



Prevent legacy to sink your software vessel

Ard Willems

Bits & Chips Smart Systems Conference, 1 October 2015





Origin of this presentation:

“This organization
is like a
container ship.
Changing its
course takes
a lot of time
and effort.”

- **This quote is from the software project manager of one of my first projects**
- **During my software engineering and architecting projects I found that software organizations made the same mistakes when growing in size; three phases can be distinguished**
- **Also during software quality assessments we do for companies I find a lot of similarities between organizations**
- **I have found that three advices help in preventing or overcoming these mistakes**
- **When adhering to these advices, the company will become more efficient, and legacy code will no longer be an issue**

NORTH ATLANTIC OCEAN
NORTHEASTERN PART

BOUNDARIES IN FATHOMS
For Symbols and Abbreviations, see Chart No. 1
RECENT REVISIONS
SCALE 1:50,000 (1" = 1 NM)

Today's Journey

3 phases



3 advices



end goal

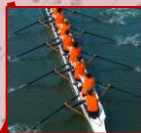


NORTH ATLANTIC OCEAN
NORTHEASTERN PART

BOUNDARIES IN FATHOMS
For Symbols and Abbreviations, see Chart No. 1
RECENT REVISIONS
SCALE 1:50,000 OF LAT 40°

Today's Journey

3 phases



The Rowing Boat

aka The Startup

Phase 1



**Business idea →
Find investors.**



Business plan



Mockup

RNoN Jageren Sleipner, April 1940

© Copyright 2009 Louis R. Coatney



Build results



Future cost
of a
(possibly)
wrong design
decision

Code inherited
from someone
else

Legacy

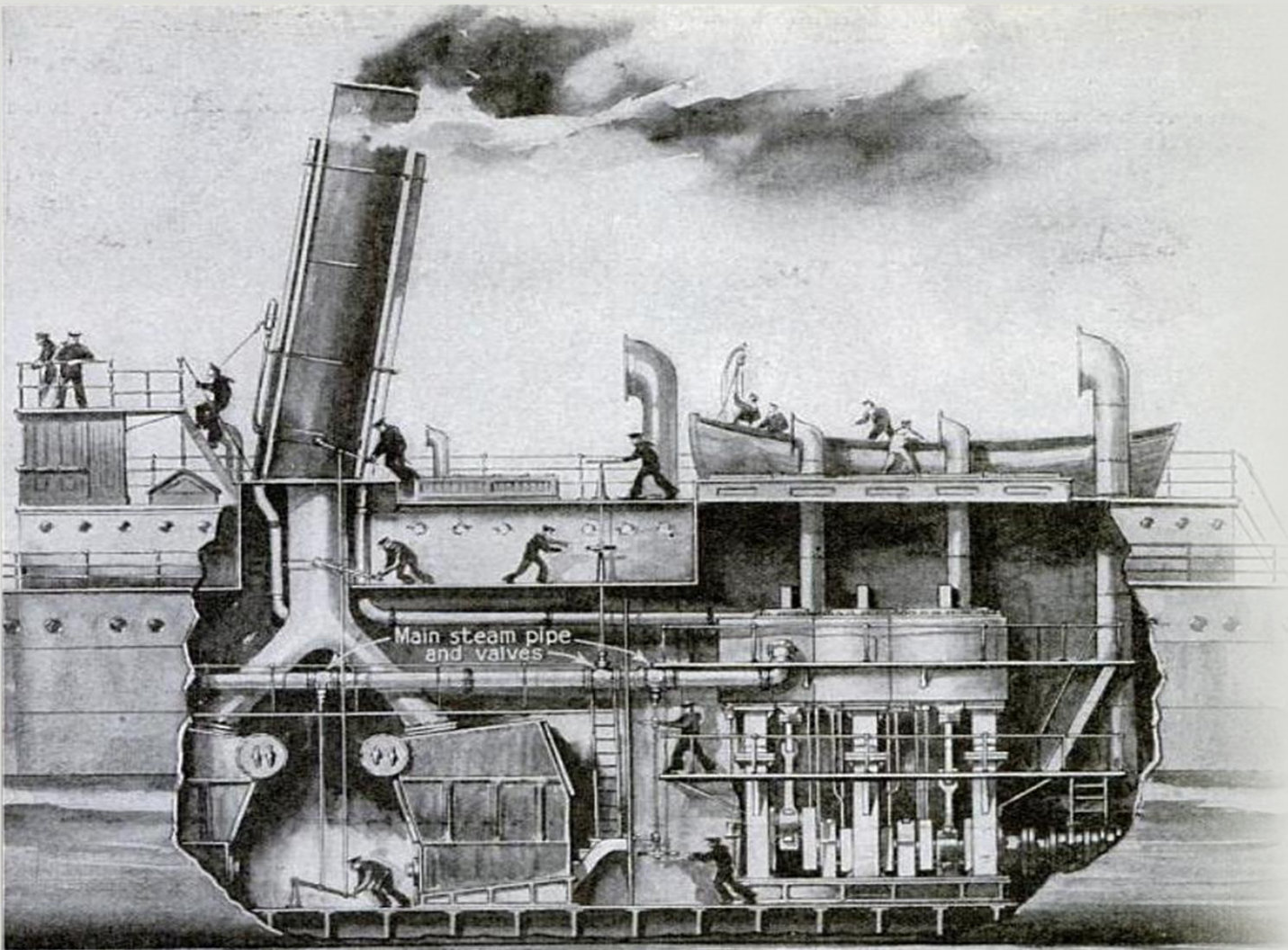
Technical Debt

The Steam Boat

aka The Medium Sized Company

Phase 2





**Growing
software
archive**

**Growing
software
team**

**Need for
processes**

Don't rush!

- Important decisions should not be made at the coffee machine
- The most experienced software engineer is often not the best software architect
- Take your time to create suitable processes



A black and white photograph of four naval officers in uniform standing in a row. They are wearing dark jackets with double-breasted buttons and white shirts. The officer on the far right has a beard and is holding a rolled-up document. The officer in the center is also holding a rolled-up document. The officer on the far left is holding a rolled-up document. The background is dark and indistinct.

Competent managers
aim for efficiency and continuous improvement

Phase 3

The Container Ship

aka The Multinational



Code complexity

- 8x more defects
- 0.5x productivity
- 10x staff turnover



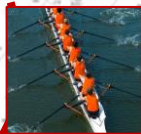
Source: **Technical Debt in Large Systems: Understanding the Cost of Software Complexity**, Dan Sturtevant PhD, MIT

NORTH ATLANTIC OCEAN
NORTHEASTERN PART

BOUNDARIES IN FATHOMS
For Symbols and Abbreviations, see Chart No. 1
RECENT REVISIONS
SCALE 1:50,000 (1" = 1.5 NM)

Today's Journey

3 advices



A large red ship is shown in a dry dock, completely encased in a complex network of metal scaffolding. The ship's hull is a vibrant red, while the upper sections are partially obscured by the grey metal framework. Several workers in safety gear are visible on different levels of the scaffolding, indicating active construction or maintenance work. In the background, yellow cranes and other industrial structures are visible under a clear blue sky.

Advice 1

Refactoring

Change design:

- Suits requirements
- Fit for future
- Doesn't change behavior

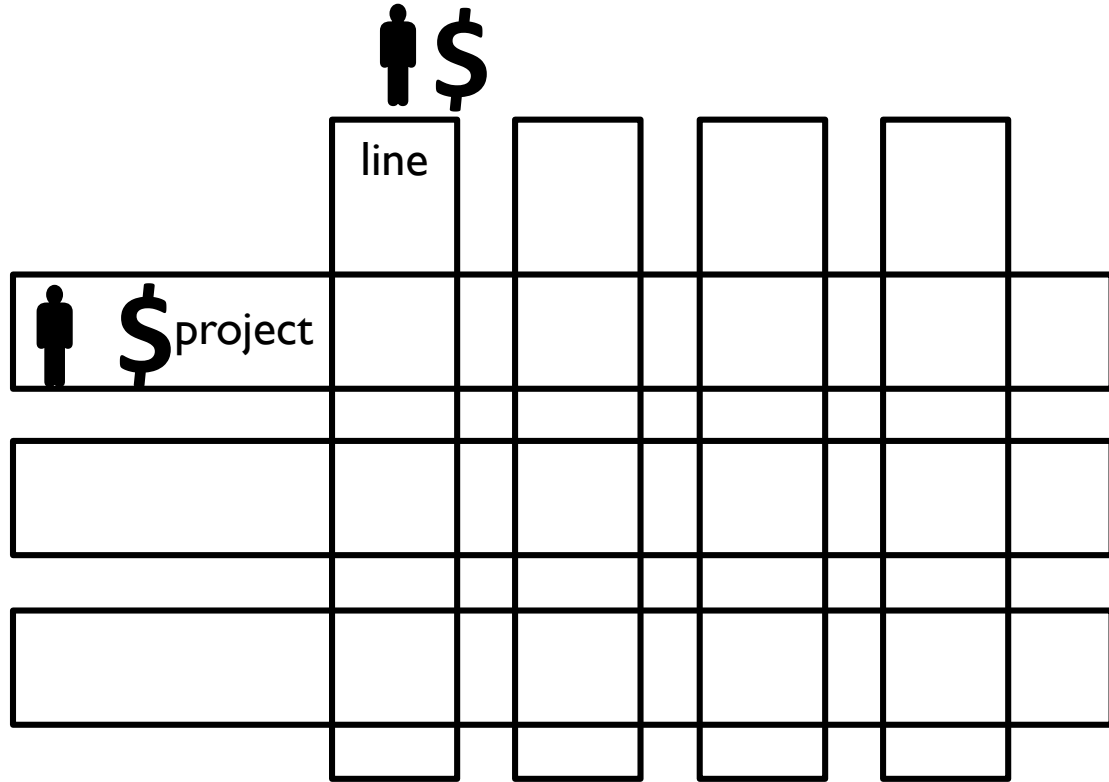
Doesn't change behavior... :

$$\text{⌚} + \$ = 0 ?$$

- Refactoring costs time and money, but does not add new features
- A project leader will therefore not benefit from refactoring for his project

Responsibility for refactoring

- A budget needs to be assigned to enable refactoring
- A line manager can be made responsible
- In an agile environment, it can be an explicit responsibility of the teams



Advice 2

Continuous Integration

An aerial photograph of a large naval fleet, including several aircraft carriers and numerous smaller warships, sailing in formation on a deep blue ocean under a cloudy sky. The ships are leaving white wakes behind them.

An example of a refactoring project

- **Software engineer proposes a changed design**
- **Estimated effort is three weeks**
- **One year later, the project is still not finished, and is canceled**

There are three causes for this setback...

Periodical Integration

- Features are added before a certain deadline
- If you want your change to be in the next release, deliver before that deadline



Long Integration Times

- After the deadline: merging, compiling, testing starts
- Test plan includes long manual testing



The polluter-pays-principle

Whoever is responsible for the failing integration, has to fix it, or withdraw his delivery



This could have been avoided with
Continuous Integration

Because of the following three aspects of CI

Merge continuously





**Test
quickly**

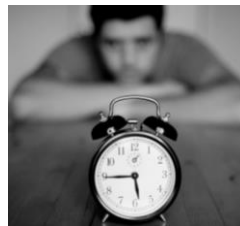


**Discoverer
pays**

Challenges of refactoring



**periodical
integration**



**long
integration
times**



**polluter
pays
principle**

Benefits of continuous integration



**merge
continuously**



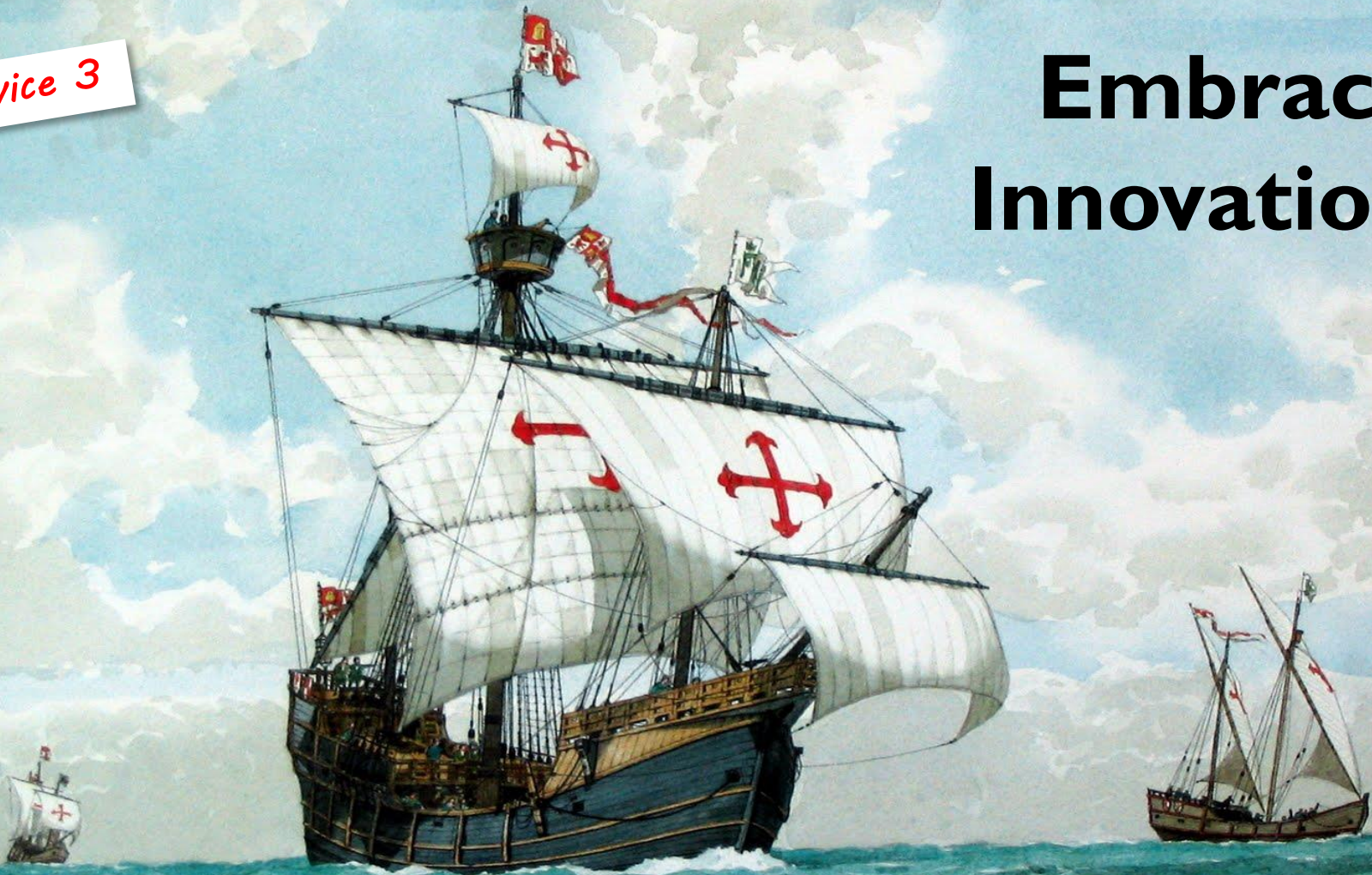
test quickly



**discoverer
pays**

Advice 3

Embrace Innovation



Software engineers are passionate people





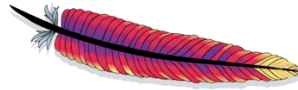
**If a software engineer asks
why a certain technology or
process is used, the worst
possible answer is:**

**“Because
we’ve *always*
done it *this*
way.”**

Let software engineers do pilots with new technology



ReSharper



Apache



Jenkins



Additional
benefit:

```
if (Developers.Happy())  
    return OnInvestment;
```

NORTH ATLANTIC OCEAN
NORTHEASTERN PART

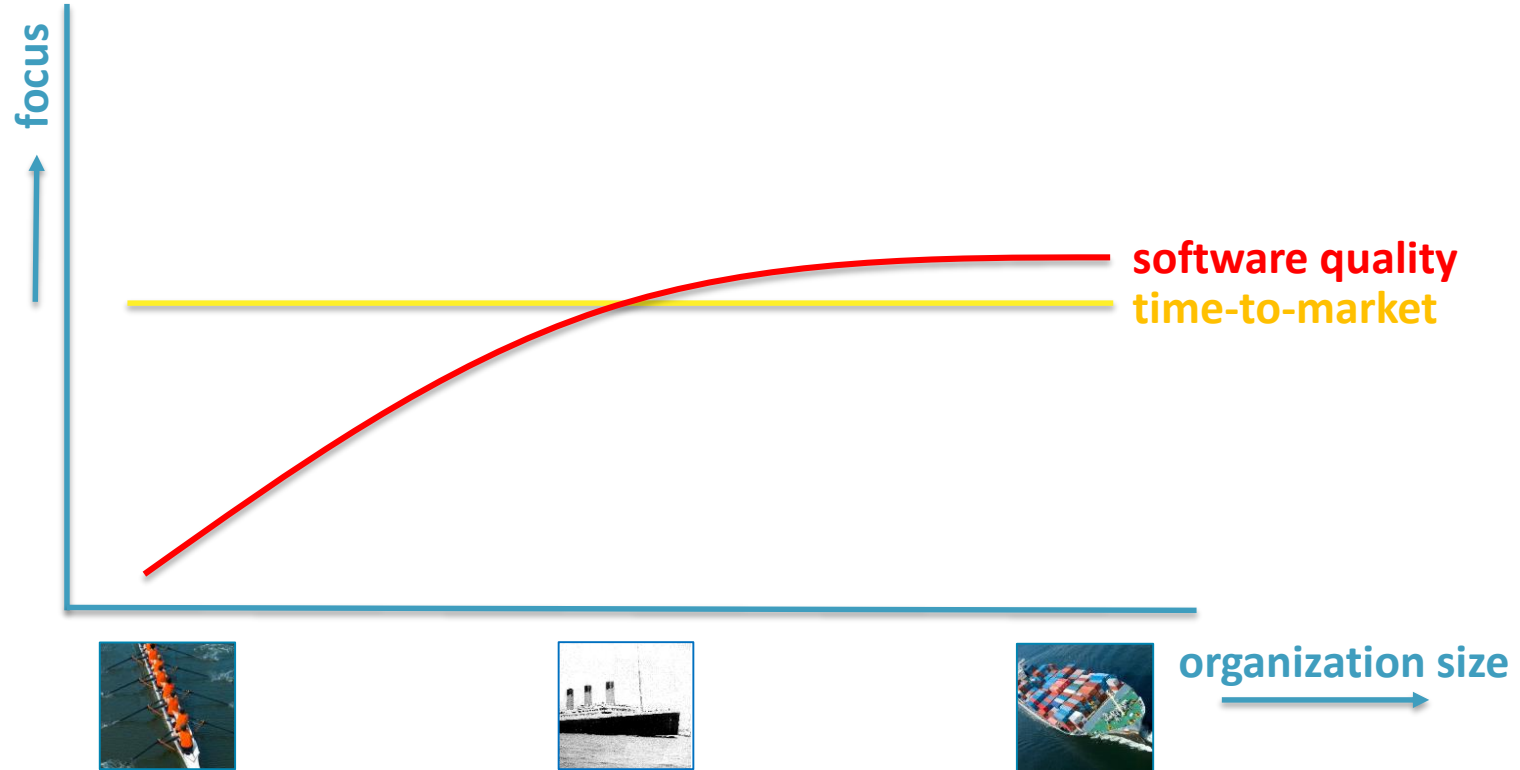
BOUNDARIES IN FATHOMS
For Symbols and Abbreviations, see Chart No. 1
NATIONAL HYDROGRAPHIC
SCALE 1:50,000 OF LAT 40°

Today's Journey

end goal



Desired focus



The end goal:



Small, flexible teams, that continuously add value to your products, with the freedom to change their course when they see fit.





Thank you
Have a safe journey

