Organizing the Technical Debt Landscape

Clemente Izurieta (Montana State University)
Antonio Vetro (Fraunhofer, Politecnico di Torino)
Nico Zazworka (Fraunhofer)
Yuanfang Cai (Drexel University)
Carolyn Seaman (UMBC)
Forrest Shull (Fraunhofer)
How Technical Debt Can be

* Organized?
* Visualized?
* Identified?
* Managed?
Existing Approaches

- Modularity Violations (tool: CLIO)
- Design Patterns and Grime Buildup
- Code Smells (tool: CodeVizard)
- ASA issues (tool: FindBugs)
Research Questions

* What are the overlaps and gaps among existing techniques?
  * Which pairs of these techniques always report problems in the same set of components?

* To what extent do existing techniques help in identifying TD?
  * Which techniques can detect components that are defective and/or change-prone?
Technical Debt Landscape

Stage 1
- Set of existing techniques and tools
- War stories of practitioners
- Gap
- Technique 1
- Technique 2
- Technique 3

Stage 2
- Industrial Software Apps
- Research Design 1
- Research Design 2
- Research Design 3
- Open Source Software Apps
- Gap
- Technique 1
- Technique 2
- Technique 3
- Technique 4

Draft of Technical Debt Landscape
- Refined Technical Debt Landscape

New emerging techniques and war stories
Research Design

* Design 1: Direct comparisons of TD identification techniques

* Design 2: Evaluating TD identification techniques for identifying real debt

* Design 3: Evaluating the relationship between types of TD and future maintenance:
Different TD techniques point to different classes and therefore to different problems

Dispersed coupling, god classes, modularity violations and multithread correctness issues are located in classes with higher defect-proneness

Modularity violations are strongly associated with change proneness.
Future Work

- Find and fill gaps

- Investigate quality factors other than defect and change proneness, such as productivity and maintenance difficulties