ORGANIZATIONAL DYNAMICS OF SOFTWARE PROBLEMS, BUGS, FAILURES, AND REPAIRS

> *"As a rule software systems do not work well until they have been used, and have failed repeatedly, in real applications."*
> -- D.L. Parnas [1990]

This is a very early, rough, and preliminary description of some aims of a new project we have undertaken with NSF support. It's intended to be communicative and informative, but not definitive; the directions below are subject to change and don't necessarily reflect our current views or the state of our research.

OVERVIEW

Worldwide communities of software developers have begun collaborating in large groups to build sophisticated computer software such as the new Mozilla web browser, the Linux operating system, and thousands of other products ranging from science tools for astrophysics to online computer games. GSLIS Professor Les Gasser and U.C. Irvine Professor Walt Scacchi are leading an international team of researchers that is investigating just how these open source collectives manage "bugs" and changes in their complex, evolving artifacts. Gasser and Scacchi have just received new ITR grants from the National Science Foundation to support the project. Researchers Bill Turner and Jean-Paul Sansonnet of LIMSI at the University of Paris/Orsay are also collaborating on the project. The Mozilla organization has contributed a large base of research data, and additional data is available in the areas of astrophysics software, academic computer science research, and computer gaming. The GSLIS-led team is using qualitative methods and automated text-analysis tools to sift through approximately a half-million online "bug reports," tracing how open source developers represent, define, negotiate, and resolve problems. The results will help the researchers design the next-generation of knowledge management tools for distributed online communities.

GOALS FOR THEORETICAL CONTRIBUTIONS FROM THIS RESEARCH

Understanding and Management of Errors, Mistakes, and Problems

Errors and mistakes are clearly an important issue in many domains. They are interesting as basic sociological and organizational issues, and they are interesting for substantive reasons, because we would like to reduce and eliminate errors and mistakes in complex and high-risk systems. (cf Hughes, Bosk, Reimer, Reason, Weick). Our approach to understanding errors and mistakes treats them and analyzes them as phenomena related to social organization in general, and specifically to the social organization of work and work processes, both intra-organizational, inter-organizational, and within extended virtual communities. The links between errors/mistakes and these kinds of social organization, including the structures and processes that constitute them, haven't yet been fully articulated in any relevant literatures. We aim to make contributions here. We expect that building a more organizationally-grounded understanding of errors will lead to better processes and better tools for managing and reducing errors and their impacts.

Development of Software, and Especially Open-Source Software

Recent research has begun to underscore the need for an expanded view of software problems and how they can best be managed. We need to grow our knowledge of the processes, impacts and cost-reduction opportunities surrounding software problems 1) for developers, owners, and many other participants, 2) when problems unavoidably occur in late-life-cycle stages and in principle can't be pushed earlier, 3) when problem conditions, definitions, value, and impacts are being negotiated, 4) when the size and variety variety of participants, organizations and technologies involved is great, and 5) when the degree and frequency of change in software, infrastructure, user-base, and environment is large, and 6) when new community and institutional forms emerge for building and using systems, and 7) when software becomes a representation of community knowledge. These needs are manifested by findings such as the following:

1. Contexts of system use change, and demands unforeseeable at design time emerge. Many aspects of software problems and resolution can't in principle be pushed to earlier lifecycle stages.

2. Multiple participants in a software process perceive, define, and value appropriate and inappropriate system behavior differently. The definition of system behaviors as problems may be up for grabs; the definition of a problem may be an outcome of complex, inter-organizational negotiations that extend over the entire lifecycle of an artifact and are never settled.

3. A large variety of individual and organizational activity occurs in dealing with software problems, failures, and repairs. These include problem discovery; documentation (synopsizing, explaining, describing system configurations); problem replication; elaboration (making links to other problems, refining descriptions), triage (establishing value, allocating resources, choosing, defending choices) attribution of responsibility for faults and for repairs (e.g., to organizational units, developers/maintainers, and/or modules); repair work (understanding, building causal models, redesigning, recoding, testing); articulation work (communicating with users and other developers; getting resources, configuring tools and test suites), and so on. It is very hard at present to relate these activities, their variety, their organized character, or their infrastructure dependencies to software process models and measurements.

4. New methods of distributed software development and use such as those found in the open-source movement, and in other active, object-sharing communities such as education, publishing, music, and online gaming, radically call into question many aspects of the concepts "software", "lifecycle" and "development organization" that are central conceptual elements of existing reliability models.

5. The iterative activities of applying information systems in real settings, understanding how they work and fail, redefining what they can and should do, and changing them, can be understood a progressive, community-wide process of knowledge capture and representation. These activities of software repair are ways of encoding knowledge of uses, people, and contexts into the very structure and operation of code [Gas1998, Spo2000]. Understanding community knowledge construction through a software medium, and how software can function as a representation of its own context can provide new axes into improving software productivity and integration, and potentially new ways of structuring software for greater in-situ reliability.

Taking these leaps entails two advances: 1) a large conceptual shift in understanding how system

development and use are bound together with the richness, variety, and temporal evolution of their socio-technical contexts, based on 2) systematic empirical research on the socio-technical processes of software problems that considers these expanded aspects in significant detail and at significant scale.

We also expect to make contributions to software process extraction and support thought knowledge management technology - see the section on Knowledge Management below.

Knowledge Management

Clearly the problem and issue repositories we are studying, along with their users and processes and embedding organizations, are processes of managing knowledge. We would like to make contributions then to the literature and practice of knowledge management (KM). This at least entails:

- Constructing relevant definitions of "knowledge management" that consider the issues of purpose, representation, construction and use of large information bases of the sort we are looking into, and consideration of the "open, multi-participant, multi-perspective systems" issues in knowledge management.

- Defining a set of key intellectual, practical, and technical issues that we can add to or use as context.

We have also begun to differentiate *episodic* or *case-oriented* knowledge management and organizational memory approaches (e.g., those that support capturing experience from prior cases and applying it in new, analogous situations) from *sensemaking* knowledge management systems. The latter support accumulation of knowledge for tracking progress and understanding relationships among large collections of situations, issues, or ongoing tasks. In this project, we are primarily focused on the latter type of KM systems.

Philosophy and Technology of Representation

The critical issue for representation that has emerged in our work concerns what seems to be a new class of representational problems. It seems to us that a conventional systematic view of representation involves three entities: the thing represented, the representation, and the relationship between thing and representation. In our work so far this conception seems to break down in a number of ways:

- We have difficulty identifying the stable thing to be represented since the artifacts under construction are highly dynamic “open” systems (in the Hewitt sense) with multiple concurrent versions and uses.

- We have difficulty identifying the stable representation since all specifications and all code are also dynamic, open, concurrent multi-version objects.

- We have difficulty identifying the relationship since the thing and the representation are neither stable nor localized, and interpretations are being continually negotiated within the community.

So we aim toward developing a social (multi-perspective, multi-participant) theory (and practice) of representation that is scalable to large, distributed entities such as software systems constructed and defined by communities.

Theory of Open, Multi-Participant, Multi-Perspective Systems

In general the theory of "open, multi-participant, multi-perspective systems" is underdeveloped, yet we find them in many places. Hewitt first introduced the open systems idea (in his early Open Systems papers such as "Office are Open Systems" [C. Hewitt, 1986] and "Open systems" [C. Hewitt, Byte Magazine, 1984?]. But these ideas haven't been fully developed as a theory *per se*. So we can make contributions here. An important avenue here would be to differentiate the standard work in organization theory (e.g. Scott's book , Organizations: Rational, Natural, and Open Systems) from a new perspective which we could introduce, using the case of software development.

Extraction and Representation of Semantic and Process Information from Text and Databases

We are be developing means to extract information from textual databases, and other more structured sources. We will also be doing some organizational process extraction and dialog extraction from these data. This will entail both human and machine-based means of developing patterns in these data and of making sense of them. We are also defining a multi-layered object-based representational architecture to capture project data, to support flexible data analyses in the areas above, and to serve as an experimental theory-driven architecture for next-generation issue-management systems. The project expects to make contributions to technology for all of these areas.

Other issues

Other issues under active investigation include developing empirical definitions and accounts of the “fit” between information systems and their contexts of use, and understanding dimensions of information quality in large dynamic repositories: how can quality be measured, how are quality measurements shaped and contextualized by different genres of information, and how can we simulate and predict the dynamics of information quality in large information bases over time.

More information can be found through the GSLIS Information Systems Research Lab, which is housing the project, at http://www.isrl.uiuc.edu.