UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

RECUPERAÇÃO DE ARQUITETURA DE SOFTWARE UTILIZANDO ALGORITMOS DE AGRUPAMENTO

ALUNO: DENNIS LESSA DOURADO

ORIENTADOR: PROF DR RODRIGO ROCHA GOMES E SOUZA

INTRODUÇÃO

- Os softwares evoluem com passar do tempo, devido a:
  - Alterações na tecnologia ou nos requisitos; e
  - Atividades de manutenção
- Segundo Garlan a arquitetura do software é fundamental
  - Se bem projetada auxilia no cumprimento de requisitos não-funcionais
  - Alterações sem um bom conhecimento da arquitetura do software pode comprometer o funcionamento dos componentes do sistema

INTRODUÇÃO

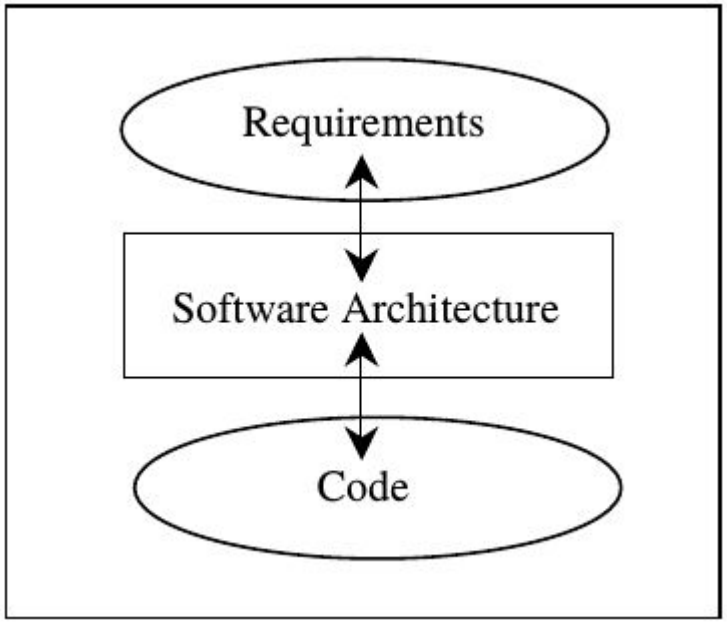
- Segundo Tzerpos e Holt muitos softwares não possuem a sua documentação de arquitetura atualizada
  - Dificultam a realização das atividades de manutenção
- Técnicas para recuperação de arquitetura tentam diminuir esse problema
  - Algoritmos de agrupamento
- O objetivo deste trabalho foi comparar algoritmos de agrupamento para a recuperação da arquitetura de software
  - Segundo o critério de *autoridade*, considerando o tempo de execução

REFERENCIAL TEÓRICO

REFERENCIAL TEÓRICO

Arquitetura de software

- Para Garlan a Arquitetura de Software:
  - Apresenta uma visão clara dos componentes do sistema, como se relacionam e suas propriedades;
  - É uma camada intermediária entre os requisitos do sistema e o código-fonte



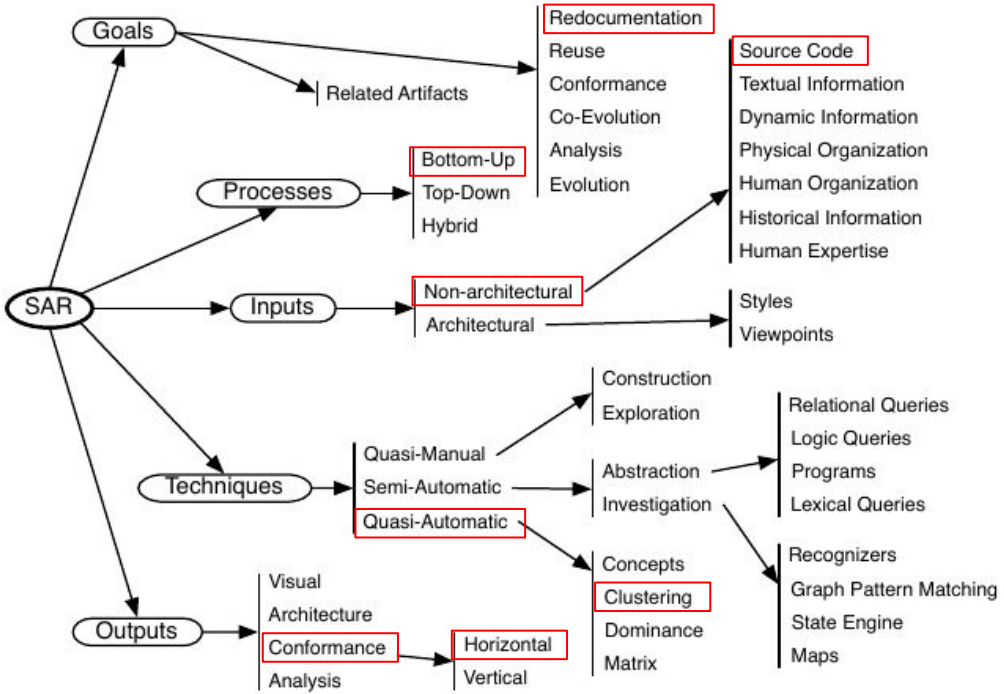
REFERENCIAL TEÓRICO

Recuperação de Arquitetura de Software

- Software Architecture Recovery (SAR)
- Ducasse e Pollet definem SAR como a reconstrução de visões arquiteturais de um sistema de software
- Eles propõem uma taxonomia para SAR em 5 aspectos:
  - Objetivos
  - Processos
  - Inputs
  - Técnicas
  - Outputs

REFERENCIAL TEÓRICO

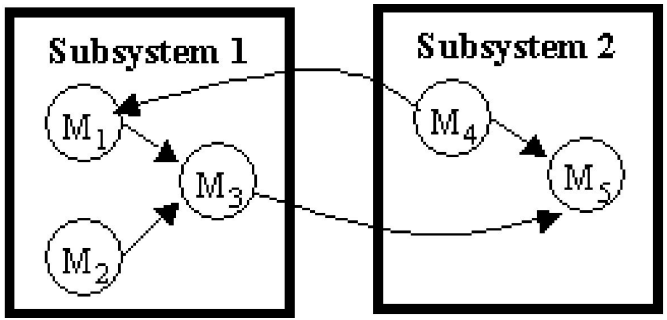
- Taxonomia de SAR segundo Ducasse e Pollet



REFERENCIAL TEÓRICO

Agrupamento de software

- O sistema é decomposto em subsistemas
  - Facilita a compreensão, manutenção e evolução do software
- Segundo Lung as técnicas de agrupamento visam maximizar a coesão e minimizar o acoplamento
  - **Coesão**: dependência de entidades de um mesmo cluster
  - **Acoplamento**: dependência entre clusters

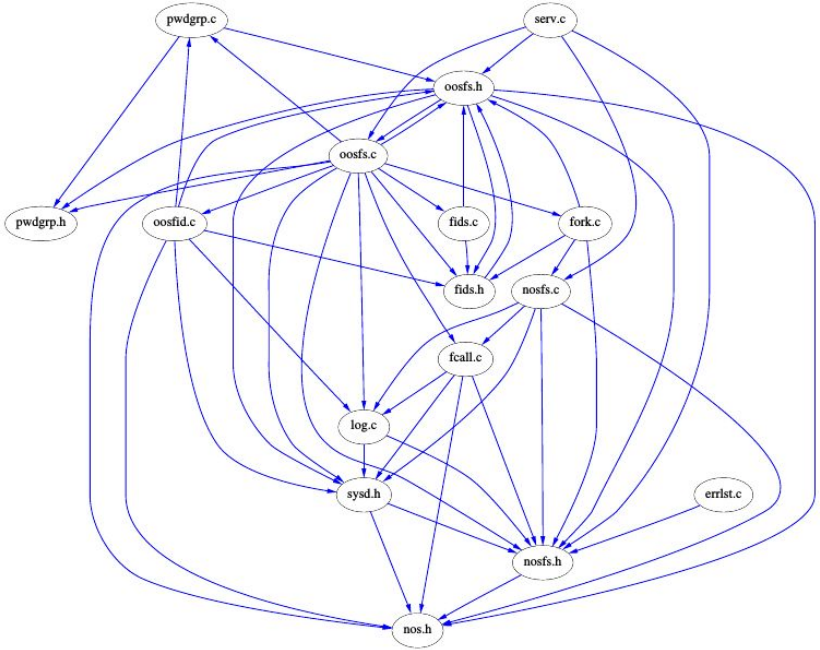


REFERENCIAL TEÓRICO

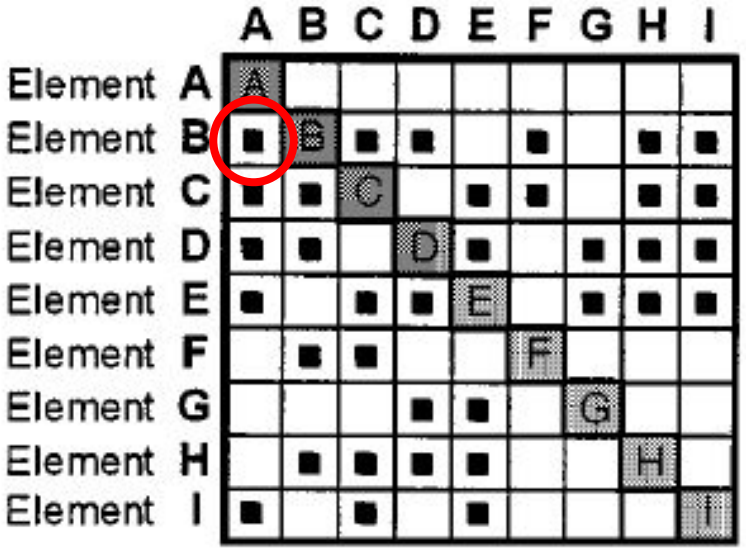
Extração de dependências

- Uma dependência é uma referência de uma entidade a outra do sistema
  - Atributos, variáveis, métodos, funções, herança
- As dependências podem ser representadas usando grafos orientados ou DSMs

REFERENCIAL TEÓRICO



GRAFOS



DSM

REFERENCIAL TEÓRICO

Algoritmos de agrupamento de software

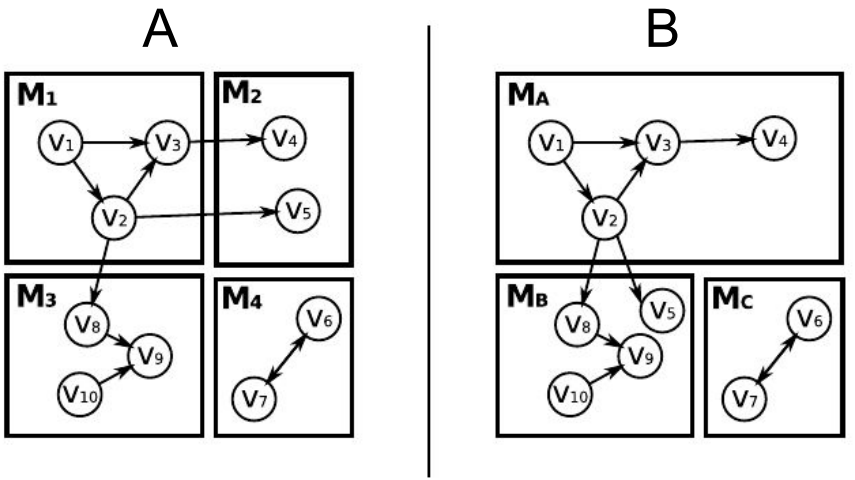
- Segundo Oliveira e Andrade os algoritmos mais utilizados são: ACDC, Bunch, LIMBO, WCA, ZBR e ARC
- Algoritmos escolhidos
  - Fácil encontrá-los implementados

Algoritmos	Determinístico	# Clusters Pré-definido
ACDC	SIM	NÃO
Bunch	NÃO	NÃO
LIMBO	NÃO	SIM

REFERENCIAL TEÓRICO

Avaliação dos algoritmos de agrupamento

- Critério de “*Autoridade*”
- Tzerpos e Holt propuseram a métrica MoJo
  - Permite apenas duas operações: **MO**ve e **JO**in;
  - Quantidade mínima de moves e joins para transformar um agrupamento em outro;
    - Quanto menor melhor



METODOLOGIA

METODOLOGIA

Visão geral

- Comparar agrupamentos gerados por algoritmos
  - Algoritmos: ACDC, Bunch (hill climbing e algoritmo genético) e LIMBO;
  - Critério: *Autoridade;*
- 3 sistemas foram escolhidos

Sistemas	Versão	Tamanho	Entidades	LOC
SIPOS	1.0.0	4,1 MB	215	19.557
InGE	1.0.0	3,8MB	227	59.723
Bash	4.2.53	6.7 MB	284	117.457

METODOLOGIA

Visão geral

- 30 execuções por algoritmo
  - 10 em cada sistema escolhido
- Dados coletados
  - Distância MoJo entre as arquiteturas recuperadas e de referência
  - Tempo de execução

METODOLOGIA

Ferramental

- Extração de dependência
  - Ferramenta Analizo
- Ferramentas foram utilizadas para a execução de cada algoritmo
  - ACDC, Bunch fornecidas pelos seus desenvolvedores
  - LIMBO -> Weka

METODOLOGIA

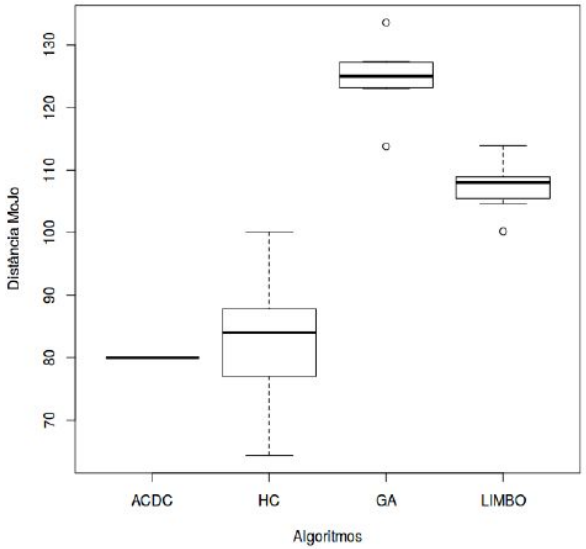
Execuções

- Primeira fase - Geração dos inputs
  - Geração da DSM
  - Conversão da DSM para o formato de input para cada algoritmo
- Segunda fase - Execução dos algoritmos
  - Configuração das ferramentas
    - LIMBO - Início com 5 cluster (incremento de 5);

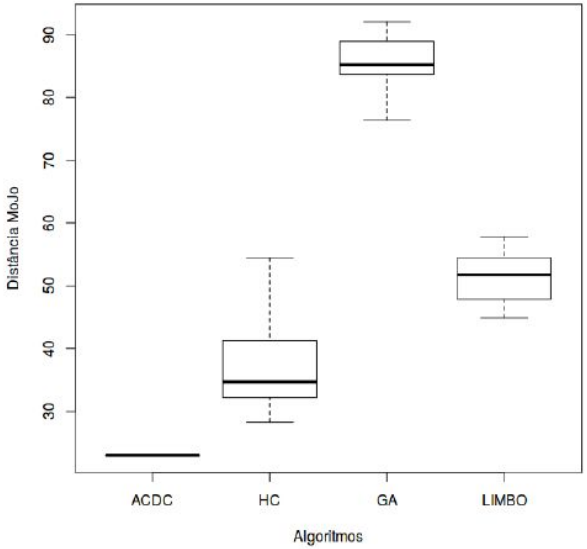
RESULTADOS E DISCUSSÃO

RESULTADOS E DISCUSSÃO

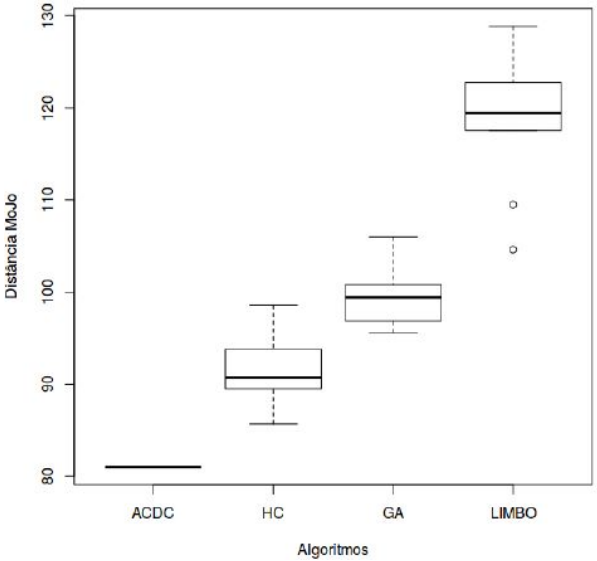
Resultados para distância MoJo



Bash



InGE



SIPOS

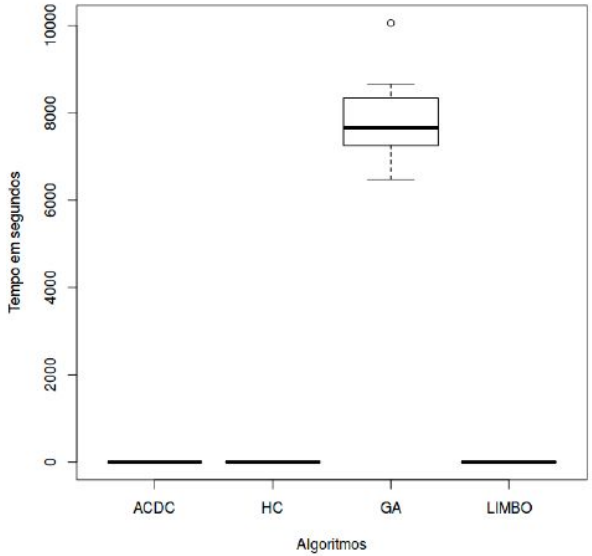
RESULTADOS E DISCUSSÃO

Resultados para distância MoJo

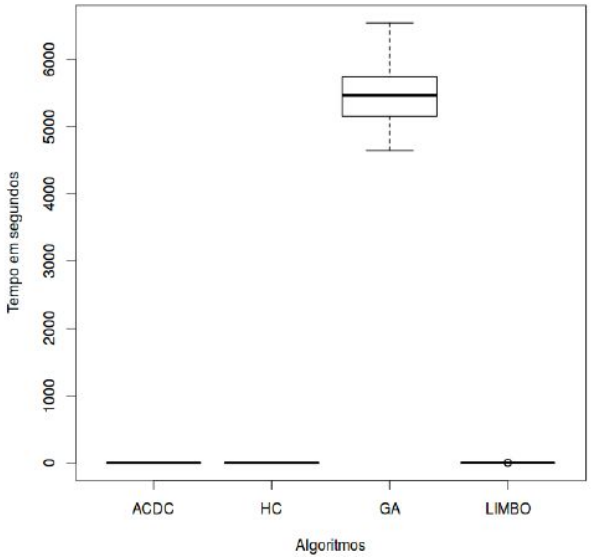
- O algoritmo ACDC foi melhor em todos os resultados
  - Seguido pelo algoritmo Hill Climbing
- O algoritmo LIMBO foi melhor que o Algoritmo Genético nos softwares:
  - Bash e InGE; e
  - Pior em SIPOS

RESULTADOS E DISCUSSÃO

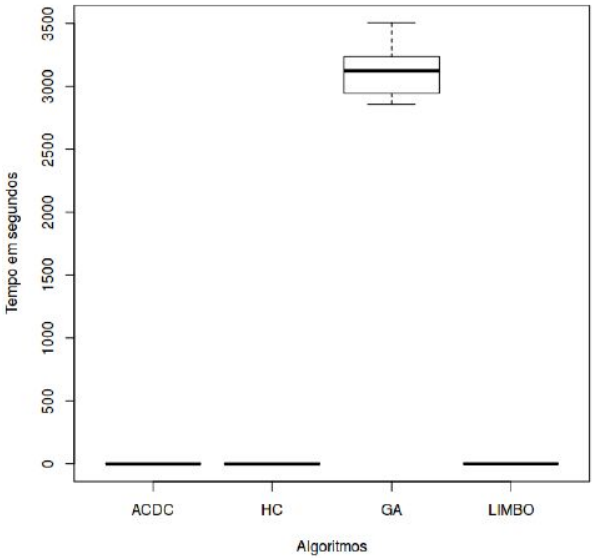
Resultados para tempos de execução



Bash



InGE



SIPOS

RESULTADOS E DISCUSSÃO

Resultados para tempos de execução

Algoritmo	Bash	InGE	SIPOS
ACDC	0,378	0,315	0,282
HC	0,315	0,141	0,109
AG	7951,433	5463,926	3039,518
LIMBO	1,084	1,063	1,063

RESULTADOS E DISCUSSÃO

Resultados para tempos de execução

- O Algoritmo Genético obteve os piores resultados
  - Altos valores para tempos de execução
- ACDC e Hill Climbing possuíram tempos de execução semelhantes
- LIMBO possuiu tempo de execução quase constante

RESULTADOS E DISCUSSÃO

Discussão

- LIMBO não se mostrou melhor que ACDC e Bunch como diz Andritsos e Tzerpos
  - Uso de ferramentas para extração da arquitetura
  - Interação com os desenvolvedores
- Os resultados obtidos mostram que Bunch é um pouco menos autoritário que ACDC, o que contradiz Wu
  - Este usou a hierarquia de diretórios

RESULTADOS E DISCUSSÃO

Discussão

- ACDC e Bunch Hill Climbing foram melhores para recuperação de arquitetura, como afirmou Wen
- Lutellier et al sugerem que:
  - LIMBO é mais apto quando se tem algum conhecimento da arquitetura; e
  - ACDC e Bunch são mais aptos para recuperar grandes arquiteturas;

CONCLUSÃO

CONCLUSÃO

- Os softwares evoluem rapidamente
  - Aumento da sua complexidade;
- Manter a documentação da arquitetura de software atualizada
  - Facilita a manutenção, não comprometendo o funcionamento do software
- Técnicas para recuperação de arquitetura de software auxiliam na obtenção da documentação
  - Utilizando algoritmos de agrupamento
- O algoritmo ACDC obteve os melhores resultados
  - Distância MoJo
  - Tempos de execução muito baixos

CONCLUSÃO

Limitações

- Comparar agrupamentos recuperados por algoritmos apenas com critério de “*autoridade*”
- Avaliação de poucos sistemas de software
  - Poucos softwares disponibilizam sua arquitetura de referência
    - Pouca variedade na quantidade de entidades

CONCLUSÃO

Trabalhos futuros

- Comparar algoritmos de agrupamento em softwares que possuem grandes variações na quantidade de entidades e usar mais critérios para avaliação
- Realizar estudo levando em consideração a organização física e código-fonte e verificando se o agrupamento sugerido pelo algoritmo está de acordo com a organização de pastas e código fonte do sistema testado

REFERÊNCIAS

1. David Garlan. Software architecture: a roadmap. In Proceedings of the Conference on the Future of Software Engineering, pages 91101. ACM, 2000.
2. Vassilios Tzerpos and Richard C Holt. On the stability of software clustering algorithms. In Program Comprehension, 2000. Proceedings. IWPC 2000. 8th International Workshop on, pages 211218. IEEE, 2000.
3. Stephane Ducasse and Damien Pollet. Software architecture reconstruction: A process-oriented taxonomy. IEEE Transactions on Software Engineering, 35(4):573591, 2009.
4. Chung-Horng Lung. Software architecture recovery and restructuring through clustering techniques. In Proceedings of the third international workshop on Software architecture, pages 101104. ACM, 1998.
5. S. S. Oliveira. L. P. T de ; ANDRADE. Um framework para recuperação arquitetural independente de plataforma, 2014.

REFERÊNCIAS

6. Periklis Andritsos and Vassilios Tzerpos. Information-theoretic software clustering. IEEE Transactions on Software Engineering, 31(2):150165, 2005.

7. Thibaud Lutellier, Devin Chollak, Joshua Garcia, Lin Tan, Derek Rayside, Nenad Medvidovic, and Robert Kroeger. Comparing software architecture recovery techniques using accurate dependencies. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 2, pages 69-78. IEEE, 2015.

8. Zhihua Wen. An optimal algorithm and extensions for the MoJo distance measure. PhD thesis, Citeseer, 2003.

OBRIGADO!