Relatório Técnico

Uma Revisão Sistemática sobre as Iniciativas Realizadas no Ensino de Arquitetura de Software

Claudia Susie Camargo Rodrigues
(susie@ufrj.br)

Cláudia Maria Lima Werner
(werner@cos.ufrj.br)

Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Rio de Janeiro, Maio de 2009

Uma Revisão Sistemática sobre as Iniciativas Realizadas no Ensino de Arquitetura de Software

Arquitetura de software é uma disciplina relativamente nova em engenharia de software, que surgiu com o crescimento de sistemas de software complexos. À medida que estes sistemas crescem, cresce também a necessidade de entender novas práticas, princípios, estratégias e padrões que atendam estes sistemas. Levando-se em consideração a importância desta disciplina na preparação dos estudantes para a prática do mercado de trabalho, este artigo descreve uma revisão sistemática sobre as iniciativas realizadas para ensinar Arquitetura de Software. Onde, revisão sistemática é uma revisão da literatura altamente estruturada usada para identificar, avaliar e interpretar toda pesquisa disponível e relevante sobre uma questão de pesquisa específica. Assim, o texto descreve como a revisão foi organizada e conduzida e apresenta os resultados segundo alguns critérios.

ABSTRACT

Software Architecture is a relatively new discipline in Software Engineering, that came up with the growth of complex software systems. As these systems increase, increase the need to understand new practices, principles, strategies and patterns that attend these systems. Taking into account the importance of this discipline in preparing students for the practice of the labor market, this article describes a systematic review of the initiatives conducted for the teaching of Software Architecture. Systematic review is a highly structured literature review used to identify, evaluate and interpret all available and relevant research to a particular research question. Thus, the text describes how the review was organized and conducted, and presents the results according to some criteria.

Sumário

1. **Introdução**

Nossa sociedade está cada vez mais dependente do software, com uma demanda grande pela sua qualidade. Esta sociedade, conectada via Internet, em busca de soluções ainda mais rápidas, requer melhores maneiras de se produzir sistemas. À medida que sistemas maiores e mais complexos vão surgindo, a Engenharia de Software (ES), assim como o seu ensino, torna-se vital. Estas são algumas das pressões sofridas pela educação de ES, que com o passar dos anos, tem se mobilizado para atender a estas novas demandas. Particularmente, a disciplina Arquitetura de Software (AS) assume um papel relevante neste cenário. A necessidade de evolução torna os produtos de software predispostos a defeitos, atrasos na entrega e custos acima do esperado. Seu escopo, complexidade e manutenção são cada vez mais significativas e exigem que profissionais da área se comuniquem por meio de componentes de software. A AS, ao incorporar a disciplina de reutilização, agrega suas vantagens, evitando retrabalho ou trabalho desnecessário, além de permitir aos engenheiros de software melhores decisões diante de alternativas de projeto (Boehm, 2006).

Pelas razões mencionadas acima, pretende investigar na literatura todo o material relevante sobre ensino de Arquitetura de Software. Para alcançar resultados com valor científico, decidiu-se realizar uma revisão sistemática, que ao contrário da revisão informal da literatura, o pesquisador segue um protocolo de revisão pré-estabelecido. Uma revisão sistemática visa estabelecer um processo formal para conduzir este tipo de investigação, evitando a introdução de eventuais vieses, ou seja, tendenciosidades, erros ou desvios sistemáticos do estudo, que se distanciam da verdade. Em contraste com o a revisão informal, a revisão sistemática investiga um tópico de pesquisa seguindo um protocolo bem definido e rigoroso de forma que outros profissionais possam reproduzir este mesmo protocolo (Biolchini *et al.*, 2005).

O presente relatório apresenta esta revisão, e discute seus resultados. O restante do texto está dividido em quatro seções. A Seção 2 descreve o objetivo do trabalho. A Seção 3 mostra o Processo para Realização da Revisão Sistemática de Literatura. A Seção 4 relata o planejamento da revisão sistemática e o protocolo preparado para a mesma. A Seção 5 descreve a condução desta revisão e os resultados obtidos. A Seção 6 apresenta os resultados da revisão sistemática, fazendo uso da categorização proposta para as iniciativas encontradas. A Seção 7 discute os resultados da análise e apresenta considerações finais.

2. Objetivo do Trabalho

Levando-se em consideração a importância da Arquitetura de Software na preparação dos estudantes para a prática do mercado de trabalho, em 2008 o grupo de Reutilização de Software da COPPE/UFRJ deu início a um projeto de pesquisa que visa investir no desenvolvimento de uma nova abordagem de ensino de ES, principalmente naquela que torna o ensino mais atraente para o aluno. A abordagem VisAr3D (Visualização de Arquitetura de Software em 3D) que tem como objetivo apoiar o ensino de AS, por meio da colaboração e troca de experiência entre alunos. Esta abordagem baseia-se no uso de Realidade Virtual e Realidade Aumentada (Azuma et al., 2001), para que os alunos possam visualizar a arquitetura de um sistema em três dimensões, interagir com ela e trocar experiências, utilizando esta tecnologia avançada e motivadora como apoio à ES. No entanto, antes de desenvolver o projeto de pesquisa é necessária a observação das práticas realizadas pela comunidade científica para ensinar esta disciplina, que serve como embasamento teórico para a pesquisa desenvolvida, contribuindo também para futuros refinamentos da abordagem.

3. Processo para Realização da Revisão Sistemática de Literatura

Conforme Biolchini (Biolchini *et al.*, 2005), esta revisão sistemática foi conduzida em três etapas: Planejamento da Revisão, Condução da Revisão e Publicação dos Resultados.

4. Planejamento da Revisão Sistemática

4.1. Questões de pesquisa e estrutura PICO

Objetivo

Identificar as iniciativas, práticas ou experiências realizadas para o ensino de ES que caracterizam uma tentativa de ensinar a disciplina AS ou parte dela, não representando, necessariamente, um curso completo de AS.

Questão de pesquisa:
Quais iniciativas foram realizadas no ensino de Arquitetura de Software?

População:
Publicação tendo em vista a Arquitetura de Software

Intervenção:
Ensino de arquitetura de software

Resultados:
Iniciativas identificadas

4.2. Fontes para busca, termos e sinônimos

Fontes:
Pesquisa nas seguintes bases de dados eletrônicas:
Compendex (em modo *Expert* Search): <http://www.engineeringvillage2.org/>
ACM (em modo Advanced Search): <http://portal.acm.org/dl.cfm>
Elsevier (em modo Search do Science Direct): <http://www.sciencedirect.com/>
IEEE (em modo Advanced Search): <httpieeexplore.ieee.org>
Springer: <http://www.springerlink.com>
Scopus: <http://www.scopus.com>

Termos e sinônimos utilizados na pesquisa:
Software Architecture - ;
Education – educational, training, teaching, learning, mentoring, course;
Initiative - experience, best practices, benefit, guideline, tool, method, technique, curriculum.

4.3. Strings de busca

A string de busca padrão utilizada nesta Revisão Sistemática foi a seguinte:

("software architecture")
AND
(education OR educational OR training OR learning OR teaching OR mentoring OR

course)
AND
(initiative OR experience OR "best practices" OR guideline OR tool OR method OR technique OR curriculum)

Para todas as bases de dados eletrônicas utilizadas, a string de busca padrão foi revisada, conforme a particularidade ou limitação de sua máquina de busca, visando observar a importância dos termos usados e a pertinênca das publicações retornadas.

Para a base da Compendex, a string de busca, previamente cercada por parêntes, foi acrecida ao final da expressão "wn KY", denotando que a busca será realizada considerando título, *subject e abstract*.

Para a base da Elsevier, adicionou-se ao início da string a expressão "title-abstr-key" após a inclusão de parêntes no início e no fim da string.

Para a base da IEEE, adicionou-se a string “<in>ab”.

Para a base da Springer, foi acrescentada no início, a expressão “su:()”, que limita a busca ao sumário.

4.3.1. Compendex, Elsevier e IEEE

string 1:
(("software architecture") AND (education OR educational OR training OR learning OR teaching OR mentoring OR course) AND (initiative OR experience OR "best practices" OR benefit OR guideline OR tool OR method OR technique OR curriculum))

string 2:

(("software architecture" OR "architecture design" OR "architectural representation" OR "architectural model") AND (education OR educational OR training OR learning OR teaching OR mentoring OR course) AND (initiative OR experience OR "best practices" OR benefit OR guideline OR tool OR method OR technique OR curriculum))

OBS: Foram acrescentados mais sinônimos de “software architecture”: “architecture design”, “architectural representation”, “architectural model”

string 3:
(("software architecture" OR "architecture design" OR "architectural representation" OR "architectural model") AND (education OR educational OR training OR learning OR teaching OR mentoring OR course) AND (initiative OR experience OR "best practices" OR benefit OR guideline OR tool OR method OR technique OR curriculum OR experiences OR initiatives OR benefits OR guidelines OR tools OR methods OR techniques OR curricula))

OBS: Foram acrescentadas palavras no plural.

4.4. Critérios para inclusão e exclusão de estudos

Os critérios definidos para inclusão e exclusão de estudo foram:

- Os documentos devem estar disponíveis na *Web*;
- Estudos sobre iniciativas que foram realizadas, especificamente, no ensino de Arquitetura de Software ou parte dela, não representando, necessariamente, um curso

completo da disciplina. .

- Os artigos que apresentavam palavras da *string* de busca no seu título ou no seu resumo.
- Os artigos escritos em inglês
- Foram excluídos artigos que apresentavam iniciativas de ensino de engenharia de software, sem o enfoque devido à Arquitetura de Software.

4.5. Processo de seleção dos estudos

1. O pesquisador executa a busca nas fontes selecionadas utilizando a *string* de busca elaborada.
2. Os artigos retornados pela busca são inseridos na ferramenta JabRef (JabRef, 2009).
3. O conjunto de artigos é selecionado a partir da verificação dos critérios de inclusão e exclusão. Esta verificação se dará pela leitura do resumo e do título do artigo.
4. Os artigos incluídos e excluídos são documentados no Formulário de Seleção de Estudos.

As cópias de todos os artigos incluídos como resultados da pesquisa inicial são revisados, inteiramente, por apenas um pesquisador.

4.6. Avaliação da qualidade dos estudos

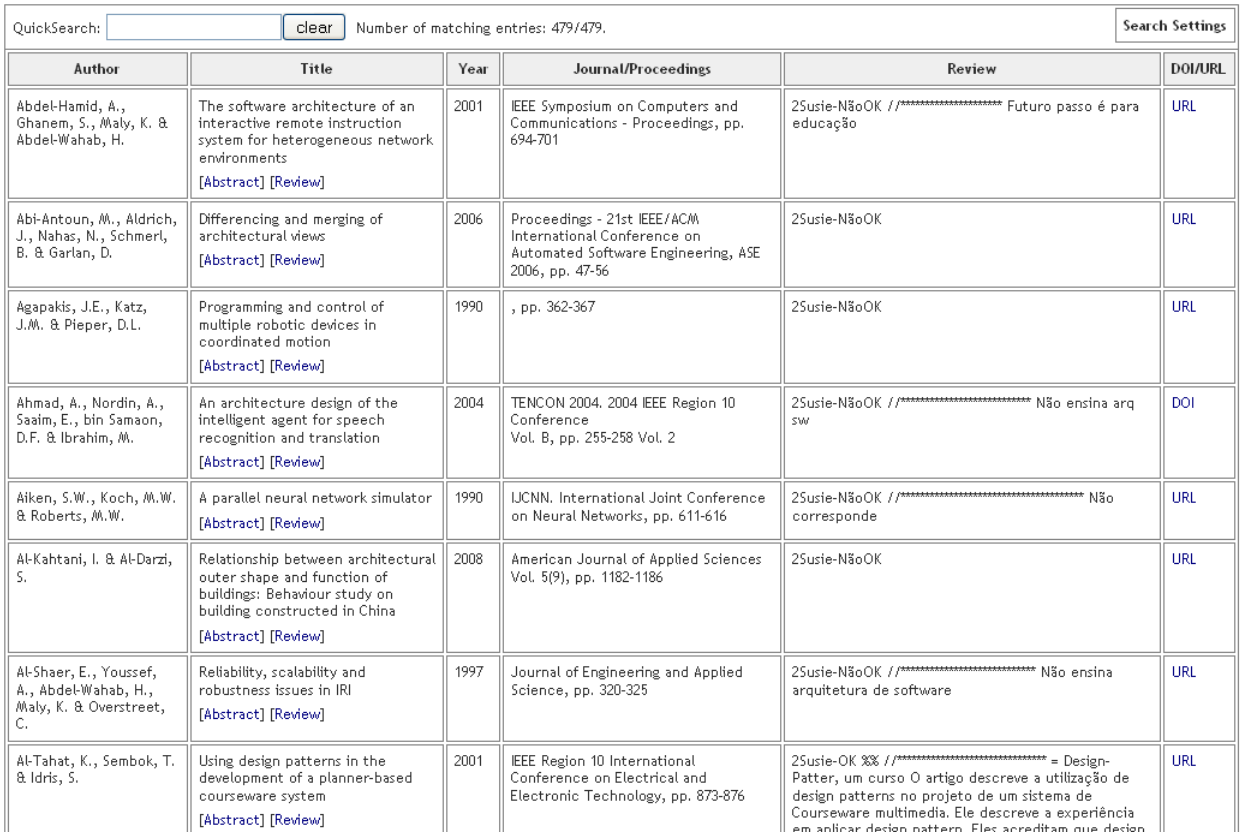
Procedimentos explícitos para avaliação da qualidade do material não foram preparados. A revisão se concentrou em procurar por estudos que descrevam iniciativas de ensino de Arquitetura de Software. A única questão considerada é que o artigo deve incluir uma descrição da prática de ensino, pois esta descrição fará parte dos dados a serem extraídos. Será considerado que as fontes dos documentos são confiáveis, e que os textos tenham passado por revisões externas que serviram de filtragem para que tenham qualidade suficiente para contribuir com a revisão sistemática.

4.7. Estratégia de extração de informações

Para cada estudo selecionado após a execução do processo de seleção, serão extraídas as seguintes informações:

- Título do documento
- Autor(es)
- Fonte
- Ano de publicação
- Pertinência do resultado da busca
- DOI/URL

A figura 1 ilustra parte do Formulário de Seleção de Estudos com as informações extraídas.



QuickSearch: clear Number of matching entries: 479/479. Search Settings

Author	Title	Year	Journal/Proceedings	Review	DOI/URL
Abdel-Hamid, A., Ghanem, S., Maly, K. & Abdel-Wahab, H.	The software architecture of an interactive remote instruction system for heterogeneous network environments [Abstract] [Review]	2001	IEEE Symposium on Computers and Communications - Proceedings, pp. 694-701	2Susie-NãoOK //********** Futuro passo é para educação	URL
Abi-Antoun, M., Aldrich, J., Nahas, N., Schmerl, B. & Garlan, D.	Differencing and merging of architectural views [Abstract] [Review]	2006	Proceedings - 21st IEEE/ACM International Conference on Automated Software Engineering, ASE 2006, pp. 47-56	2Susie-NãoOK	URL
Agapakis, J.E., Katz, J.M. & Pieper, D.L.	Programming and control of multiple robotic devices in coordinated motion [Abstract] [Review]	1990	, pp. 362-367	2Susie-NãoOK	URL
Ahmad, A., Nordin, A., Saaim, E., bin Samaon, D.F. & Ibrahim, M.	An architecture design of the intelligent agent for speech recognition and translation [Abstract] [Review]	2004	TENCON 2004. 2004 IEEE Region 10 Conference Vol. B, pp. 255-258 Vol. 2	2Susie-NãoOK //********** Não ensina arq sw	DOI
Aiken, S.W., Koch, M.W. & Roberts, M.W.	A parallel neural network simulator [Abstract] [Review]	1990	IJCNN. International Joint Conference on Neural Networks, pp. 611-616	2Susie-NãoOK //********** Não corresponde	URL
Al-Kahtani, I. & Al-Darzi, S.	Relationship between architectural outer shape and function of buildings: Behaviour study on building constructed in China [Abstract] [Review]	2008	American Journal of Applied Sciences Vol. 5(9), pp. 1182-1186	2Susie-NãoOK	URL
Al-Shaer, E., Youssef, A., Abdel-Wahab, H., Maly, K. & Overstreet, C.	Reliability, scalability and robustness issues in IRI [Abstract] [Review]	1997	Journal of Engineering and Applied Science, pp. 320-325	2Susie-NãoOK //********** Não ensina arquitetura de software	URL
Al-Tahat, K., Sembok, T. & Idris, S.	Using design patterns in the development of a planner-based courseware system [Abstract] [Review]	2001	IEEE Region 10 International Conference on Electrical and Electronic Technology, pp. 873-876	2Susie-OK %% //********** = Design-Patter, um curso O artigo descreve a utilização de design patterns no projeto de um sistema de Courseware multimedia. Ele descreve a experiência em aplicar design pattern. Eles acreditam que design	URL

Figura 1 - Formulário de Seleção de Estudo

5. Condução da Revisão

5.1. Análise dos documentos recuperados

A ferramenta JabRef version 2.4.2 (JabRef, 2009) foi o gerenciador de referências utilizado para manipular as publicações recuperadas pelas máquinas de busca. O JabRef identificou repetições, categorizou referências, priorizou leituras, recuperou o documento completo, disponibilizou campos para comentários e revisões etc. Novos campos foram customizados nesta ferramenta, bem como a elaboração de layouts específicos de exportação de relatórios.

Durante a etapa de Condução da Revisão, as fontes para a revisão sistemática são selecionadas, os estudos primários são identificados, selecionados e avaliados de acordo com os critérios de inclusão e de exclusão e de qualidades estabelecidos durante o protocolo da revisão (Mafra & Travassos, 2006).

As *strings* de busca preparadas foram executadas nas respectivas máquinas de busca das editoras selecionadas, como fontes no protocolo (IEEE, Elsevier, ACM, Springer e Compendex). No entanto, algumas máquinas de busca apresentaram limitações que impediram uma correta execução das *strings*.

A máquina de busca Springer apresentou algumas limitações ao utilizar a string de busca. A Springer limita em 10 termos as suas strings e portanto, foi necessário reduzir o número de termos utilizados na busca.

A máquina de busca da biblioteca digital da ACM não permitiu a execução das strings de busca, nem mesmo em seu módulo “Pesquisa Avançada”. As strings tiveram que ser revisadas

e divididas em 2 novas, de menor complexidade. No entanto, a pesquisa com as novas strings resultou em um número significativamente maior de artigos coletados. Por esta razão, o procedimento de execução da pesquisa foi revisto, novas buscas foram realizadas a posteriori, porém o número de artigos retornados impossibilita a execução da seleção conforme o protocolo definido.

5.2. Resultado das Buscas nas Bibliotecas Digitais

A tabela 1 expõe a quantidade de referências recuperadas de acordo com as máquinas de busca utilizada.

Tabela 1 – Artigos retornados nas máquinas de busca

Maquinas de busca	Artigos retornados
Compendex	436
Elsevier	21
IEEE	124
Scopus	430
Spring	12

A execução da revisão sistemática ocorreu no período de março a abril de 2009.

5.3. Análise dos documentos recuperados

Numa primeira avaliação superficial, foi feita a exclusão das referências sem disponibilidade de acesso pela Web e dos artigos repetidos acessados por máquinas de busca diferentes. A nova situação quantitativa ficou assim: 479 artigos. Posteriormente, em uma avaliação mais apurada e detalhada, foram selecionados os documentos candidatos a fazer parte da revisão sistemática. Foram excluídas as referências que nitidamente tratavam de outros assuntos não pertinentes à pesquisa. Finalmente, a seleção quantitativa ficou assim: 28 artigos selecionados.

5.4. Extração da informação

Após a seleção dos estudos, os dados dos estudos são extraídos e sintetizados para serem finalmente publicados durante a etapa de Publicação dos Resultados (Mafra & Travassos, 2006).

Na Tabela 2 são listados os artigos selecionados e as iniciativas de ensino realizadas.

Tabela 2 – Resultados Selecionados

(Al-Tahat *et al.,* 2001)	O artigo descreve a utilização de design patterns no projeto de um sistema de Courseware multimedia. Eles dizem que design pattern fornece um método melhor para ensinar o projeto e seu trabalho.
(Boer *et al.,* 2009)	O artigo mostra a experiência de um grupo de estudantes que aprendem sobre arquitetura de software. Eles criam uma arquitetura, discutem suas soluções com os outros estudantes e descobrem sempre que suas soluções não são satisfatórias ainda. Isto estimula a reflexão no retorno obtido e a encontrar alternativas.
(*Bucci et al., 1998*)	Ele descreve características importantes que o editor, que será construído, deve ter para ajudar os estudantes iniciantes, envolvendo noções de alto nível relacionadas à arquitetura de software.
(Chenoweth *et al.*, 2007)	É um curso de Arquitetura de Software onde prioriza a mudança de papéis dentro das equipes: clientes, arquitetos e desenvolvedores. Ele mostra os benefícios do aprendizado cooperativo enquanto os alunos são expostos a três tipos de perspectivas diferentes.
(Creighton & Singer, 2008)	O artigo diz que as responsabilidades de um futuro arquiteto de software devem estar associados não só às suas características técnicas mas também às suas habilidades sociais.
(Denzler & Gruntz, 2008)	O artigo diz que os alunos têm problemas para aplicar o conhecimento teórico em projetos concretos. Sua abordagem é utilizar design patterns em projetos contínuos para diminuir este problema.
(Dikel *et al,* 1997)	Ensina os fatores que determinam a eficiência do uso de arquitetura de software de linha de produto, tendo como base as lições aprendidas da companhia Nortel (Northern Telecom).
(Dongsun *et al.*, 2008)	Mostra a experiência de utilizar um toolkit como metodologia para ensinar arquitetura de software
(Fairbanks, 2003)	O artigo mostra a experiência de uma companhia financeira de ensinar arquitetura de software para um grupo. Ele mostra como é montado o curso e dá um destaque maior ao aproveitamento do grupo. O curso é baseado em leituras.
(Froyd *et al.*, 2007)	Este trabalho apresentou um estudo preliminar usando três design patterns para o ensino.
(Gast, 2008)	O artigo apresenta um curso em dois semestres sobre arquitetura de software com foco em padrões arquiteturais e de projeto. Eles investigaram a rastreabilidade.
(Grisham *et al.*, 2007)	O artigo mostra a experiência de um curso de arquitetura se projetado para os estudantes entenderem um projeto open source e evoluirem seu projeto arquitetural em resposta a uma série de requisitos funcionais adicionais.
(Heer & Agrawala, 2006)	O artigo apresenta um conjunto de 12 design patterns e discutem a estrutura, o uso e as suas interrelações.
(Hofmeister *et al.,* 2005)	Ele mostra sua experiência utilizando a Análise Global em quatro sistemas.
(Karam *et al.,* 2004)	O artigo apresenta um curso sobre arquitetura de software e projeto de software, dando enfoque em padrões.
(Kazman *et al.,*	O artigo apresenta um método de análise de arquitetura de

1996)	software que utiliza cenários para obter informações sobre alguns atributos de qualidade. Ele apresenta um estudo de caso, uma experiência em projeto de larga escala e as lições aprendidas.
(Lago & Van Vliet, 2005)	Este artigo relata a experiência de dois cursos de arquitetura de software com tempo de duração diferente. Eles não cobriam todos os tópicos de arquitetura de software, mas incentivavam a documentação da arquitetura, com foco nos stakeholders.
(Lee, 2003)	O artigo descreve um curso de Arquitetura de Software Web com tempos de duração diferentes. Seu objetivo é ensinar tanto fundamentos, quanto conhecimento prático para atender as necessidades da indústria.
(Maennistoe *et al.,* 2008)	O artigo descreve um curso de arquitetura de software que ensina questões de arquitetura relevantes na indústria. Foram adicionados benefícios em usar exemplos reais com restrições de sistemas existentes e aspectos não técnicos.
(McGregor *et al.*, 2007)	O artigo mostra a utilização de uma ferramenta ArchE (Architecture Expert) para produzir arquiteturas e para ensinar arquitetura de software.
(Naveda, 1999)	O artigo apresenta a estrutura de um curso introdutório de arquitetura de software e descreve sua experiência em sala de aula. Ele utiliza leituras e cursos em laboratórios. Ele enfatiza a utilização de estudos de caso. O foco do curso é em design patterns. E utiliza exemplos simples em seus projetos.
(Schauer & Keller, 1998)	O artigo foca em visualização da arquitetura com design pattern. Ele discute suas vantagens no aprendizado e mostra um protótipo.
(Svahnberg & Mårtensson, 2007)	Este artigo apresenta um guia de como se fazer um avaliação de arquitetura de software.
(Vallieswaran & Menezes, 2007)	O artigo apresenta uma ferramenta que será utilizada para o ensino de arquitetura de software: ArchKriti.
(Wang *et al.*, 2007)	O curso utilizou leituras, textos da indústria onde eram utilizados design patterns, engenharia de desempenho e arquiteturas em larga escala. Além de aplicar um experimento e a construção de um projeto de arquitetura de software em grupo.
(Wang & Stålhane, 2005)	O artigo descreve a experiência e os resultados de aplicar o método de análise “Post Mortem” em um projeto num curso de arquitetura de software.
(Wang & Sørensen, 2006)	O método "Writing as a tool for learning", que é um método pedagógico, pode melhorar o aprendizado de Arquitetura de Software que muitas vezes é difícil e não atrativo para os estudantes sem experiência prática.
Wang & Scannell, 2005)	O artigo mostra a ferramenta educativa desenvolvida (reliability modelling tool) que incorpora 4 tipos de estilos arquiteturais. Ela dá feedbacks instantâneos enquanto aprende a combinação destes 4 estilos (batch-sequential, parallel computing, fault tolerance, and client/server.).

6. Categorização das iniciativas encontradas

Posteriormente à execução da seleção dos resultados da revisão sistemática, as iniciativas encontradas foram classificadas de acordo com critérios propostos para facilitar a sua análise. Os critérios propostos foram:

- Curso (quantidade): aponta se a iniciativa proposta é apresentada no formato de um ou mais cursos.
- Projeto em larga escala (sim/não): informa se o enfoque do ensino é utilizar projetos em larga escala ou não.
- *Design Pattern* (sim/não): informa se é utilizado *Design Pattern* no curso.
- Estudo ativo (tipo de estudo ativo): aponta o enfoque dado às atividades dos alunos no processo de assimilação do conhecimento e habilidades, como a conversação dirigida, a discussão, o estudo dirigido individual e em grupo, os exercícios etc. (Libâneo, 1990).
- Abordagem utilizada (tipo da abordagem): mostra a abordagem utilizada para o ensino de arquitetura de software, que pode ser uma ferramenta, um projeto etc.

Tabela 3 – Resultados Classificados

	Referências	**Cursos**	**Projeto em larga escala**	**Design Pattern**	**Estudo Ativo**	**Abordagem Utilizada**
1	(Al-Tahat *et al.,* 2001)	1		Sim		
2	(Boer *et al.,* 2009)	1			Colaboração	
3	(*Bucci et al., 1998*)					Editor de arquitetura de software
4	(Chenoweth *et al.*, 2007)	1			Aprendizado cooperativo e mudança de papéis	
5	(Creighton & Singer, 2008)				Habilidades sociais	
6	(Denzler & Gruntz, 2008)	1		Sim		
7	(Dikel *et al,* 1997)		Sim			Experiência de uma companhia
8	(Dongsun *et al.*, 2008)	1			Instrutor, trabalho em grupo e atrativo para o aluno	Toolkit
9	(Fairbanks, 2003)	1			Colaboração	
10	(Froyd *et al.*, 2007)			Sim		
11	(Gast, 2008)			Sim		
12	(Grisham *et al.*, 2007)	1	Sim			
13	(Heer & Agrawala,			Sim		

	2006)					
14	(Hofmeister *et al.,* 2005)		Sim			Análise global
15	(Karam *et al.,* 2004)	1		Sim		
16	(Kazman *et al.,* 1996)		Sim			Método de análise baseada em cenários
17	(Lago & Van Vliet, 2005)	2		Sim		
18	(Lee, 2003)	1				Arquitetura de software web
19	(Maennistoe *et al.,* 2008)		Sim		Trabalho em grupo	
20	(McGregor *et al.*, 2007)				Instrutor	Ferramenta ArchE
21	(Naveda, 1999)	1		Sim		
22	(Schauer & Keller, 1998)		Sim	Sim		Protótipo
23	(Svahnberg & Mårtensson, 2007)					Avaliação de software
24	(Vallieswaran & Menezes, 2007)					Ferramenta Archkriti
25	(Wang *et al.*, 2007)	1	Sim	Sim		
26	(Wang & Stålhane, 2005)					Método De Análise Post Mortem
27	(Wang & Sørensen, 2006)				Comunicação	Método de ensino “Writing as a tool for learning”
28	Wang & Scannell, 2005)					Ferramenta

Na tabela 3 pode ser encontrada a classificação das iniciativas de ensino de arquitetura de software de acordo com os critérios propostos. Após a categorização das iniciativas, foi possível realizar a seguinte análise em relação a cada critério:

- Curso: Este critério destaca 12 artigos descrevendo experiência de ensino no formato de curso.
- Projeto em larga escala: 7 iniciativas que deram destaque à abrangência de sistemas complexos.
- *Design Pattern*: 10 iniciativas descreveram vantagens na utilização de *Design Pattern*.
- Estudo ativo: 8 iniciativas enfatizaram a importância de estudos ativos com destaque para maior inserção do instrutor, colaboração, aprendizado cooperativo, mudança de papéis, habilidades sociais, trabalho em grupo, atrativo ao aluno, comunicação.
- Abordagem utilizada: Ferramenta, método de ensino “writing as a tool for learning”, ferramenta Archkriti, avaliação de software, protótipo, ferramenta ArchE, método de análise baseado em cenários, arquitetura de software Web, análise global, toolkit,

experiência de uma companhia, editor de arquitetura de software.

7. Considerações finais e discussão dos resultados

Durante a sua condução, foram obtidos 479 registros. Após a execução da seleção dos resultados, feitas as devidas filtragens baseadas nos critérios definidos para inclusão e exclusão de estudos, os dados foram extraídos de 28 registros selecionados. Com este levantamento sistemático, percebeu-se que a academia já está se preparando para entrar em sintonia com as necessidades do mercado de trabalho, oferecendo cursos com projetos em larga escala. Apesar de ser predominante a utilização da sala de aula com o auxílio de material teórico e alguma prática. Iniciativas como (Dikel et al, 1997), (Grisham et al., 2007), (Hofmeister et al., 2005), (Kazman et al., 1996), (Maennistoe et al., 2008), (Schauer & Keller, 1998), (Wang et al., 2007) estão em sintonia com as idéias de Conn (2002), preparando os futuros profissionais de Engenharia de Software para o mercado de trabalho. Alguns artigos utilizam o recurso de estudo de casos para ensinar requisitos não funcionais em projetos reais.

Outro destaque foi que muitas iniciativas sugeriam a utilização de Padrões de Projeto (Design Pattern) (Coplien, 1995), com o objetivo de atingir qualidade de software em diferentes níveis de abstração, especialmente nos quesitos reutilização, modularidade e flexibilidade. Os artigos explicam que Design Pattern tem recebido muita atenção recentemente e é um tópico importante que tem sido usado de forma eficiente no ensino. Os artigos mencionam que os alunos têm problemas para aplicar o conhecimento teórico em projetos concretos e utilizando a abordagem de utilizar Design Patterns em projetos contínuos, diminui este problema. Os artigos que deram destaque ao Design Patterns foram (Wang et al., 2007), (Al-Tahat et al., 2001), (Denzler & Gruntz, 2008), (Gast, 2008), (Schauer & Keller, 1998), (Naveda, 1999) e (Lago & Van Vliet, 2005).

Alguns artigos compartilham a preocupação da monitoração mais presente por parte do instrutor. E outros voltados para o incentivo à comunicação dentro da equipe e o aprendizado cooperativo. Muitos deles utilizaram recursos que incentivam a mudança de papéis dentro do grupo e uma abordagem atrativa ao aluno. Segundo Huang & Distante (2006), os estudantes precisam aprender tanto aspectos técnicos, quanto aqueles não-técnicos no desenvolvimento de sistemas de software e os artigos selecionados enfocam a importância destes estudos ativos.

Chenoweth et al. (Chenoweth et al., 2007) descrevem um curso de Arquitetura de Software que prioriza a mudança de papéis dentro das equipes: clientes, arquitetos e desenvolvedores. Ele mostra os benefícios do aprendizado cooperativo enquanto os alunos são expostos a estes três tipos de perspectivas diferentes. O aprendizado cooperativo é um método próprio par dar este tipo de experiência porque suporta um tipo de trabalho em grupo empregado por arquitetos de software e projetistas. Este método enfatiza competências de grupos pequenos, a prática da comunicação face-a-face e reconhecer a responsabilidade individual para o sucesso do grupo. Creighton e Singer (Creighton & Singer, 2008) confirmam esta abordagem e acrescenta a necessidade de aumentar a percepção das forças e fraquezas individuais.

O artigo (Dikel et al, 1997) descreve os princípios críticos de sucesso de arquitetura de software, baseado em seus estudos: focar na simplificação, minimização e clareza; adaptar a arquitetura para as necessidades de futuros clientes, tecnologia, competição e objetivos de negócio; estabelecer um ritmo de arquitetura consistente e difundida e lançamento de produto que ajudem a coordenar as ações e expectativas de todas as partes; parceria com os stakeholders; manter uma visão da arquitetura clara; e fazer gerenciamento de riscos e oportunidades.

Num ciclo de aprendizado de arquitetura de software, os estudantes devem criar uma arquitetura de software, discutir sua solução com outros estudantes e descobrir que a sua

solução não satisfatória ainda. Segundo Boer et al.(Boer et al., 2009), isto estimula os alunos a refletirem sobre o feedback obtido e encontrar alternativas.

As iniciativas de ensino identificadas utilizavam abordagens que ensinavam a partir de métodos novos de ensino e de análise de arquitetura de software, do desenvolvimento de sistemas, de editores de arquitetura de software, de ferramentas desenvolvidas, da utilização de livros técnicos e de metodologias diversas. Cada relato contribuía com as lições aprendidas com a utilização daquela ferramenta, metodologia ou estratégia.

Bucci et al. (Bucci et al., 1998) seguem a seguinte posição: “É possível ensinar questões de nível arquitetural o mais cedo possível nos cursos de Ciência da Computação. Mas para o sucesso desta abordagem devem estar disponíveis ferramentas apropriadas para ajudar os alunos a construírem os seus modelos”. Para isso eles desejam construir um editor que ajude os estudantes nos seus modelos mentais iniciais que seja sofisticado o bastante para envolver noções de alto nível relacionadas à arquitetura de software e não à linguagem de programação.

O artigo (Lee, 2003) descreve um curso de Arquitetura de Software Web com tempos de duração diferentes. Seu objetivo é ensinar tanto fundamentos, quanto conhecimento prático para atender as necessidades da indústria.

Wang e Stålhane (Wang & Stålhane, 2005) utilizam um método chamado "Análise Post Mortem" (PMA) para suscitar pontos fortes e fracos do projeto ao avaliarem um projeto.

O artigo (Wang & Sørensen, 2006) apresenta o método "writing as a tool for learning". Este é um método simples, usado nas leituras, as quais são introduzidas pausas. Os alunos pensam no tópico e escrevem tudo que sabem sobre ele, compartilham informações com o grupo e, finalmente, comparam com o livro texto.

Wang e Scannell (Wang & Scannell, 2005) descrevem uma ferramenta de modelagem de confiabilidade que incorpora estilos arquiteturais, desenvolvida para facilitar os estudantes a utilizarem predição de qualidade. Esta ferramenta fornece aos estudantes feedbacks instantâneos e diminuem as curvas de aprendizado. Com uma interface gráfica fácil de usar, ela trabalha com os estilos: batch sequential, parallel, fault-tolerance e client/server.

Como parte de um curso, o artigo (Vallieswaran & Menezes, 2007) descreve o ArchKriti - A Software Architecture Based Design and Evaluation Tool Suite. Esta ferramenta dá suporte a passos importantes dentro do desenvolvimento baseado em arquitetura: requisitos dos stakeholders, projeto de arquitetura de um sistema, avaliação de arquitetura e documentação. Este é um projeto aberto que facilita a adição de novos estilos arquiteturais e visões.

A avaliação de arquitetura de software está se tornando uma ferramenta muito importante durante os vários processos de decisão no desenvolvimento de software. Decisões estratégicas são tomadas baseadas nos resultados das avaliações de arquiteturas de software. As avaliações são também usadas para garantir que os objetivos de qualidade sejam atingidos. Para se tornar um avaliador de arquitetura de software eficiente, é importante ganhar experiência em conduzi-las. Os artigos (Svahnberg & Mårtensson, 2007), (Boer et al., 2009), (Wang & Stålhane, 2005), (Wang & Stålhane, 2005), (Wang et al., 2007) e (Hofmeister et al., 2005) mostram experiências de cursos que dão destaque a avaliação de arquitetura de software.

O Architecture Expert (ArchE) (McGregor et al., 2007) é uma ferramenta de software que serve como um assistente de arquiteto de software que ajuda criar arquiteturas que tem níveis específicos de qualidades requeridas. O ArchE incorpora teoria de atributos de qualidade, técnicas para resolver modelos de atributos de qualidade associados a um projeto arquitetural para dar respostas para determinadas situações e a habilidade de usar projetos legados como entrada. Este artigo relata a ferramenta ArchE produzindo arquiteturas e ensinando sobre arquitetura de software.

O artigo (Hofmeister et al., 2005) utiliza a Análise Global para reduzir a distância entre a fase de requisitos e projeto de arquitetura. Ele afirma que a Análise Global serve para guiar o

processo de projeto, para capturar o raciocínio de projeto e dar suporte ao rastro entre requisitos e a arquitetura.

É importante destacar o destaque do livro ”Software Architecture in Practice, Second Edition”, utilizado nas iniciativas de ensino: (Wang et al., 2007), (Wang & Stålhane, 2005), (Wang & Sørensen, 2006) e (McGregor et al., 2007).

Os resultados representam um indicador da necessidade de aprofundar o entendimento da prática e de todas as competências ligadas à disciplina de arquitetura de software, para que os estudantes se tornem bons arquitetos. Foi observada, a importância da prática para causar o aprendizado da disciplina. Os especialistas demonstraram que ao ensinarmos os estudantes, somente métodos e técnicas, seus conhecimentos e competências se tornam frágeis. E ressaltaram, também, a necessidade de ser oferecida, a oportunidade dos estudantes projetarem sistemas mais complexos antes de saírem da faculdade, atendendo a demanda da indústria.

Este relatório descreveu a organização, a condução e os resultados de uma revisão sistemática sobre as iniciativas realizadas para ensinar Arquitetura de Software. Esta revisão serve de embasamento teórico para a pesquisa que está sendo desenvolvida pelo grupo de Reutilização de Software da COPPE/UFRJ, que visa o desenvolvimento de uma nova abordagem de ensino de Engenharia de Software, principalmente naquela que torna o ensino mais atraente para o aluno. A abordagem VisAr3D (Visualização de Arquitetura de Software em 3D) tem como objetivo apoiar o ensino de Arquitetura de Software. Todo o processo desta revisão sistemática foi desenvolvido por apenas um pesquisador, que infelizmente, aumenta o viés do estudo, mesmo com todo o critério empenhado e as iterações realizadas. Este é um ponto frágil da revisão, apesar da mesma possuir todo o protocolo formal.

Referências

Al-Tahat, K.S. and Sembok, T.M.T. and Idris, S.B., “Using design patterns in the development of a planner-based courseware system,” IEEE Region 10 International Conference on Electrical and Electronic Technology, Fac. of Info. Sci. and Technology, 43600 U.K.M. Bangi, Selangor, Malaysia: 2001, pp. 873–876.

Azuma, R.T., Baillot, Y., Behringer, R., Feiner, S., Julier, S., Macintyre, B.: Recent Advances in Augmented Reality. IEEE Computer Graphics and Applications, 21 (6), 34--47 (2001)

Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H., 2005. Systematic Review in Software Engineering. Technical Report ES 679/05. COPPE/UFRJ.

Boehm, B. A View of 20th and 21st Century Software Engineering. In: 28th International Conference on Software Engineering (ICSE’2006). pp. 12-29, Shanghai (2006).

Boer, Remco C. and Farenhorst, Rik and van Vliet, Hans, “A Community of Learners Approach to Software Architecture Education,” Software Engineering Education and Training, 2009. CSEET '09. 22nd Conference on, 2009, pp. 190-197.

Bucci, P., Long, T.J., Weide, B.W.: Teaching software architecture principles in CS1/CS2. In: Third international workshop on Software architecture.pp. 9-12, Orlando (1998).

Chenoweth, S. and Ardis, M. and Dugas, C., “Adapting cooperative learning to teach software architecture in multiple role-teams,” ASEE Annual Conference and Exposition, Conference Proceedings, Rochester Institute of Technology: 2007, pp. –.

Conn, R.: Developing software engineers at the C-130J software factory, IEEE Software, 19 (5), 25--29 (2002)

Coplien, J.O., Schmidt, D.C.: Pattern Languages of Program Design. Addison-Wesley, Reading (1995)

Creighton, O. and Singer, M., “Who leads our future leaders? on the rising relevance of social competence in software development,” Proceedings - International Conference on Software

Engineering, Siemens AG - Learning Campus, St.-Martinstr. 76, 81617 Munich, Germany: 2008, pp. 23–25.
Denzler, C. and Gruntz, D., “Design patterns: Between programming and software design,” Proceedings - International Conference on Software Engineering, UAS Northwestern Switzerland, Steinackerstrasse 5, CH 5210 Windisch, Switzerland: 2008, pp. 801–804.
Dikel, D. and Kane, D. and Ornburn, S. and Loftus, W. and Wilson, J., “Applying software product-line architecture,” Computer, vol. 30, Ago. 1997, pp. 49-55.
Dongsun, K. and Suntae, K. and Seokhwan, K. and Sooyong, P., “Software engineering education toolkit for embedded software architecture design methodology using robotic systems*,” Neonatal, Paediatric and Child Health Nursing, 2008, pp. 317–324.
Fairbanks, G.: Why can't they create architecture models like "Developer X"? An experience report. In: International Conference on Software Engineering, pp. 548 – 552, Portland (2003).
Froyd, J. and Layne, J. and Fowler, D. and Simpson, N., “Design patterns for faculty development,” Proceedings - Frontiers in Education Conference, FIE, Center for Teaching Excellence, Texas A and M University: 2007, pp. –.
Gast, H., “Patterns and traceability in teaching software architecture,” Principles and Practice of Programming in Java - Proceedings of the 6th International Conference, PPPJ 2008, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Sand 13: 2008, pp. 23–31.
Grisham, P.S. and Hawthorne, M.J. and Perry, D.E., “Architecture and design intent: An experience report,” Proceedings - ICSE 2007 Workshops:Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, SHARK-ADI'07, Empirical Software Engineering Laboratory (ESEL), University of Texas, Austin: 2007, pp. –.
Heer, J. and Agrawala, M., “Software design patterns for information visualization,” IEEE Transactions on Visualization and Computer Graphics, vol. 12, 2006, pp. 853–860.
Hofmeister, C., Nord, R.L., Soni, D.: Global Analysis: Moving from software requirements specification to structural views of the software architecture. IEE Proceedings: Software. Vol. 152, pp. 187 - 197 (2005).
Huang, S., Distante, D.: On Practice-Oriented Software Engineering Education. In: 19th Conference on Software Engineering Education and Training Workshops (CSEETW'06), pp. 15, North Shore Oahu (2006)
JabRef versão 2.4.2: http://jabref.sourceforge.net/index.php, acessado em: 04/05/2009.
Karam, O., Qian, K., Diaz-Herrera, J.: A model for SWE course "software architecture and design. In: 34th Annual Frontiers in Education (FIE 2004). Vol. 2, pp. F2C-4-8 (2004).
Kazman, R. and Abowd, G. and Bass, L. and Clements, P., “Scenario-based analysis of software architecture,” Software, IEEE, vol. 13, Nov. 1996, pp. 47-55.
Lago, P., van Vliet, H.: Teaching a Course on Software Architecture. In: 18th Conference on Software Engineering Education and Training (CSEE&T2005). pp. 35-42 (2005).
Lee, A.H., “A manageable web software architecture: Searching for simplicity,” SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education), School of Computing, University of Utah, Salt Lake City, UT 84112, United States: 2003, pp. 229–233.
Libâneo, J. C.: Didática. Editora Cortez, São Paulo (1990).
Mafra, S.N., Travassos, G.H.: Primary and Secondary Studies Supporting the Search for Evidence in Software Engineering. Technical Report [in Portuguese]. COPPE, Federal University of Rio de Janeiro (2006).
Männistö, T. and Savolainen, J. and Myllärniemi, V., “Teaching software architecture design,” 7th IEEE/IFIP Working Conference on Software Architecture, WICSA 2008, Nokia Research Center: 2008, pp. 117–124.
McGregor, J.D. and Bachman, F. and Bass, L. and Bianco, P. and Klein, M., “Using an architecture reasoning tool to teach software architecture,” Software Engineering Education Conference, Proceedings, Software Engineering Institute: 2007, pp. 275–282.

Naveda, J.F.: Teaching architectural design in an undergraduate software engineering curriculum. In: 29th Annual Frontiers in Education Conference (FIE '99). Vol. 2, pp. 12B1/1-12B1/4 (1999).

Schauer, Reinhard and Keller, Rudolf K., “Pattern visualization for software comprehension,” Program Comprehension, Workshop Proceedings, 1998, pp. 4–12.

Svahnberg, M. and Mårtensson, F., “Six years of evaluating software architectures in student projects,” Journal of Systems and Software, vol. 80, 2007, pp. 1893–1901.

Vallieswaran, V. and Menezes, B., “ArchKriti: A software architecture based design and evaluation tool suite,” Proceedings - International Conference on Information Technology-New Generations, ITNG 2007, K.R. School of Information Technology, I.I.T. Bombay, India: 2007, pp. 701–706.

Wang, W.-L. and Scannell, D., “An architecture-based software reliability modeling tool and its support for teaching,” Proceedings - Frontiers in Education Conference, FIE, School of Engineering and Engineering Technology, Penn State Erie, Behrend College, 5091 Station Road, Erie, PA 16563: 2005, pp. –.

Wang, A.I. and Sørensen, C.-F., “Writing as a tool for learning software engineering,” Software Engineering Education Conference, Proceedings, Dept. of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway: 2006, pp. 35–42.

Wang, A.I. and Stålhane, T., “Using post mortem analysis to evaluate software architecture student projects,” Software Engineering Education Conference, Proceedings, Dept. of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælandsv. 7-9, NO-7491 Trondheim, Norway: 2005, pp. 43–50.

Wang, A.I. and Arisholm, E. and Jaccheri, L., “Educational approach to an experiment in a software architecture course,” Software Engineering Education Conference, Proceedings, Dept. of Informatics, Univ. of Oslo, PO Box 1080, Blindem, N-0316 Oslo, Norway: 2007, pp. 291–298.