UNIRIO

OTIMIZAÇÃO APLICADA AO DESENVOLVIMENTO DE SOFTWARE E ALÉM

Prof. Márcio de Oliveira Barros
DIA – PPGI – UNIRIO
marcio.barros@uniriotec.br

SBSE

Search-based Software Engineering

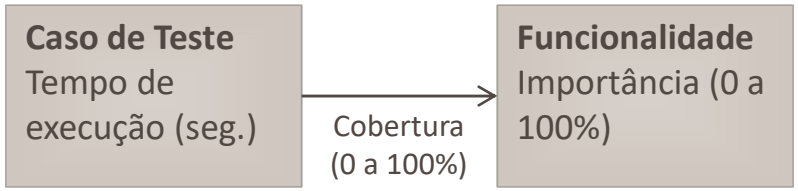
É uma área de pesquisa em que aplicamos técnicas de busca guiadas por funções de *fitness* para encontrar soluções aceitáveis para problemas de Engenharia de Software.

Search-based Software Engineering

Considere um grande conjunto de casos de teste que avaliam certas funcionalidades, demoram a rodar e atrasam o build de um sistema.

Problema: como escolher um subconjunto de testes que rode em até L minutos com o máximo de cobertura das funcionalidades?

- Modelo conceitual



- Modelo formal

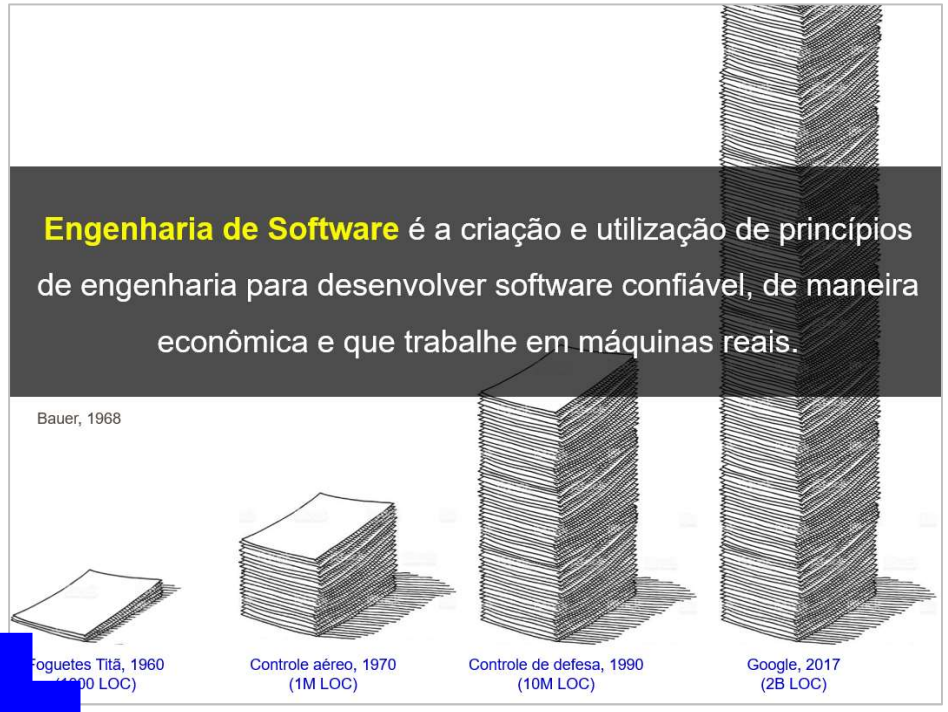
$$maximize\ coverage(T), given\ that\ exectime(T) \leq L$$

$$exectime(T) = \sum_{t \in T} exectime(t)$$

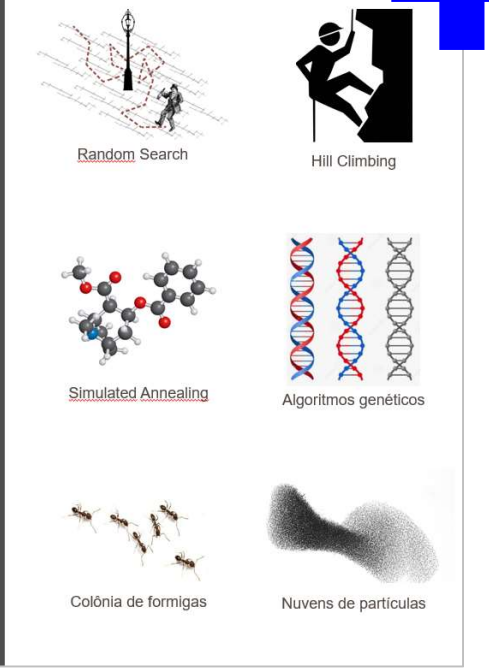
$$coverage(T) = \sum_{t \in T} \sum_{f \in F} coverage(t, f) * f.importance$$

Função de *fitness*

Diferencia uma solução boa de uma solução ruim.



Otimização heurística utiliza algoritmos baseados em uma analogia natural para resolver problemas de otimização de forma mais eficiente do que as técnicas matemáticas clássicas.



Representação

Consegue tratar um número muito grande de combinações (soluções alternativas).

Problemas típicos da Engenharia de Software

Quero atender o máximo de clientes na próxima versão do software.

Quero aumentar a produtividade da minha equipe.

Quero descobrir testes que façam a minha aplicação falhar.

Quero aumentar a coesão e reduzir o acoplamento no *design* do software.

Quero melhorar as métricas A, B e C no meu código.

Quero que meu código rode mais rápido.

Quero maior cobertura nos meus testes unitários.

Dá para perceber a relação com otimização?

Quero atender o máximo de clientes na próxima versão do software.

Quero aumentar a produtividade da minha equipe.

Quero descobrir testes que façam a minha aplicação falhar.

Quero aumentar a coesão e reduzir o acoplamento no *design* do software.

Quero melhorar as métricas A, B e C no meu código.

Quero que meu código rode mais rápido.

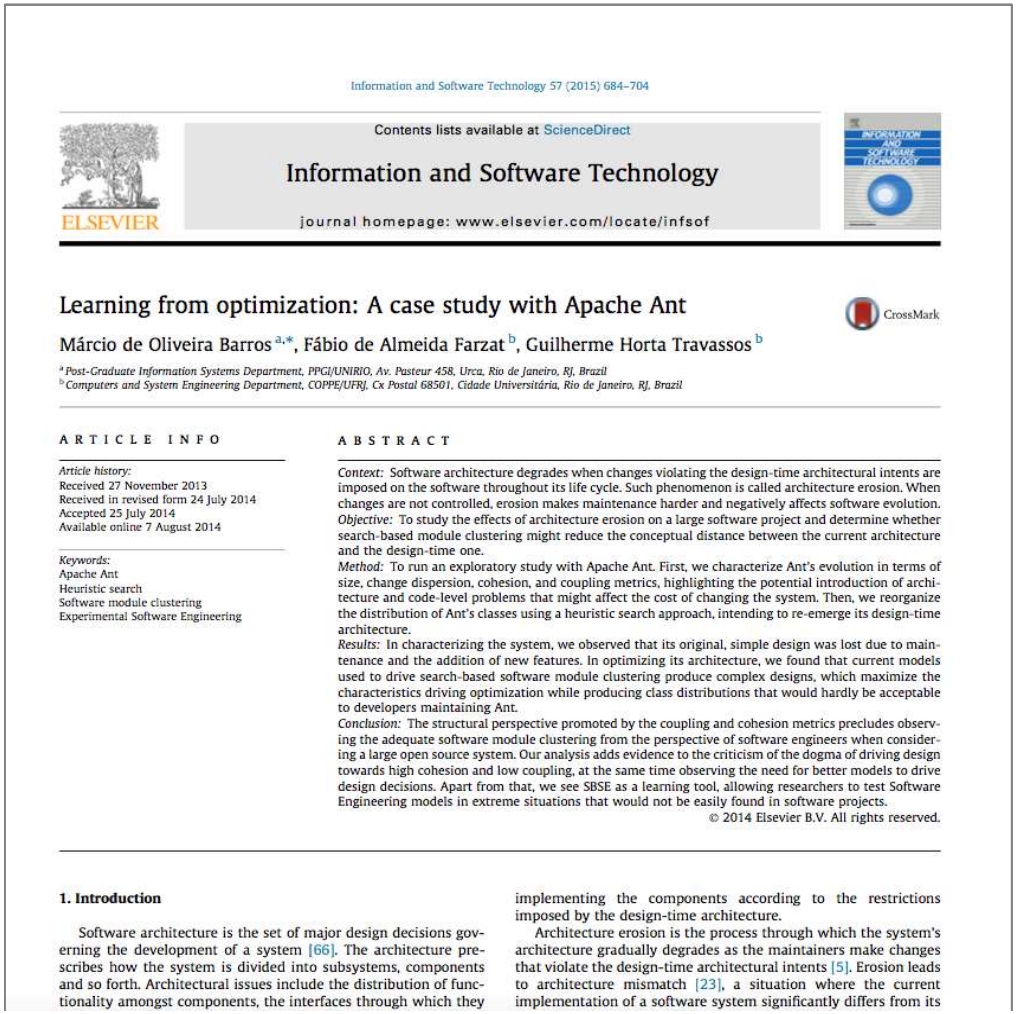
Quero maior cobertura nos meus testes unitários.

Mas com que tipo de problema o meu grupo de pesquisa trabalha?

Um exemplo: Projeto de Software

- Análise da arquitetura do Apache Ant, ferramenta de automação de *build* e CI para software desenvolvido em Java
- Arquivos XML descrevem as tarefas necessárias para compilar, testar e distribuir o software

IST, 2015



Information and Software Technology 57 (2015) 684–704

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Learning from optimization: A case study with Apache Ant

Márcio de Oliveira Barros [a,*], Fábio de Almeida Farzat [b], Guilherme Horta Travassos [b]

[a] Post-Graduate Information Systems Department, PPGI/UNIRIO, Av. Pasteur 458, Urca, Rio de Janeiro, RJ, Brazil
[b] Computers and System Engineering Department, COPPE/UFRJ, Cx Postal 68501, Cidade Universitária, Rio de Janeiro, RJ, Brazil

ARTICLE INFO

Article history:
Received 27 November 2013
Received in revised form 24 July 2014
Accepted 25 July 2014
Available online 7 August 2014

Keywords:
Apache Ant
Heuristic search
Software module clustering
Experimental Software Engineering

ABSTRACT

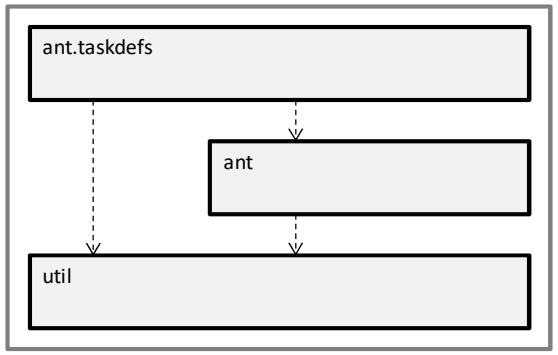
Context: Software architecture degrades when changes violating the design-time architectural intents are imposed on the software throughout its life cycle. Such phenomenon is called architecture erosion. When changes are not controlled, erosion makes maintenance harder and negatively affects software evolution.
Objective: To study the effects of architecture erosion on a large software project and determine whether search-based module clustering might reduce the conceptual distance between the current architecture and the design-time one.
Method: To run an exploratory study with Apache Ant. First, we characterize Ant's evolution in terms of size, change dispersion, cohesion, and coupling metrics, highlighting the potential introduction of architecture and code-level problems that might affect the cost of changing the system. Then, we reorganize the distribution of Ant's classes using a heuristic search approach, intending to re-emerge its design-time architecture.
Results: In characterizing the system, we observed that its original, simple design was lost due to maintenance and the addition of new features. In optimizing its architecture, we found that current models used to drive search-based software module clustering produce complex designs, which maximize the characteristics driving optimization while producing class distributions that would hardly be acceptable to developers maintaining Ant.
Conclusion: The structural perspective promoted by the coupling and cohesion metrics precludes observing the adequate software module clustering from the perspective of software engineers when considering a large open source system. Our analysis adds evidence to the criticism of the dogma of driving design towards high cohesion and low coupling, at the same time observing the need for better models to drive design decisions. Apart from that, we see SBSE as a learning tool, allowing researchers to test Software Engineering models in extreme situations that would not be easily found in software projects.

1. Introduction

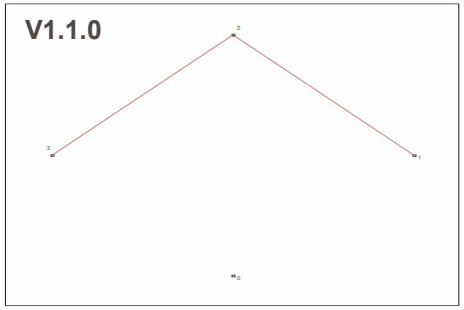
Software architecture is the set of major design decisions governing the development of a system [66]. The architecture prescribes how the system is divided into subsystems, components and so forth. Architectural issues include the distribution of functionality amongst components, the interfaces through which they implementing the components according to the restrictions imposed by the design-time architecture.

Architecture erosion is the process through which the system's architecture gradually degrades as the maintainers make changes that violate the design-time architectural intents [5]. Erosion leads to architecture mismatch [23], a situation where the current implementation of a software system significantly differs from its

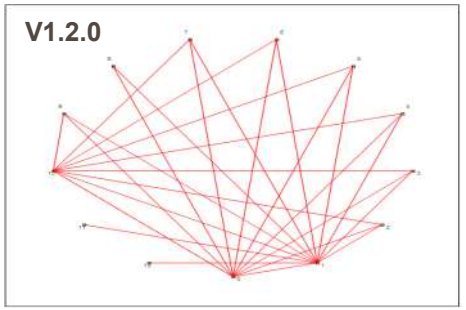
Projeto de software: arquitetura



C:102, P: 4 (07/2000)



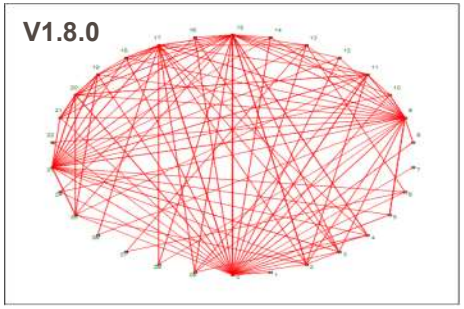
C: 173, P: 13 (10/2000)



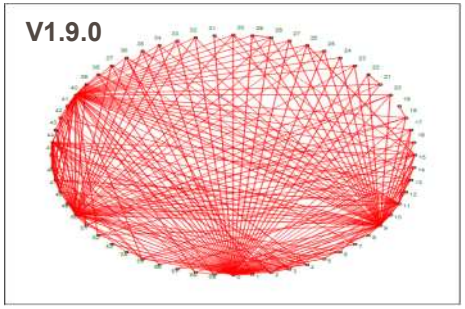
Simples, não?!

Apenas três componentes.

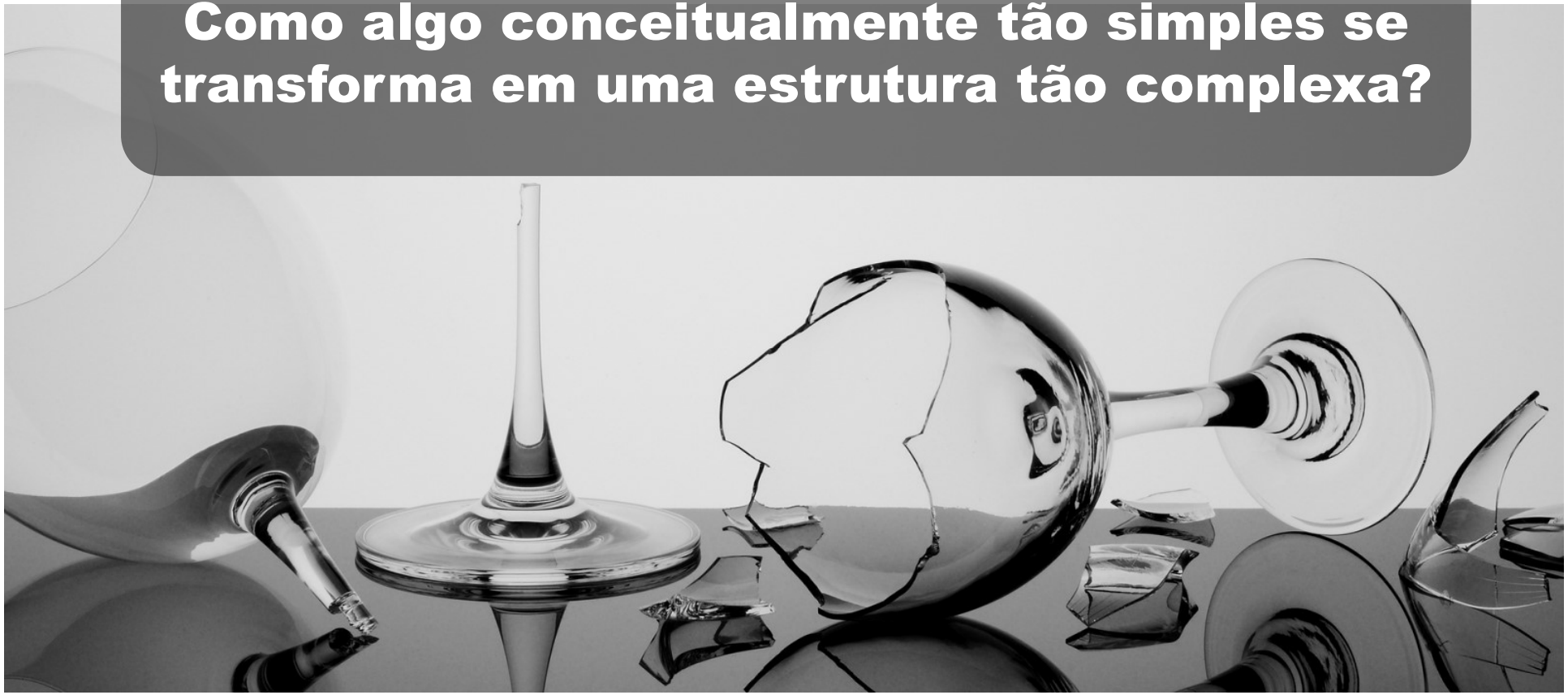
C: 870, P: 30 (10/2010)



C: 1116, P: 60 (03/2013)



Como algo conceitualmente tão simples se transforma em uma estrutura tão complexa?

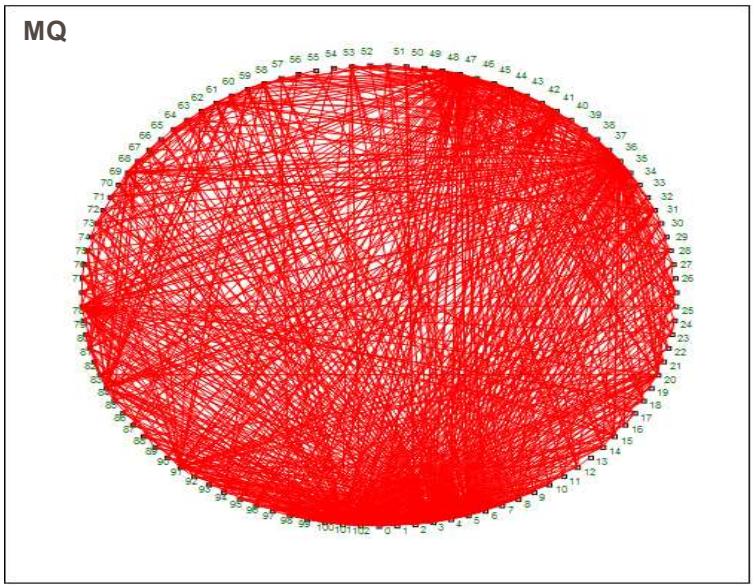


SBSE pode ajudar a recuperar a arquitetura inicial?

Resultados encontrados

Mesmo otimizando as métricas de coesão e acoplamento, a arquitetura não se torna mais simples. Muito pelo contrário ...

Original MQ: ~21 / Optimized MQ: ~100



Version	CBO ∇	AFF ∇	EFF ∇	LCOM ∇	MQ Δ	EVM Δ
Original v1.9.0	4.65	26.4	17.7	0.79	22.05	−49,639
EVM-optimized	4.48 ± 0.14	5.64 ± 0.19	5.88 ± 0.06	0.41 ± 0.01	73.62 ± 1.14	773.9 ± 7.89
MQ-optimized	7.78 ± 0.15	10.49 ± 0.18	8.68 ± 0.09	0.46 ± 0.01	101.63 ± 1.20	289.1 ± 15.27

Resultados encontrados

A otimização encontra maneiras inesperadas de melhorar a função de *fitness* (no caso, uma composição de coesão e acoplamento).

Como na programação, a otimização faz o que o pesquisador diz para fazer, não necessariamente o que o pesquisador quer!

Mas o que está além?

- A otimização leva nossas teorias e os modelos que as descrevem ao extremo ... e nos permite ver seu comportamento nestes cenários
- Mais do que resolver os problemas, a otimização pode nos ajudar a compreender a teoria subjacente da Engenharia de Software
- Quanto aprendemos sobre a teoria de desenvolvimento de software ao longo dos últimos 60 anos? O que sabemos sobre a organização ideal dos programas? Surpreendentemente pouco!
- Desenvolver software ainda é muito diferente de construir pontes, pois conhecemos pouco da ciência por trás do software
- A otimização pode ajudar a levar adiante este conhecimento!

E qual é a relação deste tipo de pesquisa com o grupo de Reutilização?

O que eu aprendi com o grupo?

- Continue pesquisando, mesmo que não haja um incentivo imediato para realizar este trabalho.

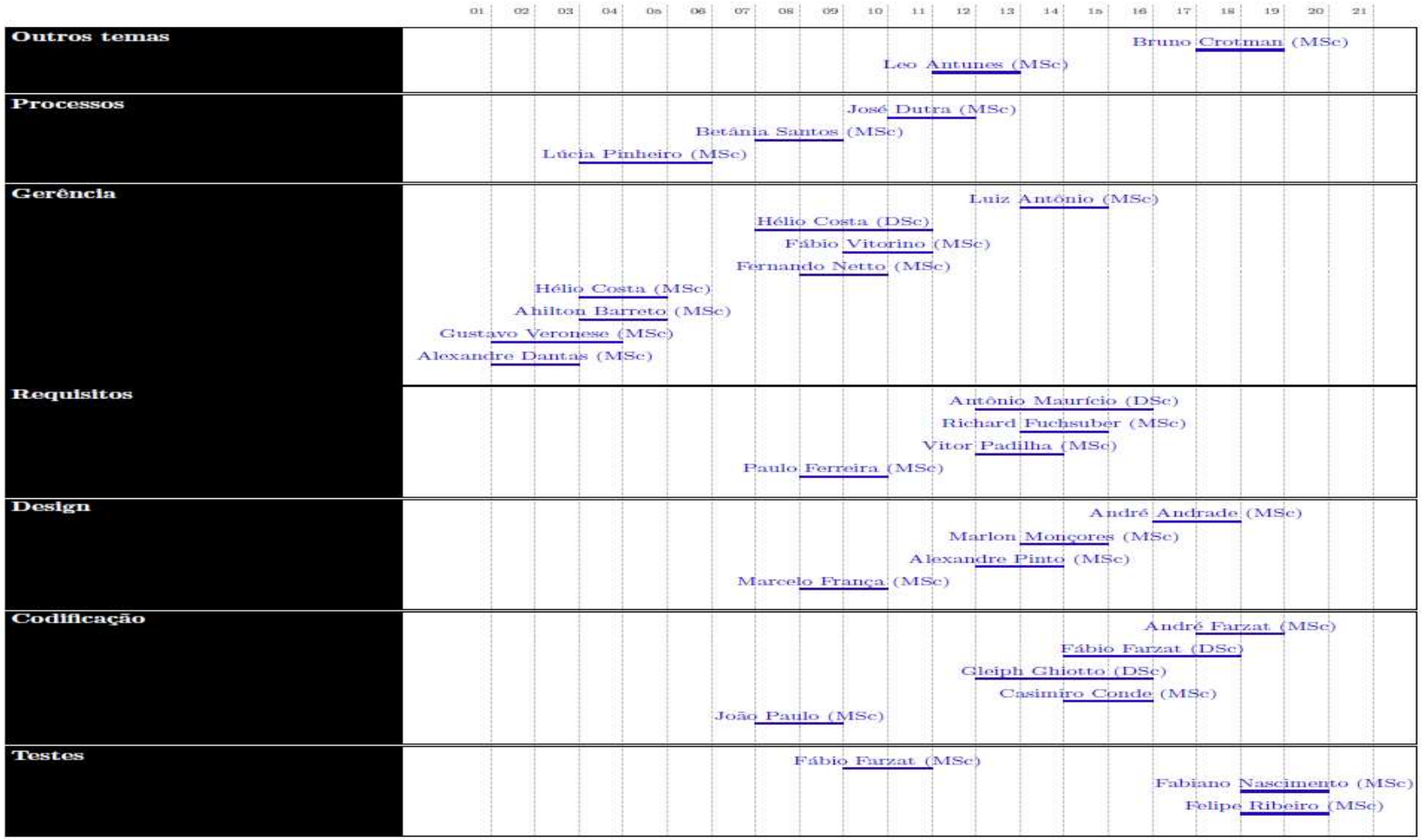
- Escolha uma área de pesquisa: é importante ser referência em uma área e fazer parte de um grupo!

- Escolha um pesquisador de referência na sua área para continuar aprendendo como se faz pesquisa.

- Avaliação experimental: oriente a sua pesquisa por ela.

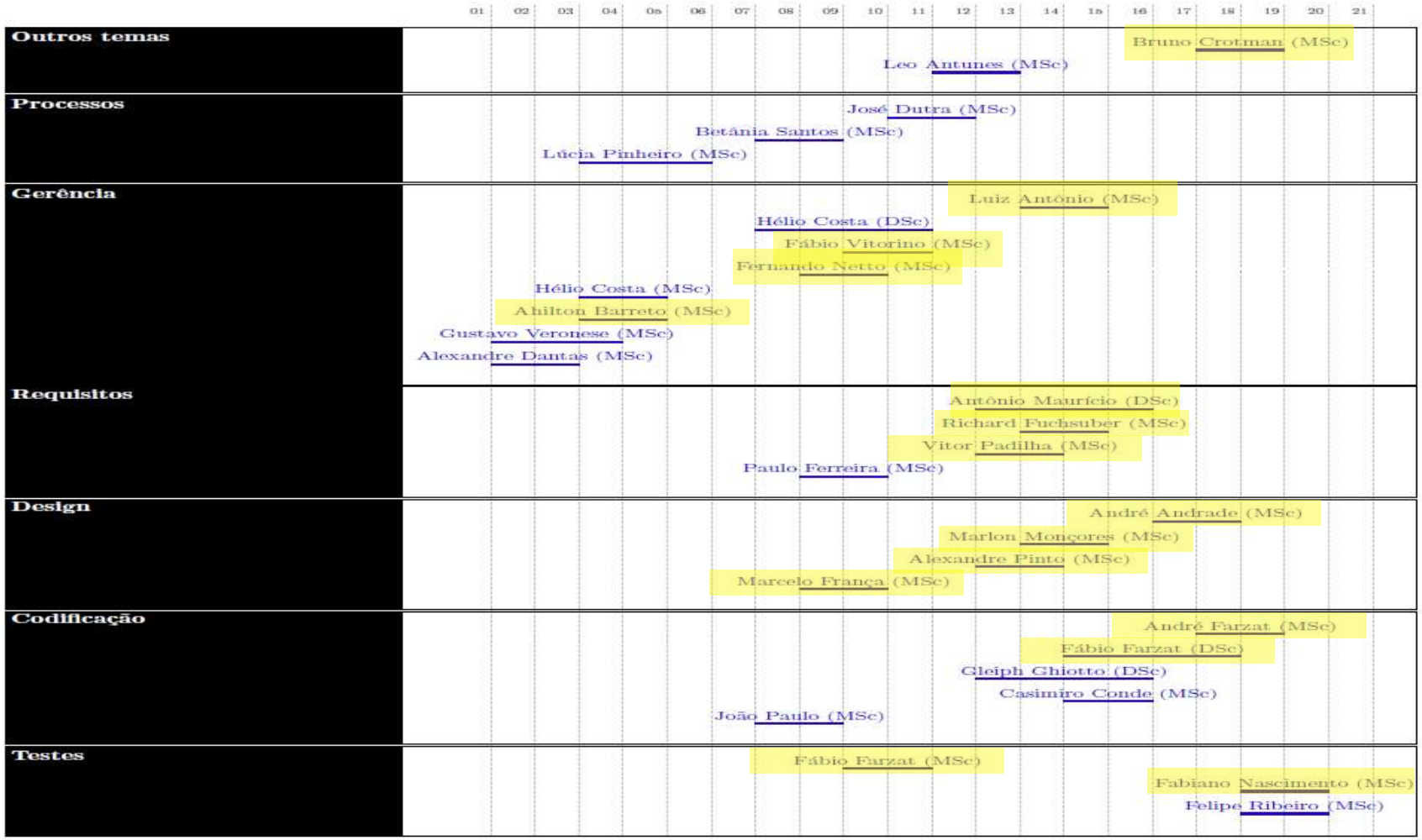
Um baile de debutantes

Apresentação dos netos do grupo: 2001 a 2020



Um baile de debutantes

Concentração na área de pesquisa de SBSE

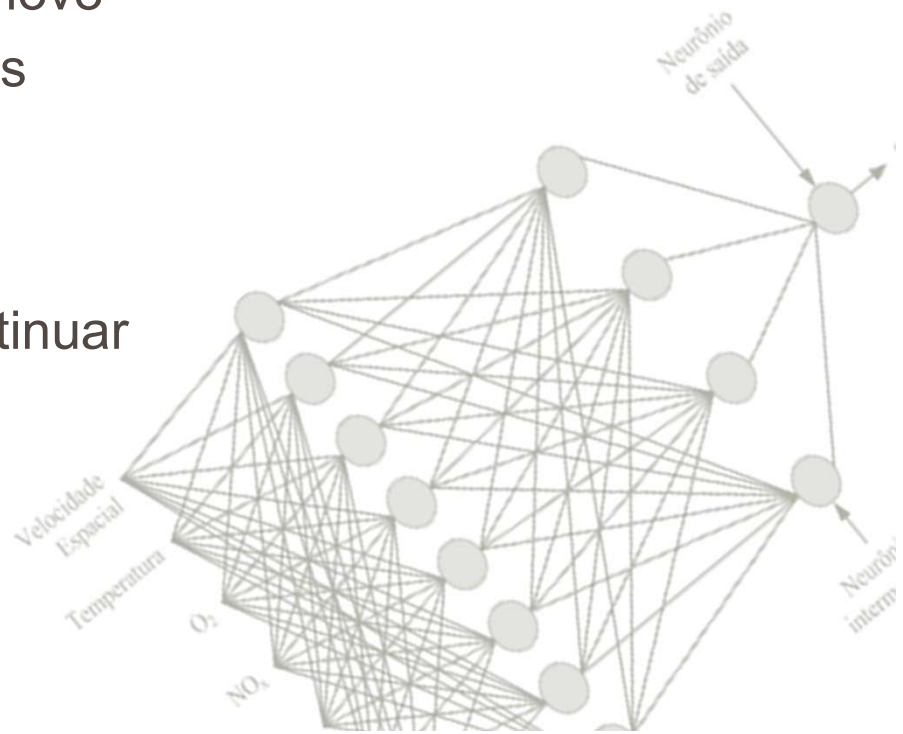


O caminho a frente ...

- De tempos em tempos, as áreas de pesquisa vão se esvaziando
  - Muitas retornam no futuro, mas perdem o interesse imediato
  - Na área de SBSE, houve uma grande fuga para a IA

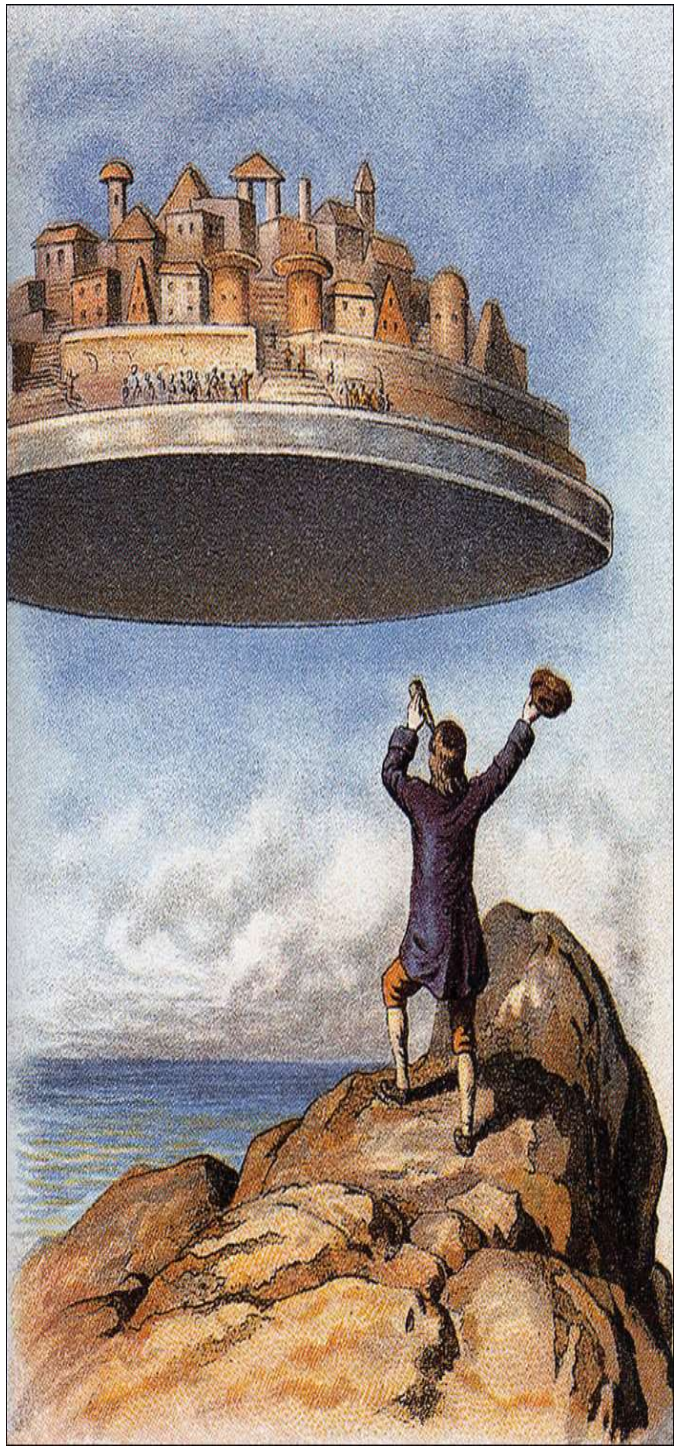
- No momento, estou em busca de um novo tema ou uma nova área para aplicar as técnicas de otimização ou NLP

- Também questiono se faz sentido continuar com tanto foco na Computação!



O caminho a frente ...

- Esta busca também tem a ver com o que a sociedade espera dos pesquisadores.
- Artigos publicados: mas de que valem os nossos artigos?
- Resultados aplicados: temos condições de desenvolvê-los e sustenta-los? Há interesse/demanda por eles?
- É difícil conseguir os dois juntos ...



Obrigado por sua atenção.

Dúvidas, comentários, críticas, ...