Titanium Sponsors

okta

Headspring

Platinum Sponsors

PAIGE TECHNOLOGIES
INTELLIGENT PAIRING. PERPETUAL SUCCESS.

EVENT STORE

Homebase

Dimensional Innovations

WellSky

DevExpress

Progress Telerik

AxonIQ

Algorand

smg service management group

Red Hat

NAIC
National Association of Insurance Commissioners

NIPR
NATIONAL INSURANCE PRODUCER REGISTRY

SAUCELABS

Veterans United
Home Loans

ROCKET Companies

trility

JOHN DEERE

ascend LEARNING

Mattermost

Gold Sponsors

touchnet
A Global Payments Company

Advantage Tech
IT Staffing & Recruiting Services

pk prokarma

ORION

TEAM Software

Netsmart

BUILDERTREND

Cerner

VMLY&R

ARTISAN TECHNOLOGY GROUP

Leggett & Platt

QUEST ANALYTICS

MOONSHOT INNOVATIONS

Jennie.ocken.org

A Product Perspective on Tech Debt

CONSUMING ENDANGERED PACHYDERMS

PERSPECTIVE

- Different sizes of company
  30 <-> 85,000 (~800 in IT)
- Different Business Pressures
  SaaS product <-> Internal Business Applications
- Different technical challenges
  Mainframe forklift <-> Startup
  Monolith <-> Microservices
- Different Roles
  Sales/Marketing <-> IT Department Head
  Single team embed <-> Head of Product

START WITH UNDERSTANDING

People create the best solution they can based on the information, resources, and constraints they had at the time.

- Building software is hard. Building products that solve customer problems is hard.
- Assumptions, whether good or bad, are required in every project.
- Things we know today change tomorrow.
- We learn things constantly.
- Not everything is under our control.

DEFINITION

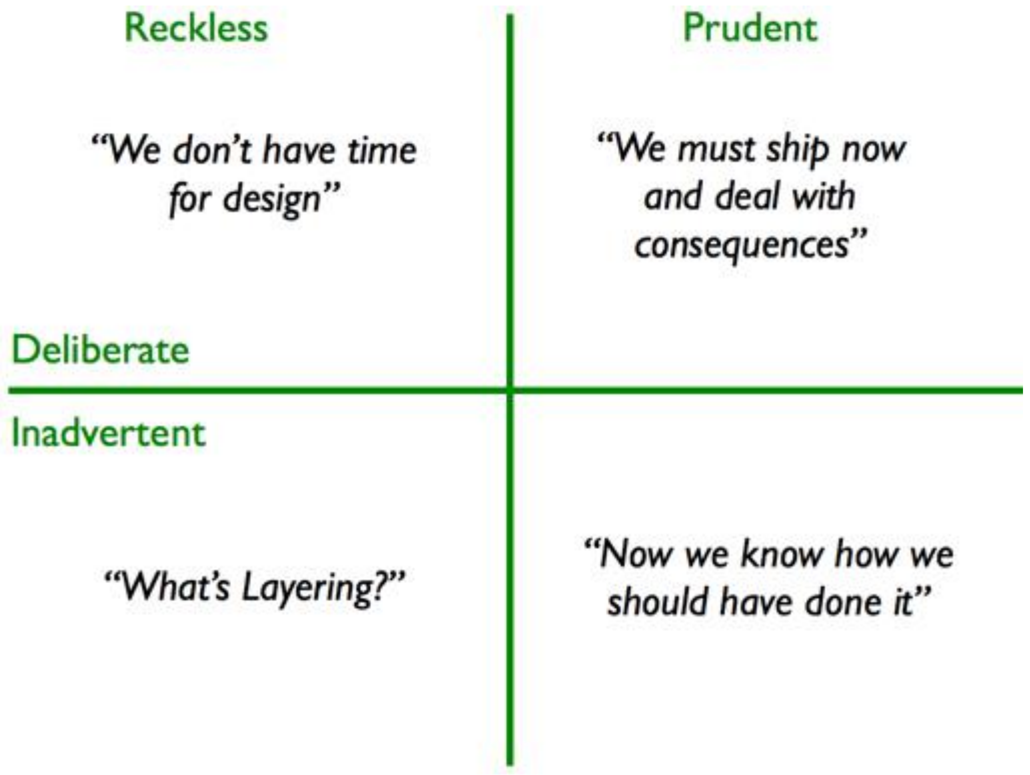
WHAT IS DEBT

- Implied cost plus interest accrued by delaying a payment

- Why delay payment?
  - Uncertainty about the right course
  - Complexity or too high a cost for right now
  - Meeting deadlines

- Debt isn't bad in and of itself, but it can become crushing over time if not paid down
  - Debts should be consolidated when possible
  - Highest interest debt should be paid down first
  - Eat the elephant one bite at a time

TECHNICAL DEBT QUADRANT – MARTIN FOWLER


Reckless
Prudent
"We don't have time for design"
"We must ship now and deal with consequences"
Deliberate
Inadvertent
"What's Layering?"
"Now we know how we should have done it"

https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

TOWARDS AN ONTOLOGY OF TERMS ON TECHNICAL DEBT

– Software Engineering Institute

- Architecture Debt
- Build Debt
- Code Debt
- Defect Debt
- Design Debt
- Documentation Debt
- Infrastructure Debt
- People Debt
- Process Debt
- Requirement Debt
- Service Debt
- Test Automation Debt
- Test Debt

https://www.researchgate.net/publication/286010286_Towards_an_Ontology_of_Terms_on_Technical_Debt

3 MAIN TYPES OF TECHNICAL DEBT AND HOW TO MANAGE THEM

– Dag Liodden @FirstMark

Deliberate

“We sometimes deliberately incur tech debt to reduce time to market”

Track and plan into the backlog

Accidental / outdated design

“As systems evolve and requirement change, you might come to realize that your design is flawed, or that new functionality has become difficult and slow to implement”

Set aside time to evaluate and refactor

Bit Rot

“A component or system slowly devolves into unnecessary complexity through lots of incremental changes”

Continuous clean up of bad code, improvement on the design, and taking the time to understand the design of the system

https://hackernoon.com/there-are-3-main-types-of-technical-debt-heres-how-to-manage-them-4a3328a4c50c

DEBT IS MORE THAN TECHNICAL

SOURCES OF DEBT

Tech Debt

Prioritizing Delivery over Implementation

Product Debt

Prioritizing Delivery over Adoption

Intentional
Actively deciding to put something off

Accidental
Things change or wrong decisions previously made

Market/Industry
External market pressures or industry standards

Intentional:

Tech Debt: Neglect
- Major runtime/platform updates
- Deprecation of 3rd party dependencies
- Neglecting security standards
- Aging/failing test suites

Product Debt: Market Opportunities
- Testing a new market segment
- Competition driven roadmap

Accidental:

Tech Debt: Technology Changes
- Committing to a tech stack that becomes unsupported
- Scope creep/shifting requirements creating unexpected complexity

Product Debt: Market Threats
- Category Killer competitive threat
- Changes in regulation or compliance
- Loss of competitive “moat”

Individual
Internal people or teams

Intentional:

Tech Debt: Band-aiding
- Culture of firefighting
- Changes made in isolation, outsourcing
- Lacking refactoring, coding standards
- Over architecting

Product Debt: Organizational alignment
- Deadline driven development
- Incomplete POC or Spike
- Shiny object syndrome
- Business making solution decisions

Accidental:

Tech Debt: Application Vision
- Lack of technical leadership / architectural
- Lack of experience, collaboration, or reviews, insufficient space for learning
- Insufficient tests, monitoring, alerting

Product Debt: Product Vision
- Loss of product vision and user intimacy
- Requirements not defined or understood up front
- Analysis Paralysis

FINDING SOLUTIONS BASED ON KIND OF DEBT

TECHNOLOGY CHANGES

ACCIDENTAL/MARKET TECH DEBT

ROOT CAUSES

- Committing to a tech stack that becomes unsupported
- Scope creep/shifting requirements creating unexpected complexity

SOLUTIONS

- Lessons
  - Choose small, internal apps for testing
  - Pivot quickly
  - Debate the tech stack and agree as a team
- Solutions
  - Regularly review architecture and stack to see if there are creeping problems
  - Capture stories to refactor or change stack
  - Technical leadership gap analysis
  - Funnel tech stack tests into 20% time POCs, commit to hardening before bringing into main application

MARKET THREATS

Accidental/Market Product Debt

ROOT CAUSES

- Category Killer competitive threat
- Changes in regulation or compliance
- Loss of competitive "moat"

SOLUTIONS

- Lessons
  - Macroeconomic situations matter
  - Pivoting requires actually committing to a path
- Solutions
  - SWOT
  - Find your moat and unfair advantage
  - Shrink and protect
  - Diversify

APPLICATION VISION

Accidental/Individual Tech Debt

ROOT CAUSES

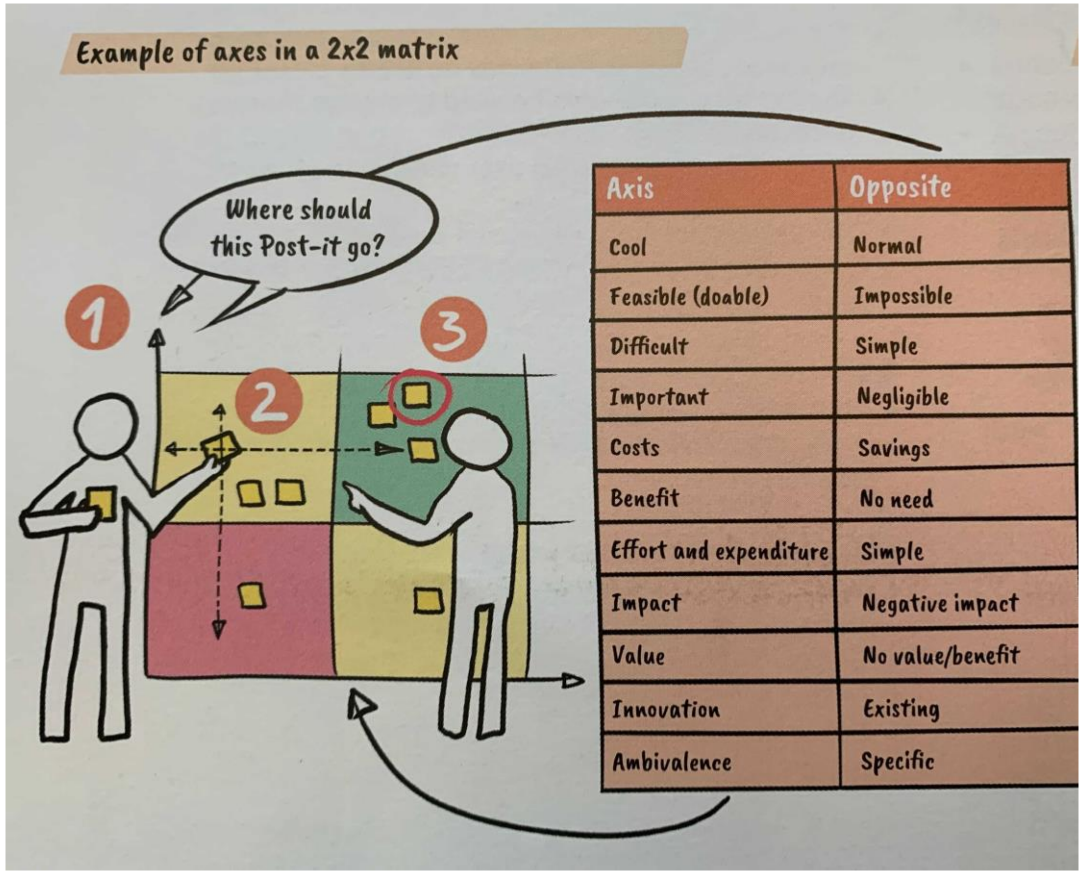
- Lack of technical leadership / architectural view
- Lack of experience, collaboration, or reviews, insufficient space for learning
- Insufficient tests, monitoring, alerting

SOLUTIONS

- Lessons
  - Everyone has blind spots
  - Applications grow based on what view we have of them (zoom out)
- Solutions
  - Mind the Gap
  - Debt Backlog

PRIORITIZING A DEBT BACKLOG

- Identify the biggest debt areas
  - Places where "code joy" diminishes
  - Features that are unused
  - Dissatisfied clients and win/loss calls
  - Proposals
- Identify the type of debt
- Quantify the pain if possible
  - Maintenance costs
  - Loss of potential markets
- Quantify the difficulty to repair
  - Level of effort
  - Complexity / amount of commitment
- Plan alongside Features


Example of axes in a 2x2 matrix
Where should this Post-it go?
1
2
3

Axis	Opposite
Cool	Normal
Feasible (doable)	Impossible
Difficult	Simple
Important	Negligible
Costs	Savings
Benefit	No need
Effort and expenditure	Simple
Impact	Negative impact
Value	No value/benefit
Innovation	Existing
Ambivalence	Specific

The Design Thinking Toolbox pg 156

PRODUCT VISION

Accidental/Individual Product Debt

ROOT CAUSES

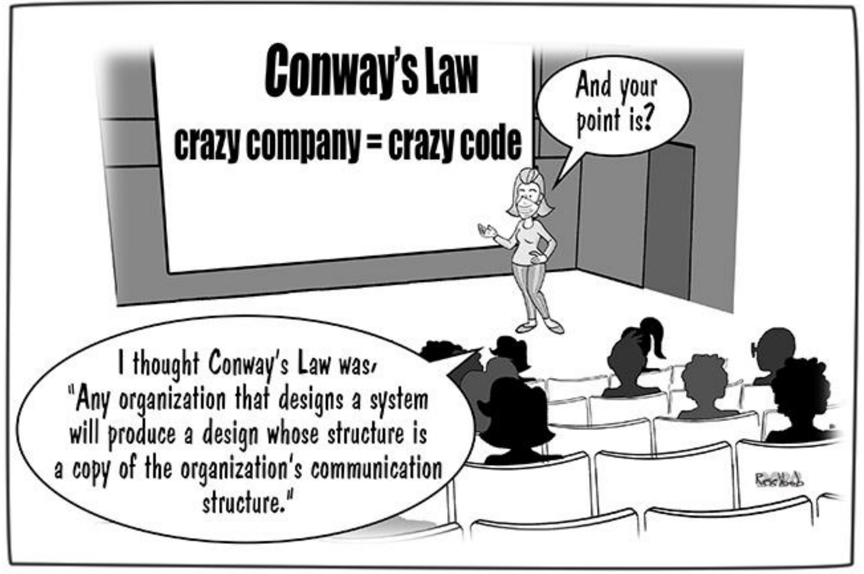
- Loss of product vision and user intimacy
- Requirements not defined or understood up front
- Analysis Paralysis

SOLUTIONS

- Lessons
  - "Global Domination" is an actionable vision.
  - Commit, observe, iterate
  - "What will our application look like in a year?" and We create _______ for __________.
- Solutions
  - Identify Market Segments and User Personas, then talk to real people to validate
  - Monitor Promises in Progress
  - Build in metrics and OKRs/KPIs

NEGLECT

Intentional/Market Tech Debt


Conway's Law
crazy company = crazy code
And your point is?
I thought Conway's Law was, "Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."
Roelbob

By Roelbob at https://devops.com/conways-law/

ROOT CAUSES

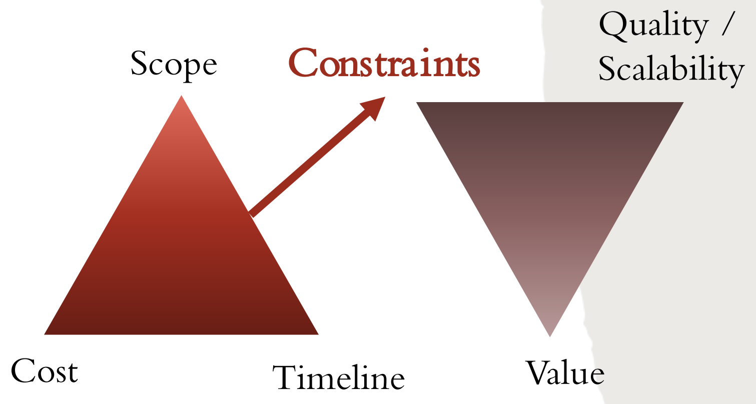
- Major runtime/platform updates
- Deprecation of 3rd party dependencies
- Neglecting security standards
- Aging/failing test suites

SOLUTIONS

- Lessons
  - False positive blindness
  - Security audits and training requires organizational desire to solve the problems discovered
  - It is part of the job to monitor industry trends and standards
- Solutions
  - Audits with action plans
  - Safe space to raise issues
  - Conway's Law vs Express Ownership

MARKET OPPORTUNITIES

INTENTIONAL/MARKET PRODUCT DEBT



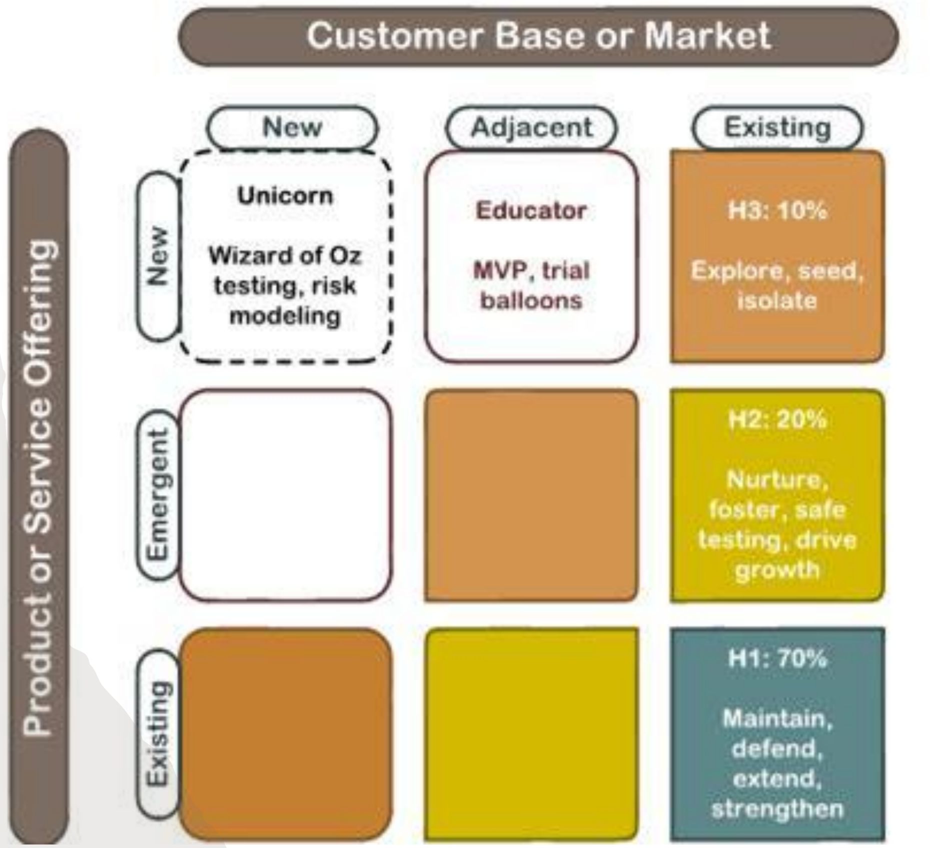
ROOT CAUSES

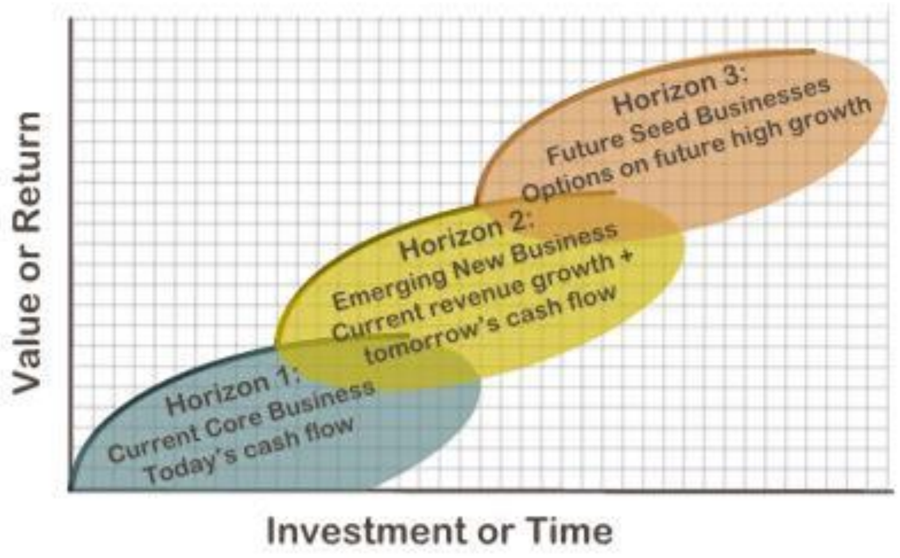
- Testing a new market segment
- Competition driven roadmap

SOLUTIONS

- Lessons
  - Know how extended you can become
  - Actively derisk
  - Document your "whys" and revaluate regularly
  - Project vs Product thinking
- Solutions
  - Design Thinking
  - Horizons Planning
  - Market Segmentation
  - Value Proposition Canvas
  - Hypothesis Testing

THE THREE HORIZONS


Customer Base or Market
New
Adjacent
Existing
Product or Service Offering
New
Emergent
Existing
Unicorn
Wizard of Oz testing, risk modeling
Educator
MVP, trial balloons
H3: 10%
Explore, seed, isolate
H2: 20%
Nurture, foster, safe testing, drive growth
H1: 70%
Maintain, defend, extend, strengthen



Iteration by Jennie aligning 3 horizons with segmentation work
Original model by Steve Coley: https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/enduring-ideas-the-three-horizons-of-growth

BAND-AIDING

Intentional/Individual Tech Debt

ROOT CAUSES

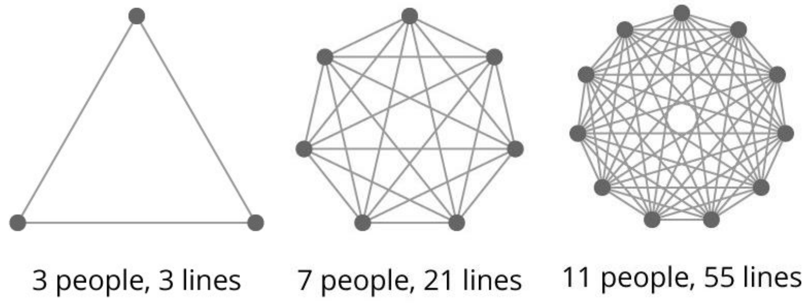
- Culture of firefighting
- Changes made in isolation, outsourcing
- Delays in refactoring
- Lack of coding standards
- Over architecting

SOLUTIONS

- Lessons
  - Root cause analysis is only valuable if it results in changes
  - People will stop trying to solve foundational issues if firefighting is the only thing rewarded
- Solutions
  - Measure what it is costing you
  - Every crises should result in a team solving the problem
  - Don’t reward firefighting, it’s not normal
  - Intentionally change perspectives in teams

ORGANIZATIONAL ALIGNMENT

Intentional/Individual Product Debt



ROOT CAUSES

- Deadline driven development
- Incomplete POC or Spike
- Shiny object syndrome
- Business making solution decisions

SOLUTIONS

- Lessons
  - Mythical Man Month / Brook's Law
  - Beware of HiPPOs – use real data whenever possible and decision-making processes
- Solutions
  - ORID retros
  - RACI or 7 levels of authority
  - Refocus on problem statements
  - Focus on learning what to do different next time
  - What vs How decisions

DEBT COMPOUNDS

WHY THE "BUSINESS" SHOULD CARE

Snowball effect

Speed of delivery

Reliability and trust

Knowing where we are going

Common team

Roles and Responsibilities

No such thing as Gold-plating

CLEANING UP DEBT

TECH

- Campsite Rule
- Pair with Features
- 20% time
- Formal backlog
- RFCs
- Support pair
- Don’t create more
- Burn the ships

PRODUCT

- Performance metrics
- Promises is progress
- Build product goals – “Who” wants “What” and “Why”
- OKRs and KPIs
- Product marketing
- Optimizing your no
- Formalize and document everything
- Burn the ships

REFERENCES

Fred Brooks, Mythical Man-Month

Steve Coley, *The Three Horizons of Growth*, https://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/enduring-ideas-the-three-horizons-of-growth

Melvin Conway, *How Do Committees Invent?*, https://www.melconway.com/Home/Committees_Paper.html

Martin Fowler, *Tech Debt Quadrant*, https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

Dag Liodden @FirstMark , *3 Main Types of Technical Debt and How to Manage Them*, https://hackernoon.com/there-are-3-main-types-of-technical-debt-heres-how-to-manage-them-4a3328a4c50c

Software Engineering Institute , *Towards an Ontology of Terms on Technical Debt*, https://www.researchgate.net/publication/286010286_Towards_an_Ontology_of_Terms_on_Technical_Debt

FURTHER READINGS

(listed from practical to theoretical)

Lewrick, Link, and Leifer, The Design Thinking Toolbox

Douglas Ferguson, Beyond the Prototype

Jeff Patton, User Store Mapping

Croll and Yoskovitz, Lean Analytics

Alexander Osterwalder, Value Proposition Design

Jim Kalbach, The Jobs to Be Done Playbook

R. Brian Stanfield, The Art of Focused Conversation

Jurgen Appelo, Management 3.0

Mik Kersten, Project to Product

Skelton and Pais, Team Topologies

Jennie.ocken.org jennie@ocken.org @jennieocken