Metode SLR untuk Mengidentifikasi Isu-Isu dalam Software Engineering

Lusiana
Jurusan Teknik Informatika STMIK-AMIK Riau
lusi_dl@yahoo.co.id

Melva Suryani
Jurusan Teknik Informatika STMIK-AMIK Riau
melva.suryani@yahoo.com

Abstrak

Software Engineering (SE) merupakan disiplin ilmu yang berkaitan dengan semua aspek produksi software yang dimulai dari tahap awal spesifikasi sampai pada pemeliharaan sistem yang digunakan [1], oleh sebab itu SE memiliki peran penting terhadap perkembangan dunia teknologi informasi. Penelitian ini mengkaji isu dan pedekatan yang digunakan dalam SE empat tahun terakhir (2010-2013) yang belum teridentifikasi dengan baik, dengan menggunakan metode SLR. SLR merupakan cara untuk mengidentifikasi, mengevaluasi dan menafsirkan semua penelitian yang tersedia dengan pertanyaan penelitian tertentu, bidang topik, atau fenomena yang menarik [2] (Kitchenham 2007). Dengan metode SLR isu-isu yang berhubungan dalam software engineering akan diidentifikasi dengan menggunakan kata kunci "what the problem" (apa masalah) dan "how to solve the problem" (bagaimana memecahkan masalah tersebut). Hasil penelitian dapat dijadikan panduan oleh mahasiswa dan pembaca lainnya untuk melakukan penelitian yang berhubungan dengan SE selain itu, penelitian ini juga dapat membantu pembaca untuk mencari topik yang dominan penelitian dalam bidang SE.

Kata Kunci : SLR, Software Engineering

Abstract

Software Engineering (SE) is a scientific discipline that is concerned with all aspects of software production started from the early stages of the spsecification to maintenance systems used [1]. Therefore, the SE has an important role on the development of information technology world. This research examines the issues and re-purposed used in SE last four years (2010-2013) which has not been identified by either method by using the Systematic Literture Review (SLR). The SLR is a way to identify, evaluate, and interpret all the research available with the specific research question, topic areas, or phenomenon of interest [2]. The SLR realated issues in software engineering will be identified using the key words "what is the problem" and how to solve the problem. Results the study guide can be used by students and other readers to do research.

1. Pendahuluan

Software Engineering (SE) merupakan disiplin ilmu yang berkaitan dengan semua aspek produksi perangkat lunak dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan [1]. Penerapan SE dalam teknologi informasi tidak dapat berdiri sendiri, banyak disiplin ilmu yang mendukung keberhasilan dari SE tersebut, antara lain: Artificial Intelligence, Computer Science, Manajemen Proyek, Electric, Ekonomi, Sosial dan lain sebagainya.

Proses SE dimodelkan dengan menggunakan System Development Life Cycle (SDLC) atau daur hidup pengembangan sistem. SDLC merupakan metodologi untuk merancang, membangun, pengujian, implementasi, pemeliharaan dan sistem industrial [3]. Oleh karena itu SE merupakan cabang ilmu yang memiliki peran penting dalam dunia teknologi dan informasi, terutama dalam menghasilkan perangkat lunak (software) yang berkualitas tinggi. Namun, isu-isu yang ada didalam SE empat tahun terakhir dari taun 2010 sampai 2013, belum terindentifikasi dengan baik. Oleh sebab itu, agar isu-isu tersebut dapat diidentifikasi dan dianalisa serta dikaji dengan baik, maka metode yang digunakan adalah metode Systematic Literature Review (SLR) [4].

Pada awalnya SLR merupakan suatu teknik dari penelitian medis (Medical Riset) dan saat ini telah diimplementasikan kedalam SE. Dimana penyajian SLR akan mengikuti struktur yang telah disajikan oleh peneliti sebelumnya [5]. Hasil dari penelitian ini diharapkan dapat memperkaya literature tentang metode SLR. Penelitian ini dapat melihat perkembangan teknologi SE. Sebagai bahan referensi bagi pembaca yang tertarik atau berminat untuk melakukan penelitian dibidang SE. Diharapkan juga dapat menjadi panduan atau guideline bagi pembaca bagaimana menggunakan metode SLR dalam penelitian ilmiah atau domain tertentu.

Beberapa contoh penelitian yang telah dilakukan menggunakan metode SLR antara lain adalah: Prosedur untuk melakukan Systematic Review [6], Spesifikasi generasi persyaratan dari model Software Engineering [7], Systematic Literature Reviews dalam

software [5], Motivasi dalam Software Engineering [8], Tantangan dalam menjelajahi Service-Oriented System Engineering [9], Systematic Literature Reviews in Software Engineering [10], Panduan untuk melakukan Systematic Literature Reviews dalam Software Engineering, [2], Systematic literature reviews dalam software engineering [5], Systematic Literature Review untuk mengidentifikasi isu-isu dalam Bidirectional Model Transformation [4].

Dari uraian latar belakang diatas maka penulis tertarik melakukan penelitian mengunakan metode SLR untuk mengidentifikasi isu-isu yang berkembang (ada) didalam *SE* saat ini.

2. Landasan Teori

2.1. Systematic Literature Review (SLR)

2.1.1. Pengertian SLR

Systematic Review merupakan istilah yang digunakan untuk merujuk pada metodologi penelitian atau riset tertentu, pengembangan yang dilakukan untuk mengumpulkan dan mengevaluasi penelitian yang terkait pada fokus topik tertentu [10]. Beberapa peneliti yang telah melakukan riset dengan *SLR*, mendefinisikan *SLR* sebagai berikut:

a. *SLR* merupakan cara untuk mengidentifikasi, mengevaluasi dan menafsirkan semua penelitian yang tersedia dengan pertanyaan penelitian tertentu, atau bidang topik, atau fenomena yang menarik [2].
b. *SLR* adalah pendekatan *evidence-based* untuk mencari studi yang relevan dengan beberapa pertanyaan penelitian yang telah ditetapkan dengan memilih, menilai, dan mensintesis temuan untuk menjawab pertanyaan penelitian [9].
c. *SLR* suatu teknik penelitian untuk menganalisis *state of-the-art* dalam bidang pengetahuan tertentu dengan secara resmi mendefinisikan pernyataan masalah, sumber-sumber informasi, string search, kriteria inklusi dan eksklusi dari makalah yang ditemukan dalam). pencarian, analisis kuantitatif yang akan dilakukan (jika perlu), dan template untuk menemukan informasi yang dikumpulkan dari kertas atau *papers* [7].
d. *SLR* merupakan suatu teknik penelitian yang digunakan untuk mengkaji atau menemukan isu-isu yang terdapat dalam *Software Engineering* [4].

2.1.2. Tujuan SLR

Penelitian *SLR* dilakukan untuk berbagai tujuan, diantaranya digunakan untuk mengindentifikasi, mengkaji, mengevaluasi dan menafsirkan semua penelitian yang tersedia dengan bidang topik fenomena menarik dengan pertanyaan penelitian tertentu yang relevan [2]. Termasuk memberikan latar belakang teoritis untuk penelitian kedepannya, yang berguna sebagai panduan, bahan penelitian tentang topik yang menarik, atau menjawab pertanyaan-pertanyaan dengan memahami penelitian yang sebelumnya telah dilakukan.

2.1.3. Kelebihan dan Kekurangan SLR

Metode SLR memiliki kelebihan dan kelemahan seperti metode lain pada umumnya. Berikut kelebihan dan kekurangan SLR:

Kelebihan dan kelemahan dari SLR [2] adalah:

a. Dapat meningkatkan bukti dari penelitian sebelumnya, dan mewakili informasi dari berbagai pertanyaan penelitian yang tersedia dalam penelitian tersebut.
b. Sedangkan kelemahannya adalah, *SLR* membutuhkan waktu cukup lama untuk memenuhi persyaratan dari pertanyaan penelitian, dan juga dalam menemukan *literature* secara menyeluruh terkadang dapat melewatkan beberapa studi penting yang dapat mempengaruhi kesimpulan.

Kelebihan dan kekurangan dari SLR [10] adalah.

1. Kelebihan dari SLR adalah:
   a. *Literature* reviews menjadi lebih serbaguna, digunakan hamper kesemua topik, dan dapat memberikan informasi yang baik dalam mengambarkan sesuatu lebih dalam
   b. *Literature review* lebih mudah dan efisien. Dapat mengumpulkan data dalam jumlah besar dan menimalkan biaya.
   c. Sumber daya yang diperlukan adalah perpustakaan yang baik, *database online* dan referensi yang kompeten
   d. *Literature review* dapat menjadi langkah awal yang baik dalam sebuah proyek atau bahan ajar untuk membuat kerangka kerja yang konseptual pada studi perencanaan lebih lanjut
2. Kelemahan dari SLR adalah:
   a. Sebuah *literature review* yang efektif membutuhkan keterampilan tingkat tinggi dalam mengidentifikasi sumber daya, menganalisis sumber-sumber untuk mendapatkan informasi yang relevan untuk membuat ringkasan.
   b. Terbatasnya informasi yang dikumpulkan tentang apa yang terjadi pada masa lalu, dan biasanya pihak organisasi yang terlibat tidak memberikan data aktual kepada peneliti, selain orang yang berkerja pada organisasi tersebut.

2.1. Software Engineering (SE)

Software Engineering (*SE*) berasal dari dua kata yaitu *Software* (Perangkat Lunak) dan *Engineering*

Rekayasa). *Software* adalah *source code* pada suatu program atau system, dan engineering adalah aplikasi terhadap pendekatan sistematis yang berdasarkan ilmu pengetahuan yang menghasilkan suatu aplikasi maupun sistem [11]. *SE* merupakan disiplin ilmu yang berkaitan dengan semua aspek produksi perangkat lunak yang dimulai dari tahapan awal spesifikasi sistem hingga pemeliharaan sistem yang digunakan [1], yang memiliki peran besar dalam teknologi informasi dan komunikasi, sehingga SE banyak dilibatkan dalam berbagai bidang seperti, pendidikan, *Artificial Intelligence*, *Computer Science*, Manajemen Proyek, *Electric*, Ekonomi, Sosial dan lain sebagainya.

Namun, agar *SE* dapat diimplementasikan kedalam kehidupan manusia. *SE* harus dapat menghasilkan *software* yang berkualitas tinggi, yang memiliki karakteristik sebagai berikut [11]:

a. *Maintainability* (dapat dirawat), perangkat lunak harus dapat memenuhi perubahan kebutuhan, dan biaya perawatan yang efisien.
b. *Dependability*, perangkat lunak harus dapat dipercaya.
c. Efisiensi, perangkat lunak harus efisien dalam penggunaan *resource* atau sumbernya.
d. *Usability*, perangkat lunak memiliki kemampuan agar dapat digunakan sesuai dengan kebutuhan. Selain itu *software* yang dihasilkan harus dapat membuat user atau pemakai merasa mudah untuk dipelajari serta digunakan dan mudah untuk diingat.

3. Metodologi Penelitian

3.1.Objek Penelitian

Objek penelitian adalah *software engineering* (Rekayasa perangkat lunak). Mengapa SE yang dipilih sebagai objek penelitian adalah sebagai berikut:

a. *SE* adalah penerapan prinsip-prinsip matematika dan ilmu komputer untuk merancang, melaksanakan, tes, dan mengevaluasi sistem perangkat lunak. Dimana *engineer* mengembangkan perangkat lunak yang terlibat dalam hampir setiap aspek masyarakat modern saat ini, termasuk game komputer, aplikasi bisnis, aplikasi militer, sistem telekomunikasi, sistem operasi, system kontrol jaringan, dan *middleware* .(http://www.mcs.uvawise.edu/sweg, agustus 2013).
b. *SE* menggusulkan beberapa dimensi penting sebagai bagian integral dari pendidikan rekayasa perangkat lunak: *interdisciplinary skills*, pengalaman praktek, komunikasi, dan keterampilan untuk melanjutkan pendidikan serta *profesionalisme.*
c. Penelitian rekayasa perangkat lunak meliputi pembangunan sistem perangkat lunak baru dan evolusi yang sudah ada ini, harus direalisasikan dengan cara yang seragam atau sama disemua level pengembangan sistem.
d. Membantu untuk memahami strategi desain penelitian *software engineers* dan melaporkan hasil yang jelas, serta juga membantu menjelaskan karakter dari penelitian *SE* untuk ilmu komputer kepada para ilmuan lainnya.

3.2 Metode Penelitian

3.2.1. Pertanyaan Penelitian (*Research Question*)

Pertanyaan penelitian dibuat berdasarkan kebutuhan dari topik yang dipilih:

RQ1. Apa masalah yang muncul dalam *SE* dari tahun 2010-2013?

RQ2. Apa metode yang digunakan dalam menyelesaikan masalah tersebut?

RQ3. Bagaimana cara menyelesaikan masalah tersebut?

3.2.2. Proses pencarian (*Search Process*)

Search Process digunakan untuk mendapatkan sumber-sumber yang relevan untuk menjawab *Research Question* (RQ), dan referensi terkait lainnya dengan menggunakan *search engine* (Mozilla firefox) dengan alamat situs http://ieeexplore.ieee.org/Xplore/home.jsp (data primer) dan https://www.google.com/ (data skunder).

3.2.3. Kriteria batasan dan pemasukan (*Inclusion and exclusion criteria*)

Tahapan ini dilakukan untuk memutuskan apakah data yang ditemukan layak digunakan dalam penelitian SLR atau tidak. Studi layak dipilih jika terdapat kriteria sebagai berikut:

a. Data yang digunakan dari rentan waktu dari 2010-2013.
b. Data yang didapat melalui *search engine* (http://ieeexplore.ieee.org/Xplore/home.jsp), dan https://www.google.com/
c. *Data* yang digunakan hanya berhubungan dengan *software engineering*

3.2.4. Kualitas penilaian (*Quality Assesment*)

Dalam penelitian SLR, data yang ditemukan akan dievaluasi berdasarkan pertanyaan kriteria penilaian kualitas diantaranya adalah sebagai berikut:

QA1. Seluruh *paper*, jurnal yang dimulai dari 2010-2013.

QA2. Apakah pada *paper* jurnal menuliskan Isu (masalah) dalam yang terdapat *SE*?
QA3. Apakah pada *paper*, jurnal menuliskan metode yang digunakan untuk menyelesaikan isu (masalah) tersebut?

Pada pertanyaan diatas diberi nilai sebagai berikut:

a. Y (Ya) : untuk masalah dan metode yang dituliskan pada paper, journal & magazine dari rentan waktu 2010-2013 dan,
b. T (Tidak) : untuk masalah dan metode yang tidak dituliskan.

3.2.5. Pengumpulan data (*Data Collection*)

A. Data Primer

Data primer adalah informasi yang dikumpulkan oleh penelitian langsung melalui survei, wawancara, observasi, yang disesuaikan dengan kebutuhan. Pada penelitian ini data primer yang diambil dari http://ieeexplore.ieee.org/Xplore/home.jsp alasan mengapa menggunakan *ieeexplorer* adalah sebagai berikut:

a. IEEEXPLORE memberikan fasilitas yang lengkap
b. Data yang ditemukan mudah dicari, karena memiliki *range* tahun yang dapat di disesuaikan berdasarkan kebutuhan peneliti.
c. Data yang ditampilkan dapat disesuaikan dengan kebutuhan.

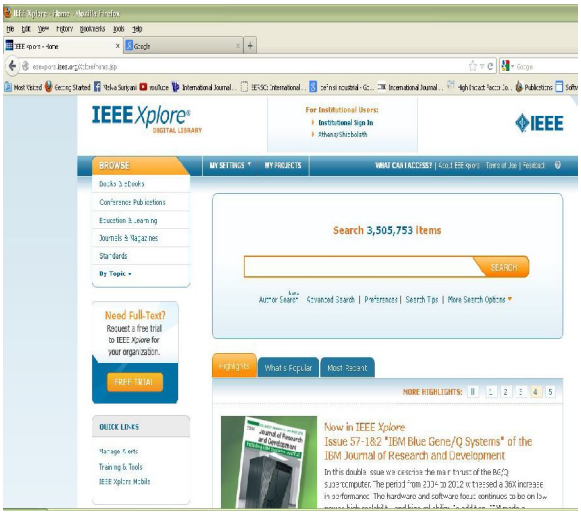
B. Data Skunder

Data skunder digunakan untuk melengkapi data primer, apa bila pada data primer hanya terdapat abstrak, maka diperlukan data sekunder untuk melengkapi data primer, dengan menggunakan *google*. Pengumpulan data dalam penelitian diperoleh dari beberapa tahap diantaranya yaitu:

a. Observasi (Pengamatan): Pengumpulan data yang diamati langsung kesumber yaitu http://ieeexplore.ieee.org/Xplore/home.jsp
b. Studi Pustaka: Melakukan studi pengkajian data terkait dengan metode *SLR* pada journal & magazine yang diperoleh dari http://ieeexplore.ieee.org/Xplore/home.jsp
c. Dokumentasi: Data yang dikumpulkan akan disimpan kedalam EndNote

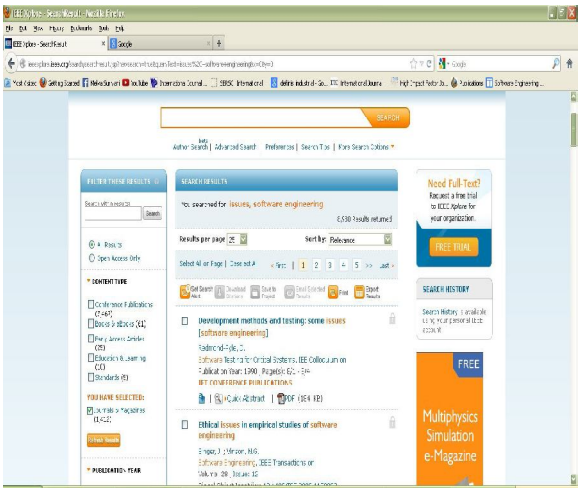
Berikut tahapan pengumpulan data mulai dari obsevasi hingga dokumentasi yang didapat melalui sumber http://ieeexplore.ieee.org/Xplore/home.jsp.

a. Masuk ke (lihat gambar 1) http://ieeexplore.ieee.org/Xplore/home. .jsp:

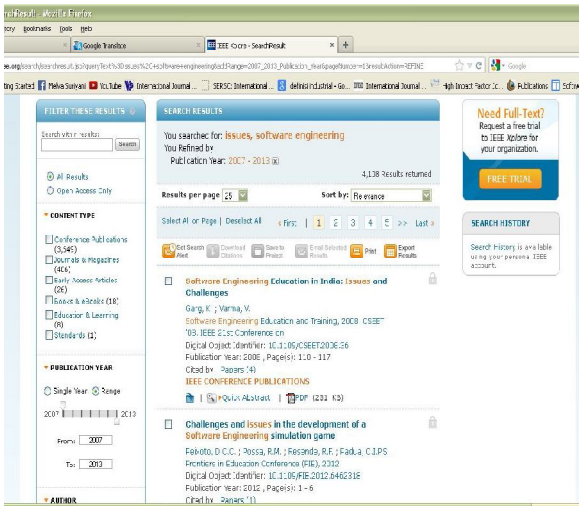


Gambar 1. Masuk ke ieeexplore

b. Masukan kata kunci "*Issues*" dan "*Software Engineering*" lalu pada Content Type pilih *Journals & Magazines* (lihat gambar 2).
c. Pada *Publication Year* pilih Range untuk menentukan sumber tahun dalam menemukan isu software engineering yang dimulai dari tahun 2010-2013, kemudian klik Refresh result. Maka akan ditampilkan judul, tahun publikasi dan nama penulis dari setiap abstrak beserta jumlah yang akan digunakan dalam penelitian (lihat gambar 3).

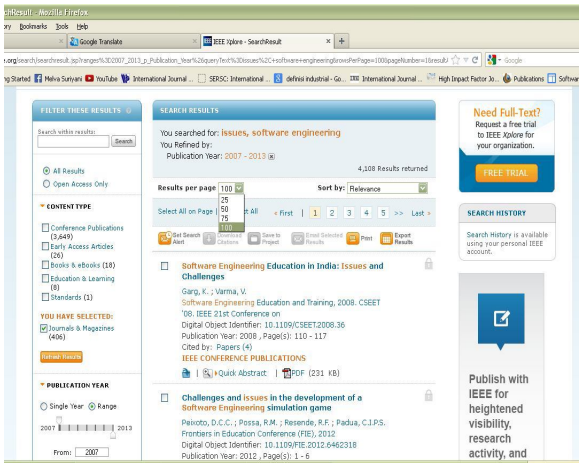


Gambar 2. Masukkan kata kunci



Gambar 3. Tahun publikasi

d. Hasil yang ditampilkan oleh *search process* ieeexplorer sebanyak 215 abstrak, maka pada *Results per page* akan dibatasi 100.



Gambar 4. Hasil searching

Setelah 100 abstrak, maka selanjutnya abstrak didapat akan disimpan menggunakan EndNote X1 sebagai media penyimpanan..

3.2.6. Analisis data (*Data Analysis*)

Pada tahapan ini data yang telah dikumpulkan akan dianalisa untuk menunjukkan:

a. Isu-isu (masalah) yang muncul dalam *SE* dari tahun 2010-2013 (mengacu pada RQ1).
b. Metode/pendekatan yang digunakan dalam *SE* (mengacu pada RQ2).
c. Berbagai metode digunakan untuk menyelesaikan masalah dalam *SE* (mengacu pada RQ3).

3.3.7. Penyimpangan Laporan (*Deviation from Protocol*)

Sebagai hasil dari kajian penulis membuat menuliskan beberapa perubahan pada *deviation from protocol:*

a. Penelitian ini mengidentifikasi isu dan pendekatan *SE*, serta menjawab pertanyaan penelitian (*Research Question*)
b. Mengumpulkan *literature* untuk menjawab serta memastikan kualitas dan menyediakan informasi yang dibutuhkan.
c. Memperluas deskripsi tentang *SLR* pada penelitian ini.

4. Hasil dan Pembahasan

4.1. Hasil *Search Process*

Hasil *search proses* tersebut akan dikelompok berdasarkan tipe jurnal untuk mempermudah melihat jenis data atau tipe jurnal yang didapat melalui *search process* yang ditampilkan pada tabel dibawah ini.

Tabel 1. Pengelompokkan berdasarkan Jurnal

No	Tipe jurnal	Jumlah
1	Power Delivery, IEEE Transactions on	2
2	Engineering & Technology	1
3	Computer	1
4	Antennas and Propagation, IEEE Transactions on	1
5	Software, IEEE	42
6	Software Engineering, IEEE Transactions on	47
7	Intelligent Systems, IEEE:	4
8	Computing in Science & Engineering	4
9	Affective Computing, IEEE Transactions on	1
10	Automation Science and Engineering, IEEE Transactions on	2
11	Oceanic Engineering, IEEE Journal of	2
12	Visualization and Computer Graphics, IEEE Transactions on	3
13	Knowledge and Data Engineering, IEEE Transactions on	3
14	Aerospace and Electronic Systems Magazine, IEEE	3
15	Internet Computing, IEEE	1
16	Proceedings of the IEEE	6
17	Software, IET	9
18	Antennas and Propagation Magazine, IEEE	3
19	Engineering Management Review, IEEE	1
20	Engineering Management, IEEE Transactions on	5
21	Reliability, IEEE Transactions on	3
22	Security & Privacy, IEEE:3	3
23	Network, IEEE	2
24	Education, IEEE Transactions on	6
25	Communications Magazine, IEEE	2
26	Biomedical Engineering, IEEE Transactions on	2
27	Neural Systems and Rehabilitation Engineering, IEEE Transactions on	2
28	Electron Devices, IEEE Transactions on	1
29	Knowledge and Data Engineering, IEEE Transactions on	3
30	Automation Science and Engineering, IEEE Transactions on	2
31	Intelligent Transportation Systems, IEEE Transactions on	1
32	Evolutionary Computation, IEEE Transactions on	1
33	IT Professional	3
34	Power Delivery, IEEE Transactions on	2
35	Annals of the History of Computing, IEEE	1

36	Reliability, IEEE Transactions on	3
37	Biomedical Engineering, IEEE Transactions on	1
38	Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on	1
39	Engineering & Technology	1
40	Industry Applications Magazine, IEEE	1
41	Selected Topics in Signal Processing, IEEE Journal of	1
42	Software, IET	9
43	Power Systems, IEEE Transactions on	2
44	Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on	1
45	Dependable and Secure Computing, IEEE Transactions on	1
46	Communications, IET	1
47	Security & Privacy, IEEE	1
48	Sensors Journal, IEEE	1
49	Information Technology in Biomedicine, IEEE Transactions on	1
50	BM Journal of Research and Development	2
51	Consumer Electronics, IEEE Transactions on	1
52	Potentials, IEEE	1
53	Computer	4
54	Robotics & Automation Magazine, IEEE	1
55	Photovoltaics, IEEE Journal of	2
56	Smart Grid, IEEE Transactions on	1
57	Geoscience and Remote Sensing Magazine, IEEE	1
58	Multimedia, IEEE Transactions on	1
59	Power Electronics, IEEE Transactions on	1
Total		215

4.2. Hasil Seleksi *Inclusion and Exclusion Criteria*

Hasil dari *search process*, akan diseleksi berdasarkan Kriteria Batasan dan Pemasukan *(Inclusion and exclusion criteria*). Proses ini menyisakan 35 journal. Kemudian kita melakukan scanning data. Dari hasil tabel *scanning* diatas akan diberikan *quality assessment* untuk melihat data yang akan digunakan (Apakah data tersebut digunakan atau tidak dalam penelitian ini).

4.3. Hasil Kualitas Penilaian *(Quality Assesment)*

Hasil seleksi dari Kriteria batasan dan pemasukan (*Inclusion and exclusion criteria*) akan dinilai kualitas (*Quality Assesment*) dari masing-masing data.

Tabel 2. Hasil Kualitas Penilaian (*Quality Assesment*)

No	Penulis	Judul	Tahun	QA1	QA2	QA3	Hasil
1.	D. P. Bernardon, V. J. Garcia, A. S. Q. Ferreira and L. N.	*Multicriteria Distribution Network Reconfiguration Considering Subtransmission Analysis*	2010	Y	Y	Y	√
2.	V. Garousi	*A Genetic Algorithm-Based Stress Test Requirements Generator Tool and Its Empirical Evaluation*	2 010	Y	Y	Y	√
3.	N. Limam and R. Boutaba	*Assessing Software Service Quality and Trustworthiness at Selection Time*	2010	Y	Y	Y	√
4.	F. Liqing, B. N. Jagdish, A. Senthil Kumar, S. Anbuselvan and B. Shung-Hwee	*Collaborative Fixture Design and Analysis Using Service Oriented Architecture*	2010	Y	Y	Y	√
5.	J. E. Pederson	*Creating a tool independent system engineering environment*	2010	Y	Y	Y	√
6.	R. Pietrantuono, S. Russo and K. S. Trivedi	*Software Reliability and Testing Time Allocation: An Architecture-Based Approach*	2010	Y	Y	Y	√
7.	R. Ramsin and R. F. Paige	*Iterative criteria-based approach to engineering the requirements of software development methodologies*	2010	Y	Y	Y	√
8.	C. G. von Wangenheim, J. C. R. Hauck, A. Zoucas, C. F. Salviano, F. McCaffery and F. Shull	*Creating Software Process Capability/Maturity Models*	2010	Y	Y	Y	√
9.	W. Zai, T. Ke and Y. Xin	*Multi-Objective Approaches to Optimal Testing Resource Allocation in Modular Software Systems*	2010	Y	Y	Y	√
10.	G. Zhongwen, C. Pengpeng, F. Yuan, J. Yongguo and H. Feng	*ISDP: Interactive Software Development Platform for Household Appliance Testing Industry*	2010	Y	Y	Y	√
11.	S. Biswas, R. Mall and M. Satpathy	*Task Dependency Analysis for Regression Test Selection of Embedded Programs*	2011	Y	Y	Y	√
12.	D. D. Black, M. E. C. Hull and K. Jackson	*Systems engineering and safety - a framework*	2011	Y	Y	Y	×
13.	G. Booch	*Unintentional and Unbalanced Transparency*	2011	Y	Y	Y	√
14.	P. Gang and M. Jifeng	*Network Structures and Online Technology Adoption*	2011	Y	Y	Y	√
15.	R. L. Glass and I. Vessey	*Naïveté Squared: In Search of Two Taxonomies and a Mapping between Them*	2011	Y	Y	Y	×
16.	N. Harrison and P. Avgeriou	*Pattern-Based Architecture Reviews*	2011	Y	Y	Y	√
17.	R. P. Jain, R. S. Poston and J. C. Simon	*An Empirical Investigation of Client Managers' Responsibilities in Managing Offshore Outsourcing of Software-Testing Projects*	2011	Y	Y	Y	×
18.	K. Moonzoo, K. Yunho and K. Hotae	*A Comparative Study of Software Model Checkers as Unit Testing Tools: An Industrial Case Study*	2011	Y	Y	Y	√
19.	R. C. Prati, G. E. A. P. A. Batista and M. C. Monard	*A Survey on Graphical Methods for Classification Predictive Performance Evaluation*	2011	Y	Y	Y	×
20.	F. Shull	*Assuring the Future? A Look at Validating Climate Model Software*	2011	Y	T	Y	×

23.	V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. Lo Presti and R. Mirandola	*MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems*	2012	Y	Y	Y	√
24.	K. Dejaeger, W. Verbeke, D. Martens and B. Baesens	*Data Mining Techniques for Software Effort Estimation: A Comparative Study*	2012	Y	Y	Y	√
25.	Kr, x030C, P. emen and Z. Kouba	*Ontology-Driven Information System Design*	2012	Y	Y	Y	√
26.	R. Marinescu	*Assessing technical debt by identifying design flaws in software systems*	2012	Y	Y	Y	√
27.	M. Shousha, L. Briand and Y. Labiche	*A UML/MARTE Model Analysis Method for Uncovering Scenarios Leading to Starvation and Deadlocks in Concurrent Systems*	2012	Y	Y	Y	√
28	F. Siddique and O. Maqbool	*Enhancing comprehensibility of software clustering results*	2012	Y	Y	Y	√
29.	S. Xiping, B. Hwong and J. Ros	*Lessons from Developing Nonfunctional Requirements for a Software Platform*	2012	Y	Y	T	×
21.	D. R. White, A. Arcuri and J. A. Clark	*Evolutionary Improvement of Programs*	2011	Y	Y	Y	×
22.	H. Azath and R. S. D. Wahidabanu	*Efficient effort estimation system viz. function points and quality assurance coverage*	2012	Y	Y	Y	√
30.	R. Baker and I. Habli	*An Empirical Evaluation of Mutation Testing for Improving the Test Quality of Safety-Critical Software*	2013	Y	Y	Y	×
31.	X. Larrucea, A. Combelles and J. Favaro	*Safety-Critical Software [Guest editors' introduction]*	2013	Y	Y	Y	√
32.	M. Piccioni, M. Oriol and B. Meyer	*Class Schema Evolution for Persistent Object-Oriented Software: Model, Empirical Study, and Automated Support*	2013	Y	Y	Y	√
33.	M. Rodriguez, G. Stahl, L. Corradini and D. Maksimovic	*Smart DC Power Management System Based on Software-Configurable Power Modules*	2013	Y	Y	Y	×
34.	K. Sagesser, B. Joseph and R. Grau	*Introducing an Iterative Life-Cycle Model at Credit Suisse IT Switzerland*	2013	Y	Y	Y	×
35.	K. J. Sauer and T. Roessler	*Systematic Approaches to Ensure Correct Representation of Measured Multi-Irradiance Module Performance in PV System Energy Production Forecasting Software Programs*	2013	Y	Y	Y	√
32.	M. Piccioni, M. Oriol and B. Meyer	*Class Schema Evolution for Persistent Object-Oriented Software: Model, Empirical Study, and Automated Support*	2013	Y	Y	Y	√
33.	M. Rodriguez, G. Stahl, L. Corradini and D. Maksimovic	*Smart DC Power Management System Based on Software-Configurable Power Modules*	2013	Y	Y	Y	×
34.	K. Sagesser, B. Joseph and R. Grau	*Introducing an Iterative Life-Cycle Model at Credit Suisse IT Switzerland*	2013	Y	Y	Y	×
35.	K. J. Sauer and T. Roessler	*Systematic Approaches to Ensure Correct Representation of Measured Multi-Irradiance Module Performance in PV System Energy Production Forecasting Software Programs*	2013	Y	Y	Y	√

Keterangan Simbol:

a. √ : Untuk Jurnal atau data yang digunakan penelitian, data tersebut dipilih karena memiliki masalah, pendekatan dan informasi yang cukup untuk pemilihan data.

b. × : Untuk Jurnal atau data yang tidak digunakan dalam penelitian. Karena data tersebut merupakan *guest editor* yang menceritakan tentang pengalaman para peneliti, masalah, pendekatan ataupun informasi yang kurang memadai untuk pemilihan data.

4.4 Analisis Data (*Data Analysis*)

Pada tahapan ini akan menjawab pertanyaan dari *Research Question* (RQ) dan membahas hasil dari metode serta pendekatan yang dominan muncul dari tahun 2010-2013.

4.4.1 Pembahasan Hasil

Bagian ini akan menjelaskan/menjawab *Research Question* (RQ)

RQ1. Apa masalah yang muncul dalam SE (2010-2013)?

Secara keseluruhan terdapat 215 jurnal melalui *search process*. Setelah data diseleksi berdasarkan *inclusion and exclusion criteria*,dengan menggunakan kata kunci (*keyword*) "*software engineering*" terdapat 35 jurnal, yang kemudian diberi kualitas penilaian (*Quality Assesment*) dari hasil *quality assessment* (QA) terdapat 24 jurnal relevan. Dimana data tersebut dikelompokkan berdasarkan isu (masalah) dan pendekatan yang digunakan untuk menjawab *research question*.

Dalam hal ini akan menjawab RQ1, yang ditampilkan pada tabel 4.

Tabel 3. Pengelompokkan Isu (Masalah)

No	Isu (Masalah)	Jumlah
1.	*Multicriteria Distribution Network Reconfiguration Considering Subtransmission Analysis*	1
2.	*Software Process Capability/Maturity Models*	1
3.	*Distributed measurement systems (DMSS)*	1
4.	*Regression Test selection*	1
5.	*Security and privacy concepts*	1
6.	*Software Product Lines*	1
7.	*Open source software development data*	1
8.	*Software Testing issue*	2
9	*Ontology-Driven information system design*	1

Dari tabel diatas menunjukkan bahwa isu (masalah) yang dominan muncul sebagai dalam *SE* adalah ***software testing***.

RQ2. Apa metode yang dominan dalam SE (2010-2013)?
Pada RQ2 menampilkan metode yang dominan dalam *Software Engineering* (2010-2013) berdasarkan data yang ditampilkan pada Tabel 4. Hasil dari RQ2 disajikan dalam Tabel 5 berikut.

Tabel 4. Pengelompokkan Pendekatan (Metode)

No	Pendekatan	Jumlah
1.	*Genetic Algorithm-Based Stress Test Requirements Generator Tool*	1
2.	*Service-Oriented Architecture (SOA)*	3
3.	*Alternative approach*	1
4.	*Iterative criteria-based approach*	1
5.	*Multi-Objective Approaches*	1
6.	*Pattern-Based Architecture Reviews*	1
7.	*Prototype*	1
8.	*Constructive Cost Model* (COCOMO)	1
9.	*Data Mining Techniques.*	1

Berdasarkan tabel 5 pendekatan dominan dalam SE (2010-2013) adalah SOA (*Service Oriented Architectures*).

4.4.2. Service Oriented Architecture

Service-Oriented Architecture (SOA) merupakan cabang ilmu baru dalam bidang *software engineering*, yang dianggap sebagai paradigma baru untuk mengembangkan solusi *SE* yang fleksibel, dimanis, menggunakan *coupling of services* dengan sistem operasi yang beragam, bahasa pemrograman dan komponen lain dari berbagai *platform* [12]. Ide dasar dari *SOA* telah mendapat perhatian besar dan kepedulian dari komunitas desain dan pengembangan perangkat lunak [13].

Teknologi SOA seperti *Web services* merupakan awal yang baik karena mulai diadopsi secara luas, dengan tujuan memungkinkan komposisi layanan aplikasi baru yang sudah ada dalam *technologically heterogeneous environment* [14]. *SOA* dapat definisikan sebagai berikut:

a. Service-Oriented Architecture (*SOA*) adalah sebuah gaya arsitektur yang berorientasikan kepada layanan (http://www.opengroup.org/soa/sourcebook/soa/soa.htm, akses 06 september 2013).
b. *Service-Oriented Architecture* adalah pendekatan desain client / server di mana sebuah aplikasi terdiri dari layanan perangkat lunak dan layanan konsumen perangkat lunak, yang dikenal sebagai klien atau *service requesters* [15].
c. *Service-Oriented Architecture* merupakan pendekatan arsitektur untuk sistem perusahaan, yang fokus kepada penawaran layanan jaringan konsumen. Contoh dari pendekatan *service-oriented* ini adalah mendefinisikan layanan antarmuka (*interface*) yang berorientasi perilaku (*service behavior*) [16].
d. *SOA* adalah sebuah arsitektur yang jauh lebih baik dibandingkan dengan didistribusikan arsitektur *client server* dan dapat membawa manfaat besar dalam bentuk penggunaan kembali kode, integrasi yang lebih baik dan meningkatkan daya tanggap terhadap kebutuhan bisnis [17].

Dalam *SOA* terdapat tiga komponen dasar (*Alliance 2006*) yang dapat dilihat pada Gambar 1:

Gambar 5. SOA Foudation:
Sumber: *Group of SOA Practitioners*, 2006

a. *Business Architecture*: Berdasarkan strategi bisnis, tujuan, prioritas dan proses. Mendapatkan hak sangat penting untuk keberhasilan pelaksanaan SOA. Salah satu manfaat utama dari SOA adalah penggunaan kembali proses bisnis yang memberikan ROI yang lebih tinggi dari pada penggunaan kembali potensi komponen Infrastruktur atau Data. Ini juga mencakup proses-proses bisnis serta implementasi aplikasi bisnis.
b. *Infrastructure Architecture: engine* yang memungkinkan SOA dan harus mengatasi semua aspek infrastruktur dari jaringan, server, pusat data, *firewall*, infrastruktur aplikasi, keamanan, pemantauan, *middleware*, dll
c. *Information and Data Architecture:* berkaitan dengan mengidentifikasi *Key Performance Indicators* dan kebutuhan informasi yang mendorong perusahaan. Arsitektur transaksi data dengan pemodelan logis, fisik dari data serta manipulasi data dan kualitas data.

SOA bukan solusi untuk semua permasalahan dalam pengembangan *software* karena memiliki kelebihan dan kelemahan diantaranya adalah sebagai berikut:

Menurut Mahmood [17] SOA menawarkan kesempatan yang lebih baik untuk aplikasi *enterprise* integrasi dengan keuntungan tambahan dari *reduction in costs*, kemudian pemiliharaan, perbaikan fleksibilitas dan skalabilitas.
Menurut Craven [18] sebagai berikut:
A. Kelebihan
a. *Loose Coupling*: Langue Independence, dan Membantu antarmuka (*interface)* dengan *legacy systems.*
b. *Distributed*: *Manage load*, failover untuk kehandalan, dan dapat didistribusikan secara geografis.
c. *Asset Management*: memanfaatkan sumber daya yang ada, menciptakan aset, dan *separate teams*
d. *Reability: SOA* lebih *realibel*

C. Kelemahan SOA

Message Overhead: XML bukan merupakan format pesan yang efisien. Karena Jumlah *byte* dibutuhkan untuk membangun pesan jauh lebih tinggi dari infomasi yang terkandung sebenarnya di dalamnya.

RQ3. Bagaimana Menyelesaikan Masalah *Software Testing*?

Pada RQ3 akan membahas penyelesaian yang terdapat pada RQ1 (*software testing*) menurut penelitian sebenarnya yang terdiri dari: *Software testing* yang pertama dibahas adalah *A Comparative Study of Software Model Checkers as Unit Testing Tools: An Industrial Case Study.* Masalah yang diangkat adalah tentang *Conventional testing methods* sering gagal untuk mendeteksi kelemahan yang tersembunyi dalam system perangkat lunak yang tertanam, seperti *device drivers* atau *file systems*. Kekurangan ini menimbulkan perkembangan yang signifikan terhadap *support/maintenance cost* dalam *manufacturers*. Sehingga *software model checkers* diusulkan untuk mengimbangi kelemahan *conventional testing methods* tersebut. Penyelesaian masalah menurut Moonzoo el al. [19] memilih dua *software model checkers* dengan alasan:
a. *Blast* dan *CBMC* relatif stabil dibandingkan dengan *tools* lainnya. Kedua tools telah berkembang selama hampir satu dekade, keduanya memiliki komunitas pengguna. Dengan demikian, ada kemungkinan bahwa banyak *bug* serta keterbatasan dalam versi asli yang telah diperbaiki, oleh karena itu USP (*Unified Storage Platform*) dapat diverifikasi menggunakan alat ini tanpa banyak kesulitan.
b. Kode sumber *CBMC* dan *Blast* tersedia untuk umum. Meskipun model checking technique diklaim merupakan teknik "*push-button*", pengguna dapat meningkatkan kinerja verifikasi atau melewati batasan dari *tool* melalui pengetahuan tentang perilaku internal *tools* yang sulit diperoleh tanpa *tools source code.*

Selain itu, terkadang perlu untuk menyesuaikan alat tergantung pada kebutuhan proyek industri. Sehingga akhirnya, peneliti memiliki keahlian baik di bidang *Blast* dan CBMC melalui beberapa tahun pengalaman. Oleh karena itu, peneliti dapat melakukan percobaan perbandingan secara adil dan *technically sound manner*. *Software testing* kedua adalah *Assessing technical debt* untuk mengidentifikasi kecacatan desain dalam sistem *software* [20]. Masalah yang diangkat adalah Kendala *time-to-market* yang sulit dan integrasi tidak terduga atau masalah evolusi pengorbanan perancang yang biasanya menyebabkan kelemahan dalam struktur sistem perangkat lunak. Dengan demikian, biaya pemeliharaan tumbuh secara signifikan.

Dampak dari keputusan desain yang memberikan manfaat jangka pendek dengan mengorbankan *system_s design integrity*, biasanya disebut *technical debt.* kemudian peneliti mengusulkan
1. *a novel framework* mengunakan teknik *assessing technical debt* untuk mendeteksi kelemahan dalam desain, yaitu *specific violations of well-established design principles and rules*. Penyelesaian masalah menurut Moonzoon et al [19] membuat *framework* komprehensif dan seimbang yang dibangun berdasarkan *metrics-based detection* dengan empat langkah yaitu:
   a. ***Select Relevant Design Flaws* (Memilih salah satu kecacatan desain yang relevan).** *Framework* dapat digunakan untuk semua jenis kecacat desain, mulai dari mempengaruhi *fine-grained* metode seperti duplikasi kode terhadap kurangnya arsitektur yang terjadi pada tingkat subsistem. Kekurangan tersebut akan disertakan secara aktual pada *framework instantiation* yang mencerminkan penilaian fokus.
   b. ***Detect Design Flaws* (Mendeteksi kecacatan desain).** Untuk mendeteksi kecacatan desain, berbagai teknik deteksi dapat digunakan. Beberapa teknik didasarkan pada Prolog untuk menggambarkan anomali struktural, sedangkan lainnya adalah berbasis metrik atau menggunakan *dedicated specification language* yang juga memungkinkan perhitungkan korelasi antar *account correlation.*
   c. ***Measure the Impact of Design Flaws* (Mengukur dampak dari kecacatan desain).** Setiap kesalahan desain mempengaruhi, sampai batas tertentu kualitas keseluruhan dari suatu sistem, tetapi tidak semua memiliki dampak negatif yang sama.
   d. ***Compute the Overall Score* (menghitung skor keseluruhan)** menghitung skor keseluruhan yang merangkum status kualitas desain dari suatu sistem dengan memperhatikan semua

pengaruh kekurangan yang terdeteksi. Untuk mendapatkan gambaran kuantitatif *debt* desain sistem, berbagai contoh kekurangan harus dikumpulkan. Tujuan ini peneliti mendefinisikan *Debt Symptoms Index (DSI).*

4.5 Software Testing

Software testing adalah kegiatan yang bertujuan untuk mengevaluasi dan juga untuk meningkatkan kualitas program, dengan mengidentifikasi kecacatan dan masalah [21]. S*oftware testing* merupakan elemen penting dalam siklus hidup pengembangan perangkat lunak dan memiliki potensi untuk menghemat waktu *money by identifying problems early* serta untuk meningkatkan kepuasan pelanggan dengan memberikan produk yang lebih bebas kecacatan. Software test dilakukan sebagai pengembangan terakhir perangkat lunak.

Menurut Pohjolainen [22[dalam s*oftware testing* terdapat *tools* setiap tool memiliki deskripsi yang mirip atau hampir sama yang terdiri dari:

a. *Test Design Tools*: *Tools* untuk membantu memutuskan apa test (pengujian) harus dieksekusi. *Test data* dan *test case generators.*
b. *GUI Test Driver*: *Tools* yang mengotomatisasi pelaksanaan tes untuk produk dengan *interface* (antarmuka) pengguna grafis. *Client / server* alat otomatisasi tes, termasuk beban penguji.
c. *Load and Performance Testing Tools*: *Tools* yang mengkhususkan diri dalam menempatkan beban berat pada sistem (terutama sistem *clientserver*).
d. *Test Management Tools*: *Tools* yang mengotomatisasi pelaksanaan tes untuk produk tanpa *interface* (antarmuka) pengguna grafis. Juga *tools* yang membantu Anda bekerja *large test suites*.
e. *Test Implementation Tools*: Miscellaneous tools membantu menerapkan tes. Misalnya, *tools* yang secara otomatis menghasilkan rutinitas, seperti halnya *tools* yang mencoba untuk membuat kegagalan lebih jelas (*assertion generators*, dll)
f. *Test Evaluation Tools*: *Tools* yang membantu Anda mengevaluasi kualitas tes Anda. Mencakup *Code coverage tools.*
g. *Static Analysis Tools*: *Tools* yang menganalisis program tanpa running. Metrics tools fall termasuk kedalam kategori ini.
h. *Regression tools*: *Regression testing tools* digunakan untuk menguji perangkat lunak setelah modifikasi.

Terdapat topik isu dalam *software testing* yang akan dibahas sekilas, diantaranya adalah sebagai berikut:

a. *Security Testing*. *Security test* adalah jenis khusus dari pengujian perangkat lunak dan difokuskan pada keamanan fungsi-fungsi yang didukung dalam system operasi yang aman [23].
b. *Cloud Testing*. *Cloud testing* adalah lanjutan dari tahap evolusi Internet. Sebuah *cloud* memiliki beberapa sifat khas yang berbeda: elastisitas dan skalabilitas, multi-tenancy, kemampuan *self-managed*, *service billing* dan *metering functions*, konektivitas *interface* (antarmuka) dan teknologi [24].

5. Penutup

5.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

a. Mengacu kepada hasil SLR yang penulis lakukan pada *Journal&Magazine* yang dipublikasi dari tahun 2010-2013. Masalah yang dominan dibahas dalam penelitian SLR ini adalah ***Software Testing*.**
b. Berdasarkan hasil dari SLR yang dilakukan pada publikasi *Journal&Magazine*, dari tahun 2010-2013 metode yang dominan muncul adalah **SOA (*Service Oriented Architecture*).**
c. Metode SLR dapat digunakan untuk mengidentifikasi isu-isu dan pendekatan yang terdapat dalam SE (2010-2013).

5.2 Saran

Berdasarkan kesimpulan dari hasil penelitian, maka penulis menyarankan hal sebagai berikut:

a. SLR dapat digunakan untuk mengidentifikasi isu-isu dan pendekatan yang ada dalam bidang ataupun domain tertentu, seperti: *Management* Proyek, *Artificial Intelligent* (kecerdasan buatan), dan lain sebagainya.
b. Hasil dari SLR yang ditulis dalam penelitian ini, dapat digunakan untuk penelitian studi lanjut, jika ingin melanjutkan pendidikan Strata-2 (S2).

Referensi

[1.] Sommerville, I (2010) Software Engineering (9th Edition)
[2.] Kitchenham (2007), "Guidelines for Performing Systematic Literature Reviwes in Software engineering", *Journal of Information and Technology*, ELSEVIER, UK
[3.] Bassil, Y. (2012). "A Simulation Model for the Waterfall Software Development Life Cycle." International Journal of Engineering & Technology **Vol. 2, No. 5**: 1
[4.] Lusiana Efrizoni, Wan M.N. Wan-Kadir, and Radziah Mohamad (2010), A Systematic Literature Review to Identify the Issues in Bidirectional Model Transformation, 2nd WRI World Congress on Software

Engineering, 19-20 December 2010, Wuhan, China. (IEEE) [abstract]

[5.] B. Kitchenham, O. Brereton, D. Budgen, M. Turnre, J. Bailey, dan S. Linkman (2009), "Systematic literature reviews in software engineering – A systematic literature review", *Journal of Information and Technology*, ELSEVIER, UK.

[6.] B. Kitchenham (July 2004), "Procedures for performing systematic reviews", *Software Engineering Group Department of Computer Science*, Keele University.

[7.] J. Nicolas, dan A. Toval (2009), "On the generation of requirements specifications from software engineering models", *Journal of Information and Technology*, ELSEVIER, UK..

[8.] S. Beecham, N. Baddoo, T. Hall, H. Robinson, dan H. Sharp (2008), "Motivation in Software Engineering: A systematic literature review", *Journal of Information and Technology*, ELSEVIER, UK.

[9.] Qing Gu, P. L. (2009). "*Exploring service-oriented system engineering challenges: a systematic literature review.*" 172

[10.] Biolchini, Jorge (2005), *"Systematic Literature Review in Software Engineering"* , Rio de Jeneiro.

[11.] Edi Eddy Prasetyo Nugroho, Komala Ratnasari, Kurniawan Nur Ramadhani, dan Budi Laksono Putro, (2009), Rekayasa Perangkat Lunak, Bandung, Telkom Polytechnic.

[12.] Avgeriou, A. K. P. (2009). An Overview of Software Engineering Approaches to Service Oriented Architectures in Various Fields. IEEE International Workshops on Enabling Technologies**:** 254.

[13.] Douglas Rodrigues, J. C. E., Kalinka R. L. J. C. Branco (2011). "Analysis of Security and Performance Aspects in Service-Oriented Architectures." International Journal of Security and Its Applications **5, No. 1**: 13.

[14.] http://www.alexu.edu.eg/Courses/CS_Advanced%20S.E/Software%20Engineering%20(9th%20Edition)%20by%20Ian%20Sommerville.pdf, akses 20 februari 2013

[15.] Liang, M. (2012) "Introduction to Service Oriented Architecture." CSCI-5828 Foundationsof Software Engineering **Volume**, DOI

[16.] Raines, G. (2009). SERVICE-ORIENTED ARCHITECTURE (SOA) SERIES. Cloud Computing and SOA, Approved for Public Release; Distribution Unlimited Case# 09-0743.

[17.] Mahmood, Z. (2007). "The Promise and Limitations of Service Oriented Architecture." INTERNATIONAL JOURNAL OF COMPUTERS **1**(2, 3): 75, 76

[18.] Craven, D. P. V. (2011). The Dark Side of SOA. Indianola, Iowa, Department of Computer Science

[19.] Moonzoo Kim, Yunho Kim and Hotae Kim (March 2011), "Compariative Study of Software Model Ceckers as Unit Testing Tools: An Idustrial Case Study, IEEE Transaction on Software Engineering, Vol 37 No 2, pp 146-160.

[20.] Marinescu, R. (2012). "Assessing technical debt by identifying design flaws in software systems." IBM Journal of Research and Development **56**(5): 9:1-9:13

[21.] S.M.K Quadri, S. U. F. (2010). "Software Testing – Goals, Principles, and Limitations."International Journal of Computer Applications (0975 – 8887) **6, No 9**:

[22.] Pohjolainen, P. (2002). SOFTWARE TESTING TOOLS**:** 6.

[23.] Gaoshou Zai dan Yaodong Li (2008), "Study and Implementation of SELinux-Like Access Control Mechanism Based on Linux, Advance Security Technology International Conference, SecTech 2008

[24.] Jerry Gao, Xiaoying Bai and Wei-Tek Tsai (2011), Claud Testing- Issues, Chalengges, Needs and Practice, Software Engineering; An International Journal (SEIJ), Vol 1 No 1

.