Seeking the Source
Software Source Code as a Social and Technical Artifact

Cleidson de Souza, Jon Froehlich, and Paul Dourish
Overview

- Uncover structures of software projects
  - Uniform treatment of artifacts and activities
  - Use code analysis and development metadata to derive views of social structure
- Qualitative study of open source software
Background

- Software architecture defines a relationship between components
  - Functional/technical influences
  - Social and organizational influences also at play
- Conway [1968]
  - “The structure of a system mirrors the structure of the organization that designed it”
- Also, Parnas’ [1972]
  - Principle of information hiding
• What happens when there is no “formal” organization
  – OSS might be an example…

• We created a tool to investigate:
  – how the relationships between software modules expose one view of the underlying social structure
Outline

• Augur, A Tool
• Visualizing Software Development
• Evolutionary Patterns
• Conclusion & Future Work
Augur, A Tool
Augur

• Visualization system for CM repositories of source code
  – Attempts to unify views of “activities” and “artifacts”
  – Contains multiple, dynamic views of data
Augur’s Data Source

• Configuration Management (CM) Systems
  – Repository for both source code and meta-data about development
    • Author information
    • Checkin information
    • Use this as a basis for analysis
  – Augur connects to existing CVS servers with no additional overhead
Simple Analysis

• Analyze which developers in the system work on which files or lines
  – Broad temporal patterns
    • e.g. clear distinctions between workday and weekend, individual contribution patterns, etc.
  – Broad collaborative patterns
    • e.g. certain authors only work alone, certain files contain many authors
Static Analysis

- Static analysis enables us to explore richer, more meaningful relationships
  - Line types
  - Containment structures
- Dependency relationships begin to emerge
  - Object extension
  - Interface implementation and extension
Dependency Analysis

• Augur extended to conduct Call Graph Analysis
  – A window into the dynamic structures of the code
  • As Parnas and Conway suggest, this structure influences the coordination of the work
int main()
{
    int temp = GetTemp();
    double fahr = ConvertTemp(temp);
    print("Temperature: \" + fahr);
}

An Example

main() -> GetTemp()
main() -> ConvertTemp(x)
main() -> print(str)
int main() {
    int temp = GetTemp();
    double fahr = ConvertTemp(temp);
    print("Temperature: "+fahr);
}

We ignore library functions (no metadata available)
Types of Projects

• Network relationships between developers
  – As derived from code dependencies

• Many different types, we’ll focus on three emergent patterns
  – Centralized
  – Densely Networked
  – Core and Periphery
Centralized

Project: iReport2
Host: Sourceforge

Directionality is important
Centralized

Project: iReport2
Host: Sourceforge

Project: jBoss
Host: Sourceforge

Project: Jakarta-Lucene
Host: Apache

Project: FreeNet
Host: SourceForge
Centralized

Project: Jakarta-Tomcat
Host: Apache
Project: Azureus
Host: Sourceforge
Densely Networked

Project: Azureus
Host: Sourceforge

Project: Megamek
Host: Sourceforge

Project: Apache WS-AXIS
Host: Apache

Project: Apache-LG4j
Host: Apache
Core & Periphery

Project: jBOSS
Host: Sourceforge
Core & Periphery

Project: WS-Axis
Host: Apache
Quick digression…
Useful to see both artifact *and* author in the same frame
– The connection between the two becomes explicit – Bipartite Graphs

Example:
04-23-2000
One Day Old
07-23-2000
Three Months Old
04-23-2001
One Year Old
10-23-2002
2.5 Years Old
Conclusion & Future Work
Conclusion

- We explored emergent social and technical structures through artifacts
- Our preliminary empirical examinations have shown that:
  - Social patterns can be revealed through CM analysis combined with artifact analysis
    - E.g. Certain modules and certain “authors” become obligatory passage points in the system
  - Suggest that tools can be built to support both technical and social structures in the system
Future Work

• Deeper investigations…
  – Interviews with project developers (compare/contrast)
• Automatic recognition…
  – Similar to Soylent e-mail system [Fisher 2004]
• Is Augur a useful tool to OSS research?
  – Is there value to the community?
• Investigate different communities
  – Commercial systems
  – Co-located development teams
• Integrate other sources of information
  – E.g. mine bug-tracking systems [Crowston 2004]
Acknowledgements

- My two co-authors:
  - Cleidson de Souza
    - Departamento de Informatica
    Universidad Federal do Para
  - Paul Dourish
    - Donald Bren School of Information and Computer Sciences
    University of California, Irvine

- Jeffrey Heer for Prefuse
- Danyel Fisher for JUNG
Questions?
Jon Froehlich
jonfroehlich@gmail.com