Vmake, Ise and Ircs -
General Tools for the Intensionalization of
Software Systems

Paul Swoboda, Dr. Bill Wadge
University of Victoria, British Columbia, Canada

Abstract

Many tools have been developed to date that employ intensionalized identifiers, markup, or files to accomplish versioning; however, most of these tools have been localized in utility. Packages such as Lemure and IHTML have effectively applied versioning to specific systems (in the case of Lemure and IHTML, to C projects and HTML files, respectively) but were not generally applicable to projects and source files in other programming environments. For example, Lemure cannot readily be used to version a library of non-C code, just as IHTML is not well-suited to versioning markup languages other than HTML. Vmake and Ise are two general tools that can be used to version arbitrary projects in a structured fashion.

Vmake is a make tool, similar in design to traditional make programs, but which operates on versioned source-files of arbitrary types, and which allows the rule names to be versioned. That is to say, just as every file can exist in multiple versions, so too can each of the rules in the make file. At every level of the resulting make tree, an intensional best-fit is made on each of the dependencies for a given rule, to determine which rule or file will be used for the resulting target. Because the rule names and file names in a make program are really identifiers, Vmake can be thought of as a general, intensional, rule-based build-language. It can be used to apply versioning to any software entity that can be constructed from source files and/or shell commands.

Ise, for “Intensional Sequential Evaluator”, is an imperative scripting language written in the spirit of Perl, but which uses versioning for all of its identifiers, including variables, files and functions. The language has most of the flow-control and data structures of Perl and C, but allows each construct to be versioned. The basic idea of Ise is that a thread of execution carries with it a current context (or global version), and every time an identifier is used in an expression, an intensional best-fit is made at run-time, against the current context, to determine which version of the identifier is being considered. Provision is also made to version plain blocks of code, and the current context can be modified at any point through either direct, sequential changes or over function calls and
scope boundaries. In addition to native scalar, array and hash datatypes, Ise has a version-hash datatype which maps version expressions to values, by intensional best-fit. This datatype can actually be used to re-implement many of Ise’s other, built-in versioning features and provides an easy way to store intensionalized data. Ise naturally has extensive support for dealing with its own flavour of version expressions, and uses overloaded operators to supply such functions as version-refinement and the addition of version-modifiers. The resulting language is ideal for the efficient construction of intensional scripting-tools. Ise has thus far been successfully tested as a CGI language and as a versioned build-tool.

Common to both Vmake, Ise, and the latest version of IHTML is a versioned file-repository system called Ircs (for Intensional Repository Control System). This collection of command-line tools allows the user to create and maintain versioned repositories in a similar fashion to the traditional Rcs tool-suite, but with intensional versions instead of revision numbers. Advantages of Ircs over the repository system used by IHTML2 include slightly faster file-access and smaller archive filenames. When spawning child processes, Ise, Vmake and IHTML pass context information to their children through the environment. Because of this context connection and the fact that all three languages use the same system of version expressions and versioned file-handling, the three packages can be easily used together. Over the next year, efforts will be made to construct several example systems that use all of these tools, to demonstrate the versioning of larger software systems.