Global SCRUM GATHERING®

Slowing down to Speed up: Agile & Technical Debt

Taghi Paksima

hetras GmbH

About me!

In IT/Software Industry since late 90s

Microsoft, Software Engineer, 2008-14

hetras GmbH, Lead Developer, since 2014

Agile practitioner since 2008, avid Agile evangelist since 2013

Founder, Organiser: ‘Agile Munich’

What is Technical Debt?

"**Technical debt** is a metaphor referring to the consequences of corners being cut throughout a software project or poor software architecture and software development within a codebase." Wikipedia

"Both the intentional and un-intentional violations of good architectural and coding practices." Steve McConnell

All the mess you leave behind when you think you are done

Dire Consequences

Technical Debt:

- Creates financial burden
- Kills productivity
- Makes maintenance difficult
- Or can even bring a project to a halt

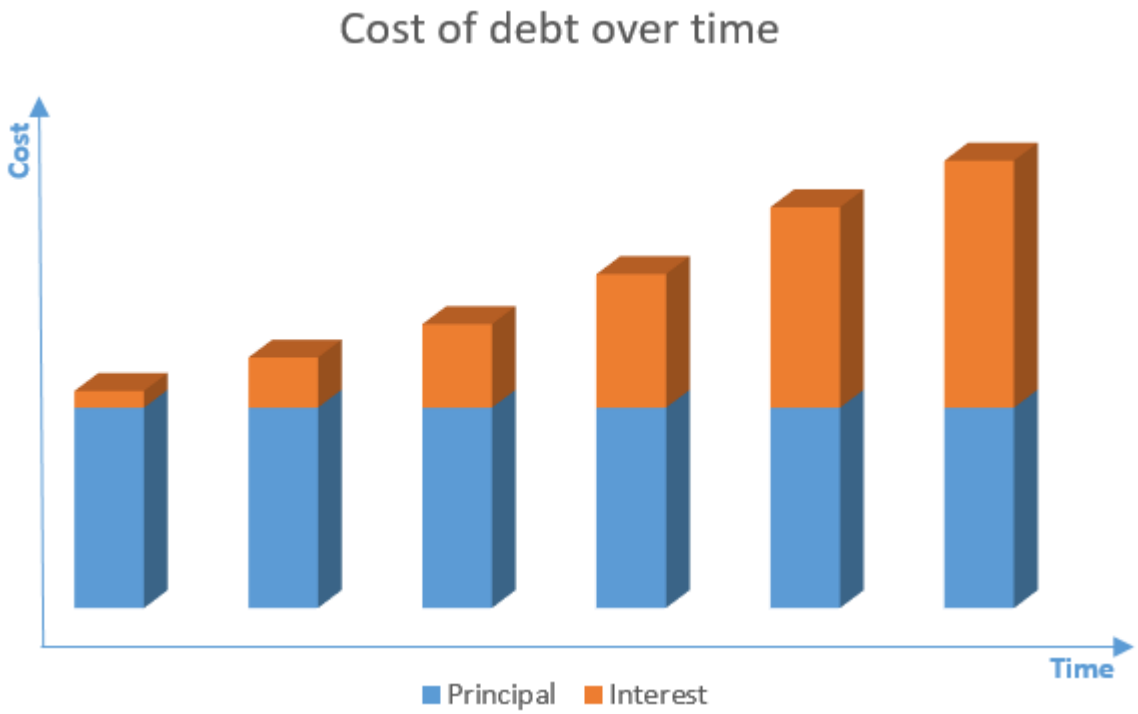
Cost of Debt

Principal:

The actual cost of fixing the shortcuts taken

Interest:

The recurring cost of delaying the payback

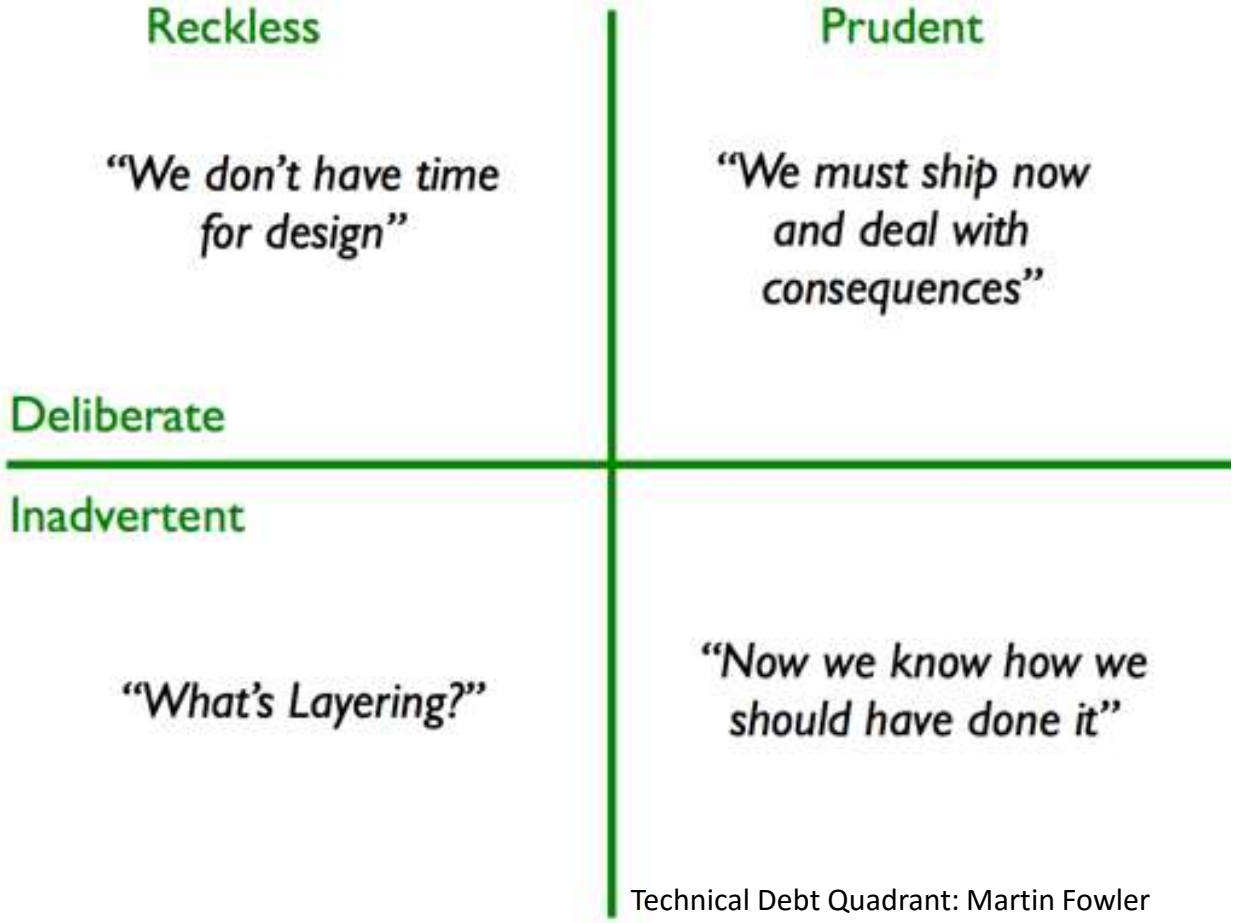


Technical Debt: The Causes

Business pressure
Lack of tests
Agile Misconception
Lack of processes & engineering practices
Lack of knowledge
Lack of planning
Lack of collaboration/communication
Parallel development
Delayed refactoring

But, debt is inevitable

Technical Debt: The Reasons



Technical Debt Quadrant: Martin Fowler

The Good, the Bad, and the Ugly

- Conscious decision to facilitate achieving a goal, but you'd better have a payback plan!
- Tech debt increases entropy: That's what keeps the CTO awake at nights
- Regression of the Tribal Culture:

  From code churn to developer churn

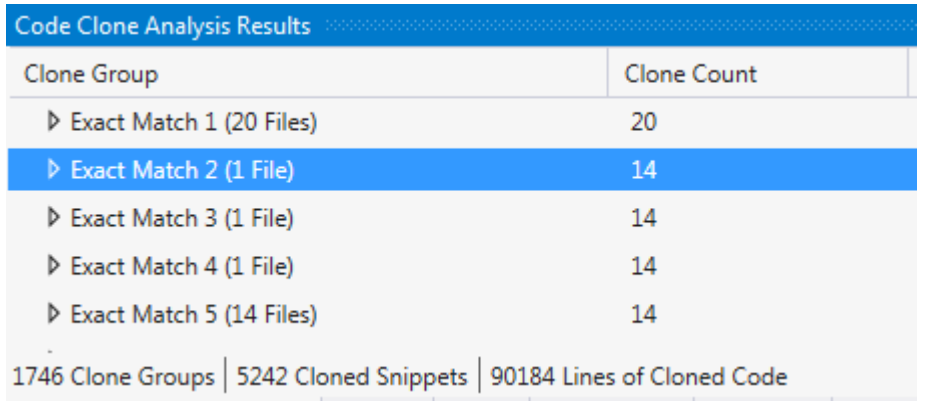
The Cost of Debt

$3.61 Average technical debt per line of code (over $1 million per application) [CAST Research Labs, 2013]

50% Of an application cost throughout its life cycle is spent on maintenance and support [Gartner, 2013]

Detecting Debt

- Low Test Coverage
- Code Quality Metrics
- Defect Density
- Code Smell
- Code Churn
- Declining Velocity*
- Aging software without refactoring



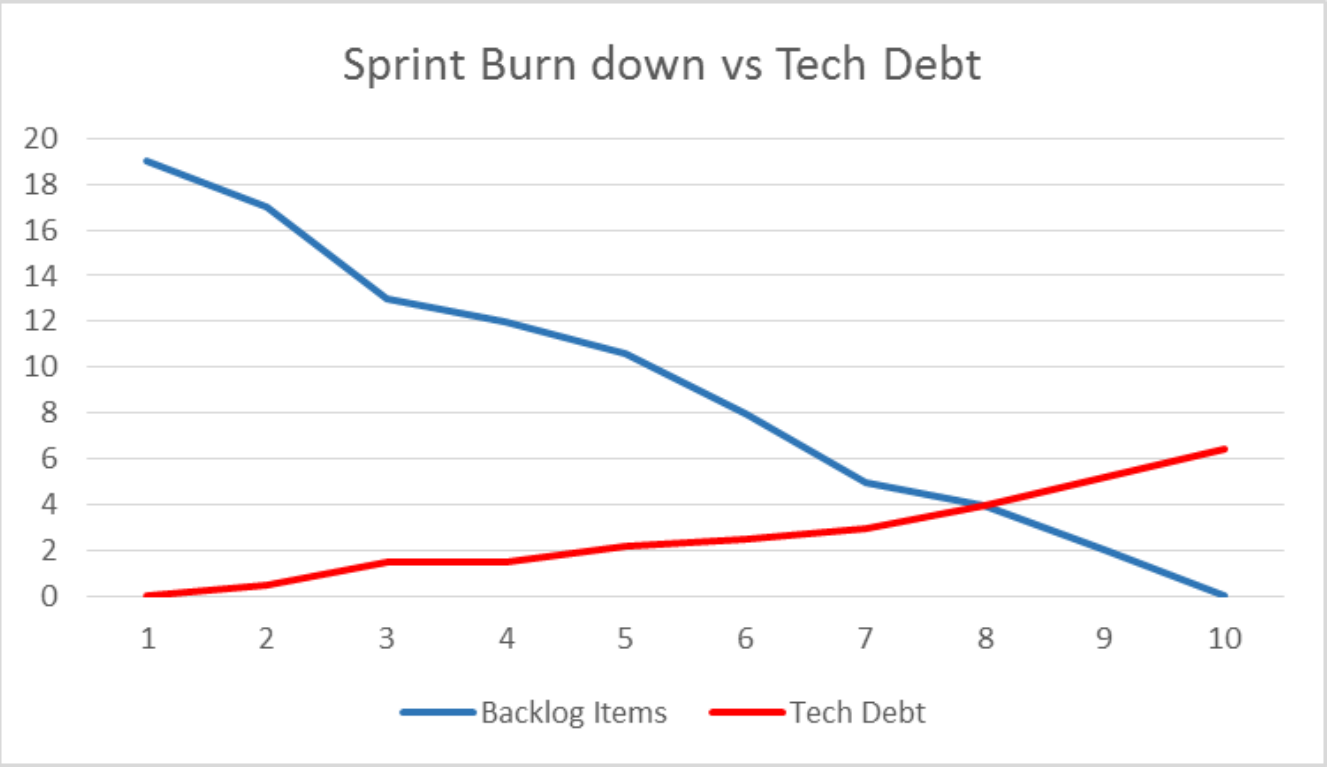
Detecting Debt: Tools

Static Analysis Tools
nDepend
SQALE (SonarCube)

Definition of Done

- Define what *Done* actually means for your team
  - Code Complete
  - Test Complete
- Anything deferred, as against DoD = Tech Debt
- Train the POs on the implications of debt
- Teams self-organise to manage Tech Debt
- DoD will be evolving over time

Almost Done!


Sprint Burn down vs Tech Debt
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
Backlog Items
Tech Debt

Make Debt Visible

Make sure tech debt is tracked in the project backlogs

Highlight tech debt state in Sprint reviews, management reports, etc

Visually represent debt metrics to the engineering team

Technical Debt Metrics

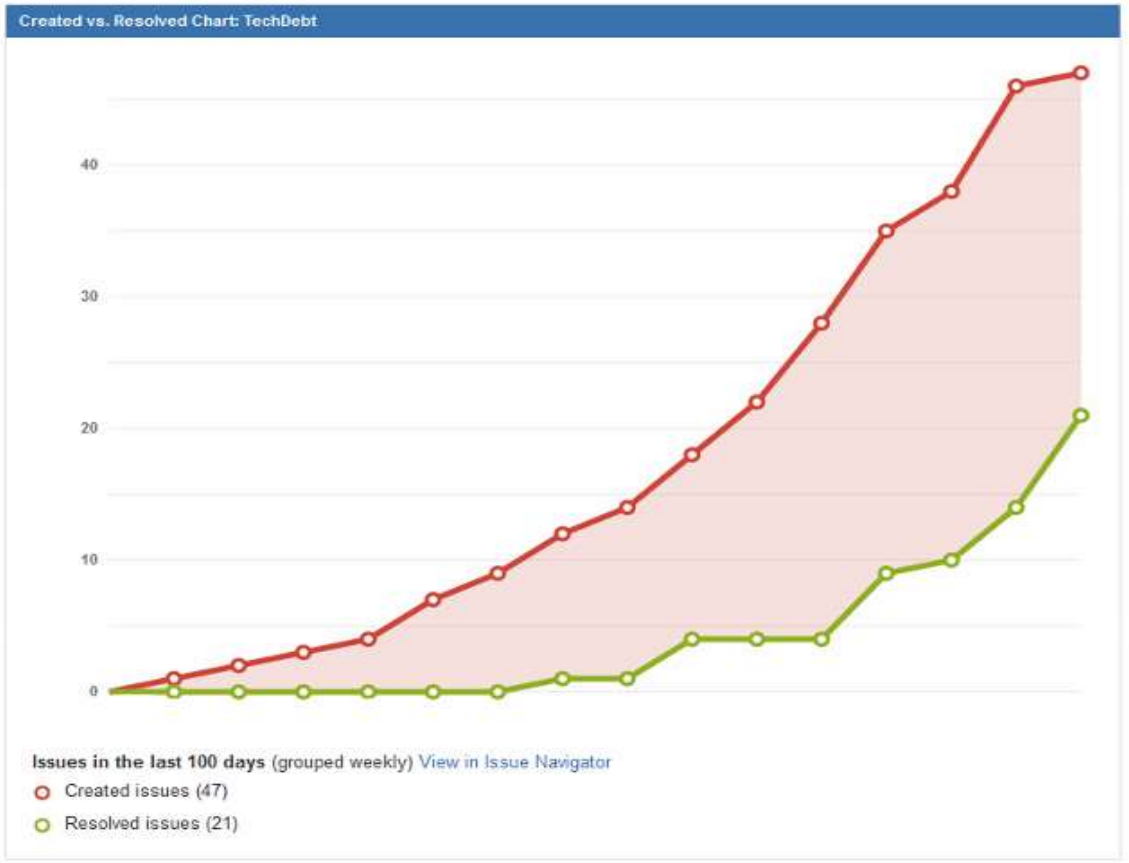
KISS Metrics

Debt ratio per iteration

Debt resolution rate

Pro Metrics

SQALE Indices



Tackling Technical Debt

- Sweep it under the carpet!
- Refactor/Pay-as-you-go
- Buffering: Allocate part of the team capacity in each iteration to reducing technical debt
- Debt Bash: Dedicate one iteration to fixing technical debt
- Dedicated Team: One team focuses on a backlog of top priority debt items

Debt Relief Recipe

Define, detect & track technical debt: Starting from DoD

Make debt visible: Communicate technical debt state regularly

Minimise new debt: quality gates throughout ALM

Train & empower teams to determine their pay back strategy

Agile vs. Fast

Agility in Agile is more about sustainable pace and lateral nimbleness than hasty development.

From Agile Principles:

Continuous attention to technical excellence and good design enhances agility.

Teams should be able to maintain a constant pace indefinitely.

Slow down to Speed up

Taghi Paksima

http://linkedin.com/in/paksima
http://agilemunich.org/
taghi@agilemunich.org
@TaghiPaksima

I'm looking forward to your feedback and questions