6 Future Challenges

MDWE methods are evolving to be properly adapted to the continuous evolution of Web system requirements and technology. In the last few years a new type of Web applications, known as Web 2.0 has emerged. These applications let people collaborate and share information online. Murugesan [2] says the *People-Centric Web* and the *Read/ Write Web*, offers smart user interfaces and built-in facilities for users to generate and edit content presented on the Web and thereby enrich the content base.

Service Oriented applications and the Web 2.0 are providing a clear infrastructure to integrate multiple software services under a rich user interface. AJAX-based (Asynchronous Javascript and XML) Rich Client Applications or RIAs, Service Mashups, REST or XML Web Services allow integrating current Web applications with third party services, portals, and with legacy systems. RIAs are changing the browser from a static request-response interface to a dynamic, asynchronous interface. RIAs promise a richer user experience and a set of benefits that affect Web Engineering methods [24].

The wide adoption of mobile devices allows users to access the Web using handheld devices (pocket PCs, PDAs, smartphones, etc). Mobile Web applications offer some additional features compared to traditional desktop Web apps such as location-aware services, context-aware capabilities and personalization. We are not forgetting that Web applications development is a complex task that also needs to take into account many different aspects and non functional requirements such as scalability, reliability, availability, evolution and maintainability, usability, security, accessibility, mobility, localization, personalization, adaptivity, etc.

Web Usability is one of the most relevant quality factors that should be taken into account by current and future model driven Web engineering methods. Usability should be integrated in each step of the software development method in a mandatory way, because generated Web apps should meet the diverse expectations and needs of many types of users, including able and disabled users, with different skills. The continuing appearance of new technologies like RIAs and the Mobile Devices does not guarantee that current design guidelines and usability tests will work for those new approaches to assure a better user experience.

However, MDA is still a relatively young approach and there are some issues around the MDA-based software development not yet sufficiently clarified. In particular, MDA is originally intended for new developments and it is not clear how MDA handles the production of releases, patches or updates, an important part of the ongoing maintenance of Web applications. Regarding non-functional requirements, MDA deals well with functional properties but it is less capable of dealing with non-functional requirements. In this sense, there is a debate about whether the specification of non-functional requirements is within the scope of MDA. In summary, while the principles underlying MDA are sound, there remain issues that must be solved for MDA to realize its full potential, particularly in the Web applications domain.

References

[1] H.W. Gellersen, M. Gaedke. Object-Oriented Web Application Development. IEEE Internet Computing, 1999. Vol 3(1), pp. 60-68.

[2] G. Rossi, O. Pastor, D. Schwabe, L. Olsina (eds.). Web Engineering: Modelling and Implementing Web Applications. Springer Human-Computer Interaction Series, 2007. ISBN: 978-1-84628-922-4.

[3] P. Cáceres, V. De Castro, J.M. Vara, E. Marcos. Model Transformations for Hypertext Modelling on Web Information Systems. Proc. of the ACM/SAC 2006 Track on Model Transformations (MT2006), Dijon, France, 2006, pp. 1256–1261.

[4] S. Meliá, J. Gómez. The WebSA Approach: Applying Model-Driven Engineering To Web Applications. Journal of Web Engineering (JWE), 5(2), 2006, pp. 121–149.

[5] M.J. Escalona, N. Koch. Metamodelling the Requirements of Web Systems. Web Information Systems and Technologies: Int. Conferences WEBIST 2005 and WEBIST 2006. Revised Selected Papers, Springer LNBIP, Vol. 1, 2007, pp. 267-280.

[6] N. Koch, A. Kraus. Towards a Common Metamodel for the Development of Web Applications. Proc. 3rd Int. Conf. Web Engineering (ICWE 2003), LNCS 2722,

497-506. Springer Verlag, 2003.
[7] N. Moreno, A. Vallecillo. Towards Interoperable Web Engineering Methods. Accepted for publication at the Journal of the American Society for Information Science and Technology (JASIST), 2008.
[8] J. Bettin. Measuring the Potential of Domain-Specific Modeling Techniques. Proc. of the Second Domain-Specific Modeling Languages Workshop, OOPSLA, Working Papers W-334. Helsinki School of Economics, 2002 , pp. 39-44.
[9] N. Koch. Transformation Techniques in the Model-Driven Development Process of UWE. Proc. of the 6th Int. Conf. on Web Engineering (ICWE 2006), ACM Vol. 155, Palo Alto, California, 2006.
[10] P. Valderas, V. Pelechano, O. Pastor. A Transformational Approach to Produce Web Application Prototypes from a Web Requirements Model. International Journal of Web Engineering and Technology (IJWET), Vol. 3, No. 1, 2007, pp. 4-42.
[11] C. Cachero, S. Melia, M. Genero, G. Poels, C. Calero. Towards Improving the Navigability of Web Applications: A Model-Driven Approach. European Journal of Information Systems, 2007, Vol. 16, pp. 420-447.
[12] A. Kraus. Model Driven Software Engineering for Web Applications. PhD Thesis, Institut für Informatik, Ludwig-Maximilians-Universität München, 2007.
[13] A. Knapp, N. Koch, G. Zhang, H.-M. Hassler. Modeling Business Processes in Web Applications with ArgoUWE. Proc of Int. Conf. Unified Modeling Language (UML 2004), Springer LNCS 3273, 2004, pp. 69-83.
[14] S. Meliá, J. Gómez, J.L. Serrano. WebTE: MDA Transformation Engine for Web Applications. Proc. 7th Int. Conf. Web Engineering (ICWE 2007), Springer LCNS 4607, Como, Italy, 2007.
[15] P. Diaz, S. Montero, I. Aedo. Modelling Hypermedia and Web Applications: the Ariadne Development Method. Information Systems, Vol.30(8), 2005, pp. 649-673.
[16] A. Langegger, J. Palkoska, R. Wagner. DaVinci - A Model-driven Web Engineering Framework. J. Web. Infor. Syst Vol.2(2), 2006 pp. 119-132.
[17] J.M. Vara, V. De Castro, E. Marcos. WSDL Automatic Generation from UML Models in a MDA Framework. International Journal of Web Services Practices, Vol.1, No.1-2, 2005, pp. 1-12.
[18] J. Gómez, A. Bia, A. Parraga. Tool Support for Model-Driven Development of Web Applications. Web Information Systems Engineering (WISE 2005), pp. 721-730.
[19] Webratio. <http://www.webratio.com>.
[20] J. Fons, V. Pelechano, O. Pastor, P. Valderas, V. Torres. Applying the OOWS Model-Driven Approach for Developing Web Applications. The Internet Movie Database Case Study.
[21] R. Acerbis, A. Bongio, M. Brambilla, S. Butti. WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. Web Engineering, 2007, pp. 501-505.
[22] J.M. Vara, M. Didonet Del Fabro, F. Joualt, J. Bezivin. Model Weaving Support for Migrating Software Artifacts from AUTOSAR 2.0 to AUTOSAR 2.1.Int. Conf. on Embedded Real Time Software (ERTS 2008), Toulose (France), 2008.
[23] M. Wimmer, A. Schauerhuber, W. Schwinger, H. Kargl. On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges. Workshop on Model-driven Web Engineering (MDWE 2007), held in conjunction with ICWE, Como, Italy, 2007.
[24] J.C. Preciado, M. Linaje, F. Sánchez. Designing Rich Internet Applications with engineering methodologies. Proc. of the 9th IEEE Int. Symposium on Web Site Evolution. IEEE Computer Society, 2007, pp. 23-30.

UPENET
UPGRADE European NETwork
The network of CEPIS member societies' publications

High Performance Computing

The TOP500 Project: Looking Back over 15 Years of Supercomputing

Hans Werner Meuer

The TOP500 project was launched in 1993 to provide a reliable basis for tracking and detecting trends in high performance computing. Twice a year, a list of the sites operating the world's 500 most powerful computer systems is compiled and released. The best performance on the Linpack benchmark is used as the measurement for ranking the computer systems. The list contains a variety of information including the systems' specifications and major application areas. Information on all 30 TOP500 lists issued to date is available at: www.top500.org

Keywords: Benchmark, Cluster Computing, HPC, Supercomputing, Performance.

Author

Prof. Dr. Hans Werner Meuer is the Managing Director of Prometeus GmbH, and the General Chairman of ISC'08 in Dresden. He is Professor Emeritus of Computer Science at the University of Mannheim, Department of Mathematics and Computer Science. Hans Meuer received his doctorate in mathematics in 1972 from RWTH Aachen University, Germany. He served as specialist, project leader, group and department chief during his 11 years at the Research Center in Jülich, Germany, from 1962–1973. For the following 26 years, he was Director of the Computer Center at the University of Mannheim, Germany, and Professor of Computer Science for 28 years. <hans.meuer@supercomp.de>

1 Mannheim Supercomputer Statistics 1986–1992 and TOP500 Project Start in 1993

From 1986 through 1992, the Mannheim supercomputer statistics were presented to participants of the Supercomputer Seminars at Mannheim University, and we noticed an increased interest in this kind of data from year to year [1]. The statistics simply counted the vector computer systems installed in the U.S., Japan and Europe, since in the mid-80s a supercomputer was synonymous with a vector computer. Counting the vector computers installed worldwide primarily depended on the input provided by the manufacturers of the systems, which made the statistics less reliable. Whereas we knew well which vector systems existed in the U.S. and Europe, information regarding systems in Japan was much more difficult to collect. We therefore contacted the three Japanese vector computer manufacturers – Fujitsu, NEC and Hitachi – for information on all systems installed in Japan and used their data as the basis for our yearly estimations.

In 1992, we released the last Mannheim statistics, counting 530 supercomputers installed worldwide. Figure 1 shows the result of our 7-year activity regarding the share of the different manufacturers in the supercomputer market. Cray clearly led with a constant share of about 60%; the second U.S. manufacturer, CDC (Control Data Corporation), had been doing rather well with just under 10% – until the end of the 80s when their share started to drop, and they were completely out of the supercomputer business in 1991. The Japanese vector computer manufacturers Fujitsu, NEC and Hitachi entered into our statistics in 1986 with a combined share of 20%, and were able to expand their share to about 40% in 1992, with Fujitsu clearly in the lead at 30% of all vector computers installed worldwide.

Figure 2 illustrates the shares of vector computer installations by coun-

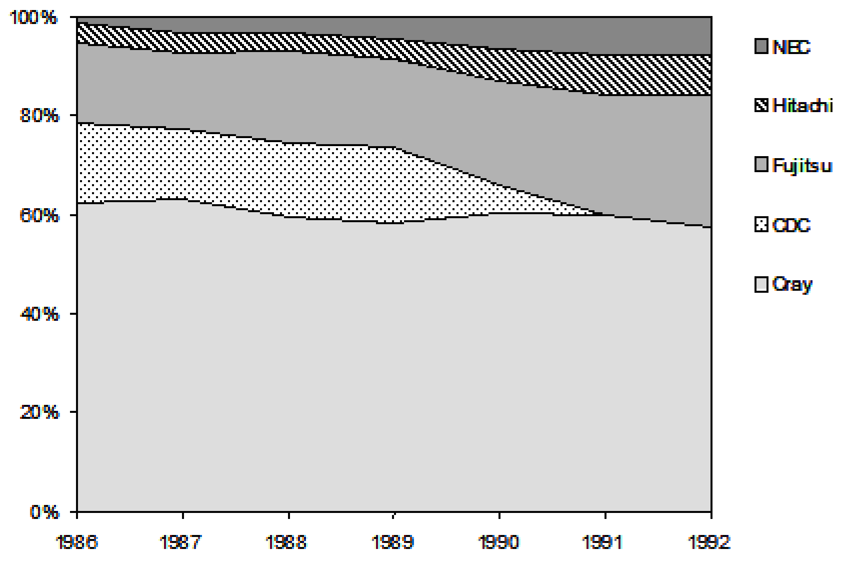


Figure 1: Manufacturers' Shares.

tions that we would like to answer here:

■ Why is it "the 500 most powerful computers"? One reason is that the last time we counted the supercomputers worldwide in 1992, we ended up with 530. And another reason surely is the (emotional) influence of the Forbes 500 lists, e.g. of the 500 richest men or the 500 biggest corporations in the world.

■ "Most powerful" is defined by a common benchmark, for which we had chosen Linpack. But why Linpack? Linpack data, above all Rmax, are well known and easily available for ALL systems in question. Strictly speaking, TOP500 lists computers only by their ability to solve a set of linear equations, A x = b, using a dense random matrix A.

■ An alternative to updating the TOP500 list twice a year would be to continuously update the list. Why don't we do this? First, updating the TOP500 list is a time-consuming and complex process. Second, we thought that a bi-annual publication would be a much better way to show significant changes, which the HPC community is primarily interested in, and this has proven to be true over the years.

TOP500 authors are Hans Werner Meuer, Erich Strohmaier, now Lawrence Berkeley National Laboratory (LBNL), USA, and Jack Dongarra, the "Father of Linpack", University of Ten-

try. The U.S. clearly led in 1986 with a share of 50%, but which dropped to 35% in 1992, however. In Japan, the situation developed in the opposite direction, with a share of about 20% in 1986 and already 40% in 1992, surpassing the U.S. share. Europe had a constant share of between 25% and 30% over the seven years, with Germany leading slightly ahead of France and the U.K.

Though useful, the Mannheim supercomputer statistics were not perfect, as they lacked a reliable database. Additionally, the so-called entry level vector computer systems such as Fujitsu's VP30/50 became more and more popular in Japan. But were these systems really supercomputers in terms of performance? And how should mini-supercomputers such as the Convex C1/2 from the U.S. be rated? We had to carefully consider which systems qualified as supercomputers and therefore should be listed in the Mannheim statistics. From the early 90s on, vector computers were no longer the only supercomputer architecture; massively parallel systems such as the CM2 of Thinking Machines (TMC) had entered the market. What we therefore needed was a method to define what constituted a "supercomputer" and could be updated on a yearly basis.

This is why Hans Werner Meuer and Erich Strohmaier started the TOP500 project at the University of Mannheim/Germany, in spring 1993. Here are its simple guiding principles:

■ Listing of the 500 most powerful computers in the world.

■ Rmax, the best Linpack performance, is used as the benchmark [2].

■ The TOP500 list is updated and published twice a year, in June at ISC in Germany and in November at SC in the U.S.

■ All TOP500 data is publicly available at www.top500.org

There are some immediate ques-

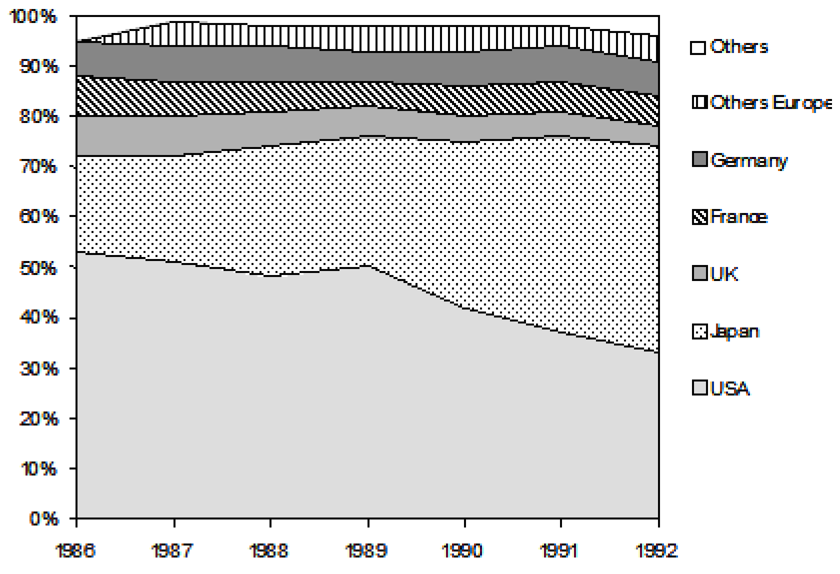


Figure 2: Countries' Shares.

1st TOP500 List, 06/1993

Country	Count	Share
USA	225	45.0%
Japan	111	22.2%
Germany	59	11.8%
France	26	5.2%
U.K.	25	5.0%
Australia	9	1.8%
Italy	6	1.2%
Netherlands	6	1.2%
Switzerland	4	0.8%
Canada	3	0.6%
Denmark	3	0.6%
Korea	3	0.6%
Others	20	4.0%
Total	500	100.0%

30th TOP500 List, 11/2007

Country	Count	Share
USA	283	56.6%
Japan	20	4.0%
Germany	31	6.2%
France	17	3.4%
U.K.	48	9.6%
Australia	1	0.2%
Italy	6	1.2%
Netherlands	6	1.2%
Switzerland	7	1.4%
Canada	5	1.0%
Denmark	1	0.2%
Korea	1	0.2%
China	10	2.0%
India	9	1.8%
Others	55	11.0%
Total	500	100.0%

Figure 3: 1st and 30th TOP500 Lists – Countries.

nessee, USA. The fourth author, Horst Simon, LBNL, had supported the TOP500 project from the very beginning and joined the project officially in 2000. In 1999, only six years after starting the project, the authors published their experiences with TOP500 [3].

30 TOP500 lists have been published up to now:

- First TOP500 list was published on June 24, 1993, at ISC'93 in Mannheim, Germany.
- 29th TOP500 list was published on June 27, 2007, at ISC'07 in Dresden, Germany.
- 30th TOP500 list was published on November 12, 2007, at SC07 in Reno, USA.

The release dates of the next three TOP500 lists are:

- 31st TOP500 list will be published on June 18, 2008, in Dresden, Germany.
- 32nd TOP500 list will be published on November 18, 2008, in Austin, USA.
- 33rd TOP500 list will be published on June 24, 2009, in Hamburg, Germany.

After 15 years and 30 lists, we have managed to establish TOP500 among HPC users, manufacturers and the media as THE instrument for analyzing the HPC market.

2 Competition between Countries, Manufacturers and Sites

One of the most important reasons for TOP500's success is that we foster competition between countries, manufacturers and computing sites.

2.1 Competition between Countries

From our 7th Mannheim supercomputer statistics published at the Mannheim Supercomputer Seminar in 1992, we expected a neck-and-neck race between the U.S. and Japan for our first TOP500 list (see Figure 3). However, the "Japanese danger" was grossly overestimated, as the first TOP500 list showed the U.S. clearly leading with 45% of all TOP500 installations, and Japan was far behind with only 22%.

If we look at the 30th TOP500 list published in November 2007 at SC in Reno/USA, we see that the dominance of the U.S. is even bigger today than 15 years ago: Now they have a share of 56.6 % of all systems installed, and Japan holds a share of only 4%. Even the U.K., with a 9.6% share, and Germany, with a 6.2% share, are ahead of Japan, which is followed closely by France with 3.4%.

The overall development of the various countries' share through the past 30 TOP500 lists is also very interesting (see Figure 4). In 1993, the

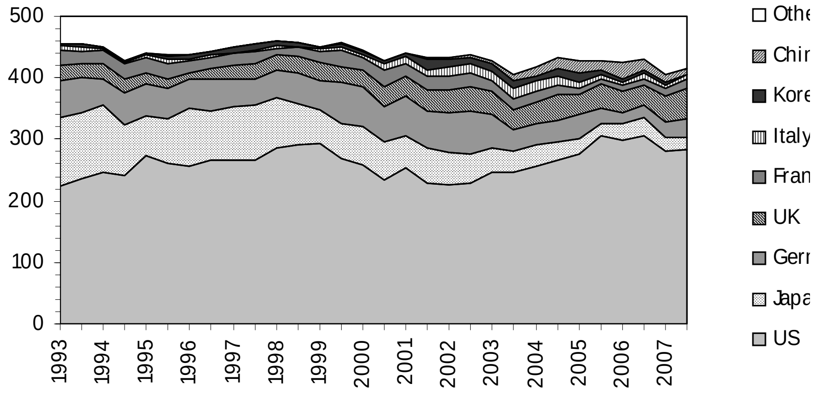


Figure 4: Supercomputer Installations Worldwide

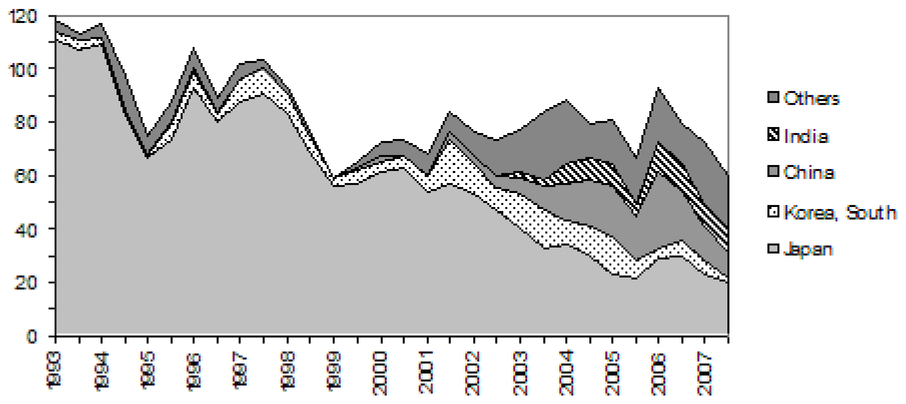


Figure 5: Supercomputer Installations in Asia.

U.S. started with a huge share of 45%, which they have even managed to expand slightly. Japan, however, started with a 22% share but has fallen back significantly. In Europe, Germany, which had always clearly been ahead of the U.K., is now far behind the U.K.

Figure 5 illustrates the development of the supercomputer installations in Asia since 1993. It shows the rapid drop in Japan's share and indicates that China and India will enter the HPC market as new players in the medium term. But we will have to wait until the next TOP500 lists to see how this plays out.

2.2 Competition between Manufacturers

If we focus on the manufacturers (see Figure 6), Cray Research was the clear leader on our first TOP500 list with a 41% share, ahead of Fujitsu with 14 %. Third place was already held by TMC – a non-vector supercomputer manufacturer – with 10.8%, ahead of Intel with 8.8%. At that time, Intel still had its Supercomputer Division, which also produced non-vector supercomputers. Surprisingly, today's leading HPC manufacturers, IBM and Hewlett-Packard, were not represented on the first TOP500 list at all.

In the 30th TOP500 list of November 2007, IBM has the clear lead with a 46.4% share. The second position is held by Hewlett Packard with 33.2%, and the leader of 1993, Cray Research (now Cray Inc.), is now down to 2.8%.

If we look at the development of the manufacturers since 1993 (see Figure 7), we notice that the HPC market has been very dynamic: in only 15 years, the market has seen a complete transformation. Cray has turned from the clear market leader in the general HPC market, including the industrial customer segment, into a niche player for high-end government research

1st TOP500 List, 06/1993

Manufacturer	Count	Share
Cray Research	205	41.0%
Fujitsu	69	13.8%
Thinking Machines	54	10.8%
Intel	44	8.8%
Convex	36	7.2%
NEC	32	6.4%
Kendall Square Res.	21	4.2%
MasPar	18	3.6%
Meiko	9	1.8%
Hitachi	6	1.2%
Parsytec	3	0.6%
nCube	3	0.6%
Total	**500**	**100.0%**

30th TOP500 List, 11/2007

Manufacturer	Count	Share
Cray Inc.	14	2.8%
Fujitsu	3	0.6%
Thinking Machines	–	–
Intel	1	0.2%
Hewlett-Packard	166	33.2%
NEC	2	0.4%
Kendall Square Res.	–	–
MasPar	–	–
Meiko	–	–
Hitachi/Fujitsu	1	0.2%
Parsytec	–	–
nCube	–	–
IBM	232	46.4%
SGI	22	4.4%
Dell	24	4.8%
Others	35	7.0%
Total	**500**	**100.0%**

Figure 6: 1st and 30th TOP500 Lists – Manufacturers.

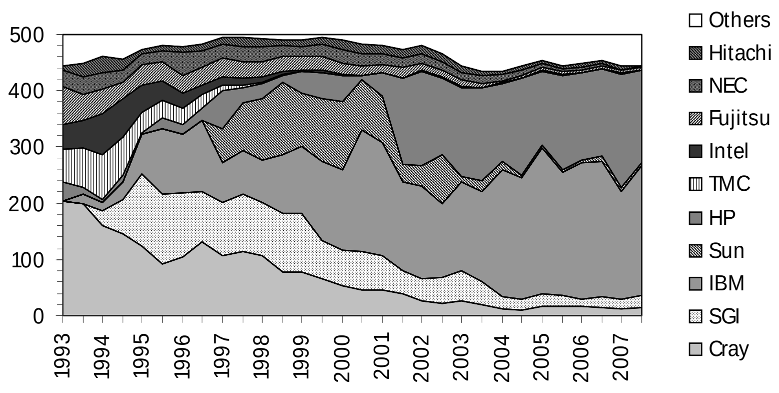


Figure 7: Manufacturers / Systems.

laboratories and academic customers. IBM on the other hand, which was of virtually no importance in the HPC market in the early 90s, has become the dominant market leader in all market segments, including industrial and commercial customers. Hewlett-Packard – once a small HPC manufacturer represented in the first TOP500 lists only by Convex, which they later took over – has established itself as number two, right after IBM. Sun Microsystems, which used to be number two among the HPC manufacturers a couple of years ago, has fallen back dramatically in the TOP500. But Sun is now trying to catch up with the other HPC manufacturers. And also a re-invigorated Cray might be back in the general HPC arena again: They currently have three hybrid supercomputers in the TOP10, which shows that they have successfully left behind their approach of pure vector computing. With only very few systems in the overall TOP500 list, however, Cray will have to work hard to replicate its earlier success.

2.3 Competition between Sites

Figure 8 lists the 20 most powerful sites through 30 TOP500 lists. The percentage in the right-hand column is a site's relative contribution to the Rmax total of the average list of 30. In this list, the U.S. leads with two-thirds of the sites (14) ahead of Japan, with four centers (20%). The fact that the U.S. has the four most powerful sites in the world also shows its dominance as a consumer and producer of HPC systems. Europe is represented by Germany (Forschungszentrum Jülich, FZJ) at position 18 and by the U.K. (ECMWF) at position 15. (Note that ECMWF is a European and not purely a U.K. site.)

3 My Favorite Supercomputer in All TOP500 Lists so far

We have published 30 TOP500 lists with a total of 15,000 systems, and our data base has even twice as many entries. So it might sound strange that I have just one favorite system. I would like to emphasize, however, that there

Rank	Site	Country	Over time
1	Lawrence Livermore National Laboratory	USA	5.39%
2	Sandia National Laboratories	USA	3.70%
3	Los Alamos National Laboratory	USA	3.41%
4	Government	USA	3.34%
5	The Earth Simulator Center	Japan	1.99%
6	National Aerospace Laboratory of Japan	Japan	1.70%
7	Oak Ridge National Laboratory	USA	1.39%
8	NCSA	USA	1.31%
9	NASA/Ames Research Center/NAS	USA	1.25%
10	University of Tokyo	Japan	1.21%
11	NERSC/LBNL	USA	1.19%
12	Pittsburgh Supercomputing Center	USA	1.15%
13	Semiconductor Company (C)	USA	1.11%
14	Naval Oceanographic Office (NAVOCEANO)	USA	1.08%
15	ECMWF	U.K.	1.02%
16	ERDC MSRC	USA	0.91%
17	IBM Thomas J. Watson Research Center	USA	0.86%
18	Forschungszentrum Jülich (FZJ)	Germany	0.84%
19	Japan Atomic Energy Research Institute	Japan	0.83%
20	Minnesota Supercomputer Center	USA	0.74%

Figure 8: TOP20 through 30 TOP500 Lists.

Rank	Manufacturer	Computer	Rmax [GF/s]	Site	Country	Year	#Proc
1	Intel	ASCI Red	1,068	Sandia National Laboratories	USA	1996	7264
2	Hitachi	CP-PACS/2048	368.2	Center for Computational Science	Japan	1996	2048
3	Fujitsu	Numerical Wind Tunnel	229	National Aerospace Laboratory	Japan	1996	167
4	Hitachi	SR2201/1024	220.4	University of Tokyo	Japan	1996	1024
5	Cray	T3E	176	Forschungszentrum Jülich	Germany	1996	512
6	Cray	T3E	176	Government	USA	1996	512
7	Cray	T3E	176	Max-Planck-Gesellschaft MPI/IPP	Germany	1996	512
8	Cray	T3E	176	NASA/Goddard Space Flight Center	USA	1996	512
9	Cray	T3E	176	Pittsburgh Supercomputer Center	USA	1996	512
10	Cray	T3E	176	University of Stuttgart	Germany	1996	512

Figure 9: Top 10 Sites of the 9th TOP500 List, June 1997.

are many systems that impressed me over the past 15 years. Before I reveal my favorite supercomputer, I would like to highlight another one here first: It was number 259 on our 9th TOP500 list published at the Mannheim Supercomputer Seminar (ISC'97) in 1997. This system, named "Deep Blue", was installed at the IBM Watson Research Center in Yorktown Heights and had a best Linpack performance of 11.37 Gigaflop/s; it was an IBM SP2 P2SC with 32 processors and a clock rate of 120 MHz. But the floating point performance was not what really mattered: Each of the 32 processors was equipped with 15 special-purpose VLSI chess chips. Deep Blue was the first chess computer to beat a reigning world chess champion, Garry Kasparov [4]. Ten years after this event, no chess player stands a chance against any kind of computer, not even against a simple home computer. One year ago, in November/December 2006, Deep Fritz played a six-game match against reigning world chess champion Wladimir Kramnik in Bonn. Deep Fritz won 4–2 [5].

3.1 My All-time Favorite: Intel's ASCI Red

During the 1996 acceptance tests at Sandia National Laboratories in Albuquerque/USA, Intel's ASCI Red showed an Rmax performance of 1,068 Gigaflop/s. Thus it was the first Teraflop/s computer to enter the HPC arena, and it immediately grabbed the first place on our 9th TOP500 list of June 1997 (see Figure 9).

"ASCI" stands for "Accelerated

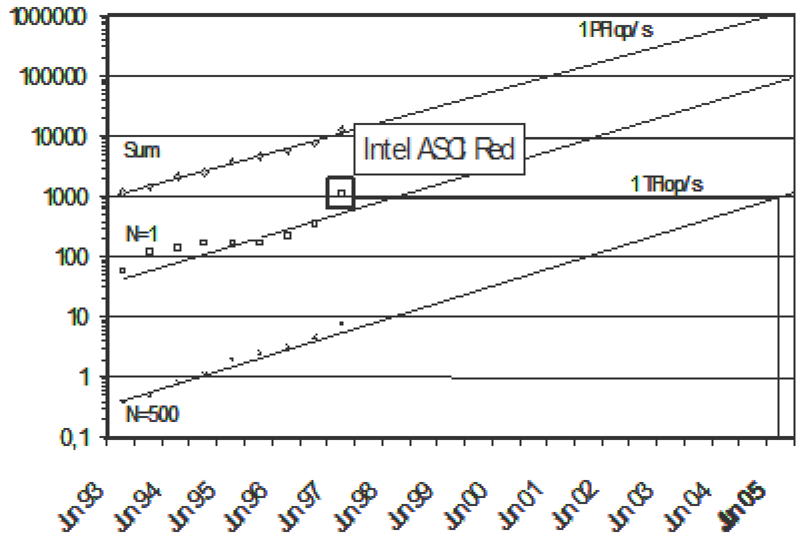


Figure 10: Performance [Gigaflop/s].

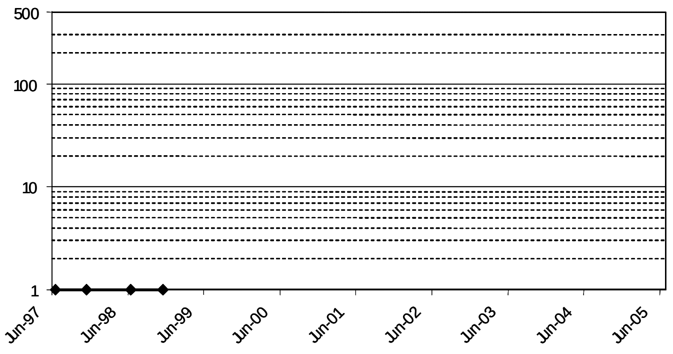


Figure 11a: Number of Teraflop/s Systems in the TOP500 Lists between 1997 and 2005: 1997–1998.

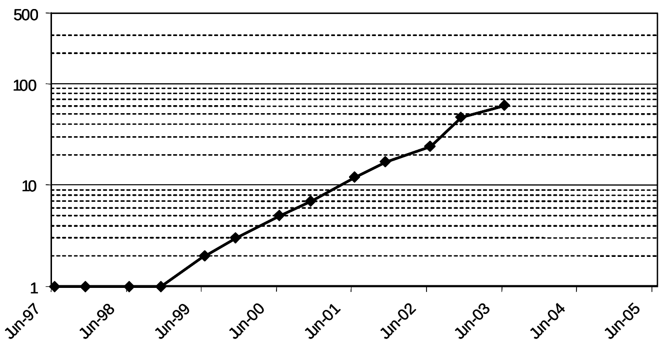


Figure 11b: Number of Teraflop/s Systems in the TOP500 Lists between 1997 and 2005: 1997–2003.

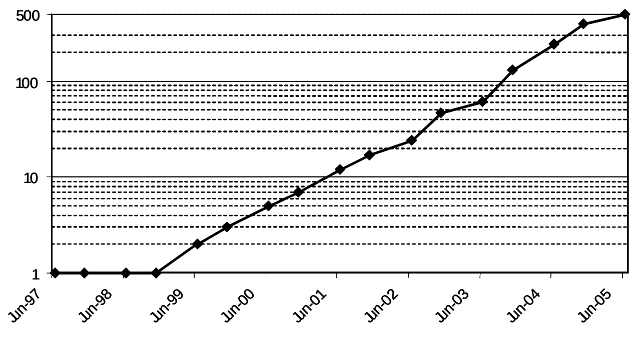


Figure 11c: Number of Teraflop/s Systems in the TOP500 Lists between 1997 and 2005: 1997–2005.

Strategic Computing Initiative". This initiative, under the auspices of the Department of Energy (DOE), was the U.S. response to France's nuclear weapons tests at the Mururoa atoll, where France conducted 41 atmospheric and 147 underground nuclear tests between 1966 and 1996. DOE's "Accelerated Strategic Computing Initiative" applies advanced capabilities in scientific and engineering computing to one of the most complex challenges in the nuclear era: maintaining the performance, safety, and reliability of the U.S. nuclear weapons arsenal without physical testing. ASCI was established in 1996 in response to the government's commitment to pursue a comprehensive ban on nuclear weapons testing.

ASCI Red was the last supercomputer designed and assembled solely by Intel; Intel's Supercomputer Division had already been closed down when ASCI Red was launched [6].

3.2 Eight-year Forecast

At the opening session of the 12th Supercomputer Conference in Mannheim in June 1997, we released our 9th TOP500 list with the new number one system: ASCI Red (see Figure 10). Considering the measured data from the nine TOP500 lists from June 1993-June 1997 and assuming that the increases in performance would continue as before, we extrapolated the performance of future systems using linear regression on the logarithmic scale. We fit exponential growth to these three levels of performance, the sum of all 500 systems, the number one system and the entry level system.

Based on the extrapolation of these fits, I announced at ISC'97 that, eight years from then, i.e. in 2005, there would be only Teraflop/s systems on the TOP500 list, even though the 1997 list had only one: Intel's ASCI Red.

Probably many of the ISC'97 attendees thought I was crazy to make a forecast eight years out for such a dynamic market. And I do have to admit that I did not feel too comfortable myself – going out on a limb like that in front of many of the world's top HPC experts. You certainly can imagine that I was extremely anxious to see the next TOP500 lists. (see Figures 11a, 11b and 11c.)

The TOP500 lists of November 1997, June 1998 and November 1998 had still only one Teraflop/s system: ASCI Red. I started to get really nervous...

... but fortunately, the TOP500 list of June 1999 showed a new Teraflop/s system: SGI's ASCI Blue Mountain at Los Alamos National Laboratory. It was the second system worldwide to break the Linpack Teraflop/s barrier, and with 1.6 Teraflop/s, it took second place on the June 1999 TOP500 list. ASCI Red was slightly over 2 Teraflop/s then, as Intel had managed to increase the number of processors to more than 9,000.

And in June 2005 – as I had predicted eight years earlier – there were only Teraflop/s systems in the TOP500

Rank	Manufacturer	Computer	Rmax [TF/s]	Site	Country	Year	#Cores
1	IBM	BlueGene/L eServer Blue Gene	478.2	DOE/NNSA/LLNL	USA	2007	212,992
2	IBM	JUGENE BlueGene/P Solution	167.3	Forschungszentrum Jülich	Germany	2207	62,536
3	SGI	SGI Altix ICE 8200	126.9	New Mexico Computing Center	USA	2007	14,336
4	HP	Cluster Platform 3000 BL460c	117.9	Computational Research Laboratories, TATA SONS	India	2007	14,240
5	HP	Cluster Platform 3000 BL460c	102.8	Swedish Government Agency	Sweden	2007	13,728
6	Sandia/Cray	Red Storm Cray XT3	102.2	DOE/NNSA/Sandia	USA	2006	26,569
7	Cray	Jaguar Cray XT3	101.7	DOE/ORNL	USA	2007	23,016
8	IBM	BGW eServer Blue Gene	91.3	IBM Thomas Watson	USA	2005	40,960
9	Cray	Franklin Cray XT4	85.4	NERSC/LBNL	USA	2007	19,320
10	IBM	New York Blue eServer Blue Gene	82.2	Stony Brook/BNL	USA	2007	36,864

Figure 12: Top 10 Sites of the 30th TOP500 List, November 2007.

list. The entire Teraflop/s market had gained momentum since 1999. Whereas the June 2001 TOP500 list showed only a few more than 10 Teraflop/s systems, the 2003 list already had more than 100 such systems.

3.3 Why Intel's ASCI Red is my Favorite System in the TOP500

Here are the reasons why ASCI Red is my favorite system:

1. We saw the CDC 7600 (a predecessor model of the later vector computers) break the Megaflop/s barrier with 1.24 Megaflop/s in 1971, and we also saw the legendary Cray 2 exceed the Gigaflop/s barrier in 1986, boasting 2 GB of memory and a best Linpack performance Rmax of 1.7 Gigaflop/s. But what impressed me the most was when Intel's ASCI Red broke the Teraflop/s barrier in 1997. Of course, I hope to see the breaking of the Petaflop/s barrier this year (2008). I will thus have personally witnessed an increase in supercomputer performance of 10 orders of magnitude over the years.

2. Intel's ASCI Red marked the beginning of a new supercomputer era. In the mid-90s when vector computers started to become less important, DOE's ASCI initiative, which focused on defense applications, opened up a completely new source of funds. ASCI Red was the first product of this initiative and laid the foundation for the U.S. dominance in the production and implementation of supercomputers. ASCI Red was also a remarkable supercomputer from a technical point of view:

It was a mesh-based (38 X 32 X 2) MIMD massively parallel machine initially consisting of 7,264 compute nodes, 1,212 gigabytes of total distributed memory and 12.5 terabytes of disk storage. The original incarnation of this machine used Intel Pentium Pro processors, each clocked at 200 MHz. These were later upgraded to Pentium II OverDrive processors. The system was upgraded to a total of 9,632 Pentium II OverDrive processors, each clocked at 333 MHz. It consisted of 104 cabinets, taking up about 2,500 square feet (230 m^2). The system was designed to use commodity mass-market components and to be very scalable.

3. In June 1997, I predicted that, eight years later, there would be only Teraflop/s systems on the TOP500 list, even though just a single Teraflop/s computer existed at that time, and this was ASCI Red. Never before – and probably never again – has one of my forecasts proven so accurate. ASCI Red was retired from service in September 2005, after having been on 17 TOP500 lists over eight years. It was the fastest computer on the TOP500 list from June 1997 to June 2000 and was replaced as the number one by IBM's ASCI White at Lawrence Livermore National Laboratory on the November 2000 list.

4. And finally, ASCI Red was the reason I got invited to give an interview on German TV (ZDF, Zweites Deutsches Fernsehen). Shortly before Christmas 1996, Nina Ruge, a German TV journalist, interviewed me on her late-night TV show "Heute Nacht" about the first computer to exceed the Teraflop/s barrier: Intel's ASCI Red.

4 Highlights of the 30th TOP500 List and Bell's Law

4.1 Highlights of the 30th List [7]

Among the 10 top sites on the 30th TOP500 list are five new systems and one substantially upgraded system (marked gray in Figure 12). The main changes have taken place among the first five places. Place number one again goes to BlueGene/L, a joint development of IBM and the Department of Energy's (DOE) National Nuclear Security Administration (NNSA), which is installed at DOE's Lawrence Livermore National Laboratory in Livermore, CA/USA. BlueGene/L had been in first place since November 2004; however, the current system has been significantly upgraded so that it now achieves a Linpack benchmark performance of 478.2 Teraflop/s (trillions of calculations per second) compared to a performance of 280.6 Teraflop/s six months ago, before its upgrade.

Place number two is held by a brand-new first installation of a newer version of the same type of IBM system. It is a BlueGene/P system installed at the Forschungszentrum Jülich (FZJ), Germany, with a performance of 167.3 Teraflop/s.

The number three system is not only a new one, but also the first system of a new supercomputing center: the New Mexico Computing Applications Center (NMCAC) in Rio Rancho/ USA. The system, built by SGI and based on the Altix ICE 8200 model, reaches a speed of 126.9 Teraflop/s.

For the first time ever, India has been able to place a system in the Top10, at number four. The Computational Research Laboratories, a wholly owned subsidiary of Tata Sons Ltd. in Pune/India, installed a Hewlett-Packard Cluster Platform 3000 BL460c system. They integrated this system with their own innovative routing technology and achieved a performance level of 117.9 Teraflop/s.

The number five system is also a new Hewlett-Packard Cluster Platform 3000 BL460c system, installed at a Swedish government agency. It was measured at 102.8 Teraflop/s.

The last new system in the Top10 is a Cray XT4 system installed at the National Energy Research Scientific Computing Center (NERSC) at DOE's Lawrence Berkeley National Laboratory in Berkeley, CA/USA. With a Linpack performance of 85.4 Teraflop/s, it ranks ninth.

Processor Architecture / Systems

Figure 13 illustrates that vector computers are on the retreat in the TOP500: On the 30th TOP500 list of November 2007, there are only four vector computer systems, two from Cray Inc. and two from NEC, among them the worldwide number one of 2002–2004, the Earth Simulator in Yokohama/ Japan, which has fallen to number 30.

Operating Systems / Systems

Up to a couple of years ago, UNIX in all its variations was the prevalent operating system on supercomputers, but now Linux has taken over this role. Despite Microsoft's effort to break into this market, Windows plays no role (see Figure 14).

Processor Generations / Systems

354 out of 500 systems (70.8%) use Intel processors, whereas six months ago, only 289 systems (57.8%) had Intel processors. This is the largest share for Intel chips in the TOP500 ever. Especially successful are the Dual-Core Woodcrest and the Quadcore Clovertown processors with a share of 43% and 20.4% respectively. The AMD Opteron family, which left the IBM Power processors behind a year ago, still remains the second-most-common processor family, even though the number of systems using this processor went down from 105 (21%) to 78 (15.6%). 61 systems (12.2%) run on IBM Power processors, compared to 85 systems (17%) half a year ago (see Figure 15).

Interconnect Family / Systems

Due to its widespread use by industrial customers, Gigabit Ethernet is still the most widely utilized internal system interconnect technology (270 systems). It is followed by InfiniBand technology with 121 systems. Myrinet, which dominated the market a couple of years ago, has fallen even further back (see Figure 16).

Architectures / Systems

The most widely used architecture is the cluster architecture, as 406 out of 500 systems (81.2%) are labeled as clusters. Times are long gone when clusters only appeared in the second half of the TOP500 list. There are even two cluster systems among the TOP10, including the most powerful system in Asia at number four, the Cluster Platform system of HP. Constellations have dropped to 0.6%. MPPs hold an 18.2% share, with eight systems in the TOP10 (see Figure 17).

4.2 Bell's Law (1972)

Bell's Law of Computer Class formation was discovered about 1972 [8]. It states that technology advances in semiconductors, storage, user interfaces and networking take place approximately every decade enabling a new, usually lower-priced computing platform to form. Once formed, each class is maintained as a quite independent industry structure. This explains mainframes, minicomputers, workstations and personal computers, the web, emerging web services, palm and mobile devices and ubiquitous interconnected networks. We can expect home and body area networks to follow this path. Bell's Law states that important classes of computer architectures come in cycles of about ten years. It takes about a decade for each of the phases:

- Early research.
- Early adoption and maturation.
- Prime usage.
- Phase out past its prime.

When I was preparing my presentation for the IDPT conference in Antalya in June 2007 [9] Erich Strohmaier and I asked ourselves: Can we use Bell's Law to classify computer architectures in the TOP500?

We make an attempt by introducing the following architectures/computer classes:

- Data parallel systems
  Vector (Cray Y-MP and X1, NEC SX etc.)
  SIMD (CM-2 etc.)
- Custom scalar systems
  MPP (Cray T3E and XT3, IBM SP etc.)
  Scalar SMPs and Constellations (Cluster of big SMPs)
- Commodity clusters
  NOW, PC cluster, Blades etc.
- Power-efficient systems
  BG/L or BG/P as first examples of low-power systems. They might be able to form a new class, but we do not know this yet.

When analyzing all TOP500 lists from the very beginning in 1993 up to now, we find the following computer classes over time, as shown in Figures 18 and 19.

The HPC computer classes "Data

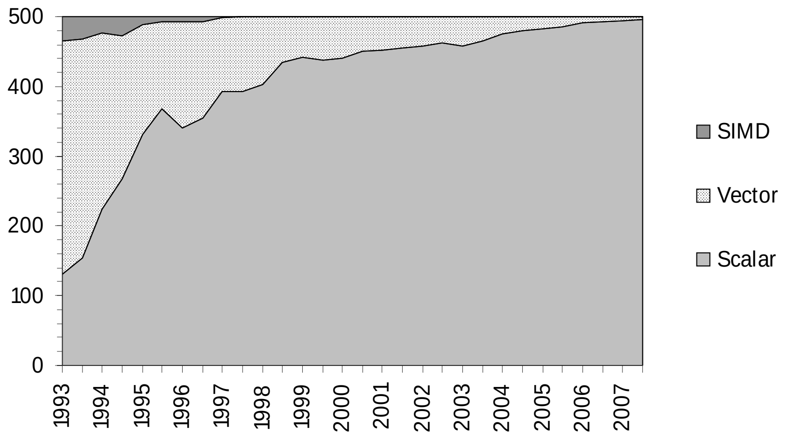


Figure 13: Processor Architecture / Systems.

parallel systems", "Custom scalar systems" and also "Commodity clusters" follow nicely the 10-year cycle of Bell's Law and confirm this law for HPC computer architectures, see Figure 20. For simplicity, we have left out the "Early research" phase of Bell's Law. Of course, we have to wait and see whether or not the 10-year "Past prime usage" phase of the "Commodity clusters" class will really start around 2010. And there is even more speculation about which way the "Power-efficient systems" class, represented by IBM's BG series, will go. A couple of smaller such systems were delisted from the 30th TOP500 list due to poor performance. The next TOP500 lists will show whether or not the "Power-efficient systems" class will really be a class of its own.

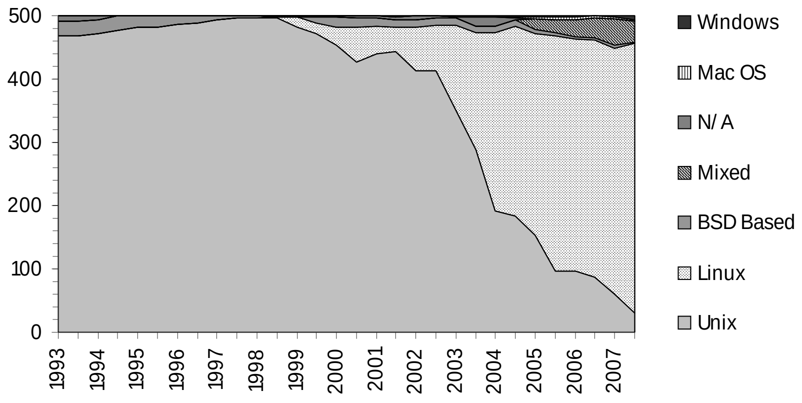


Figure 14: Operating Systems / Systems.

5 Performance Development and Performance Projections

In Figure 21, we have plotted the performance over the last 15 years at position N=500 (entry level performance), at the leading position (number one), as well as the total accumulated performance, the sum of all 500 systems. As can easily be seen, all these curves show an exponential growth. The scale on the left-hand side is a logarithmic scale.

If we compare this growth with Moore's Law, we find that, even though Moore's Law assumes a doubling in performance every 18 months for microprocessors, our growth is larger. We have a doubling for the sum in approximately 14 months, for the number one position in approximately 13 months and even for the number 500 position in a little less than 13 months. There are two main reasons for this larger growth in performance: processor performance and number of processors used.

Also note that the curves at positions one and 500 are quite different: At number one, we typically see a step function. Once a system has made number one, it remains there in the next couple of TOP500 lists. That was true for the "Numerical Wind Tunnel – NWT", Intel's ASCI Red and also for the "Earth Simulator", which ranked first from June 2002 through June 2004. And it also proves true for the

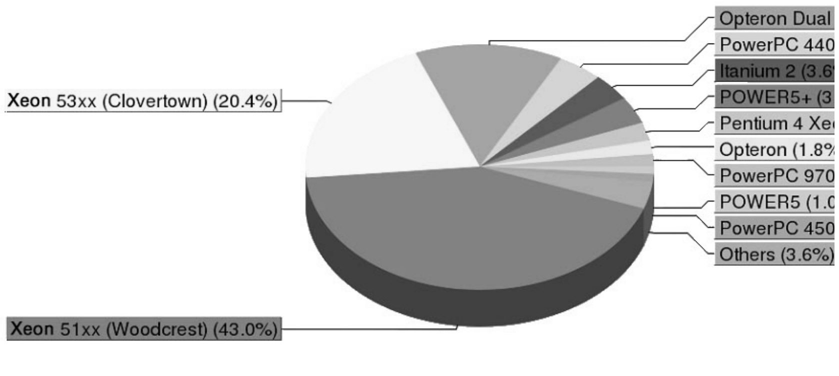


Figure 15: Processor Generations / Systems.

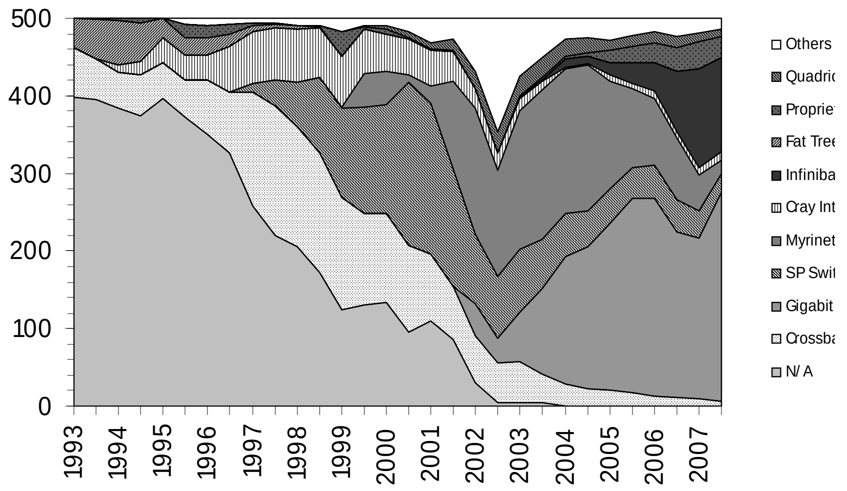


Figure 16: Interconnect Family / Systems.

means that it will have a Teraflop/s performance in the year 2014, i.e. in less than 18 years after the first Teraflop/s system, ASCI Red, entered the HPC arena.

Generally, it will take six to eight years for any system to move from position one to 500 and eight to ten years to move from position 500 to notebook level.

The Linpack Petaflop/s threshold will be reached in 2008. One of the hot candidates for the first Petaflop/s system to enter the TOP500 list is IBM's RoadRunner at Los Alamos National Laboratory, USA. In 2015, there will be only Petaflop/s systems in the TOP500 list. Our projection also shows that the first Exaflop/s computer will

current number one supercomputer (since November 2004), IBM's BlueGene/L at Lawrence Livermore National Laboratory (LLNL), holding this position at different stages of expansion.

If we include a powerful notebook in this figure, we notice that its performance has reached 7 Gigaflop/s now and has thus grown by a factor of 10 within three years.

Again, as when discussing Intel's ASCI Red, we have done a projection into the future, based on 30 lists of real data, by a least square fit on the logarithmic scale (see Figure 22). For a powerful notebook, for example, this

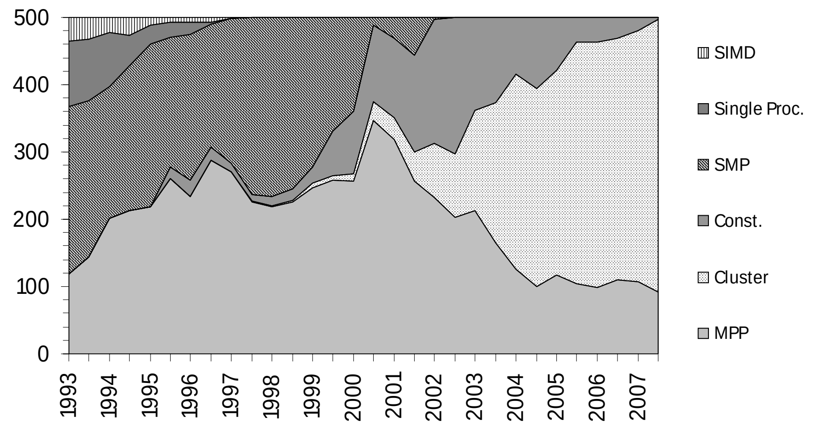


Figure 17: Architectures / Systems.

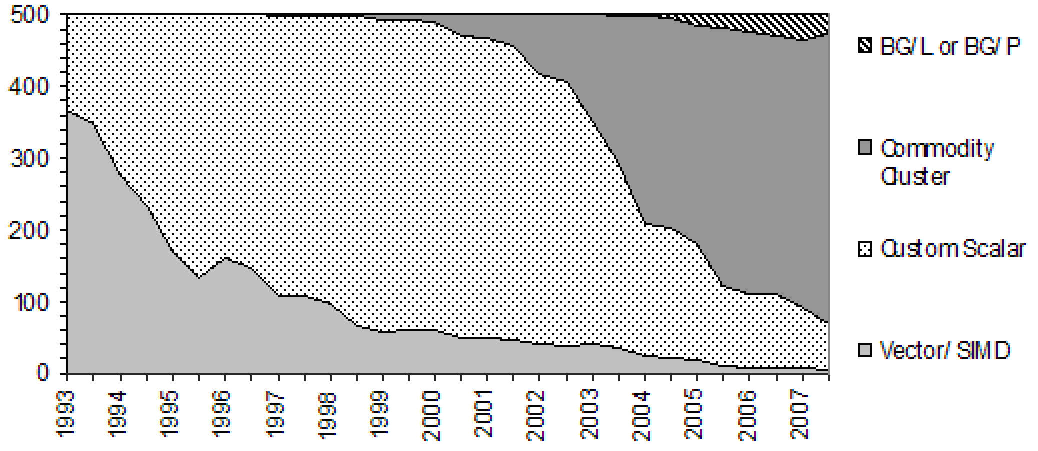


Figure 18: Computer Classes in HPC Based on the TOP500: Computer Classes / Systems.

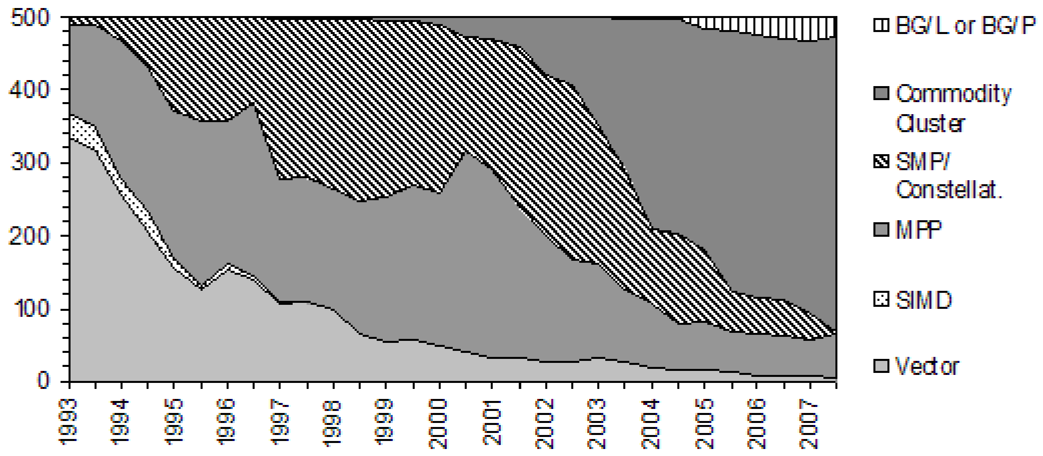


Figure 19: Computer Classes in HPC Based on the TOP500. Computer Classes – Refined / Systems.

enter the TOP500 list in 2019, and only one year later, in 2020, there will be the first notebooks with a performance of 100 Teraflop/s.

The rule seems to be that system performance increases by a factor 1,000 every eleven years: Cray 2 broke the Gigaflop/s barrier in 1986 (in pre-TOP500 times); Intel's ASCI Red exceeded the Teraflop/s barrier in 1997; the first Petaflop/s system will enter the TOP500 list this year (2008); and, according to our projection, the Exaflop/s threshold will be reached in 2019.

6 Top500 in the Future

6.1 The TOP500 Website

The results of ALL 30 TOP500 lists and a variety of additional information are available on our TOP500 website: www.top500.org. This site draws remarkable traffic – more than 20K page impressions per day and is kept up to date by Anas Nashif, our Technical Manager, who is also responsible for the TOP500 data base. The website has been improved and relaunched recently and offers many interesting features: access to sublists, list charts, list statistics and up-to-date information on the HPC market in general. The TOP500 project is financed – but not influenced – by advertising on the website.

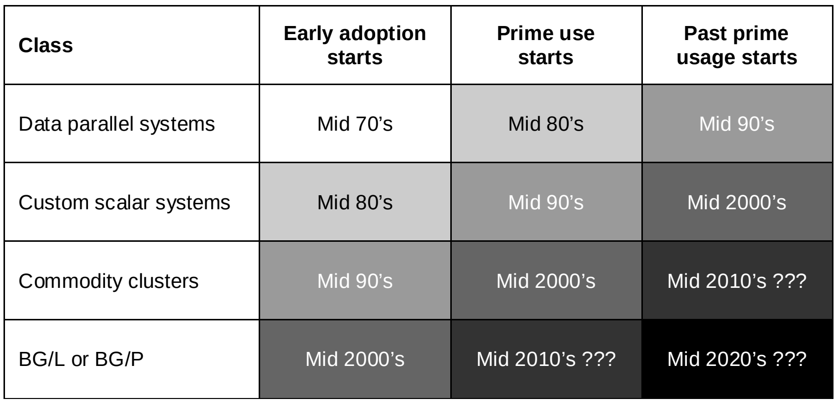
6.2 Summary after 15 Years of Experience

The TOP500 corrected the deficits of the Mannheim supercomputer statistics, which we had used for seven years at the ISC conferences of 1986-92, and has proven to be a reliable tool ever since. Its simple but successful approach based on the Linpack benchmark, though often criticized, is able to get trends right, as far as processors, architectures, manufacturers, countries and sites are concerned. And as shown in Chapter 3, "My Favorite Supercomputer in all TOP500 Lists so far", its performance predictions turn out remarkably correct, even over such a long period as eight years.

The TOP500 lists should only be seen as a source of information for general trends. As such, they are extremely useful and much more reliable than the predictions of market research companies such as IDC, Diebold, etc. However, we have always advised people not to use the TOP500 lists to answer specific questions such as: Is system X at position 254 better suited for a certain application than system Y at position 344? For these cases, you would have to run your own benchmarks and applications sets on the systems in question.

With the TOP500 lists, it is also not possible to estimate the size of the HPC market (e.g., in US $), as we just do not know how much the systems at the different positions cost. We have often been asked for the systems' prices, and we have to admit that this information would be of great interest to the HPC community. But at the very beginning of the TOP500 project we decided not to include this kind of more or less unreliable and vague data in our lists.

When analyzing the TOP500 lists, we find that systems of the upper half of the list used to remain there for a



Class	Early adoption starts	Prime use starts	Past prime usage starts
Data parallel systems	Mid 70's	Mid 80's	Mid 90's
Custom scalar systems	Mid 80's	Mid 90's	Mid 2000's
Commodity clusters	Mid 90's	Mid 2000's	Mid 2010's ???
BG/L or BG/P	Mid 2000's	Mid 2010's ???	Mid 2020's ???

Figure 20: HPC Computer Classes and Bell's Law.

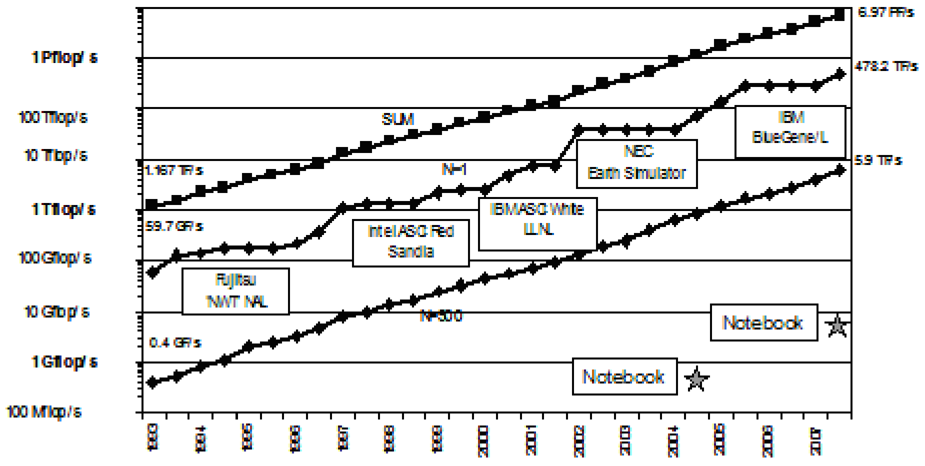


Figure 21: Performance Development.

couple of periods and smooth out seasonal fluctuations. And there are the short time entries, which are often on a list for only six months, since the turnover is very high. Figure 24 illustrates that, on average, approximately 200 systems drop out after six months, not making it into the following list due to poor performance.

6.3 Motivation for Additional Benchmarks

Figure 25 summarizes the pros and cons of the Linpack benchmark:

6.4 HPC Challenge Benchmark

We clearly need something more than Linpack for the TOP500, e.g., HPC Challenge Benchmark and others. At ISC'06 in Dresden/Germany, Jack Dongarra gave the Friday keynote presentation on "HPC Challenge Benchmarks and the TOP500" [10] (see Figures 26 and 27). The conference attendees voted his excellent talk one of the two best ISC'06 presentations.

The HPC Challenge Benchmark basically consists of seven different benchmarks, each stressing a different part of a computer system. Of course HPL, the High Performance Linpack benchmark, is also part of these seven benchmarks and stands for the CPU. We do not have the advantages of a single figure of merit any longer, and the results of the HPC Challenge Benchmark are much more complex so that so-called Kiviat charts are needed. With these charts, it will be much harder for journalists, for example, to report on new systems entering the HPC arena than when they are evaluated only with Linpack.

Dongarra's conclusion is that we will certainly still see Linpack as the TOP500 benchmark for a while. However, it needs to be expanded to produce lists using another yardstick. The HPC Challenge Benchmark could become a standard in the U.S. when it comes to selecting an HPC system.

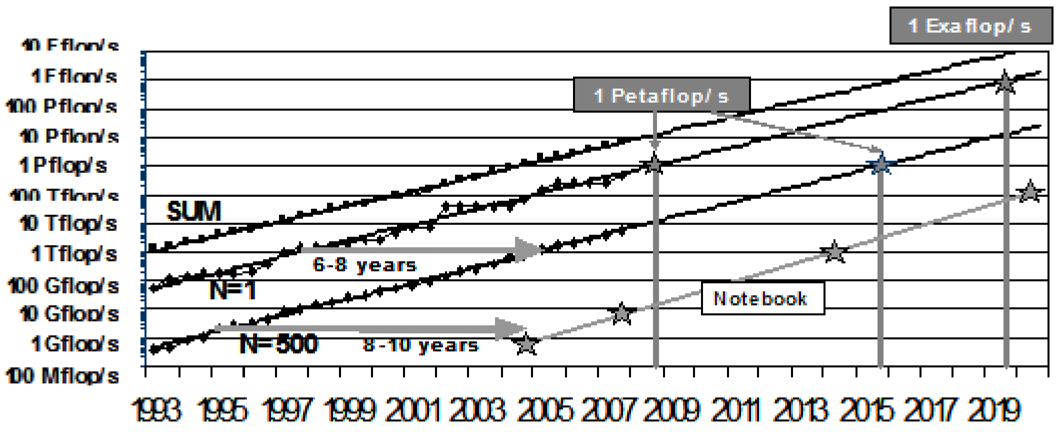
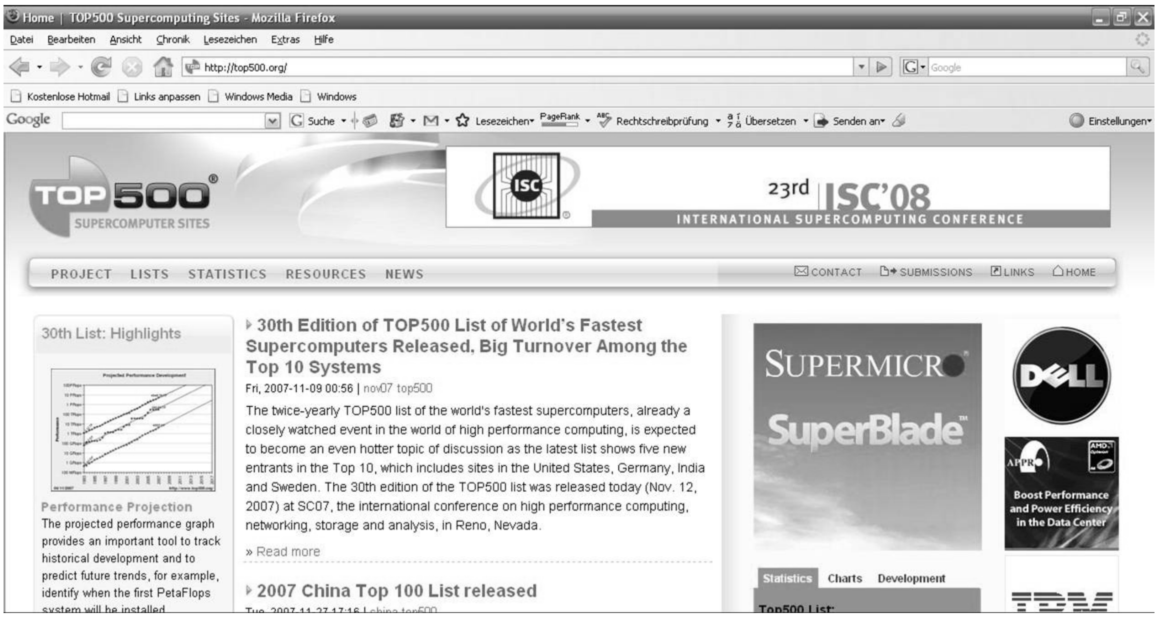


Figure 22: Performance Projection.


Home | TOP500 Supercomputing Sites - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://top500.org/
TOP500 SUPERCOMPUTER SITES
23rd ISC'08 INTERNATIONAL SUPERCOMPUTING CONFERENCE
PROJECT LISTS STATISTICS RESOURCES NEWS
CONTACT SUBMISSIONS LINKS HOME
SUPERMICRO SuperBlade
DELL
Boost Performance and Power Efficiency in the Data Center
Statistics Charts Development

30th List: Highlights

Performance Projection

The projected performance graph provides an important tool to track historical development and to predict future trends, for example, identify when the first PetaFlops system will be installed

30th Edition of TOP500 List of World's Fastest Supercomputers Released, Big Turnover Among the Top 10 Systems

Fri, 2007-11-09 00:56 | nov07 top500

The twice-yearly TOP500 list of the world's fastest supercomputers, already a closely watched event in the world of high performance computing, is expected to become an even hotter topic of discussion as the latest list shows five new entrants in the Top 10, which includes sites in the United States, Germany, India and Sweden. The 30th edition of the TOP500 list was released today (Nov. 12, 2007) at SC07, the international conference on high performance computing, networking, storage and analysis, in Reno, Nevada.

» Read more

2007 China Top 100 List released

Figure 23: TOP500 Website.

6.5 The Green500 List

The Green500 list, overseen by Wu-chun Feng and Kirk W. Cameron of Virginia Tech/USA, is another approach to ranking supercomputers [11]. Its purpose is to list the most energy-efficient supercomputers in the world and serve as a complementary view to the TOP500. However, the latest Green500 list is far from being complete, as it does not even include all TOP500 systems. The Green500's 10 most energy-efficient supercomputers are all IBM systems, see Figure 28. This is probably one of the reasons why IBM strongly supports the project.

The TOP500 authors basically support the idea of a Green500 list, but they reserve the right to launch their own independent and more thorough project, if necessary.

7 Conclusion

The TOP500 project was launched in 1993 to improve and renew the Mannheim supercomputer statistics, which had been in use for seven years. Our simple TOP500 approach does not define "supercomputer" as such, but we use a benchmark to rank systems and to decide on whether or not they qualify for the TOP500 list. The benchmark we decided on was Linpack, which means that systems are ranked only by their ability to solve a set of linear equations, A x = b, using a dense random matrix A. Therefore, any supercomputer – no matter what its architecture is – can make it into the TOP500 list, as long as it is able to solve a set of linear equations using floating point arithmetic. We have been criticized for this choice from the very beginning, but now, after 15 years, we can say that it was exactly this choice that has made TOP500 so successful – Linpack therefore was a good choice. And there was, and still is, no alternative to Linpack. Any other benchmark would have been similarly specific, but would not have been so easily available for all systems – a very important factor, as compiling the TOP500 lists twice a year is a very complex process.

One of Linpack's advantages is also its scalability in the sense that it has allowed us in the past 15 years to benchmark systems that cover a performance range of 10 orders of magnitude. It is true that Linpack delivers performance figures that occupy the upper end of any other application performance. In fact, no other realistic application delivers a better efficiency (Rmax/Rpeak) of a system. But using the peak performance instead of Linpack, which "experts" have often

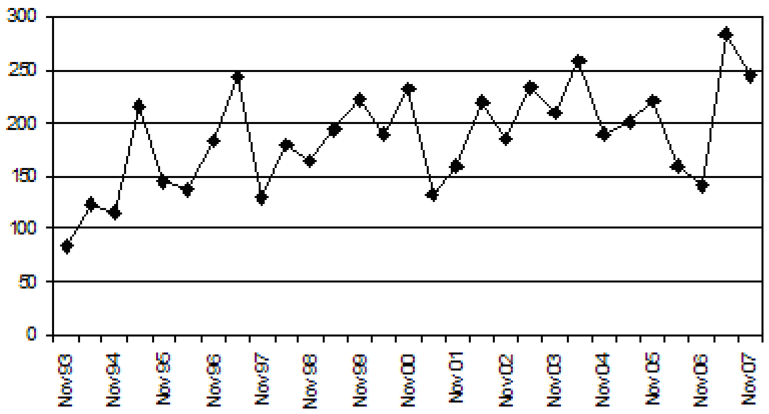


Figure 24: TOP500 Replacement Rate.

Pros	Cons
• Only one figure of merit • Simple to define and rank • Allows problem size to change with machine and over time • Allows competition	• Emphasizes only “peak” CPU speed and number of CPUs • Does not stress local bandwidth • Does not stress the network • No single figure of merit can reflect the overall performance of an HPC system

Figure 25: Linpack Benchmark – Pros and Cons.

recommended to us, does not make any sense. We have seen a lot of new systems that were not able to run the Linpack test because they were not stable enough. For example, it takes more than six hours to run the Linpack test on the current number two system on the TOP500 list, BlueGene/P at Forschungszentrum Jülich (FZJ). Therefore, running Linpack to measure the performance is kind of a first reliability test for new HPC systems.

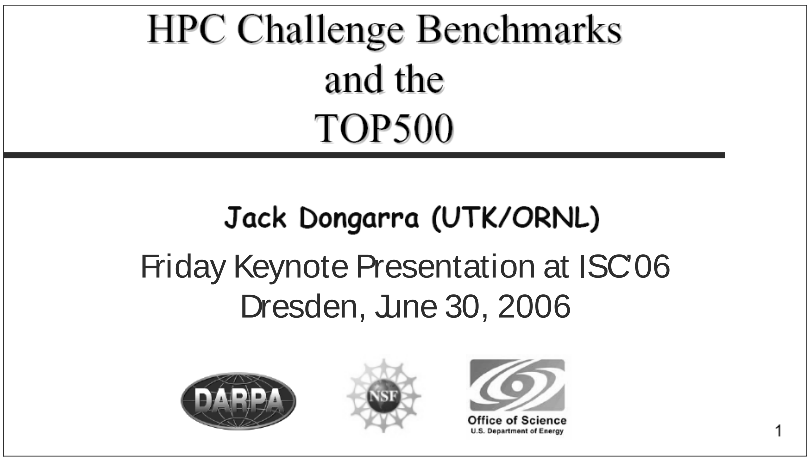
The misinterpretation of the TOP500 results has surely led to a negative attitude towards Linpack. Politicians, for example, tend to see a system’s TOP500 rank as a general rank that is valid for all applications, which of course is not true. The TOP500 rank only reflects a system’s ability to solve a linear set of equations, and it does not tell anything about its performance with respect to other applications. Therefore, the TOP500 list is not a tool for selecting a supercomputer system for an organization. In this case, you would have to run your own benchmark tests that are relevant to your own applications. In this context, an approach such as the "HPC Challenge Benchmark" consisting of seven different benchmarks, which test different parts of a supercomputer, is critical. As experts run the "HPC Challenge Benchmark" tests and interpret their results, it is not a problem not to have only one single figure of merit. For this reason, the HPC Challenge Benchmark has already become a certain criteria in the U.S. when it comes to buying HPC systems.

The TOP500 lists’ success lies in compiling and analyzing data over time. Despite relying solely on Linpack, we have been able to correctly identify and track ALL developments and trends over the past 15 years, covering manufactures and users of HPC systems, architectures, interconnects, processors, operating systems, etc. And above all, TOP500’s strength is that it has proved an exceptionally reliable tool for forecasting developments in performance.

It is very unlikely that another benchmark will replace Linpack as basis for the TOP500 lists in the near future. And in any case we would stick to the concept of a single benchmark because this is the easiest way to trigger competition between manufacturers, countries and sites, which is extremely important for the overall acceptance of the TOP500 lists. Of course, we appreciate it if alternative benchmarks are introduced to complement Linpack. In fact, we are working on this already and encourage other HPC experts to come up with constructive suggestions, too.

Acknowledgement

I would like to thank my colleague, Erich Strohmaier, for his support, his advice and his inspiration, as well as for preparing the data for the figures in this paper and the TOP500 project in general. My thanks go also to my two colleagues, TOP500 authors Jack Dongarra and Horst Simon, for the many interesting discussions on TOP500 topics over the past 15 years, which have been very helpful for this paper, too. I would also like to thank Reiner Hartenstein, University of Kaiserslautern, for inviting me to give a talk on this subject at the World Conference on Integrated Design & Process Technology (IDPT) in Antalya/Turkey on June 7, 2007. He is the person who suggested using "Bell’s Law" to classify computer


HPC Challenge Benchmarks
and the
TOP500

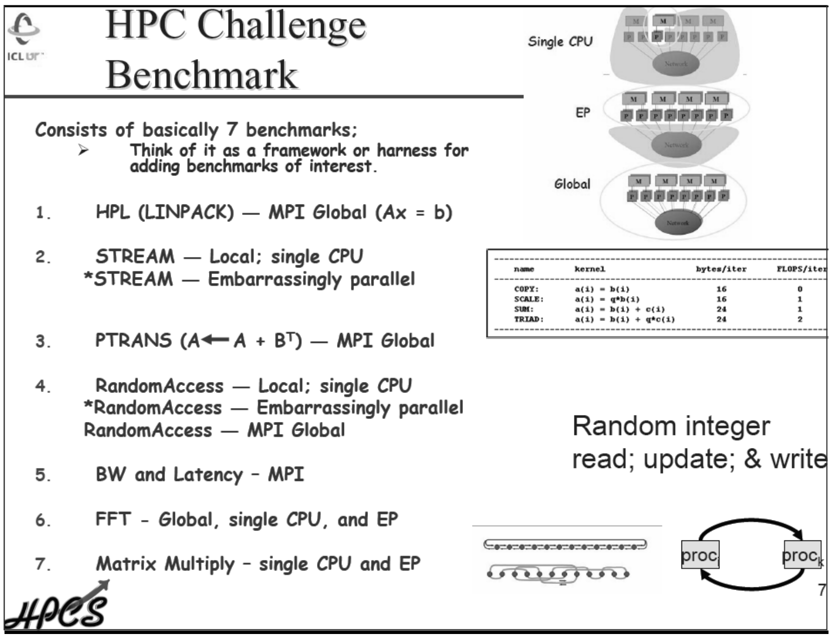
Jack Dongarra (UTK/ORNL)

Friday Keynote Presentation at ISC’06
Dresden, June 30, 2006

DARPA
NSF
Office of Science
U.S. Department of Energy

1

Figure 26: ISC’06 Keynote Presentation.



HPC Challenge Benchmark

Consists of basically 7 benchmarks;
- Think of it as a framework or harness for adding benchmarks of interest.

1. HPL (LINPACK) — MPI Global (Ax = b)
2. STREAM — Local; single CPU
   *STREAM — Embarrassingly parallel
3. PTRANS ($A \leftarrow A + B^T$) — MPI Global
4. RandomAccess — Local; single CPU
   *RandomAccess — Embarrassingly parallel
   RandomAccess — MPI Global
5. BW and Latency – MPI
6. FFT - Global, single CPU, and EP
7. Matrix Multiply - single CPU and EP

```
-----------------------------------------------------------------
 name      kernel                    bytes/iter     FLOPS/iter
-----------------------------------------------------------------
 COPY:     a(i) = b(i)                  16              0
 SCALE:    a(i) = q*b(i)                16              1
 SUM:      a(i) = b(i) + c(i)           24              1
 TRIAD:    a(i) = b(i) + q*c(i)         24              2
-----------------------------------------------------------------
```

Random integer
read; update; & write

Figure 27: HPC Challenge Benchmark.

Green500 Rank	MFlop/s/W	Site	Computer	Total Power (kW)	TOP500 Rank
1	357.23	Science and Technology Facilities Council – Daresbury Laboratory	Blue Gene/P Solution	31.10	121
2	352.25	Max-Planck-Gesellschaft MPI/IPP	Blue Gene/P Solution	62.20	40
3	346.95	IBM – Rochester	Blue Gene/P Solution	124.40	24
4	336.21	Forschungszentrum Jülich (FJZ)	Blue Gene/P Solution	497.60	2
5	310.93	Oak Ridge National Laboratory	Blue Gene/P Solution	70.47	41
6	210.56	Harvard University	eServer Blue Gene Solution	44.80	170
7	210.56	High Energy Accelerator Research Organization / KEK	eServer Blue Gene Solution	44.80	171
8	210.56	IBM – Almaden Research Center	eServer Blue Gene Solution	44.80	172
9	210.56	IBM Research	eServer Blue Gene Solution	44.80	173
10	210.56	IBM Thomas J. Watson Research Center	eServer Blue Gene Solution	44.80	174

Figure 28: Green500's Top10.

architectures in the TOP500 lists. My thanks go to Hans Günther Kruse, University of Mannheim, for checking the manuscript and for his suggestions for improvement. And last but not least, I would like to thank Heike Walther, Prometeus GmbH, for translating most of this paper and Jon Bashor, LBNL, for giving it the final polish.

References

[1] H. W. Meuer, The Mannheim Supercomputer Statistics 1986–1992, TOP500 Report 1993, University of Mannheim, 1994, p. 1–15.

[2] See: http://www.top500.org/project/linpack.

[3] E. Strohmaier, J. J. Dongarra, H. W. Meuer, H. D. Simon, The Marketplace of High-Performance. Computing, Parallel Computing 25, 1999, p. 1517–1544.

[4] See: http://en.wikipedia.org/wiki IBM_Deep_Blue

[5] See: http://www.spiegel.de/netzwelt/tech/0,1518,452 735,00. html

[6] See: http://www.sandia.gov/ASCI/Red/

[7] See: http://www.top500.org/

[8] See: http://research.microsoft.com/~GBell/Pubs.htm

[9] World Conference on Integrated Design & Process Technology (IDPT), Antalya/Turkey, June 3–8, 2007, http://www.fpl.uni-kl.de/IDPT/IDPT2007-final.pdf

[10] See: http://www.netlib.org/utk/people/JackDongarra/SLIDES/isc-talk-2006.pdf

[11] See: http://www.green500.org/

Project Management

Critical Factors in IT Projects

Marco Sampietro

© Mondo Digitale, 2007

This paper was first published, in its original Italian version, under the title "*I fattori critici nei progetti informatici*", by ***Mondo Digitale*** (issue no. 4, December 2007, pp. 3-9, available at <http://www.mondodigitale.net/>). ***Mondo Digitale***, a founding member of **UP**ENET, is the digital journal of the CEPIS Italian society AICA (*Associazione Italiana per l'Informatica ed il Calcolo Automatico,* <http://www.aicanet.it/>.)

Managing IT projects is certainly a complex and very challenging activity; every project has its distinct features and there are no recipes for success that can be repeated in every situation. Nevertheless there are some factors that have shown a high rate of repetitiveness and a considerable impact on project performances. These factors are called "Critical Success Factors". This article analyses, in differing degrees of detail, the factors that most commonly impede the management of IT projects.

Author

Marco Sampietro is Professor of the Information System Unit of the SDA Bocconi School of Management in Milan (Italy). He is the Director of the course on Information System Project Management. He is Professor at the Part Time Executive MBA for Project Management and Process Driven Organization courses. He also teaches Project Management at Bocconi University. He coordinates the Master in Information Systems Management at Bocconi University. <marco.sampietro@sdabocconi.it>

(Keywords added by the Editor of the **UP**ENET section)

Keywords: Critical Factors, IT Projects, Project Management.

1 The Difficulty of Achieving Success in IT Projects

As pointed out by some researches [1][2][4][18][19][20], IT project initiatives often have only a moderate success rate. Some signs of improvement have been seen over the years, but the percentage of projects that are heavily delayed or suffer from an excessive absorption of economic resources or major qualitative shortcomings still remains high (Table 1).

It is interesting to note that problems related to project management are, in a sense, "democratic"; they actually affect every kind of firm, regardless of size or industry. The less than flattering list of the more striking cases of IT project failures includes some companies that are considered excellent: such as Avis Europe which, in 2004, abandoned the installation of its ERP system after having spent 54 million euros; or Ford Motor Co. which, in 2004, interrupted its project after having spent 400 million euros on analysis and investment in the purchasing system; or McDonald's which cancelled an innovative information acquisition project in 2002 after having spent 170 million euros. There are many other more or less well known and more or less expensive examples. The purpose of this paper is not to list the problems and misfortunes of others but rather to point out which factors normally contribute to the failure of IT projects.

For each project it is possible to draw up a relatively long list of factors that have negatively influenced its performance. From an individual company's point of view, the discussion of these factors is a very useful exercise as it allows areas of improvement to be highlighted, maintaining a direct link with the specific situation. However, in the absence of past data carved out in a specific situation, we need to adopt an alternative approach (the one which forms the basis for this paper) in order to obtain useful information that can be used to improve project performance. This approach, known as Critical Success Factors, is based on the study of those variables that are seen to exert a strong influence on project results in different situations, irrespective of the specific information technology to be implemented. Critical Success Factors in IT projects are well studied in literature, and various researches [1][2][5][7][8][9][11][12][13] [14][15][17] come to very similar conclusions, thereby underlining that, at least from the point of view of the most problematic areas and those that need to be improved, the situation is quite clear and shared. However this does not mean that managing these critical areas is an easy task.

2 Critical Success Factors in IT Projects

The first conclusion that clearly emerges from studies on Critical Success Factors is that difficulties in projects are seldom caused by technologies. This may seem strange, because technologies are often considered to be complex and the source of difficulties. If this is accepted as true as a first approach, it is also true that the solution to technological problems is more deterministic; that is, when confronted with a technological problem, information retrieval and the training and competence of professionals are determinant variables for its resolution. Moreover, interactions between

Research	Main results
Standish Group-Chaos Report (1994 and 2004)	In 1994 the success rate of IT projects was 16%; in 2004 it was 24%. In 2004, 51% of projects were considered problematic; 15% failed.
Robbins Gioia Survey (2001)	51% of respondents considered the ERP project unsatisfactory. 46% of respondents reported that the firm did not understand the system's potential and therefore there had not been any substantial improvement.
Conference Board Survey (2001)	40% of respondents stated that no benefits were apparent a year after the implementation of an ERP system.
Oxford University (2003)	16% of IT projects were considered successful, 74% were problematic, and 10% were abandoned.
Royal Academy of Engineering and the British Computer Society (2004)	Only 16% of IT projects could be considered successful.

Table 1: Statistics on the Success of IT Projects.

technological components, even complex ones, can be objectively represented and schematized, and so be resolved by scientific methods.

The same does not apply to other variables that influence the course of a project. Relationships between people, the influence of technological changes, and stakeholder expectations are all high impact variables, but ones which can never be represented with certainty and objectivity. The margins of indeterminacy are always present and project management therefore becomes the management of uncertainty and dynamism.

Studies suggest that IT project Critical Success Factors can be classified into two groups: factors linked to the organization method, management and control of the IT project (and therefore to expertise in project management application) and relational factors, linked to the ability to understand ourselves and others, and to manage the numerous relationships involved in an IT project in the best way possible.

In Table 2 we propose a list of Critical Success Factors that derive from various researches carried out in the IT industry. For people who manage IT projects, many of these factors will not be new. The aim of this paper is to provide operational indications on the management of some factors that, even if often quoted, are rarely analysed in detail. Since the topic is complex, it is impossible to address all the factors with the same level of analysis in one paper. The choice of which factors to address in depth stems from the observation that their impact is very often underestimated and that they are sometimes not even considered as variables highly influential in the project. Therefore the factors highlighted in bold in the table will be more closely addressed, while the others will be given a brief description in the table, underlining the most important aspects.

In this section we are going to analyse three critical factors in particular: having clear shared goals, generating realistic expectations, and producing realistic planning. For these factors we are not only going to describe the negative impacts but we are also going to provide operational indications on how to manage them in the best possible way.

3 Clear Shared Goals

This factor is one of the most quoted factors in researches. At times it can be very difficult to come up with well-defined goals, but very often, under the pressure of time, we settle for starting without having the situation clear and this invariably generates a great many difficulties. We should not confuse goals with functional or technical specifications: goals are the destination while specifications are part of the means whereby we achieve those goals. Goals are rarely of a technical nature, but they are linked to business results. It is important to have excellent functional and technical specifications however; if they are not part of a wider context, they risk being read without a critical eye and can quickly become obsolete, producing instability in the project.

Moreover, it is fundamental to communicate goals clearly, at least to the people that are most involved in the project. Ensuring that everybody understands the goals will narrow the latitude for personal interpretation.

Only communicating goals that strictly relate to the area for which a person or group of collaborators is responsible and therefore preventing them from appreciating the project as a whole has proven to be a bad habit. Nevertheless, this behaviour, which is often rooted in good faith in that it aims to reduce the complexity perceived by the collaborators, is a mistake for various reasons. Firstly because of the following organizational principle: "People are much more motivated and effective if they understand the context in which they are operating". In other words, understanding how their actions

Clear shared goals (see the text of the article)
Presence of the sponsor and ability to influence the project. A sponsor must have the organizational leverage to be able to support the project when necessary and s/he must be present at the most important moments like the kick-off meeting and meetings where important decisions are taken. The sponsor's contribution to the project is limited from the point of view of time spent, but when s/he intervenes s/he must be incisive. Knowing that the sponsor is ready to intervene if needed provides a strong motivational drive.
Generating realistic expectations (see the text of the article)
Realistic planning (see the text of the article)
Understanding and adapting to the project's natural life cycle. Projects have their own distinct features that we cannot alter. For example, if we have to implement a Decision Support System, it is physiological that users actually using the system are going to learn to appreciate it and that they will discover new needs they did not even know they had before. If we consider this idea to be physiological, we should adopt project life cycles that are congruent with this idea; therefore a large number of interactions should be anticipated and planning cycles should be very short. If, on the other hand, we adopt paradigms that are not contextual to the situation, such as the application of a waterfall life cycle, we would find ourselves in the situation of starting out with some very detailed planning which would prove to be barely applicable, and we would judge the users to be incompetent because they would not be able to communicate all their needs from the outset. Adopting and following the correct life cycle of a project is a fundamentally important issue because it allows us to represent and manage a phenomenon in the way best suited to its nature.
Considering past experience. An ever increasing number of companies are implementing actions to pool and systematize experiences in past projects. In fact, strong repetitiveness has been noticed in factors that negatively influence projects. Making them available and paying due attention to them is certainly a winning strategy. Unfortunately we have noticed that in the Italian IT industry there still is an almost pedagogical vision of mistakes: everyone must also learn through their mistakes. Perhaps this approach can be shared from the point of view of personal growth, but less so from the point of view of keeping costs down and ensuring the success of a project or company.
Understanding the reference context. A project is part of a relational network, albeit a very complex one. It is important to understand who are the key players that can affect the project through their actions, understand their attitude towards the project, and therefore adopt behaviours that are appropriate for the situation, while avoiding the creation, even unintentionally, of situations that are adverse to the project. Users deserve a leading role in the reference context as they are too often trivialized in their role as IT end users. Users represent the end customers of the project efforts and therefore it is possible to obtain interesting results from the careful evaluation of their characteristics and their active involvement. Users are often not involved much in an attempt to "limit" changes. This strategy can be very dangerous because working in isolation and with a very different mentality from that of the users can lead to a significant departure from their expectations, which can in turn lead to extreme instances such as project rejection.
Recognizing complexity. The underestimation of a project's complexity generates not only macroscopic planning mistakes but also the use of inadequate methodologies and tools. One factor that contributes to complexity is scale; i.e. the sheer dimension of the project. In fact it has been noticed that projects with similar content but very different dimensions have led to new problems and to completely different results.
Arranging planning and control systems. Planning and monitoring project activities means knowing exactly where we are and how far away we are from our goal. Improvisation, often adopted in projects that appear easy at first glance, often turns out to be inadequate. An alarm bell for a project manager is being unaware of the current state of progress of the various activities and the problems that are arising in the project.
More frequent project milestones. If the scale factor in projects has a great influence on the complexity and therefore the management of a project, a good method is to try to represent a complex project as the sum of smaller projects. By doing so the focus shifts from the planning and management of one big project to the management of interfaces between various smaller projects. There are no projects that cannot be divided up, but that is not to say that it is easy to adopt this approach. This is a project management skill that is not easy to find.
Having clear project borders. However, even with clear goals there are many interpretations as to exactly where a project intervention should end; in other words, when you can state with

Table 2: Critical Success Factors in IT Projects.

relate to the actions of other colleagues or interlocutors helps people adopt behaviours that are more suited to the specific situation and allows individuals to feel they are part of an important complex initiative and not only mere executors of small activities without clear links to important results. Furthermore, the underestimation of complexity on the part of collaborators leads to intolerance of possible delays or difficulties in the project. In fact, if the perceived complexity is low, the people involved may believe that the difficulties are linked to incompetence or disorganization and they will therefore be less inclined to accept these situations.

However, having well-described, clear, and shared goals is still not enough. Even well-described goals, understood by everybody, can be grossly distorted with the passage of time. Personal expectations, the division of work time into different projects and between projects and functional activities, and the exchange of opinions and information with other colleagues, lead to the modification and reinterpretation of project goals, even if these modifications have not in fact been requested by the client. We are not talking about deviations that lead to the performance of completely different projects, but rather about people performing activities that were not requested and so failing to perform other more urgent activities and dedicating themselves, in different ways, to the same activities, or to activities that may even partly conflict with one another, thereby causing friction and loss of time, a very precious factor in projects.

4 Realistic Expectations

In IT projects the matter of expectation management is very important. Unfortunately we are experiencing a phenomenon of technology trivialization, where the perception of those who do not have IT skills leads them to believe that everything can be resolved with the touch of a mouse in a very short time. The extremely intangible nature of IT itself reinforces this perception: complex projects may be resolved by writing software that can be physically recorded on a CD ROM and this certainly does not help customers or users who are unaccustomed to technology to perceive its underlying complexity.

The problem of creating unrealistic expectations lies in the fact that people will judge the project on the basis of initially incorrect premises incorrect factors. While the raising of expectations is a factor that people experience positively, the lowering of expectations during the project may have serious consequences since it can lead to the alienation of people, the lowering of their willingness to collaborate, and the creation of harsher judgements that do not reflect the efforts of those collaborating on the project.

Given that in an ideal world everyone would be able to appreciate the complexity underlying our efforts, and thereby formulate realistic expectations, a pragmatic approach must instead consider what can realistically be done and not what it would be nice to have. It is actually unrealistic to think that we can provide everybody with all the skills necessary to fully appreciate the complexity of IT projects.

It is therefore necessary to work on other fronts, in particular on our capacity to induce the formation of consistent expectations. While we stress the fact that the problem cannot be fully resolved, we do propose some partial correctives.

In this respect it is important to objectify as far as possible the work performed and to be performed. In other words, from the initial phases of the project, it is useful to provide information that can be easily interpreted by our interlocutors. A first step might be to draw up a clear project plan that can be understood by anyone (see the Critical Success Factor "Common Vocabulary" with regard to this), in which the required commitment in terms of resources and time can be easily appreciated. Moreover, it can be useful to translate technical elements into terms that can be more easily understood by people without IT skills. For example, to explain that in order to develop a program it is necessary to write a quantity of code that, if put on paper, would cover 1000 pages, can be much more immediate for a user than such technical aspects as how much memory is required to run that program or lines of code.

In order to create realistic expectations, the initial phase of the project (the so-called Conception Phase), when ideas and proposals are gathered together and discussed, is essential. In this phase, when looking into the involvement of users or the people that may support the project, we often tend to overestimate the features of the project, thereby contributing to the creation of unrealistic expectations and thus unintentionally laying the basis for more difficult project management. In this respect, the questions put to users during typical interviews are very important. For example, if we wish to improve the services available on a company portal, it could be dangerous to ask about users’ interest regarding specific technological functions because, if they see later on that those functions have not been implemented, they may be dissatisfied. It is better to focus on the needs that users have expressed and to omit any mention of implementation modes.

5 Realistic Planning

To plan means providing oneself a method for achieving desired goals. Planning is useful to those involved in the project for organizing, coordinating with others, and reaching desired goals. If we want planning to support our work it has to be realistic; that is, it must be able to represent, albeit in a simplified way, the hot points of the project and so be useful when performing project activities. In many IT projects we have noticed that planning is completely detached from real project implementation and management. The moment planning becomes a simple "formal code of good conduct" for the project, merely documentation that must be shown in order to demonstrate professionalism, it may even become an obstacle to the success of the project as it requires time to be developed but does not provide any support to project management.

Even those who arm themselves

with good will and try to develop a project plan not only to demonstrate correct reporting, but also to create a real action guide, often discover that their project plans are not very useful and are very different from reality, and so will tend to abandon or underestimate the real importance of good planning. So, why is realistic planning so difficult?

One problem is in the planning activity itself, which consists of the shaping, and therefore the simplification, of future events. Unfortunately, when we approach planning we find it natural to replace timing, costs and the use of resources not with what is likely to occur in the specific case, but rather with average, idealized, and "normal" values. This immediately produces a huge gap between what could realistically occur and what we anticipate will happen. Risks are often completely underestimated and activities idealized, under the assumption that they will be performed without hitches, interruptions or mistakes. However, good planning must consider these factors in order to be as representative as possible of reality as it might happen, and not as we would like it to happen. An innate problem with human beings with regard to our ability to attribute realistic schedules to activities is the difficulty of mentally representing activities over a long time horizon. An example of this can be seen in an experiment that shows how the way we ask ourselves questions about the time required to perform activities has a very strong impact on the realism of planning. Some project managers were asked how many months they needed to perform certain activities; the answer was about one month. They same question expressed in terms of days rather than months caused the managers to reconsider the previous answer and say that two months would most likely be needed. This happens because human beings are "conditioned" in their perception of time by the sleep/ wakefulness cycle. The day is thus our physiological planning unit; the month is only a convention. Only the habit of planning can change this method of relating activities to time.

When planning, therefore, we must always be aware of our natural limits. This aspect is stronger still in projects characterized by profiles with strong technological competences, as often happens in IT projects. As Jerry Madden, NASA project and program manager, has outlined in his study on Critical Success Factors, technically oriented people have a tendency to be optimistic and tend to underestimate the presence and impact of difficulties. To be optimistic is undoubtedly a positive quality, but it should not cause us to be unrealistic, because we run the risk that our projects will run systematically behind schedule, not only because of external causes, but also due to the underestimation of real difficulties that may be encountered and their consequences in terms of time. For example, in Microsoft development departments, many managers normally double the schedule set by programmers. This is questionable from a purely project management point of view, but it is a concrete response to an existing problem.

6 Conclusion

Perhaps to effectively intercept, monitor, and manage all the factors that act as obstacles in the achievement of desired project outcomes is a utopian goal. Typically, information system projects are characterized by strong dynamism and this makes it difficult to effectively identify and control all the variables that can affect project performances. Nevertheless, it is possible to identify and focus efforts on those factors, typically few in number, that may have a significant impact on the success of the project. As we have seen, many factors are linked to project management abilities and organizational sensitivity, more so than to pure technological competencies. This does not mean that technological competencies are secondary for IT project managers; they remain essential as they allow them to have a closer understanding of technological problems which are otherwise often trivialized and underestimated, and they allow them to establish effective relations with the collaborators who have to accomplish tasks of a more operational nature.

Technological, organizational, and project management competencies must all be present and balanced in order to address IT project management related difficulties in an effective manner.

Although there are studies that provide a classification of the importance of different Critical Success Factors derived from qualitative surveys, it is advisable not to trust these reports uncritically but rather to personalize the approach based on the specific reality of each specific project. True average values are not actually much use; it is the project manager, project team, and users' task to collaborate in order to establish the project's strengths and weaknesses and thereby leverage the first and limit the second in order to obtain successful results.

References

[1] Chaos Report, Standish Group, 1994

[2] Chaos Report, Standish Group, 2003

[3] Charette, R.N.: Why Software Fails. IEEE Spectrum, September 2005.

[4] Conference Board Survey, 2001

[5] Cusing K.: Why Projects Fail. Computer Weekly, November 2002.

[6] Dunning D., Heath C., Suls J.M.: Picture Imperfect. Scientific American Mind, December 2005.

[7] Ewusi-Mensah K.: Critical Issues in Abandoned Information System Development Projects. Communications of the ACM, Vol.40, no.9, 1997.

[8] Fielding R.: IT projects doomed to failure. Computing, November 2002.

[9] Fortune J., Peters G.: Information Systems: Achieving Success by avoiding failure. Wiley, 2005.

[10] Hale D.P., Sharpe S., Hale J.E.: Business-Information Systems Professional Differences: Bridging the Business Rule. Information Resources Management Journal, April-June 1999.

[11] Jaques R.: UK wasting billions on IT projects. Computing, April 2004.

[12] IT Project Management: Challenges and Best Practices. Kellogg

School of Management, 2003.

[13] Kelly L.: Government reexamines IT failures. Computing, July 2003.

[14] Liebowitz J.: Information Systems: Success or Failure? The Journal of Computer Information Systems, Vol. 40, no. 1, 1999.

[15] Lyytinen K., Hirschheim R.: Information System failures: A survey and classification of the empirical literature. In Oxford Research in Information Technology, Oxford University Press, 1987.

[16] Martinsons M.G., Chong P.K.C.: The influence of human factors and specialist involvement on information systems success. Human Relations, Vol. 52, 1999.

[17] Common Causes of Project Failure. National Audit Office and the Office of Government Commerce, 2002.

[18] Robbins Gioia Survey, 2001.

[19] Common Methodologies for Risk Assessment and Management, and The Societal Aspects of Risk. The Royal Academy of Engineering, London, 2002.

[20] Sauer C., Cuthbertson C.: The state of IT project management in the UK. Templeton College, Oxford University, November 2003.

[21] Wynekoop J.L., Walz D.B.: Revisiting the Perennial Question: Are IT People Different? Database for Advances in Information Systems, Vol. 29, 1998.

CEPIS Projects

Selected CEPIS News

Fiona Fanning

Euro-Inf Project

The Euro-Inf project aims to create a framework to set up a European system for accrediting informatics education at the first and second cycle levels, and in doing so to improve the quality of informatics education programmes, provide a recognized label for accredited programmes, and ultimately to increase the mobility of European graduates in accordance with the Lisbon Strategy. Partners in this project include ASIIN (the lead partner), CEPIS, the Hamburg University of Applied Sciences, and the University of Paderborn.

Following a meeting of the project board in late 2007, Euro-Inf adopted a refined version of the Framework Standards and Accreditation Criteria for Informatics Programmes. This revised version is available on the Euro-Inf project website <http://www.euro-inf.eu/>.

With the project entering its last phase, the final conference will take place on September 4 and 5, 2008 at Cagliari, Sardinia.

Due to limited space, the conference is aimed at the following participants:

- representatives of European Higher Education Institutes (HEI) interested in quality assurance or obtaining a European Quality Label for their study programmes;
- international informatics experts participating as future Euro-Inf auditors (on invitation);
- representatives from industry and professional societies having an interest in raising the quality of informatics higher education;
- participants from countries in the Bologna/Socrates Area (EU + Iceland, Norway, Liechtenstein, Turkey).

The conference will deal with issues of subject-specific quality assurance for informatics higher education. In particular, it focuses on the activities, results and future of the Euro-Inf project. Further information on this conference will be available shortly on the project website: <http://www.euro-inf.eu/>.

E-Skills Foresights Scenarios project

Although this project is finished and has been submitted to the European Commission, the results are gaining considerable acclaim. The report examined the key trends that will play a role in influencing the supply and demand of each of the three types of e-skills, as well as the off-shoring of ICT work that is of growing political interest and could substantially affect future demand levels. It then examined, qualitatively and quantitatively, how things could develop.

A list of ninety "change drivers" covering social, technological, economic, environmental, political and values-related forces were examined. The main impact of each driver on the demand for ICT Practitioner skills was analysed, then three factors that are strong determinants of this demand and are impacted most significantly by the ninety drivers were identified: the rate of ICT innovation (technological change), economic growth (both within the EU and beyond), and the degree of off-shoring undertaken within the industry.

The project concluded that estimates of supply and demand levels indicate that the EU could run seriously short of people with the right e-skills by as early as 2010. Demand is calculated to be as much as 250,000 per year by then, drawing on an insufficient supply pool of just 180,000. Europe therefore, could face an annual shortfall in skilled IT personnel of 70,000 by 2010. Such a shortfall would cause downstream negative effects and restrict opportunities for economic development. In turn, this could cause the EU to lose its competitive advantage on the world stage and shift ICT activity to other rival global regions.

The report recommends that the ICT industry must not be allowed to develop "organically". It must be managed to prevent the industry from suffering the consequences of an inadequate workforce. This inadequacy could take two forms: too few people and/or not the right depth of skills. To tackle the potential shortage of skilled ICT professionals, the research group recommends the following joint actions and strategies for the industry and policymakers:

- Promotion of better understanding within the ICT industry and public bodies of current quantitative and qualitative skill levels in Europe.
- Creation of awareness of the threats and opportunities in the growth of globalization of ICT activity, and the benchmarking of EU skill levels against competitor economies.
- Seeking of public-private initiatives and investment by ICT industry players and the European Commission to estimate future levels of demand. This would create better understanding of the impacts of cyclical market effects on practitioner supply and demand.
- Training must be used more consistently and it should be ensured that ICT skills are transferred throughout business.
- Concentration on the quality

aspects of skill shortages, not just the quantity. The ICT industry needs skill elites and people with the right level of excellence.

To get the balance of supply and demand right, education policymakers, in regional and national governments and at EU level, must address our future ICT workforce needs now. A greater supply of ICT professionals must be stimulated by investing in the right research and development, by creating a clear and pragmatic immigration policy, and through cooperation between professional bodies, trade unions and policymakers. The ICT industry itself can play an important role in coordinating the efforts of these groups by working closely with educational institutions.

CEPIS welcomes your comments on this report. For the Executive summary and full report of this work please see: <http://www.cepis.org/index.jsp?b=0-636-638&pID=648&nID=717>.

In the future, CEPIS plans to publish a post project report that will take new data into account as well as the changing economic conditions.

Other CEPIS News

On April 12, the CEPIS Spring Council brought together 28 European Informatics Societies for a day of productive discussion in the beautiful city of Ljubljana, Slovenia. Guest speaker, Ljudmila Tozon, Counsellor of the Permanent Representation of Slovenia to the EU for Telecommunications and Information Society, provided an update on the European Union's i2010 strategy, with a preview of the upcoming mid-term review as well as an overview of the Slovenian Presidency priorities.

The following SINs and Working Groups provided updates on their work: Legal and Security SIN, Education and Research, Professionalism, and Informatics Students and Young Professionals.

If you are interested in learning more about the work of these CEPIS groups, please contact Fiona Fanning, CEPIS Policy and Communications Executive, <fanning@cepis.org>.