

CMPS 290g, Fall 2003, Article Summary Review: **A Layered Approach to Software Design**

Joerg Meyer, jmeyer@soe.ucsc.edu

November 18, 2003

1 Original Paper

I. P. Goldstein and D. P. Bobrow, *A Layered Approach to Software Design*, in Interactive Programming Environments, D. R. Barstow, H. E. Shrobe, and E. Sandewall, Eds. New York, NY: McGraw-Hill, 1984, pp. 387-413.

2 Summary

Goldstein and Bobrow present a layered network approach to represent changes in software design and development. The fundamental difference to a file based version control system is that software is represented as network in which the nodes represent software modules (e.g., products, components, classes, class methods, etc.) and the edges denote a containment hierarchy. Changes in this system are represented by layers. A layer is similar to traditional delta files such that it only contains the changes to the network layer it was derived from. The layered network approach was implemented in PIE (Personal Information Environment), which can be viewed as an extension to the standard Smalltalk class browser.

3 Reviewer Ratings

	Arithmetic Mean	High	Low
Importance	4.14	6	2
How well written?	3.85	5	2

Table 1: Rating of 6 (best) to 1 (worst).

4 Reviewer Commentary

4.1 General Commentary

- Reviewers viewed the layered network approach as change-centric being comparable to change based systems because a layer reflects a set of changes.

- Reviewers understood the layered networks as a generic model that can cover all kinds of artifacts, such as a class, a method, a module or even a product. This generic model allows for fine-grained modelling of source code, so that changes can be tracked on a more fine-grained level than files.
- Reviewers noted that PIE was the predecessor to a system called Envy and that PIE enabled multiple inheritance in Smalltalk, which was not part of the Smalltalk standard.

4.2 Positive Commentary

- Reviewers found the inherent change-based nature of the network model to be more natural than the file centric view of other SCM systems. Changes can easily be located by comparing layers because layers represent changes and contexts represent variants, similar to change-sets.
- The network model and its fine-grained representation of a design object allow easy and better visualization of changes by identifying changes in a method vs showing differences in files by comparing lines.

4.3 Negative Commentary

- Quite a few reviewers doubt the applicability to other programming languages than Smalltalk. It seems fairly integrated and may require significant effort to allow the maintenance of networks while allowing the use of file based design and development tools(e.g., editor, compiler).
- An important issue in software design is collaboration and the ability to share changes. The paper is very vague on collaboration and network support which allow users to share their changes by “sending layers”. This perception is mostly due to the tight integration into the Smalltalk class browser.
- Besides the tight integration into the Smalltalk IDE, quite a few reviewers felt that the layered network approach has severe limitations in terms of visualizing large amounts of versions and changes resulting in quite complicated and complex networks. Furthermore, the PIE browser presented in the paper was regarded as too flat, being a textual representation of a virtually 3-dimensional tree (layers of 2 dimensional networks(trees)).
- While it seemed obvious to the reviewers on how to merge layers that modified different parts of the previous layer, the paper provided no detail on merging layers that changed the same node.
- Last but not least, there is no evolutionary history maintained in a layered network, which poses the question of the order of layers. The reviewers felt that the evolution (history) is one of the more important features of an SCM system and that the PIE browser does not seem to provide that functionality.