Technical debt conceptualizes the tradeoff between the short-term benefits of rapid delivery and the long-term value of developing a software system that is easy to evolve, modify, repair, and sustain. In this work, we are developing tools that integrate data from multiple, commonly available sources to pinpoint problematic design decisions and quantify their consequences in a repeatable and reliable way for uncovering technical debt.

**Technical Debt Analysis Workflow**

Our research approach includes:
- Automatically extract potential TD issues from issue trackers using data mining techniques
- Enrich these clusters by incorporating information from code analyses and software repositories.
- Rank each TD cluster in terms of accumulating rework to date (e.g., total change and bug churn, number of related issues).

**Categorization and classification:**
- Created a classifier that utilizes n-gram feature engineering and gradient boosting
- Our data shows that developers do identify long-lasting design issues as technical debt
- We are in the process of improving the classifier to help identify technical debt.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>Test Count</th>
<th>Training Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.60</td>
<td>0.42</td>
<td>0.50</td>
<td>510</td>
<td>157</td>
</tr>
<tr>
<td>2</td>
<td>0.71</td>
<td>0.32</td>
<td>0.44</td>
<td>411</td>
<td>256</td>
</tr>
<tr>
<td>3</td>
<td>0.77</td>
<td>0.42</td>
<td>0.54</td>
<td>311</td>
<td>356</td>
</tr>
<tr>
<td>4</td>
<td>0.74</td>
<td>0.70</td>
<td>0.72</td>
<td>161</td>
<td>506</td>
</tr>
</tbody>
</table>

**Results:**
- Analysis across 8 projects reveals that our approach highlights hidden issues.
- Our models are improving, especially in recall (more true debt is identified).

**TD clusters**

Design problems, frequently the result of optimizing for delivery speed, are a critical part of long-term software costs. Detecting such design issues with tool support is a high priority for technical debt analysis. We developed an approach where existing static analysis rules are mapped to design issues. The goal of the analysis is to help teams focus on rework causing issues earlier in the lifecycle.

Government acquisition managers need capabilities to assess what kind of technical debt is created throughout the software lifecycle. The SEI team has been a pioneer in advancing the research agenda in this regard. Our ongoing work is focused on creating analytical tools towards an integrated software analytics approach.