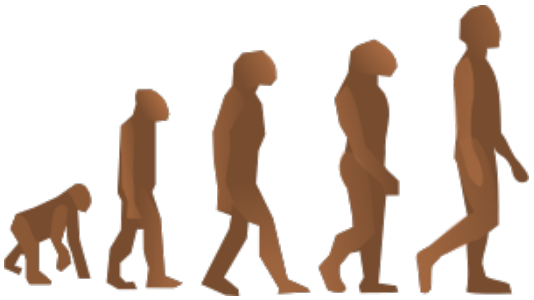
Darwin



Warten auf .... Godot

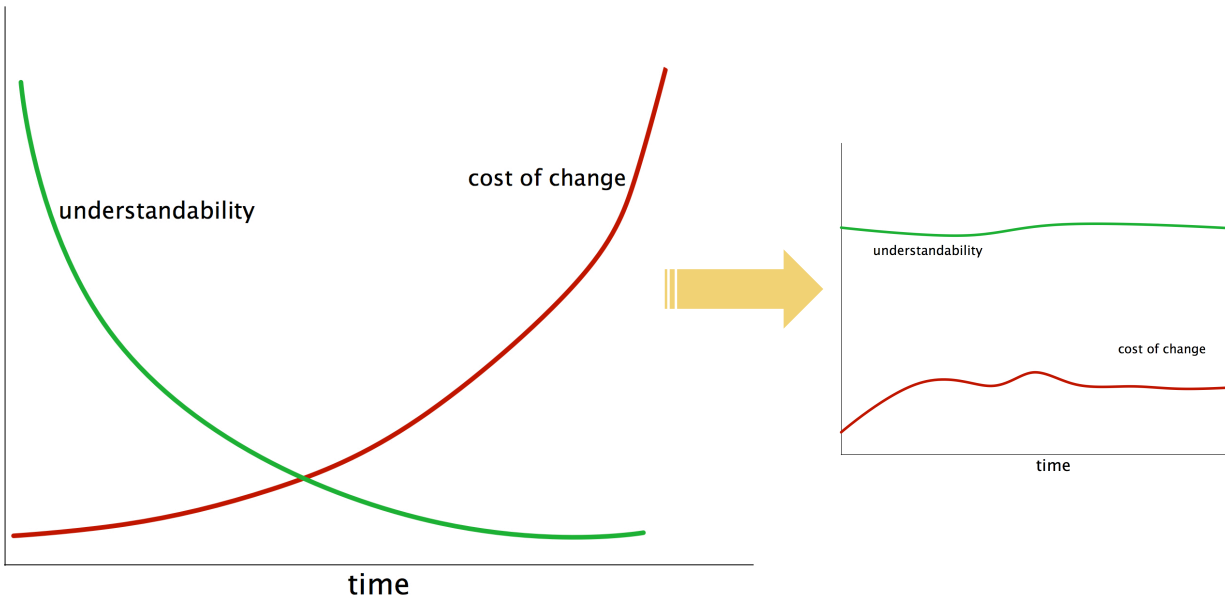
(Samuel Beckett, 1952)

Kategorie „Absurdes Theater“

Gernot Starke

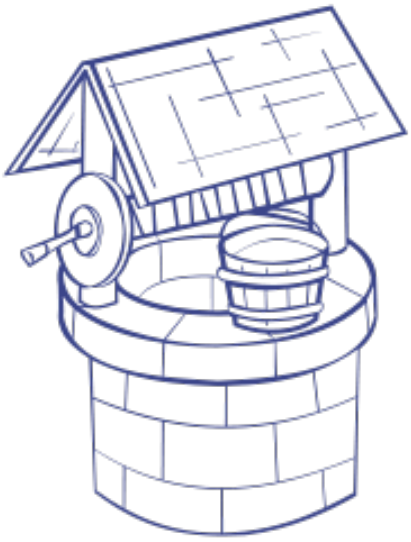
Follow @gernotstarke

Das Problem...



© 2014 Dr. Gernot Starke / innoQ / aim42

Informatiker müssen angeln...



Kinder aus Brunnen angeln („retten“) ...

These:

Informatik

Ausbildung fokussiert

von Systemen

auf Neubau

80 : 20 Regel

- 80% unserer Zeit ändern wir,
  20% bauen wir neu.

In der Ausbildung:

- 100% der Zeit lernen wir neu bauen.
- In der restlichen Zeit lernen wir ändern.

Wartbare Software benötigt „Ordnung“

- konzeptionelle Integrität
  - Substantielles Investment in
    „innere Ordnung“

- Verständlichen Code
- Überblicksdokumentation

These:

Informatik

Praxis benötigt mehr Änderungskompetenz

an Systemen

These:

Änderungen an Systemen sind durch Geld motiviert

Gründe für Änderung an Software

- Neue / geänderte Anforderungen
- Änderungen im Kontext
  - Externe Schnittstellen, Datenformate
  - Technologie
  - Organisation
- Aufgetretene Probleme
  - Fehler
  - Verletzung von Qualitätsanforderungen
- Hohe Betriebs- oder Änderungskosten

Geld!

- Intrinsische Motivation von Entwicklern

These:

Budgetverantwortliche ignorieren Architekturprinzipien

© 2014 Dr. Gernot Starke / innoQ / aim42

These:

Verbesserung an Systemen ist mehr als Refactoring einzelner Klassen

Architecture Improvement Method

Gernot Starke & aim42-Contributors

http://aim42.org

© 2014 Dr. Gernot Starke / innoQ / aim42

Darum aim42

Methodischer Rahmen für Optimierungs- und Veränderungsprojekte

Darum aim42

Gibt Sicherheit bei Änderungen

© 2014 Dr. Gernot Starke / innoQ / aim42

Darum aim42

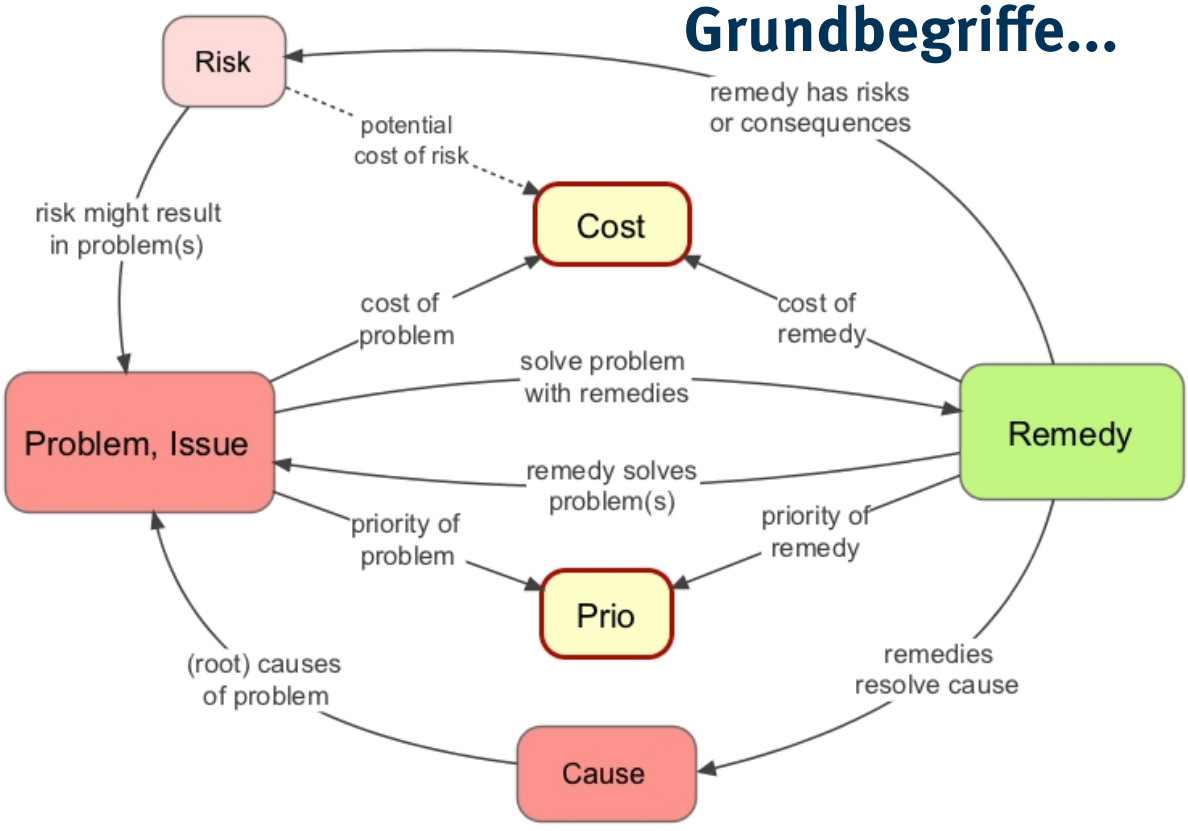
Adressiert Business und Technik

Darum aim42

frei, flexibel, open-source

Methodik

Grundbegriffe...

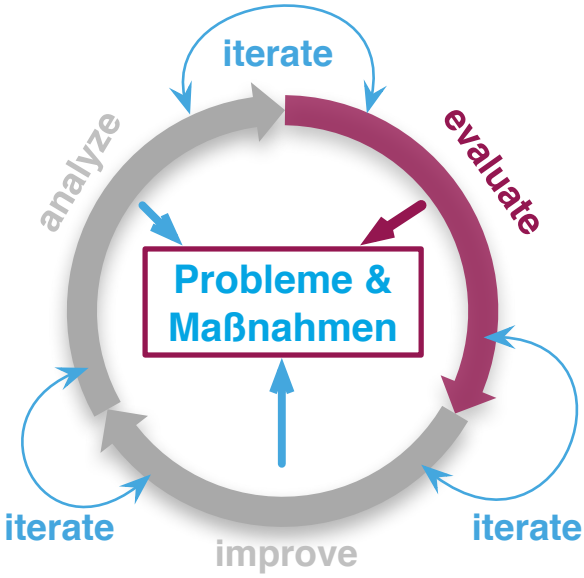


Iterative Phased Improvement

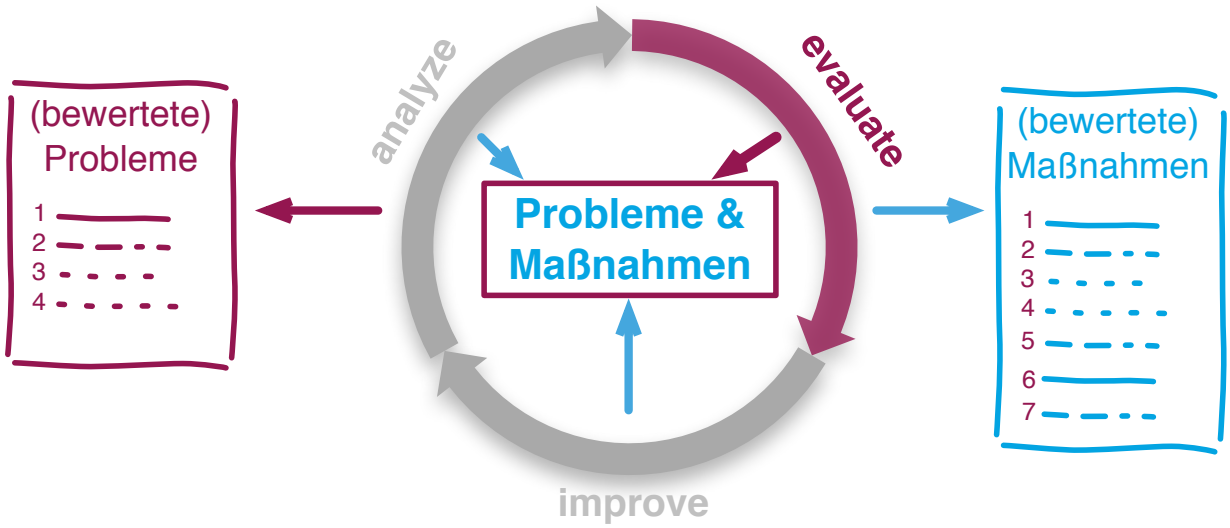
- architecture
- code
- runtime
- organization

Estimate „value“ of problems / risks / issues and their remedies

Crosscutting-Activities (1)



Crosscutting-Activities (2)



Practices and Patterns

aim42

Analyze Practices and Patterns

- ATAM
- Capture Quality Requirements
- Context-Analysis
- Data-Analysis
- Development-Process-Analysis
- Documentation-Analysis
- Issue-Tracker-Analysis
- Pre-Interview Questionnaire
- Profiling
- Qualitative Analysis
- Quantitative-Analysis
- Questionnaire
- Root Cause Analysis
- Runtime-Artifact-Analysis
- Software Archeology
- Stakeholder-Analysis
- Stakeholder-Interview
- Static Code Analysis
- Use-Case-Cluster
- View Based Understanding

Analyze Practices and Patterns...

- ATAM
- Context-Analysis
- Development-Process-Analysis
- Issue-Tracker-Analysis
- Questionnaire
- Software Archeology
- Stakeholder-Interview
- Static Code Analysis

Sample Practices from ANALYZE

- ATAM: Architecture Tradeoff Analysis Method. Systematic approach to find architectural risks and tradeoffs (compromises) .
- DATA ANALYSIS: Analyse and inspect the data created and manipulated by the system for its content, structure, quantity and size.
- PRE-INTERVIEW-QUESTIONNAIRE: Prior to interviewing stakeholders, present them with a written questionnaire, so they can reflect in advance.
- STATIC CODE ANALYSIS: Analyse source code to identify building blocks and their dependencies, determine complexity, coupling, cohesion and other structural properties.

Evaluate Practices and Patterns

- Estimate in Intervall
- Estimate Problem Cost
- Estimate Remedy Cost
- Failure Mode and Effect Analysis
- Impact Analysis

Estimate-in-Intervalls

- Schätze IMMER in Intervallen
- Breite des Intervalls ==
  Eigenes Vertrauen in Schätzung

[15..2]

Improvement Practices and Patterns

- Anticorruption-Layer
- Assertions
- Automated-Tests
- Branch-For-Improvement
- Extract-Reusable-Component
- Group-Improvement-Actions
- Improve-Code-Layout
- Introduce Boy Scout Rule
- Interface Segregation Principle
- Isolate Changes
- Keep-Data-Toss-Code
- Never-Change-Running-System
- Quality-Driven-Software-Architecture
- Refactoring
- Refactoring-Plan
- Remove-Nested-Control-Structures
- Sample-For-Improvement
- Schedule-Work
- Untangle-Code
- Use Invariants To Kill Zombies

Improvement Practices and Patterns

- Automated-Tests
- Quality-Driven-Software-Architecture
- Sample-For-Improvement
- Introduce Boy Scout Rule
- Isolate Changes

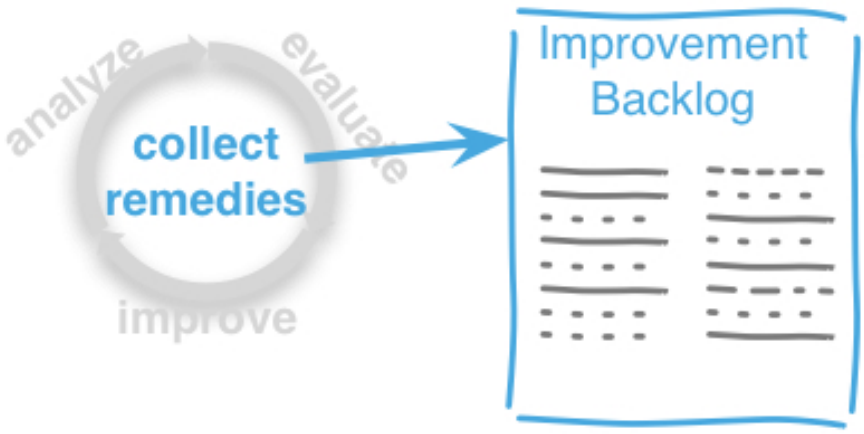
Introduce Boy-Scout Rule

- Im bestehenden Code die BSC einführen...
- Klärung:
  - welche Code-Regeln zukünftig gelten sollen
  - Prioritäten der Veränderung (z.B. Bezeichner, Vereinfachung, Sonar-Findings, Formatierung)
  - Kommunikation im / über Code

Crosscutting Practices and Patterns

- Explicit Assumption
- Fast Feedback
- Collect Problems
- Collect Opportunities for Improvement
- Improvement Backlog

Improvement Backlog



Improvement Backlog (kompakte Fassung)

- Probleme mit Maßnahmen zur Behebung
- Inklusive Kosten
  - Probleme: Kosten pro Zeit/Auftreten
  - Maßnahmen
- Risiken der Behebung

Prio	Problem	Maßnahmen (Remedies)	Kosten (Problem)	Kosten (Behebung)	Risiken
1					
2		....			
3		.....			

Sample Crosscutting Practices and Patterns

- COLLECT PROBLEMS: Maintain a central list or overview of known problems, together with their cost/effort evaluation.
- COLLECT OPPORTUNITIES FOR IMPROVEMENT: In all AIM42 phases, one should identify remedies for the currently known problems or their causes.
- IMPROVEMENT BACKLOG: A list or collection of remedies and their cost/effort/risk estimation.
- In the sense of lean and agile, teams shall try to FAST FEEDBACK: The later a lack of quality is identified the higher are the costs to fix it.

Management überzeugen

Management überzeugen

You need to talk business!

(Martin Fowler, OOP 2014)

© 2014 Dr. Gernot Starke / innoQ / aim42

Management überzeugen...

- Problem Cost ermitteln
  - Technische Probleme haben einen Preis
  - Schätzen mit expliziten Annahmen

Beispiel: Mehraufwand „Heterogenität“

- Technologiezoo: System aus ›20 Subsystemen in 8 Technologien
- Techniker: „aufwändig, komplex“
- Management: was kostet das?

Kosten der (übertriebenen) Heterogenität

- Mehraufwände in Lebenszyklus-Phasen schätzen
  - Analyse, Architektur, Implementierung, Test, Betrieb
  - Unterstützt durch reale Aufwandsmessungen

Mehrkosten „Heterogenität“ [20%..2%]

	Anteil	Mehraufwand min	Mehraufwand max	1.000 € min	max
				1.017,78 €	1.204,56 €
Requirements	**7%**			70 €	70,00 €	70,00 €
Design/Architektur	**6%**			60 €	60,42 €	61,20 €
10% Zusatzaufwand Schnittstellen		5%	15%		0,30	0,90
10% übergreifende Entscheidungen		2%	5%		0,12	0,30
80% Sonstiges						
Programmierung	**12%**			120 €	122,40 €	145,68 €
2% Setup, Updates-Umgebung		5%	100%		0,12	2,40
2% Einarbeitung, Recherche		5%	20%		0,12	0,48
10% Fehlersuche, Testing		3%	100%		0,36	12,00
5% Effiziente Lösung von Detailproblem		-10%	-40%		-0,60	-2,40
10% Lösung von Standardproblemen		10%	50%		1,20	6,00
20% Team-interne Abstimmung		5%	30%		1,20	7,20
51% Sonstiges						
Integration / Test	**8%**			80 €	83,40 €	113,80 €
5% Komponenten integrieren		5%	100%		0,20	4,00
30% Integrationstests durchführen		5%	50%		1,20	12,00
20% Integrationstests auswerten		10%	50%		1,60	8,00
10% Testumgebung bereitstellen/erhalten		5%	80%		0,40	6,40
35% Sonstiges						
Maintenance / Operations	**67%**			670 €	681,56 €	813,88 €
3% Vorhalten von Entwicklerkapazität		5%	20%		1,01	4,02
5% Entwickler finden/einarbeiten		10%	30%		3,35	10,05
1% Versions- und Security-Updates		3%	10%		0,20	0,67
1% Auswahl / Beschaffung Laufzeitumgebungen		10%	100%		0,67	6,70
3% Konfiguration, Installation		5%	70%		1,01	14,07
0,50% Monitoring, Logging		5%	10%		0,17	0,34
5% Fehlersuche und -Behebung		1%	100%		0,34	33,50
2% Skalierung/Clustering		5%	15%		0,67	2,01
1% Paketierung, Deployment-Vorbereitung		2%	10%		0,13	0,67
30% Erweiterungen/Änderungen vornehmen		2%	30%		4,02	60,30
49% Sonstiges						

Praxis

Praktisch eingesetzt...

- Automotive: „Multimedia-Framework“
- Rail-Service „Infrastruktur“
- Mobilfunk „Billing“
- Airport-Operations „Luggage Handling“
- Systemsoftware für Maschinenbau / Lebensmittelindustrie
- 2014:
  - Europäische Bahn (Audit OnlineTicket)
  - ERP-Hersteller (Audit und Rebuild Kernsystem)

© 2014 Dr. Gernot Starke / innoQ / aim42

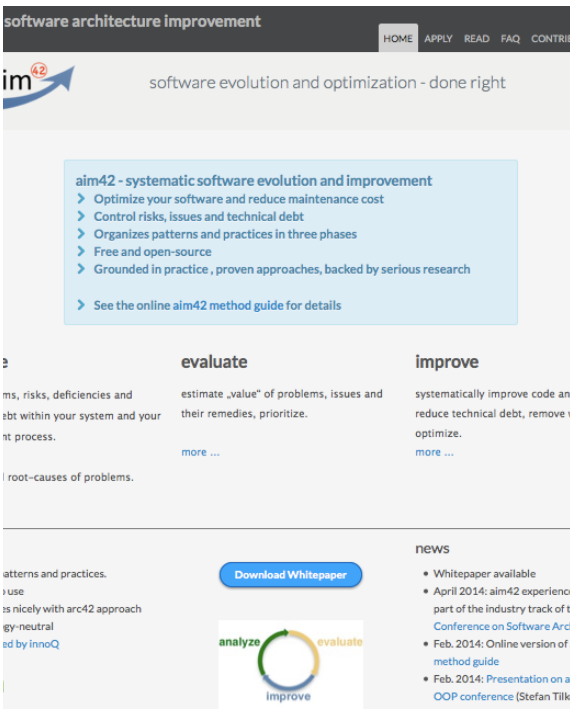
Projekt

Build & DevelopmentProcess

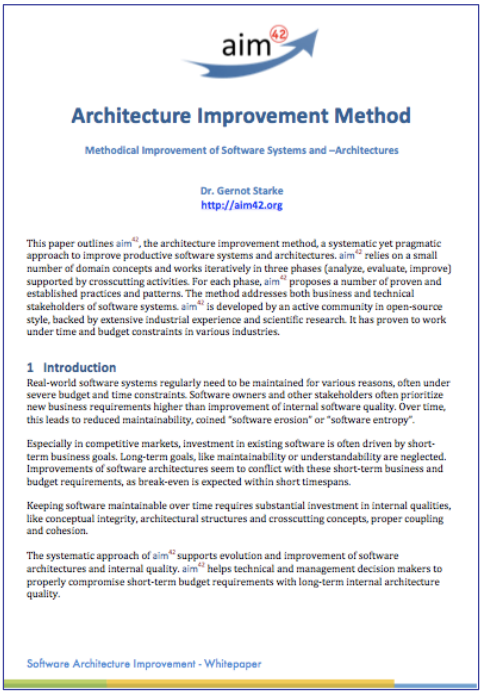
- **Method Guide**: AsciiDoc
- Gradle, Travis-CI
- Ergebnis:
  - HTML: http://aim42.github.io

© 2014 Dr. Gernot Starke / innoQ / aim42

Website



Whitepaper



aim42

Architecture Improvement Method

Methodical Improvement of Software Systems and –Architectures

Dr. Gernot Starke
http://aim42.org

This paper outlines aim42, the architecture improvement method, a systematic yet pragmatic approach to improve productive software systems and architectures. aim42 relies on a small number of domain concepts and works iteratively in three phases (analyze, evaluate, improve) supported by crosscutting activities. For each phase, aim42 proposes a number of proven and established practices and patterns. The method addresses both business and technical stakeholders of software systems. aim42 is developed by an active community in open-source style, backed by extensive industrial experience and scientific research. It has proven to work under time and budget constraints in various industries.

1 Introduction

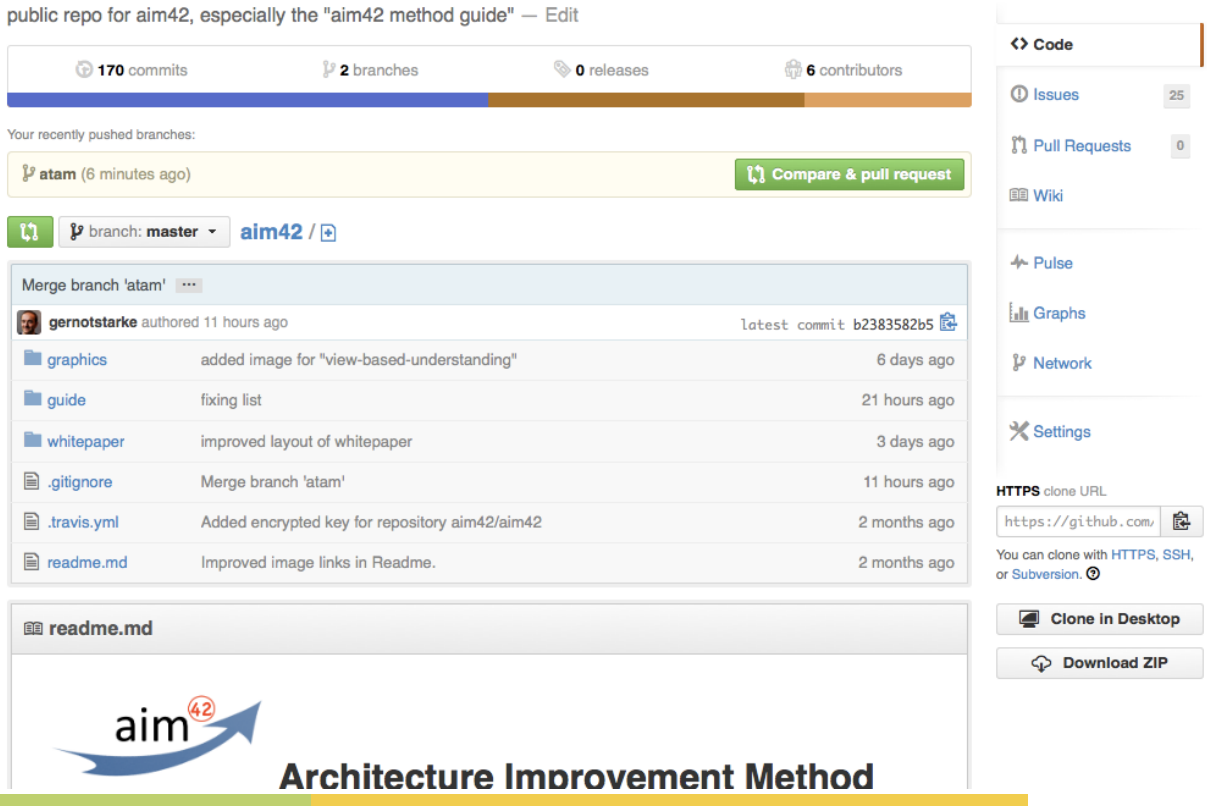
Real-world software systems regularly need to be maintained for various reasons, often under severe budget and time constraints. Software owners and other stakeholders often prioritize new business requirements higher than improvement of internal software quality. Over time, this leads to reduced maintainability, coined "software erosion" or "software entropy".

Especially in competitive markets, investment in existing software is often driven by short-term business goals. Long-term goals, like maintainability or understandability are neglected. Improvements of software architectures seem to conflict with these short-term business and budget requirements, as break-even is expected within short timespans.

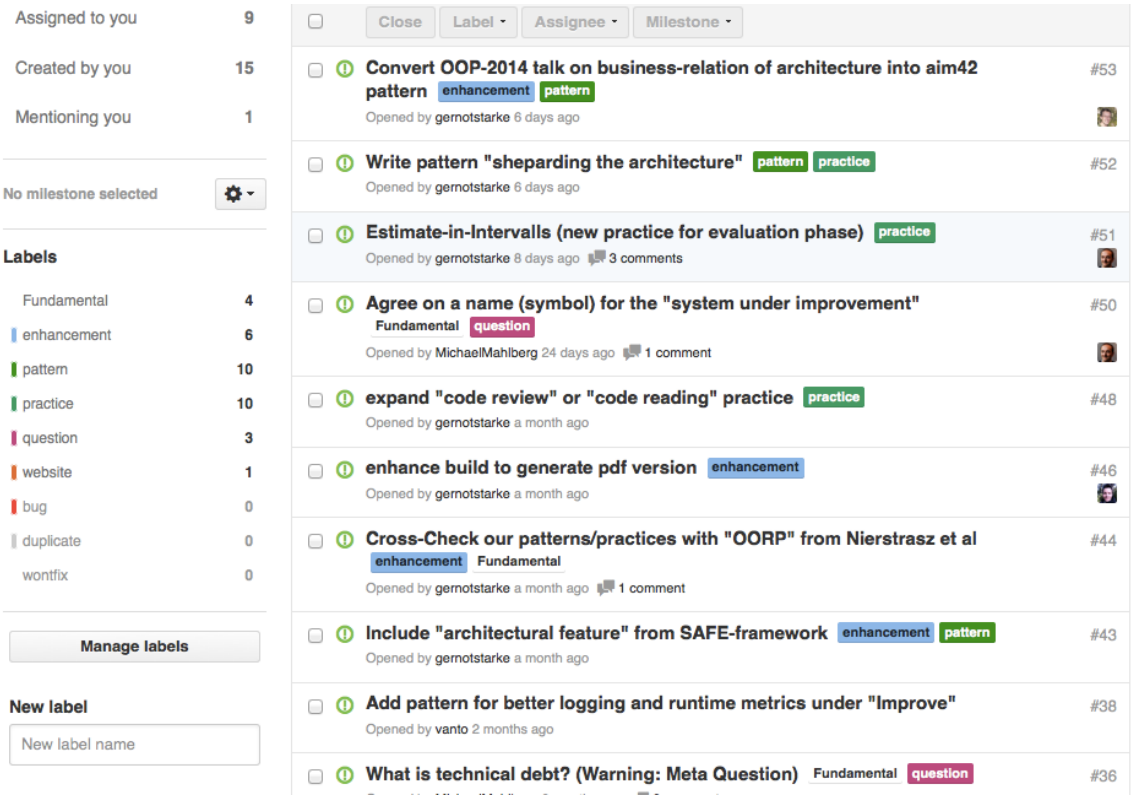
Keeping software maintainable over time requires substantial investment in internal qualities, like conceptual integrity, architectural structures and crosscutting concepts, proper coupling and cohesion.

The systematic approach of aim42 supports evolution and improvement of software architectures and internal quality. aim42 helps technical and management decision makers to properly compromise short-term budget requirements with long-term internal architecture quality.

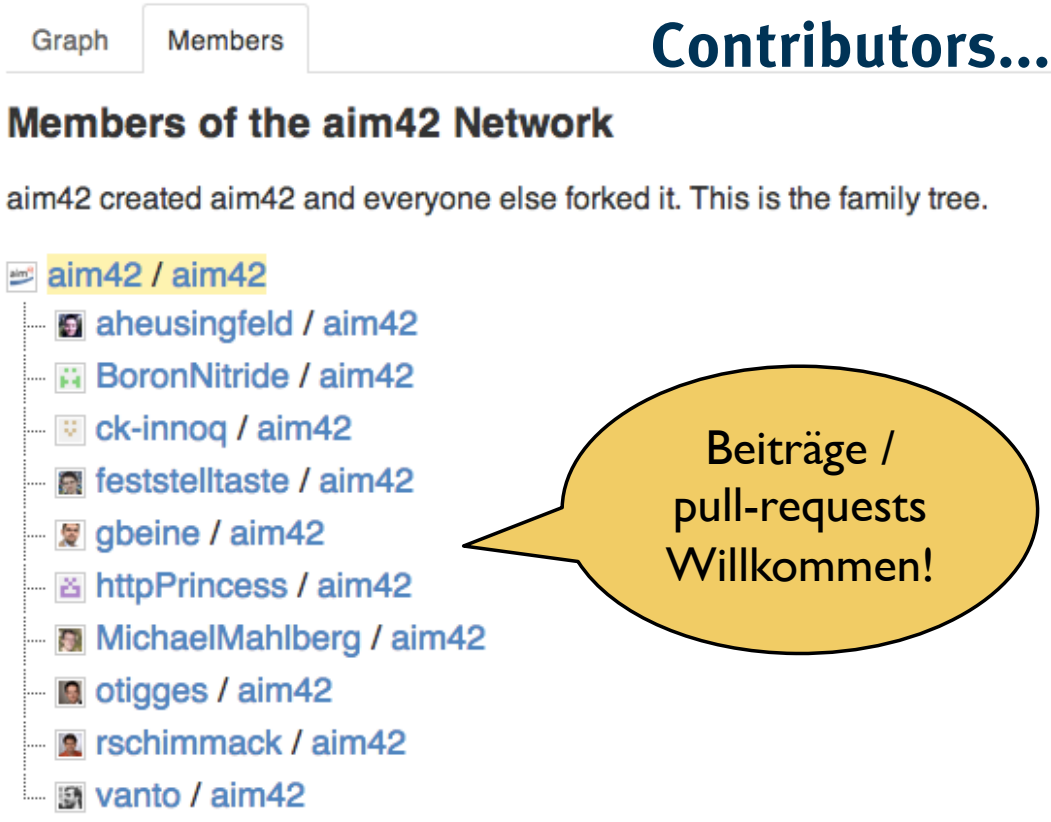
Code: github



Issues: viele...



Contributors...



Publicity

Publicity... (1): BT-Magazin Mai 2014

Publicity... (2): ECSA 2014 (August)
European Conference on Software Architecture

Banner © by ECSA Conference

ECSA 2014

The European Conference on Software Architecture (ECSA) is the premier European software architecture conference, providing researchers, practitioners, and educators with a platform to present and discuss the most recent, innovative and significant findings and experiences in the field of software architecture research and practice. In 2014, the conference will feature keynotes, research track, industrial track, workshops, tutorials, doctoral symposium, and tool demonstrations and poster presentations.

ECSA 2014 will take place at the University of Vienna, Austria, from August 25 to 29, 2014.

Contributions welcome

- Method guide: **http://aim42.github.io**
  - Source: https://github.com/aim42/aim42
- https://github.com/aim42/aim42/issues
- Twitter: @arc_improve42
- Mailing list: aim42@lists.innoq.com

Disclaimer & Legal Notice

- Graphics by
  - openclipart.org
  - aim42.org
- aim42 logo by Gernot Starke

- Licensed under
  Creative Commons Sharealike 4.0
- https://creativecommons.org/licenses/by-sa/4.0/