Considering Software Preservation

Brian Matthews, Arif Shaon, Juan Bicarregui, Catherine Jones, Esther Conway and Jim Woodcock

Software is a class of electronic object which is by its very nature digital, and the preservation of software is often a vital prerequisite to the preservation of other electronic objects. However, software has many characteristics that make preserving it substantially more challenging than for many other types of digital object. Software is inherently complex, normally composed of a very large number of highly interdependent components and often forbiddingly opaque for people other than those who were directly involved in its development. Software is also highly sensitive to its operating environment, with the typical software artefact depending on a large number of other items including compilers, runtime environments, operating systems, documentation and even the hardware platform with its built-in software stack. Preserving a piece of software thus involves preserving much of its context as well.

Handling these challenges is a major barrier to the preservation of software, so much so that the preservation of software is often seen as a secondary activity, less critical than the preservation of the data it manipulates. This is despite the fact that in many cases, such data becomes unusable without the software to handle it; and recreating software from partial information can be a near-impossible task.

Software preservation is thus a relatively underexplored topic and there is little practical experience in the field of software preservation as such. The e-Science Centre, Science and Technology Facilities Council (STFC) has undertaken preliminary studies sponsored by the UK Joint Information Systems Committee (JISC) into the *Significant Properties of Software* for preservation (2007), and subsequently in a development project on *Tools and Guidelines for Preserving and Accessing Software Research Outputs* (2007-09). Given the relative immaturity of the field, the studies became explorations of the notion of software preservation, looking at the stakeholders and motivations behind software preservation as much as identifying methods and technology.

Different communities have different motivations for preserving software, such as libraries and archives, managers of data archives who have a need to preserve associated software, and software developers who themselves maintain and reuse software over the long term. We therefore considered different approaches to software preservation, ranging from a strong emphasis on preserving software executables directly, which uses hardware preservation and emulation, to an emphasis on preserving the essential behaviour of software in a new context via migration and porting. The choice of preservation approach depends on the nature of the software artefacts available, the extent to which the original operating environment of the software can also be preserved or reproduced, and legal restrictions such as software licensing.

The project identified some concepts useful for a software preservation methodology, discussing the stages of retrieval, reconstruction and replay which need to be passed through to reproduce a usable performance of a software product. We defined a notion of adequacy of preservation, an aspect of the authenticity of preservation which tests the future performance of software against specified preservation properties once it has been reconstructed into a working version in the new environment. We developed a new software preservation framework to categorise software components and identified the properties required to ensure adequate preservation. This software preservation framework uses concepts from the Open Archival Information System (OAIS) information model; indeed the framework can be seen as a specialization of OAIS for the case of software.

We tested the software preservation framework in collaboration with the British Atmospheric Data Centre (BADC). This included assessing the overall efficiency of the framework against a variety of BADC software, specifically in terms of its relevance (to the software) and sufficiency (of the information recorded) for long-term preservation of the software. The cost-effectiveness of the framework must be considered within the context of the BADC's approach to accommodating changes in the technological environment to ensure effective long-term software maintenance and reuse.

Software engineering best practice shares many of the concerns of software preservation in producing quality software that can be maintained and reused in the future, such as providing version control, dependency analysis and good documentation. Software preservation could be integrated into the software life cycle to systematically capture those properties required for preservation and an adequate replay of the software. We have provided a Java-based tool called Significant Properties Editing and Querying for Software (SPEQS), developed in view of our analysis of the BADC use case (Figure 1). SPEQS demonstrates the feasibility of capturing preservation properties identified in the software preservation framework and integrating these with the software development life cycle, to aid in long-term preservation.

Link:

British Atmospheric Data Centre: http://badc.nerc.ac.uk/home/index.html

Please contact:

Brian Matthews, e-Science Centre, Science and Technology Facilities Council, *UK*

E- mail: brian.matthews@stfc.ac.uk

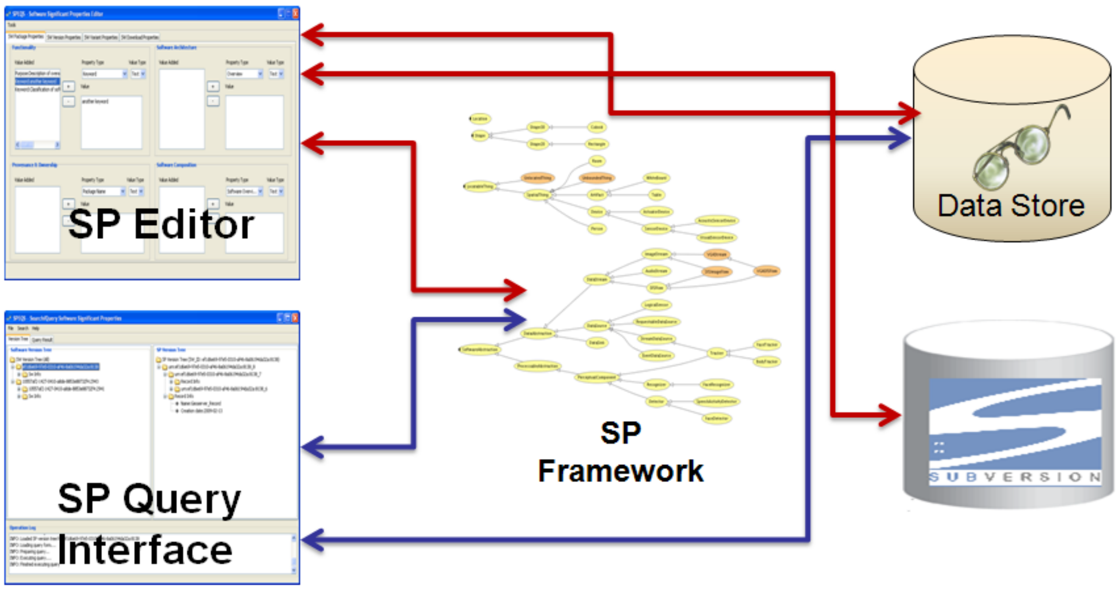

Figure 1: Using the SPEQS tool to capture preservation properties of software.