PDF Support for All – Source-to-source Compilation Between Smalltalk Dialects by Example

Master's Project Proposal, Summer Term 2022
Software Architecture Group
Prof. Dr. Robert Hirschfeld

Project Scope and Goals

In this project, students will explore the design space of source-to-source compilation through the firsthand experience of bringing the project "PDFtalk" from VisualWorks over to Squeak/Smalltalk via automatic translation. The following goals form the starting point for this project and are likely to be refined and extended:

- *Identify and compare* the basic components of two different (but similar) Smalltalk systems, namely Cincom's VisualWorks and Squeak/Smalltalk. Begin with simple domain concepts such as Color and DateAndTime to then dive into more advanced techniques such as exception handling and the meta-object protocol.
- *Apply and extend* the existing "Smalltalk Transform" project, which is already capable of translating sources from VisualWorks to Gemstone/S. Write and test new transformation rules until the solid "PDFtalk" test suite indicates that the project can fully run in a Squeak environment.
- *Design and implement* new interactive tools tailored to Squeak's Morphic framework so that users can directly manipulate PDF documents without having to leave the environment.

As a starting point, the following transformation rules illustrate the abstraction level that students will encounter:

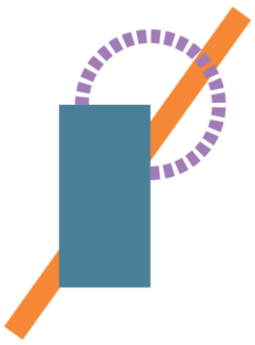
```
PackageChange
  unusedClasses: #(
    #{Smalltalk.GeneralBindingReference}
    #{Core.LimitedPrecisionReal})
  newSuperclasses: (Valuemap
    with: #{Timestamp} -> #DateAndTime
    with: #{Smalltalk.ColorValue} -> #Color)
  hierarchyChanges: (Array with: (ClassChange
    classReference: #{Smalltalk.Value}
    classChanges: (Array with: (Rewrite
      method: #localSpecification
      rule: #replaceLiteralBindingWithPath)))) ...)
```

The PDF specification (ISO standard PDF 32000-1:2008) is part of this project's domain. A basic understanding of its contents can help locate the corresponding source code in "PDFtalk." Identifiers from the specification may be present in transformation rules:

```
PackageChange localChanges: (Array
  with: (ClassChange
    classReference: #{PDFtalk.Fonts.Font}
    instanceChanges: (OrderedCollection new
      add: (Rewrite
        method: #fontbody
        rule: #replaceGemstoneErrorClasses);
      add: (Ignore method: #screenFont);
      add: (Ignore method: #screenFontDescription);
      add: (Ignore method: #screenFamilyName);
      add: (Ignore method: #isScreenFontInstalled);
      yourself)
    classChanges: (Array
      with: (Replace method: #loadedFonts code: #_gs_loadedFonts)
      with: (Replace method: #fontClasses code: #_gs_fontClasses))) ...)
```

Context and Motivation

There are several different Smalltalk dialects, and each has often its own Smalltalk community. After the original development of Smalltalk-80 at Xerox Parc, other companies continued the efforts to evolve the base system as a product. These Smalltalk-80-based systems include Apple Smalltalk, ObjectWorks, VA Smalltalk, VisualWorks, and Squeak. Since Smalltalk is not just a programming language but a whole programming system, evolution means not only adapting syntactical elements but also the standard library and hence the messages that objects can understand. As a result, different Smalltalk dialects emerged, all sharing the same roots.



While objects and messaging remain at the heart of any Smalltalk dialect (and system), all kinds of applications were developed that are not cross-dialect compatible per se. These applications can be executed on different platforms such as Windows, macOS, and Ubuntu/Linux thanks to byte-code abstraction and virtual machines. While dialect-specific applications may reach some end-users this way, however, they do not reach the respective Smalltalk communities, each living in their favorite dialect and system. A Smalltalk programmer can only really enjoy a framework, library, or application if its sources are compatible, that is, if the objects can talk directly to the rest of the system. Consequently, objects that must overcome inter-process boundaries (i.e., instances of different Smalltalk VMs) represent a challenge for programmers because there is no convenient way to manage a distributed object space/memory yet. As of today, Smalltalk programmers rely on the concept of a shared object space with live, interactive tools where they can directly explore, understand, and modify applications with ease.

Therefore, source-to-source compilation can help to combine the efforts of different Smalltalk communities. Programmers can continually work with the dialect they have the most experience. They can rely on the transformation pipeline to make the results available for other communities. Transformation rules usually target a specific dialect version and are largely independent from a specific project or application. Yet, since most Smalltalks are dynamically typed, project-specific knowledge may be required to configure selected message sends correctly. Also, as dialects evolve, transformation rules need to evolve as well. Therefore, programmers can benefit from tool support to identify breaking changes and explore possible steps to move forward.

normal**bold***italic****boldItalic***Times***boldItalic***

This master's project covers such source-to-source compilation by example. The "PDFtalk" library, implemented in VisualWorks, should be made available to other Smalltalk dialects. There is a working transformation framework called "Smalltalk Transform," and a solid test suite to verify the transformation rules. While the students will focus on Squeak/Smalltalk as target dialect, there is ongoing efforts to support other dialects such as VA Smalltalk, too. Consequently, it is expected that the "Smalltalk Transform" project will evolve along the way to support features that are yet unknown but required for successful source transformation. Finally, students will implement interactive tools around PDF exploration and modification from scratch since transforming VisualWorks' GUI framework into Squeak's Morphic is out of scope. The existing "PDFtalk Snooper" may serve as inspiration for such tool support.

Project Values
In this project, students are expected to follow certain practices and represent certain values so that all emerging ideas and solutions can outlive this project's timebox:

- **Explorable implementation**

  The project's solution will be implemented partially in VisualWorks (i.e., the transformation rules) and partially in Squeak/Morphic (i.e., the PDF explorer). Especially the Squeak portion should remain explorable in the sense that all graphical and non-graphical software artifacts can easily be worked on using Squeak's interactive toolset.

- **Backwards compatibility**

  The proposed solution should work for existing applications, too. This means that students have to learn about the existing baseline (i.e., mostly names of existing concepts) to be able to identify potential conflicts when transforming VisualWorks code to Squeak.

- **Cross-platform compatibility**

  The proposed solution should work on recent versions of Microsoft Windows, Apple macOS, and established Linux distributions. Consequently, students have to be careful when relying on platform resources such as external fonts to be processed in the context of PDF documents.

- **Community contributions**

  On a regular basis, parts of the proposed solution should be packaged as contributions to the Squeak Trunk as well as other involved projects. On the one hand, this can include fixes, code refactorings, and feature additions to Squeak's base system and Morphic. On the other hand, this can include changes to the VisualWorks projects "PDFtalk" and "Smalltalk Transform" as explained above.

Further Reading

The "PDFtalk" and "Smalltalk Transform" projects
→ https://www.youtube.com/watch?v=BksxQIuj_SM (Talk at ESUG'11)
→ https://wiki.pdftalk.de/doku.php?id=start
→ https://wiki.pdftalk.de/doku.php?id=smalltalktransform
→ https://wiki.pdftalk.de/doku.php?id=pdftalknonnamespacefileout (this project)

PDF 1.7 specification (ISO standard PDF 32000-1:2008)
→ https://wiki.pdftalk.de/lib/exe/fetch.php?media=pdf:pdf32000_2008.pdf

A brief history of Smalltalk
→ https://en.wikipedia.org/wiki/Smalltalk#History

T. Schrader and C. Haider, "Complex values in Smalltalk," in *Proceedings of the International Workshop on Smalltalk Technologies (IWST'09)*, 2009.
→ https://wiki.pdftalk.de/doku.php?id=complexvalues

M. Taeumel and R. Hirschfeld, "Evolving User Interfaces from Within Self-supporting Programming Environments: Exploring the Project Concept of Squeak/Smalltalk to Bootstrap UIs," in *Proceedings of the Programming Experience 2016 (PX/16) Workshop*, 2016, pp. 43–59.
→ Includes an overview of basic components in Squeak

Organization

A group of three to five (3-5) students may participate in the project. Organization and tasks will be explored and mainly determined by the project participants. The project will be carried out at the Hasso Plattner Institute in Potsdam. Project participants are expected to communicate with the advisors on a regular basis. Communication tools include GitHub issues, projects, wiki, and e-mail. There will also be talks to prepare so that students can collect feedback from the entire Software Architecture Group. Communication is likely to be conducted in English and German. Expected results include a working software accompanied by appropriate documentation.

Contact

Dr. Marcel Taeumel (C-E.11, marcel.taeumel@hpi.uni-potsdam.de)
Patrick Rein (C-E.5, patrick.rein@hpi.uni-potsdam.de)
Prof. Dr. Robert Hirschfeld (C-E.7, hirschfeld@hpi.uni-potsdam.de)

Software Architecture Group
http://www.hpi.uni-potsdam.de/swa