Technical debt in a Xamarin.Forms project

XAMARIN ASSEMBLE

3º EDICIÓN

¿Quién soy?

Diego Bonilla

CEO & cofounder - Nareia

Co organizer - XamarinUY

diego.bonilla@nareia.com.uy

/dbonillanareia

nareia.com.uy

meetup.com/xamarinuy

XAMARIN ASSEMBLE

Agenda

01 Motivation

02 Technical debt

03 Why?

04 Solutions

05 Case study

06 Conclusions

1 Motivation

Motivation

- Little mentioned in the Xamarin communities
- Rarely studied, it's more like a feeling
- At Nareia, many third party legacy projects and a lot of WTF!
- Quality of life

2 Technical debt

Debt metaphor

- Ward Cunningham
  - OOPSLA ’92, The WyCash Portfolio Management System
- **THE** metaphor
- Financial analogy

  > *And that said that if we failed to make our program align with what we then understood to be the proper way to think about our financial objects, then we were gonna continually stumble over that disagreement and that would slow us down which was like paying interest on a loan.*
- Clarification, 2011
  - Metaphor, Debt, Speed, Burden, Agility

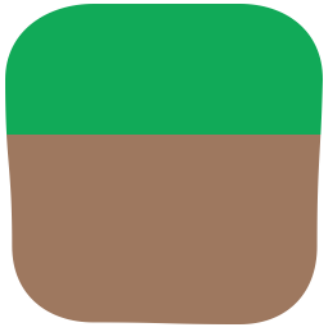
Cruft

Martin Fowler

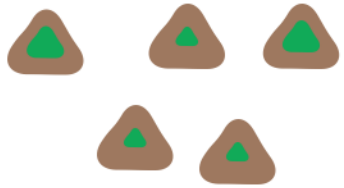
Badly designed, unnecessarily complicated, or unwanted code

*Any software system has a certain amount of **essential** complexity required to do its job...*

*... but most systems contain **cruft** that makes it harder to understand.*

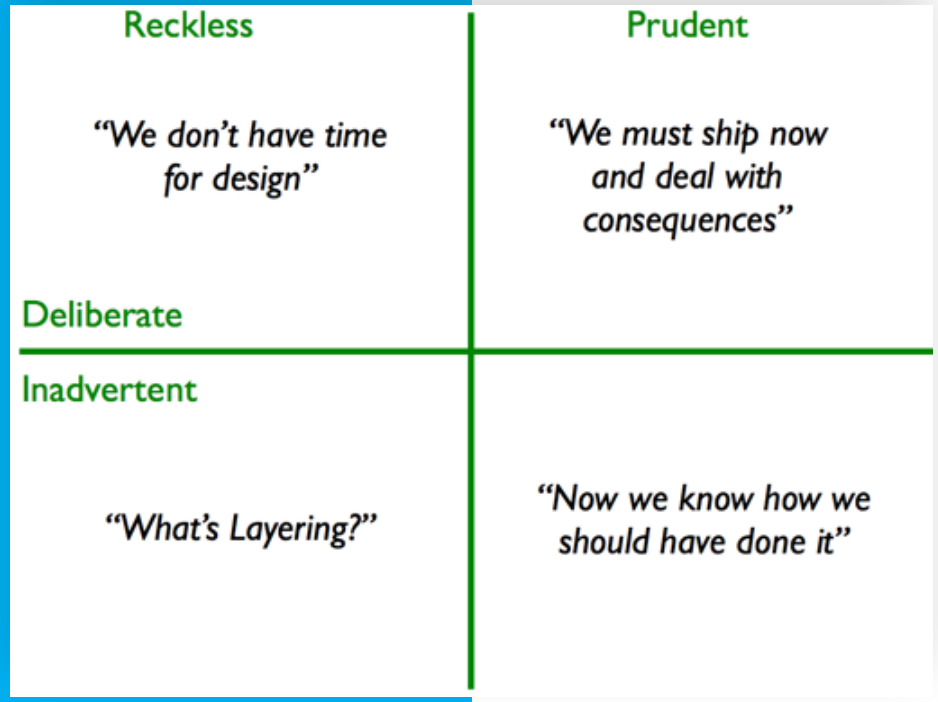


*Cruft causes changes to take **more effort***



The technical debt metaphor treats the cruft as a debt, whose interest payments are the extra effort these changes require.

Debt or non-debt



"A mess is not a debt"

Robert Cecil Martin (Uncle Bob)

2009

TechnicalDebtQuadrant

Martin Fowler

- ***prudent*** *and* ***reckless*** *debt*
- ***deliberate*** *and* ***inadvertent*** *debt*

XAMARIN ASSEMBLE

Other contributions

- Aging

is the decay of component and inter-component behavior when the application functionality meets a minimum standard of satisfaction for the customer.

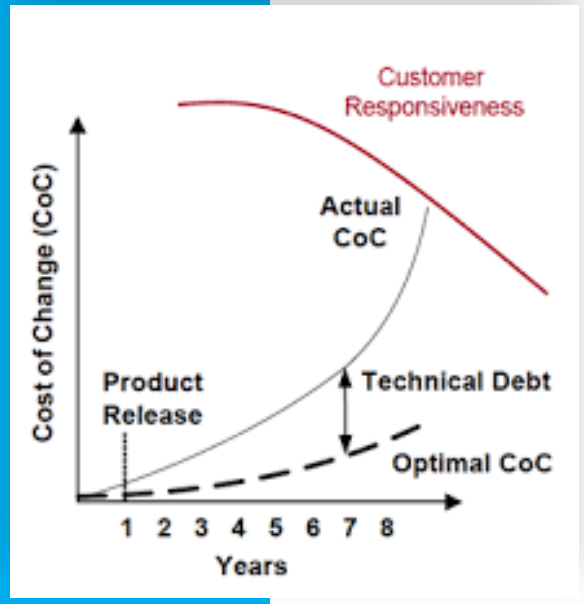
Chris Sterling

- Code level debt vs Structural debt

A design or construction approach that is expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now.

S. McConnell, 2011

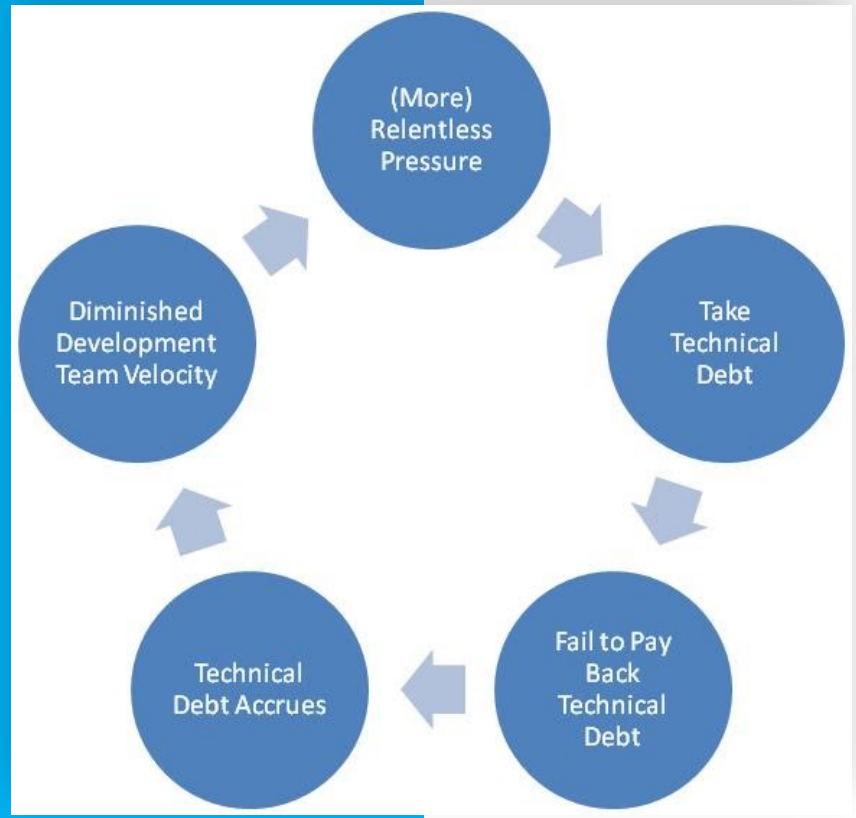
Other contributions



Cost of Change

Jim Highsmith

XAMARIN ASSEMBLE



Other contributions

Vicious circle

Israel Gat

XAMARIN ASSEMBLE

Consequences

- Long Delivery Times
- Lot of production bugs
- Rising Development Costs
- Do not touch me!
- Poor estimation
  - Estimating is nearly impossible
- Angry with the customer
- Poor customer responsiveness
- Frustrated teams
- Poor Performing teams

- **Unhappiness**

3 Why?

Why? Reckless Inadvertent

- Xamarin.Forms is not just C#
  - Platforms
    - Lifecycle
    - Backgrounding
    - AOT / JIT compilation
    - Linker
    - Performance
    - Manifest, property lists
  - MVVM != Model + ViewModel + View
  - Architecture, design patterns, SOLID, GRASP, code sharing strategies
- High team turnover
- No documentation

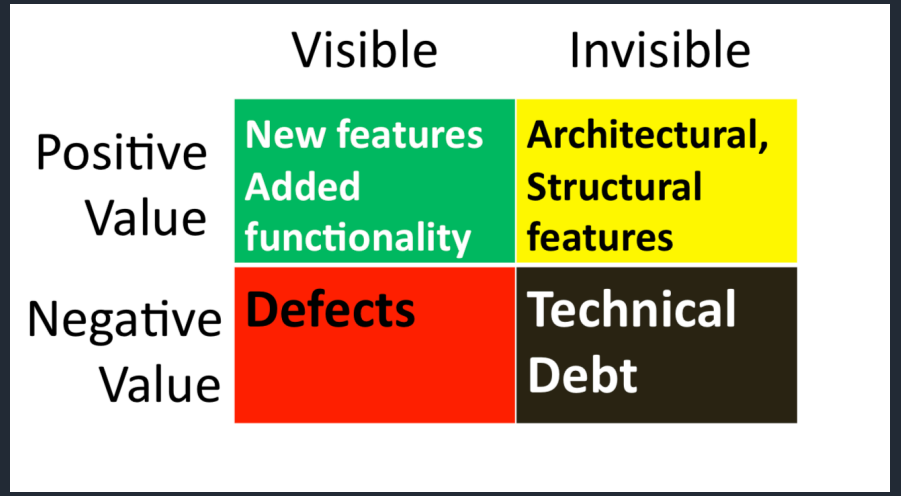
Why? Reckless Deliberate

- Business pressure
- It only matters what you see- frontend / backend effect
- The obsession to deliver value asap
- Misguided agility

Why? Prudent

- In times of customer value
- Architecture has no externally visible “customer value”
- Iteration planning is driven by “customer value”
- Ergo: architectural activities are often not given attention

Invisible / negative value



Philippe Kruchten

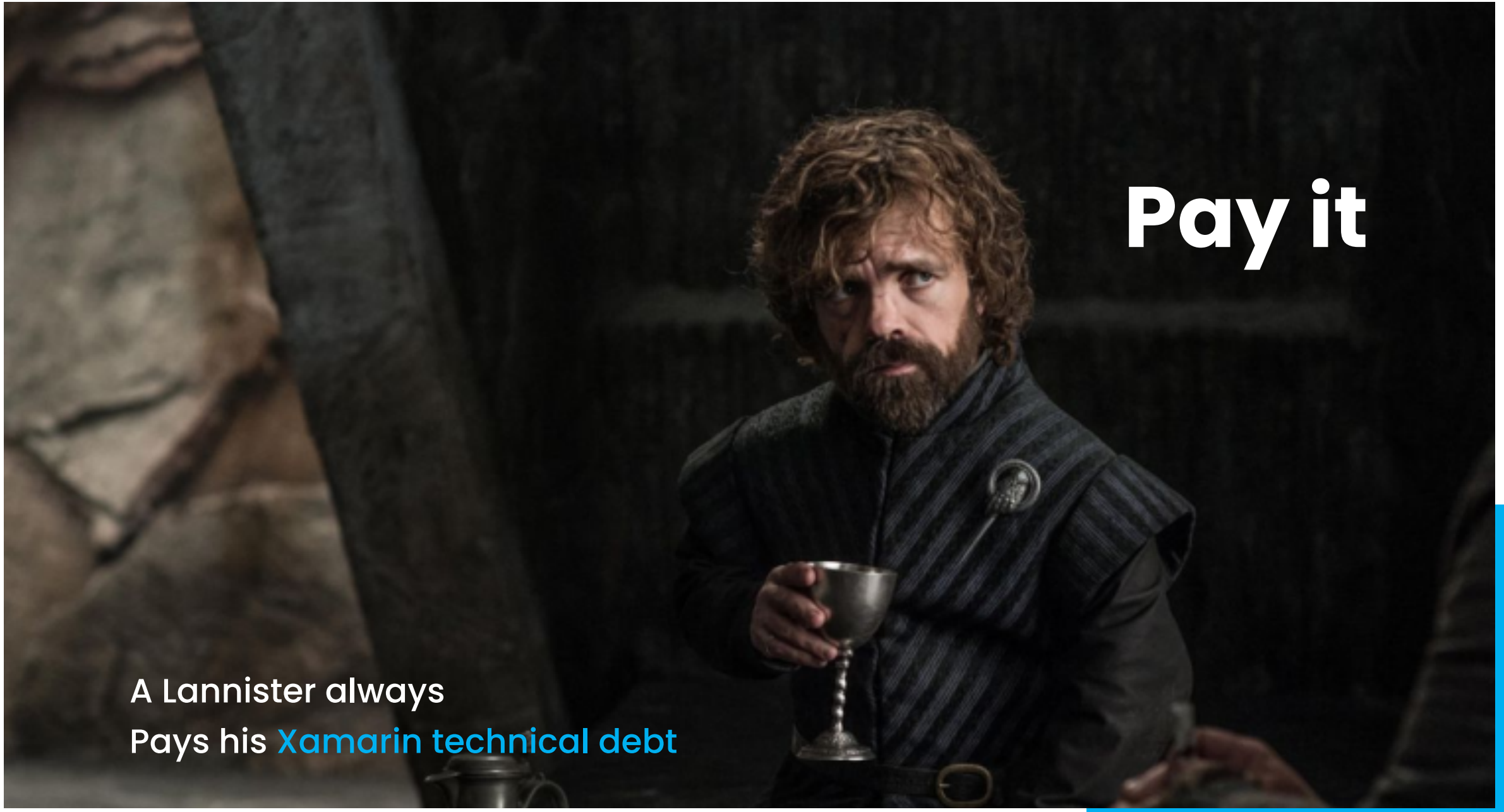
4 Solutions

Solutions

- Information
- Documentation, papers, books
- **Community**

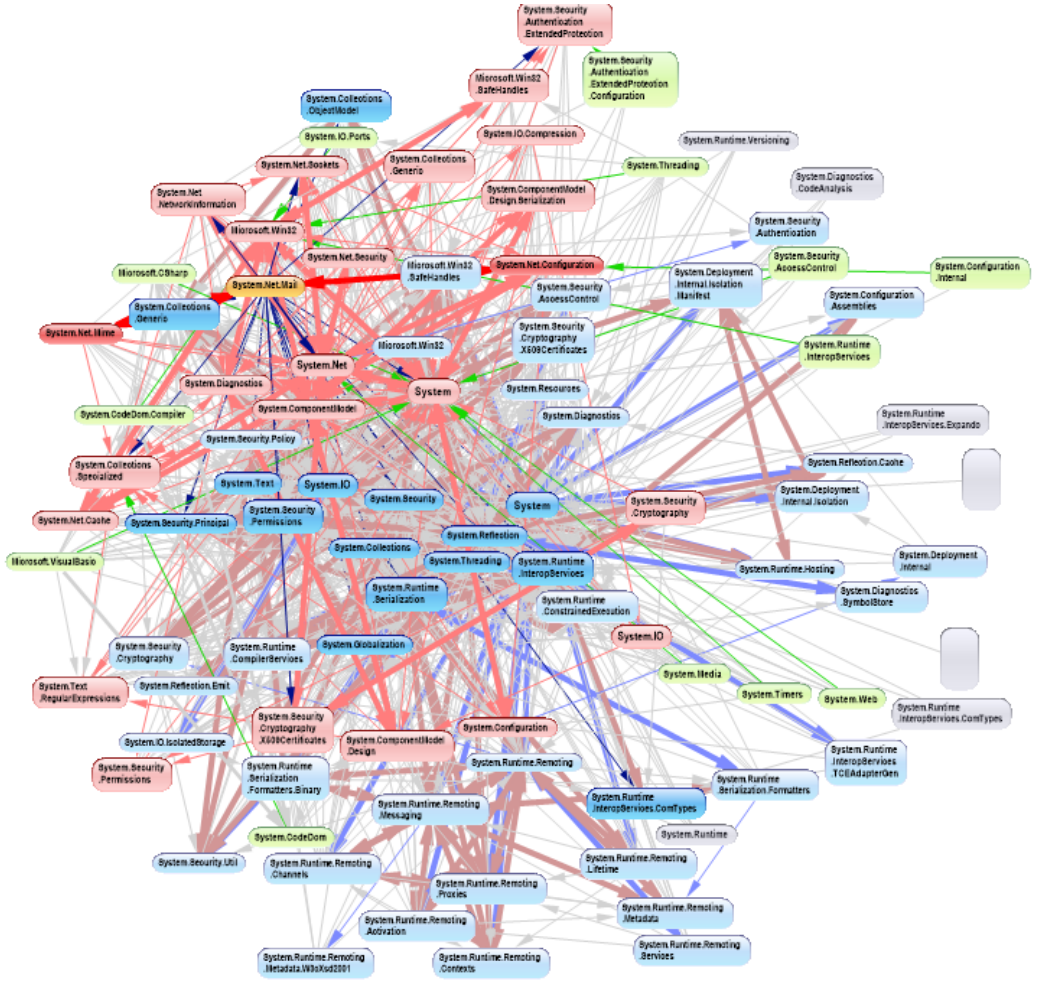
Solutions

- Identify debt, name it!
  - Make technical debt a visible item on the backlog
    - Prioritize with other backlog items
  - Make it visible outside of the software dev. organization
- Incorporate debt reduction as a regular activity
- Use buffer in longer term planning for yet unidentified TD
- Buffer for debt repayment
- Characterize **objectively** and **quantitatively** the amount of TD
  - Xamarin **architecture** level
  - Xamarin **code** level


Pay it
A Lannister always
Pays his Xamarin technical debt

Architectural level

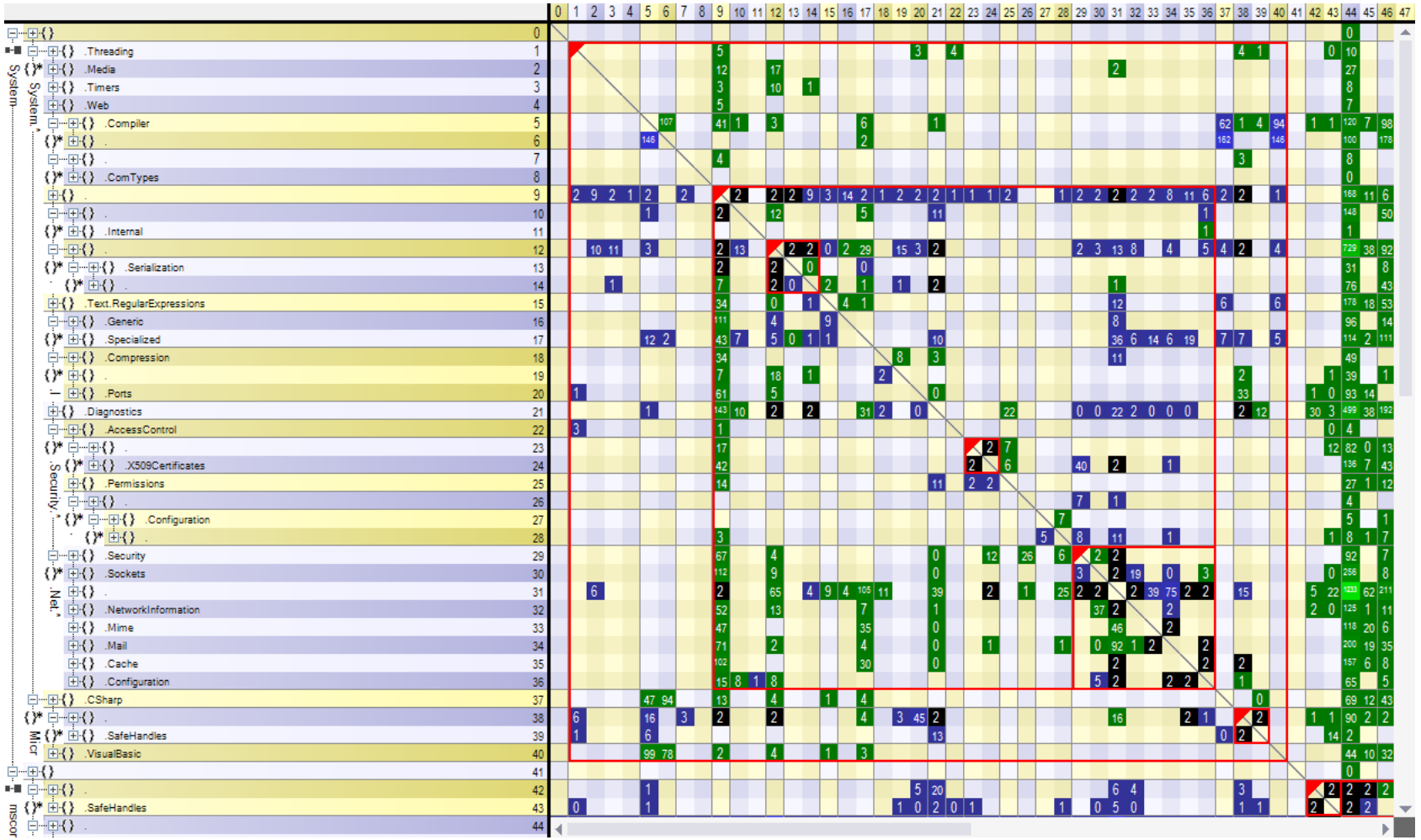
- Harder to detect with tools
- Less research?
- Dependency (design) Structure Matrix
  - **Cycles**
- Domain Mapping Matrix
  - Multiple domain



Dependencies

Source: NDepend

XAMARIN ASSEMBLE



Dependency Structure Matrix

Example (NDepend)

XAMARIN ASSEMBLE

Code level

- Code smells
- Code duplication
- Code coverage
- Cyclomatic Complexity
- Many tools to do static code analysis
  - NET Compiler Platform ("Roslyn") analyzers
  - NDepend
  - SonarLint
  - Resharper

Cyclomatic Complexity

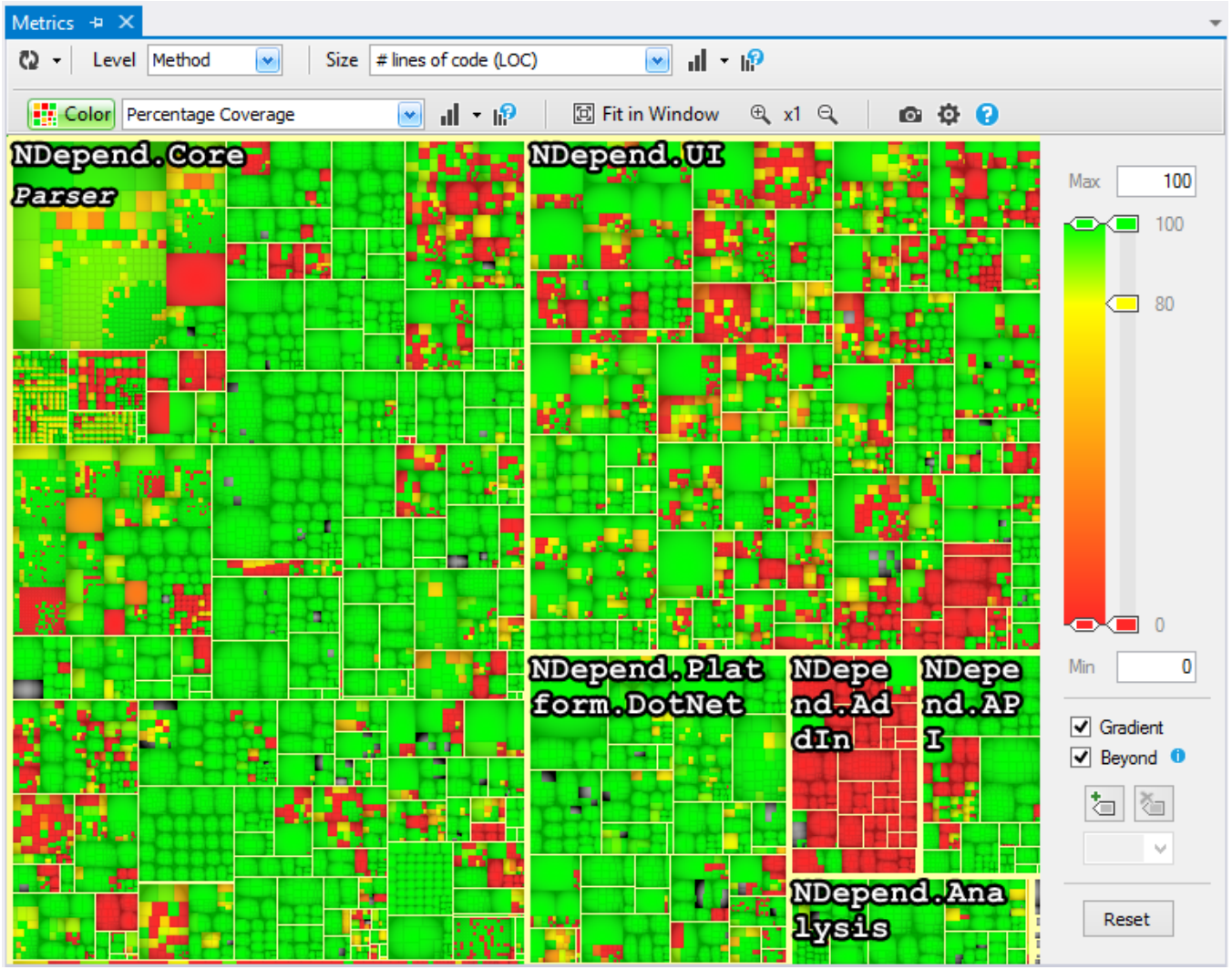
- number of decisions that can be taken in a procedure (1976)

  1 + {the number of following expressions found in the body of the method}:
  if | while | for | foreach | case | default | continue | goto | && | || | catch | ternary operator ?: | ??

  Following expressions are not counted for CC computation:
  else | do | switch | try | using | throw | finally | return | object creation | method call | field access

- Methods where CC is higher than 15 are hard to understand and maintain. Methods where CC is higher than 30 are extremely complex and should be split into smaller methods (except if they are automatically generated by a tool).

XAMARIN ASSEMBLE



Cyclomatic Complexity

Source: NDepend

XAMARIN ASSEMBLE

Others Why/Solutions

- Xamarin limitations
  - old times?
  - support Lifecycle: up-to-date
- Legacy code
  - harder
  - ask project experts
- Project maturity
- Unclear requirements
- Cutting back on process
  - code reviews
- No history of design decisions
- Making bad assumptions

Others Why/Solutions

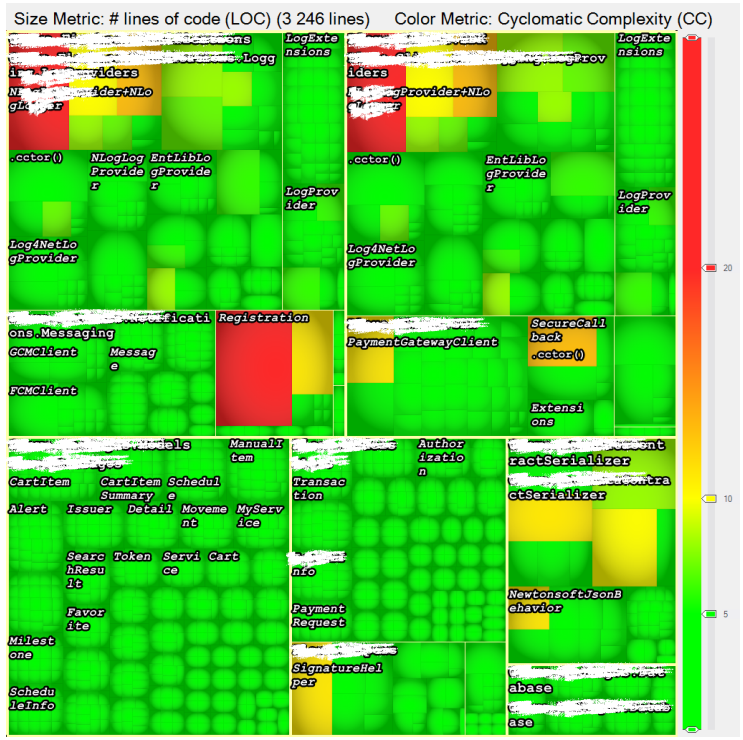
- Poor leadership/team dynamics
- Superstars - egos get in the way
- Little knowledge transfer
- Schedule and budget constraints
- Poor communication between developers and management
- Changing priorities
- Lack of vision, plan, strategy
- Unclear goals, objectives and priorities
- Trying to make every customer happy

5 Case Study

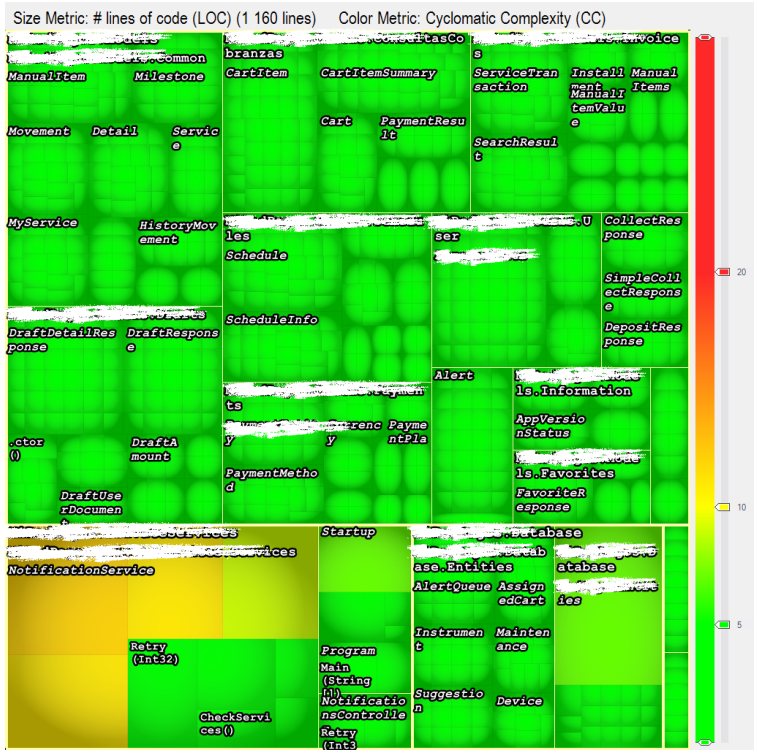
Xamarin.Forms project – Nareia

	Before	Now
Rating	C	A
Issues Critical %	0.3	0
Issues High %	3.8	0.5

Xamarin.Forms project – Nareia



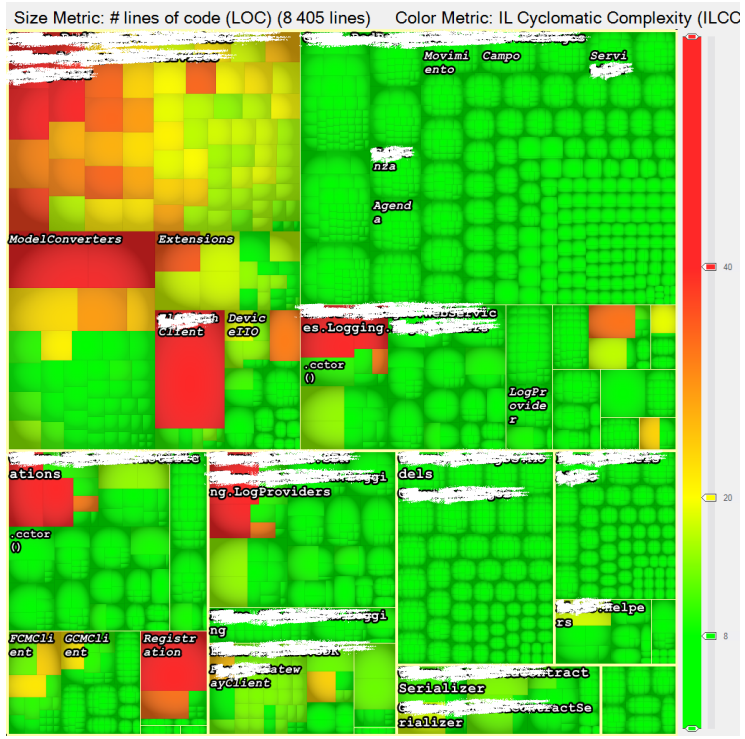
Before



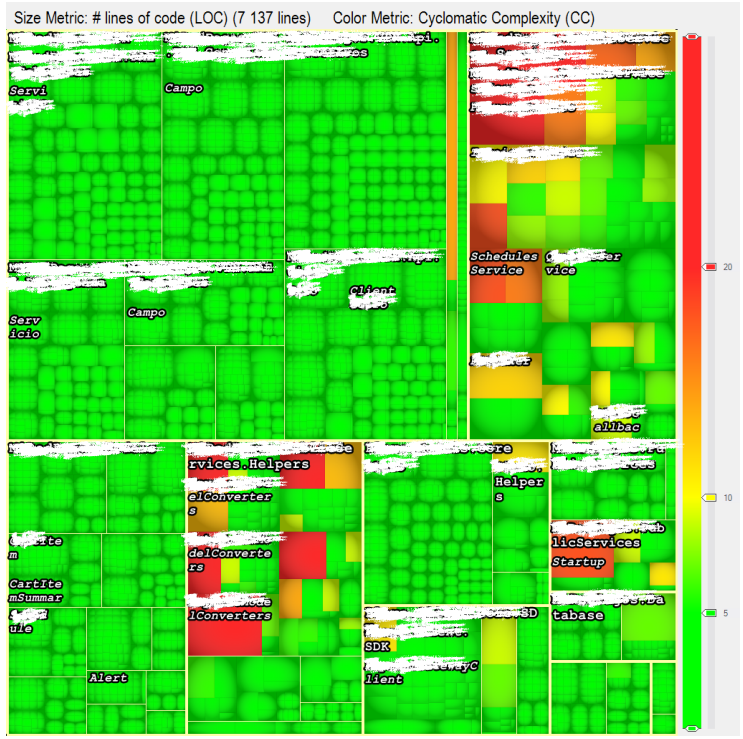
Now

XAMARIN ASSEMBLE

Xamarin.Forms project – Nareia



Before



Now

XAMARIN ASSEMBLE

6 Conclusions

Conclusions

- It’s only a metaphor, but also much more
- The whole point of good design and clean code is to make you go faster (Fowler)
  - Better *bilities*
- There **will always** be technical debt
- Identify debt, characterize objectively and quantitatively, and pay!
  - Architecture level
  - Code level
  - Refactoring + redesign
- Stay up to date
- Call to dev revolution!

XAMARIN ASSEMBLE

References

XAMARIN ASSEMBLE

¡MUCHAS GRACIAS!

Diego Bonilla

CEO & cofounder - Nareia
Co organizer - XamarinUY
diego.bonilla@nareia.com.uy

/dbonillanareia