Report from 2nd International Workshop on Developing Tools as Plug-ins (TOPI 2012)

Diego Garbervetsky
Departamento de Computación, FCEyN
Buenos Aires, Argentina Hong Kong
diegog@dc.uba.ar

Sunghun Kim
UBA Hong Kong Univ. of Science and Tech.
China
hungkim@cse.ust.hk

DOI: 10.1145/2382756.2382775
http://doi.acm.org/10.1145/2382756.2382775

ABSTRACT

The International Workshop on Developing Tools as Plug-Ins (TOPI) is a venue for researchers and practitioners interested in plug-in development. The main interest is understanding the opportunities and challenges of developing tools as plug-ins, and thus, we seek for discussions regarding the characteristics of good plug-ins, interoperability requirements to making tools available across platforms, recent successful tools as plug-ins as well as foreseen medium and long term challenges of tools as plug-ins. The second edition of this workshop, TOPI 2012 was co-located with the International Conference on Software Engineering (ICSE 2012). TOPI 2012 received a total of 32 submissions. Among them, 14 were accepted as full papers and 4 as short papers. The audience during the whole workshop ranged from 25 to 30 participants. The final program comprised position papers including new proposals for plug-in architectures as well as their interaction with development environments and run-times, and papers discussing the implementation of different kind of tools as plug-ins. This report describes the main results of TOPI 2012.

1. INTRODUCTION

Our knowledge as to how to solve software engineering problems is increasingly being encapsulated in tools. These tools are at their strongest when they operate in a pre-existing software development environment. This approach allows integration with existing elements such as compilers, debuggers, profilers, visualizers as well as numerous other development and often, runtime tools.

However, building tools as plug-ins can be challenging: How do they interact with the core environment? How do they interact with one another, especially since each developer may choose a different set of plug-ins. How can we share tools across different and future core development environments?

This series of workshops is intended to all those interested in developing tools as plug-ins that can be integrated into IDEs, middle-wares and/or browsers.

Submission process for TOPI 2012: To achieve a reasonable spread of attendees, we called for 6-page papers covering topics like characteristics of good plug-ins, examples of successful tools as plug-ins, and challenges in incorporating tools as plug-ins (e.g., cross platforms, architectural limitations, etc.).

Each submission had to address research, ongoing work, ideas, concepts, and critical questions related to the engineering of software tools as plug-ins. We were also looking for interesting case studies, position papers and/or technology prototypes.

We received 32 submissions from authors of 17 different countries. Those papers were reviewed by at least 3 members of our program committee, composed by 13 experts from academia and industry. After the reviewing process, we ended up accepting 14 submissions as full papers and 4 as short papers. The best 7 papers were also invited to submit an extended version for inclusion in the Software Practice and Experience journal. The short papers had to present their work during a demo session. In addition, authors of accepted full papers who presented concrete plug-ins or tools were also invited to showcase them during the demo session.

The final program comprised position papers including new proposals for plug-in aware architectures, discussions about plug-in ecosystems, and interaction with their development environments and run-times. We also accepted papers explaining the design and implementation of different kind of tools as plug-ins.

Organization: The second edition of this workshop was held in Zurich on June 3 2012, co-located with the International Conference on Software Engineering (ICSE 2012).

The workshop program was organized in four sections: three main sessions devoted to full paper presentations, one for general discussions about plug-in development, one for plug-ins integrated in development environments, and the last one for presentations of more heterogeneous plug-ins. Each speaker had a slot of 15 minutes for presenting her work and a few minutes for clarification questions. At the end of each section, there were 10 minutes for open discussions between the audience and all the speakers. The workshop included a demo session featuring brief presentations for short papers and several demos.

The audience during the whole workshop varied along the day between about 25-30 participants[1].

2. KEYNOTE

Harald C. Gall, a professor of software engineering in the Department of Informatics at the University of Zurich, Switzerland delivered the TOPI 2012 keynote, and it was well received.

He discussed the potential of integrating software evolution analyses as pluggable services into development environments. In this setting, he presented the SOFAS (SOFtware Analysis Services) approach, a distributed and collaborative software analysis platform to allow for interoperability of software analysis tools across platform, and geographical and organizational boundaries. Such tools are mapped into a software analysis taxonomy and adhere to specific meta-models and ontologies for their category of analysis. They also offer a common service interface that enables their composite use on the Internet These distributed analysis services are accessible through an incrementally augmented software analysis catalog, where organizations and research groups can register and share their tools.

As an example, he presented SmellTagger for detecting code smells using a multitouch interface. SmellTagger uses the SOFAS web service to detect and retrieve code smell information.

Finally, he discussed current limitations and potential new horizons for software evolution research. He proposed the use of software analyses as services with semantically defined resources, supporting an agile development process, leveraging new user interfaces and multitouch devices (like tablets or the Microsoft Surface) for interacting with code and other software artifacts.

3. WORKSHOP SESSIONS

3.1 Plug-In Development

During this session, we discussed different approaches for plug-in development and interaction with their ecosystems.

The first work presented in this session entitled "Playing Cupid: the

[1] In http://topi2012.dc.uba.ar there are pictures of the speakers and the audience as well as information about the program committee.

IDE as a matchmaker for plugins" [16], addressed the problem of interacting plug-ins which is a main concern in plug-in development. This work introduced an approach in which data-driven plug-in capabilities play a central role. The authors proposed a plug-in architecture when the IDE permits other plug-ins to exchange/subscribe to data in a functional style. This enables the construction of new plug-ins by composing the input/output of other plug-ins in a pipeline style.

Sharing the same idea on how to take advantage of the existing functionally and data of other plug-ins, the paper "IDEs Need Become Open Data Platforms (as need Languages and VMs)" [10] stated that IDEs and their plugins should make their data available to other plug-ins. The author argued that data from IDE or other plugins is often hidden and not trivial to access, making it very hard to use the available data and functionality. After this presentation, there was an interesting discussion about how to balance open access with maintainability, which is usually enforced by information hiding.

The paper "Simplicity Principles for Plugin Development: The jABC Approach" [12] discussed the advantages of using JABC for plug-in development. The jABC which is a graph-drawing plugin framework has been evolving during the last 10 years. The authors claimed that, using this framework plug-in developers can deliberately focus on the actual functionality, like providing semantics to graphs, without having to deal with tedious but semantically irrelevant issues like user interfaces. Plug-in functionality can be itself conveniently modeled as a workflow within the jABC. To justify the claim they showed some examples.

The session included also the paper "A Reference Architecture for Integrated Development and Run-time Environments" [17] which presented a reference architecture for programming environments offering support both at development time and at run time. During the presentation the speaker analyzed various existing IDEs with respect to this the reference architecture.

Finally, in "XML Development with Plugins as a Service" [9] the author discussed how plug-ins can be developed as a service based on their experience with an XML tool called XMLStyleHelper. He carefully outlined benefits and challenges of their service-based approach. Examples of benefits are seamless updates, faster delivery of new features and platform independence and better data collection on how a plug-in is used. Some of the challenges he mentioned are related with managing the confidentiality, user behavior, privacy and provision of platform-independent access.

3.2 Session 2: Verification Tools as Plug-ins

This session was devoted for plug-ins related with verification or program visualization tools. We analyzed examples of successful integration of tools using plug-ins. In particular, we wanted authors to share their experiences and lessons learnt in order to help us understanding the advantages and obstacles and they found working in current architectures.

The first work discussed the implementation of "TacoPlug: An Eclipse plugin for TACO"[4], which is a plug-in to support TACO, a bounded verification tool for Java programs annotated with JML that relies on a SAT-solver. One of the challenges of working with bounded verifiers is that the output is often difficult to understand. To address this issue, the plug-in provides several views (notably a heap visualization view) for assisting developers in finding the actual part of the code that violates the specification. The authors showed how they leverage on existing IDE features that programmers are used to deal with in order to present the results of the analysis in a way that is easier to understand.

The presentation of the paper "Integrating a Set of Contract Checking Tools into Visual Studio" [6] pinpointed the main lessons learnt when the team incorporated CodeContracts, a language-agnostic way to express and check coding assumptions in .NET programs, into Visual Studio. The authors gave a series of recommendations. Among others, they said is better to embed new language features into source languages using only existing language features (such as attributes and calls to special libraries) rather than new syntax or stylized comments; give semantics to the new features by rewriting the compiler output (the target language), rather than the source input; perform analysis of code at the target language level, rather than the source level; for IDE integration, write plug-ins that are reusable across many tools rather than a single specific one. This means that generic plug-ins should be also pluggable with other tool specific extensions.

The next presentation was for the paper "The EventB2Dafny Rodin Plug-In" [3]. EventB2Dafny is a plug-in for the Rodin platform which provides support for program refinement and mathematical proof for the EventB language. The plug-in used Dafny, an imperative language that includes an automatic verifier of assertions and contracts, as intermediate language to discharge proof obligations. This enabled Rodin to use all the power of the Dafny verifier, including its connection with automated theorem provers.

Finally, the paper "Plugging In and Into Code Bubbles"[14]. Code Bubbles is an IDE front-end that, instead of displaying programs as large chunks of text, relies on displaying information in context inside connected smaller windows called bubbles. The author described the main features of a "code bubbles" programming environment, how it was built as a set of plugins on top of Eclipse. He also described the plug-in architecture supported by CodeBubbles itself that uses message passing as a mean to communicate between plug-ins with the core system. He showed how this architecture can support various types of plug-ins.

3.3 Session 3: Tools as Plug-Ins

This final session presented plug-ins targeting non-traditional applications rather than IDEs.

The authors of the paper "An Architectural Blueprint for a Plug-gable Version Control System for Software (Evolution) Analysis" [7] proposed an approach for attaching plug-ins to version control systems in order to monitor what is going on in a project. Events such as commit, tag, branch, or merge are made available, and can be monitored by plug-ins to provide awareness. For that, they presented a model of version control history supporting releases, branches, files, versions, and change set. They showed a series of usage scenarios including continuous code quality monitoring, extracting fine-grained changes, and querying data.

The next paper entitled: "SSELab: A Plug-in-based Framework for Web-based Project Portals" [8] presented a framework to assist in the migration of tools from desktop to servers. The idea is leveraging the available processing power of current server infrastructures. They showed readily available clients that can be used in different contexts, either from the command line or from within IDEs such as Eclipse, even after migrating development tools from desktop to server environments.

In the context of model based analysis the authors of the paper "SAML goes Eclipse -Combining Model-Based Safety Analysis and High-Level Editor Support" [11] proposed an eclipse plugin that allows the modeling of a system using the SAML modeling framework. They used model transformers to translate that model into various formats required by existing engines for safety analysis. The authors also provided an overview of the design choices and requirements for the tool implemented as plug-in. For instance, instead of showing the output of the underlying analysis tool, they need to implement for each engine a translator of the error message to the SAML model.

The last two presentations addressed the problem of dealing with data sources. In the paper "Surfacing Scientific and Financial Data with the Xcel2RDF Plug-in" [13] the authors presented an Excel plug-in for converting spreadsheet data into RDF triples. The plug-in assists the user in the selection of RDF vocabularies, in the discrimination between data and metadata, assigning semantics to data and including provenance information. With similar goals in mind, the work "OLAP2DataCube: An Ontowiki Plugin for Statistical Data Publishing" [15] described a tool that translates large analytical databases into linked data suitable for further processing. The tool uses Online Analytical Processing (OLAP) as input and generates RDF in the Data Cube vocabulary as output. The author showed how the tool was used to port on a large dataset related to open government data in Brazil.

3.4 Short paper and Demos Session

During lunch time, we had a very exciting demo session.

The first group of demos was for the papers accepted as short papers. They presented their work with a poster and a live demo. They were: "A Modular Environment for Software Development and Re-Engineering" [2], "Developing a Plugin Tool to Make OneNote an e-Textbook" [5], "Rumadai: A Plug-in to Record and Replay Client-side Events of Web Sites with Dynamic Content" [18] and "IBM Software Development Kit for Power-Linux" [1].

There were some free time slots to show other great demos from the workshop participants. In particular, we enjoyed a tours of CodeBubbles and its Eclipse plug-in, and CodeContracts and its Visual Studio extensions.

4. CONCLUSIONS

We believe this second edition of the workshop was successful, maintaining a significant number of submissions, good quality papers and an active audience. This showed there is a great interest from the software engineering community in these kind of topics.

Even there were some exceptions, in these two editions of the workshop, there were a clear bias to discussions about plug-ins extending IDEs functionality. In the following years we expect (hope?) to receive more submissions related to other kind of systems like runtime systems, compilers and, specially, browsers.

We would also like to have more interaction with key industrial players. With this idea in mind, we plan to invite industry leaders involved in the design and development of mainstream IDEs, browsers and run-time engines to participate by either submitting experience papers or by being part of the program committee. We also plan to invite these main players to be keynote speakers in future editions of this workshop.

5. ACKNOWLEDGMENTS

The workshop organizers would like to thank all the contributors to this workshop, starting from the authors, participants, the program and steering committee, the ICSE 2012 conference and workshop organizers, student volunteers and colleagues for their advices. A special thanks to Prof. Dr. Harald Gall for accepting to be the keynote speaker and bring us an excellent presentation.

6. REFERENCES

[1] R.F. Araujo, D.H. Barboza, O.B. Pontes, R.M. Teixeira, R.S. Joao, W.S. Moschetta, and V.H.S. Durelli. IBM software development kit for powerlinux. In TOPI2012, pages 86 –87, June 2012.

[2] S. Campana, A. Poli, L. Spalazzi, and F. Spegni. A modular environment for software development and re-engineering. In TOPI2012, pages 90 –91, June 2012.

[3] N. Catano, K.R.M. Leino, and V. Rivera. The eventb2dafny rodin plug-in. In TOPI2012, pages 49 –54, June 2012.

[4] M. Chicote and J.P. Galeotti. Tacoplug: An Eclipse plug-in for taco. In TOPI2012, pages 37 –42, June 2012.

[5] J. Cristy and J.G. Tront. Developing a plug-in tool to make onenote an e-textbook. In TOPI2012, pages 84 –85, June 2012.

[6] M. Fahndrich, M. Barnett, D. Leijen, and F. Logozzo. Integrating a set of contract checking tools into visual studio. In TOPI2012, pages 43 –48, June 2012.

[7] G. Ghezzi, M. Wursch, E. Giger, and H.C. Gall. An architectural blueprint for a pluggable version control system for software (evolution) analysis. In TOPI2012, pages 13 –18, June 2012.

[8] C. Herrmann, T. Kurpick, and B. Rumpe. Sselab: A plug-in-based framework for web-based project portals. In TOPI2012, pages 61 –66, June 2012.

[9] S. Karus. Xml development with plug-ins as a service. In TOPI2012, pages 25 –30, June 2012.

[10] A. Kuhn. Ides need become open data platforms (as need languages and vms). In TOPI2012, pages 31 –36, June 2012.

[11] M. Lipaczewski, S. Struck, and F. Ortmeier. Saml goes eclipse: Combining model-based safety analysis and high-level editor support. In TOPI2012, pages 67 –72, June 2012.

[12] S. Naujokat, A. Lamprecht, B. Steffen, S. Jorges, and T. Margaria. Simplicity principles for plug-in development: The jabc approach. In TOPI2012, pages 7 –12, June 2012.

[13] M.L. Pesce, K.K. Breitman, and M.A. Casanova. Surfacing scientific and financial data with the xcel2rdf plug-in. In TOPI2012, pages 73 –78, June 2012.

[14] S.P. Reiss. Plugging in and into code bubbles. In TOPI2012, pages 55 –60, June 2012.

[15] P.E.R. Salas, M. Martin, F.M. Da Mota, S. Auer, K.K. Breitman, and M.A. Casanova. Olap2datacube: An ontowiki plug-in for statistical data publishing. In TOPI2012, pages 79 –83, June 2012.

[16] T.W. Schiller and B. Lucia. Playing cupid: The IDE as a matchmaker for plug-ins. In TOPI2012, pages 1 –6, June 2012.

[17] H. Tajalii and N. Medvidovic. A reference architecture for integrated development and run-time environments. In TOPI2012, pages 19 –24, June 2012.

[18] A. Yildiz, B. Aktemur, and H. Sozer. Rumadai: A plug-in to record and replay client-side events of web sites with dynamic content. In TOPI2012, pages 88 –89, June 2012.