



Diego Bonilla
CEO & Co-Founder
@dbonillanareia
Co-organizer @XamarinUY

Uruguay



```
1
{
  "FaceRectangle": {
    "Top": 85,
    "Left": 164,
    "Width": 89,
    "Height": 89
  },
  "Scores": {
    "Anger": 0.9456818,
    "Contempt": 1.37124937E-06,
    "Disgust": 0.001311498,
    "Fear": 0.0113821821,
    "Happiness": 0.01080296,
    "Neutral": 0.000253919832,
    "Sadness": 0.00151579734,
    "Surprise": 0.0290504545
  }
}
1
```



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

Debt metaphor

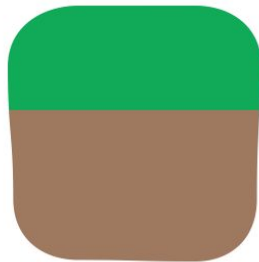
- Ward Cunningham, 1992
- The metaphor
- Refactoring
- Financial analogy
 - *And that said that if we failed to make our program align with what we then understood to be the proper way to think about our financial objects, then we were gonna continually stumble over that disagreement and that would slow us down which was like paying interest on a loan.*
- Speed

Cruft

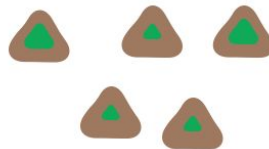
- Badly designed, unnecessarily complicated, or unwanted code
- Martin Fowler:

*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***

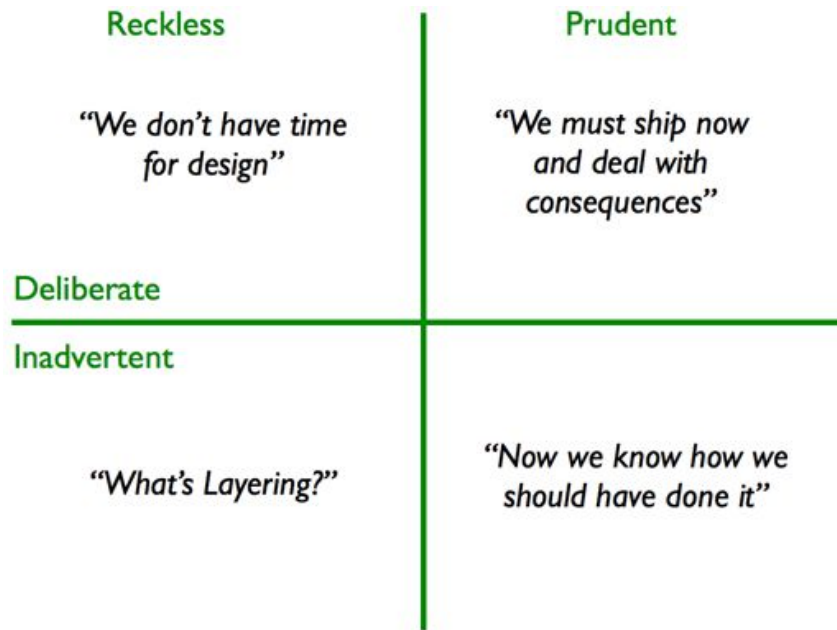


The technical debt metaphor treats the cruft as a debt, whose interest payments are the extra effort these changes require.

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

Technical Debt Quadrant



Why? Reckless

- Xamarin is not just C#
 - Platforms
 - Lifecycle
 - Backgrounding
 - AOT / JIT
 - Linker
 - Performance
 - Manifest, property lists
 - And more
- MVVM != Model + ViewModel + View
- Architecture, design patterns, SOLID, GRASP, code sharing strategies
- Inexperience

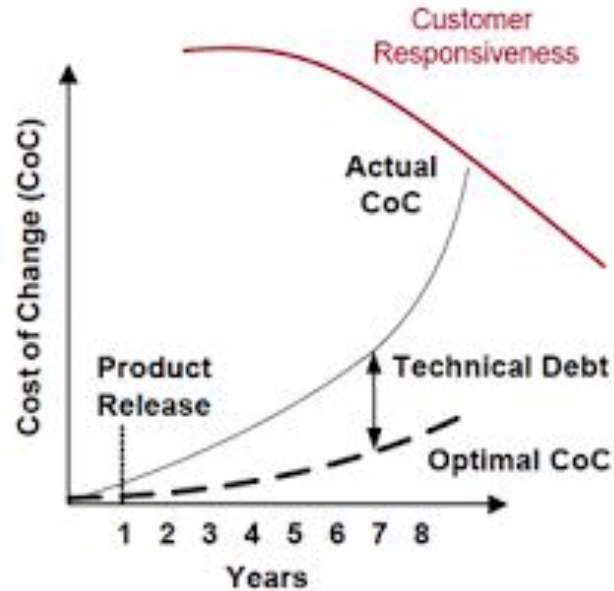
Solutions

- Documentation
- Papers, books
- **Community**

Other contributions

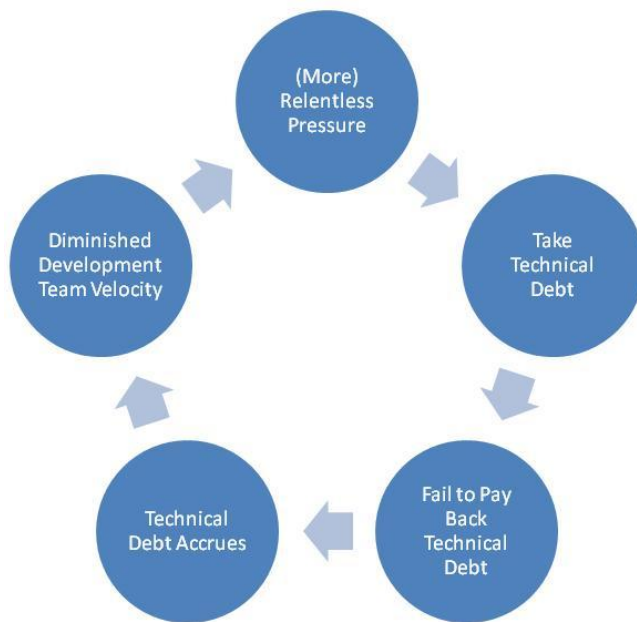
- Aging
- *is the decay of component and inter-component behavior when the application functionality meets a minimum standard of satisfaction for the customer.*
 - Chris Sterling
- Code level debt vs Structural debt
- *A design or construction approach that is expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now.*
 - S. McConnell, 2011

Cost of Change



Jim Highsmith

Vicious circle



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**

Why? Invisible


- 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
- Cost of development is not identical to value

	Visible	Invisible
Positive Value	New features Added functionality	Architectural, Structural features
Negative Value	Defects	Technical Debt

- Philippe Kruchten

Solutions

- Identify debt, name it!
 - Make technical debt a visible item on the backlog
 - Prioritize with other backlog elements
 - 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 technical debt
- Buffer for debt repayment
- Characterize **objectively** and **quantitatively** the amount of TD
 - Xamarin **architecture** level
 - Xamarin **code** level

A still from the television series Game of Thrones. Tyrion Lannister, played by Peter Dinklage, is shown from the chest up. He has curly brown hair and a full beard. He is wearing a dark blue and black striped tunic with a circular brooch on the left shoulder. He is holding a small, ornate silver goblet in his right hand. The background is dark and textured, possibly a cave or a stone wall. The lighting is dramatic, highlighting his face and the goblet.

A Lannister always...
Pays his Xamarin technical debt

Architectural level

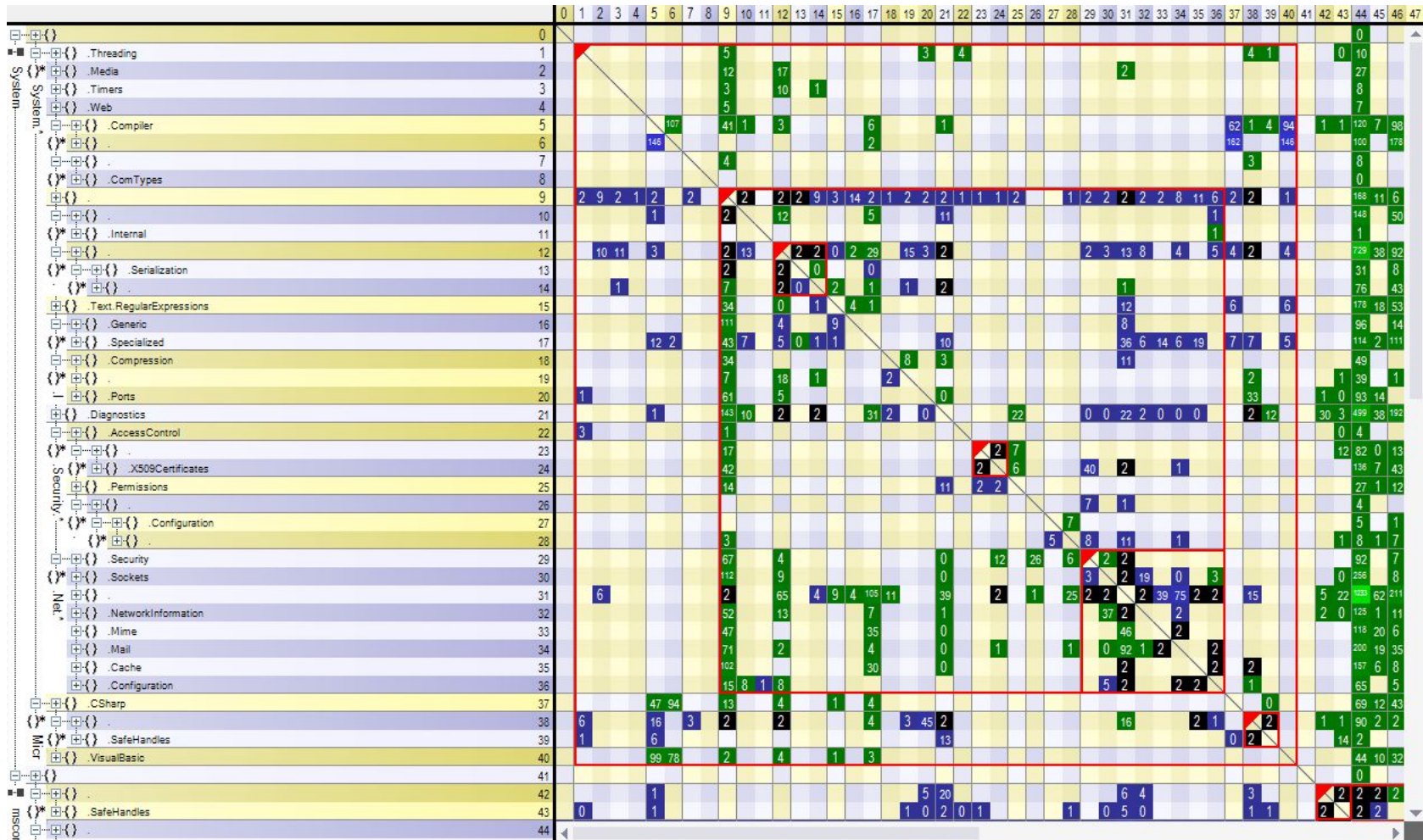
- Harder to detect with tools
- Less research?
- Dependency (design) Structure Matrix
 - **Cycles**
- Domain Mapping Matrix
 - Multiple domain

Dependencies



Dependency Structure Matrix

Example (NDepend)



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).

Cyclomatic Complexity



Source: NDepend

Others Why/Solutions

- Xamarin limitations
 - old times?
 - 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

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!



References



Gracias!

Diego Bonilla
@dbonillanareia

