IBM

An Enterprise Perspective on Technical Debt

Your flight attendant today

Tim Klinger, **Peri Tarr**, Patrick Wagstrom, and Clay Williams

(Technical) Debt as an Enterprise Proposition

- **Debt**: An amount of money borrowed by one party from another, usually paid back by a specified date with interest.
  - *It's not debt if you don't have to pay it back, or if you don't have to pay interest*
- **Leverage**: Use borrowed capital for (an investment), expecting the profits made to be greater than the interest payable.
- **Our questions**:
  - Is technical debt due to bad decisions, or is it treated as leverage?
    - *Incur debt to pursue options organization couldn't otherwise afford?*
  - Financial risk and value are managed in aggregate. Is technical debt?
    - *Who is responsible for deciding whether to incur, pay off technical debt?*

Technical Debt ≠ Low Quality

Enterprise Software Delivery

- **Numerous technical and non-technical stakeholders**
  - Testing, strategy, legal, marketing, sales, support, …
- **Product delivery strongly affected by non-technical concerns**
  - Regulatory requirements
  - Meeting customer obligations and needs
  - Capacity to provide support
- **Product delivery managed in aggregates**
  - Portfolio, brand, geography, …
  - *Global maximization may lead to local minima*

Enterprise Technical Debt

- **Acquiring technical debt may offer strategic, non-technical benefits**
  - Meeting critical delivery date may ensure product success
  - Taking on debt in one product may give portfolio “upsell” opportunities
  - *When to incur technical debt to gain leverage for investment elsewhere?*
- **After delivery, enterprise must decide when to repay (some) technical debt**
  - Cost of remediation (principal) vs. cost of maintenance (interest)
  - *When is the right time to pay down the debt?*

Our Informal Technical Debt Hypotheses

- Technical architects are aware of, and willing to choose to incur, technical debt
- Decisions to incur technical debt are strongly influenced by non-technical stakeholders
- Technical debt is not perceived as a persistent debt, but rather a debt that has a good probability of having to be paid back
  - *It's not debt if you don't have to pay it back, or if you don't have to pay interest*

Understanding the Enterprise Perspective

- **Conducted a brief qualitative study**
- **Interviewed four technical architects at IBM**
  - Usual limitations apply: small sample size, same firm, etc.
- **Sought out architects working on a variety of projects**
  - IBM internal projects
  - IBM-originated products
  - Acquisitions
  - External projects/products

The Subjects

- Subject A: Architect for a new product that fits into an existing product line at IBM
- Subject B: Experienced architect on many different product lines. Moved into a recently acquired tool and bring it toward integration with existing tools.
- Subject C: Established architect who worked on a project before it was acquired by IBM and was responsible for integration.
- Subject D: A very experienced architect who has worked on many large scale and prominent systems inside and outside of IBM.

Subject A: New IBM Product in Existing Portfolio

- **Technical decision: Which API to support?**
  - Choice 1: Support a new standard API
    - Offers numerous technical improvements over current API
    - Would be required in future versions of the portfolio
  - Choice 2: Implement new features using old API
    - Customer dissatisfaction a real danger
    - Will necessitate migration to new API (technical debt)
- **Choice 1 was technically superior, but the architect had to take Choice 2**
  - Choice 1 required assistance from a partner team, which declined
    - *Supporting new API could endanger advertised ship date*
  - Other products with which this product needed to integrate still used old API
  - *No recourse*

Decision by external and largely unrelated stakeholder caused Subject A to incur technical debt

Subject B: Experienced IBM Architect Shepherding New Acquisition

- **Prior to acquisition, customer needs drove prioritization of requirements**
  - Team had made commitments to customers for new features
- **After acquisition, some low-priority features (e.g., internationalization, integration) became much more important**
  - Compatibility with larger portfolio
  - Compatibility with IBM's more global presence, wider distribution of software
- **Regulatory requirements necessitated reprioritization of features such as internationalization to very high**
  - Delayed some customer commitments
  - Removed many planned new features

Being acquired induced technical debt

Subject C: Architect from IBM-Acquired Product

- **Prior to acquisition, customer needs drove prioritization of requirements**
  - Team had made commitments to customers for new features
  - Integration with other products was not a high priority for customers
- **After acquisition, integration architecture was a key priority**
  - Product was purchased to be integrated into an existing portfolio
  - Worked with IBM staff to develop appropriate integration approach
    - The best integration approach had architectural implications elsewhere
    - "Elsewhere" was understaffed and overworked
    - Decision: Go with existing, "good enough" mechanism and rewrite acquired and existing products to new integration architecture later…maybe?
- **Is it technical debt? No one was sure.**
  - Subject C and the existing staff were very aware that debt might have been acquired, but they weren't certain whether it would have to be repaid

Unanticipated requirements induced technical debt…or did they?

Subject D: Experienced and Broadly Knowledgeable Architect

- *Provided numerous examples from outside of IBM*
- **Very successful external desktop product had accrued lots of technical debt**
  - Team was unable to anticipate some of the ways people used the program
  - Existing code base was increasingly impairing ability to deliver new features or perform adequately *(high interest)*
- **Time to repay the debt. Or not.**
  - A team was put together to rewrite the product from scratch
  - They learned they no longer fully understood the customer requirements
    - Many customers relied on peculiarities of product's architecture, features
    - Paying off technical debt would result in loss of customers
  - Without a clear understanding of hidden customer requirements, it was impossible for the team to remediate all of the product's debt

Paying off technical debt had high probability of inducing unacceptable financial loss

IBM

Subject D: Experienced and Broadly Knowledgeable Architect

- **Successful and profitable web firm**
  - Moderate-sized code base, excellent engineers
  - Profits from leveraging technical debt allow them to hire enough staff to periodically rewrite the entire system
  - *This model has sustained them for more than five years*

Successful, intelligent, and strategic acquisition and leverage of technical debt

Common Experiences of Subjects

- **Induced and unintentional debt are significant challenges**
  - Architects intentionally make decisions about technical debt
  - Architects have debt foisted off onto them unwittingly, induced by
    - Other stakeholders in the project or across the portfolio
    - Situational evolution ⇒ unanticipated/architecturally significant requirements
- **Decisions are managed in an ad-hoc manner**
  - Often made without formalization or quantification of impact of decisions
  - Architects consider cost of repayment. Enterprises do not appear to do so.
  - *Financial costs are obvious and revisited, but costs of technical debt are not*
- **Stakeholders lack effective ways to communicate, reason about debt**
  - Decisions to incur debt often made by stakeholders who
    - Do not fully comprehend ramifications on those who will have to service it
    - Are optimizing locally, rather than globally
  - Major communication gap between stakeholders
    - Financial or customer related vs. technical stakeholders
    - No channels, vocabulary to explain technical debt to other stakeholders

IBM

Findings and Recommendations

- **Collectives matter for both products and people**
  - Enterprises optimize benefits across their portfolio of investments
    - *Decisions may cascade to have negative impact on individual projects*
  - BUT, enterprises cannot currently optimize globally over technical debt
    - *Insufficient global view, communication channels*
    - *Those stakeholders who make decisions across portfolios are not technical*
- **The ability to assess debt matters**
  - Decisions regarding technical debt were rarely, if ever, quantified
  - BUT, it is hard to quantify debt where the principal and interest—even the existence of the debt—can fluctuate over time
    - *Technical debt is relative to goals, requirements, stakeholders, ecosystem*
    - *Interest is hard to quantify; multiple stakeholders may pay it (support, development, sales, …)*
- **Bridges across enterprise gaps matter**
  - Stakeholders choosing to incur, repay technical debt do not service it
  - Technical architects could not communicate risks to decision makers
  - Lack of quantification leaves people unable to reason about ramifications

Last Thoughts

- **Not everything that costs money in SWD is debt. To qualify, must quantify:**
  - Principal (cost to get rid of debt)
  - Interest (cost to carry debt)
- **Quality issues are a subset of the technical debt story**
  - Other kinds include architectural debt
  - Not all quality problems are part of this subset
    - E.g., complex module that never gets touched incurs no interest
  - *Debt is only important when it is "active"*
- **Hard to know whether some problems will become "active" as debt and what they will cost if they do become active**
  - Reasoning based on uncertainty is essential to any realistic approach to debt

IBM

Issues for Future Study

- Opportunity to apply concepts from finance to thinking about technical debt
- Tools from decision science may provide assistance for managing technical debt
- Technical debt is often invisible to non-technical stakeholders
  - Need a way to share, quantify, compare this information
    - Relative measures may work better than absolute (debt points, rating, …)
- Dynamic forces may result in the need to incur unintentional technical debt

QUESTIONS/DISCUSSION