Software Engineering Research Trends

Romi Satria Wahono
romi@romisatriawahono.net
http://romisatriawahono.net
08118228331

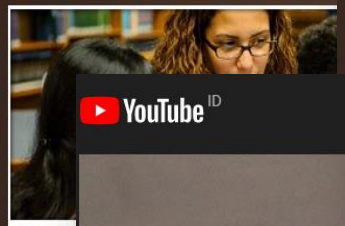
Romi Satria Wahono

- SMA Taruna Nusantara Magelang (1993)
- B.Eng, M.Eng and Ph.D in Software Engineering
  Saitama University Japan (1994-2004)
  Universiti Teknikal Malaysia Melaka (2014)
- Core Competency in Enterprise Architecture, Software Engineering and Machine Learning
- LIPI Researcher (2004-2007)
- Founder and CEO:
  - PT Brainmatics Cipta Informatika (2005)
  - PT IlmuKomputerCom Braindevs Sistema (2014)
- Professional Member of IEEE, ACM and PMI
- IT and Research Award Winners from WSIS (United Nations), Kemdikbud, Ristekdikti, LIPI, etc
- SCOPUS/ISI Indexed Q1 Journal Reviewer: Information and Software Technology, Journal of Systems and Software, Software: Practice and Experience, Empirical Software Engineering, etc
- Industrial IT Certifications: TOGAF, ITIL, CCAI, CCNA, etc
- Enterprise Architecture Consultant: KPK, RistekDikti, INSW, BPPT, Kemsos Kemenkeu (Itjend, DJBC, DJPK), Telkom, FIF, PLN, PJB, Pertamina EP, etc

RomiSatriaWahono.Net

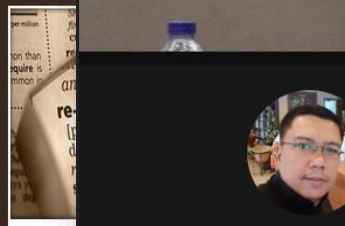
Lecture Notes in Software Engineering, Computing Research and Technopreneurship

HOME ABOUT ME IN THE NEWS LECTURES



BAGAI

Pada ar
penelitia
to do Go
artikel y



TAHAP
PENEL
MAHAS

Jujur, se
kecewa
mahasis
masuk k

RomiSatriaWahono.Net

YouTube ID

Search

Romi Satria Wahono

3.55K subscribers

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Uploads PLAY ALL

Data Mining untuk Mahasiswa Galau

268 views • 9 hours ago

Menjadi Programmer Technopreneur

4.3K views • 5 years ago

Kuliah 10 Menit tentang Enterprise Architecture

10K views • 5 years ago

Kuliah 20 Menit tentang Metodologi Penelitian

136K views • 5 years ago

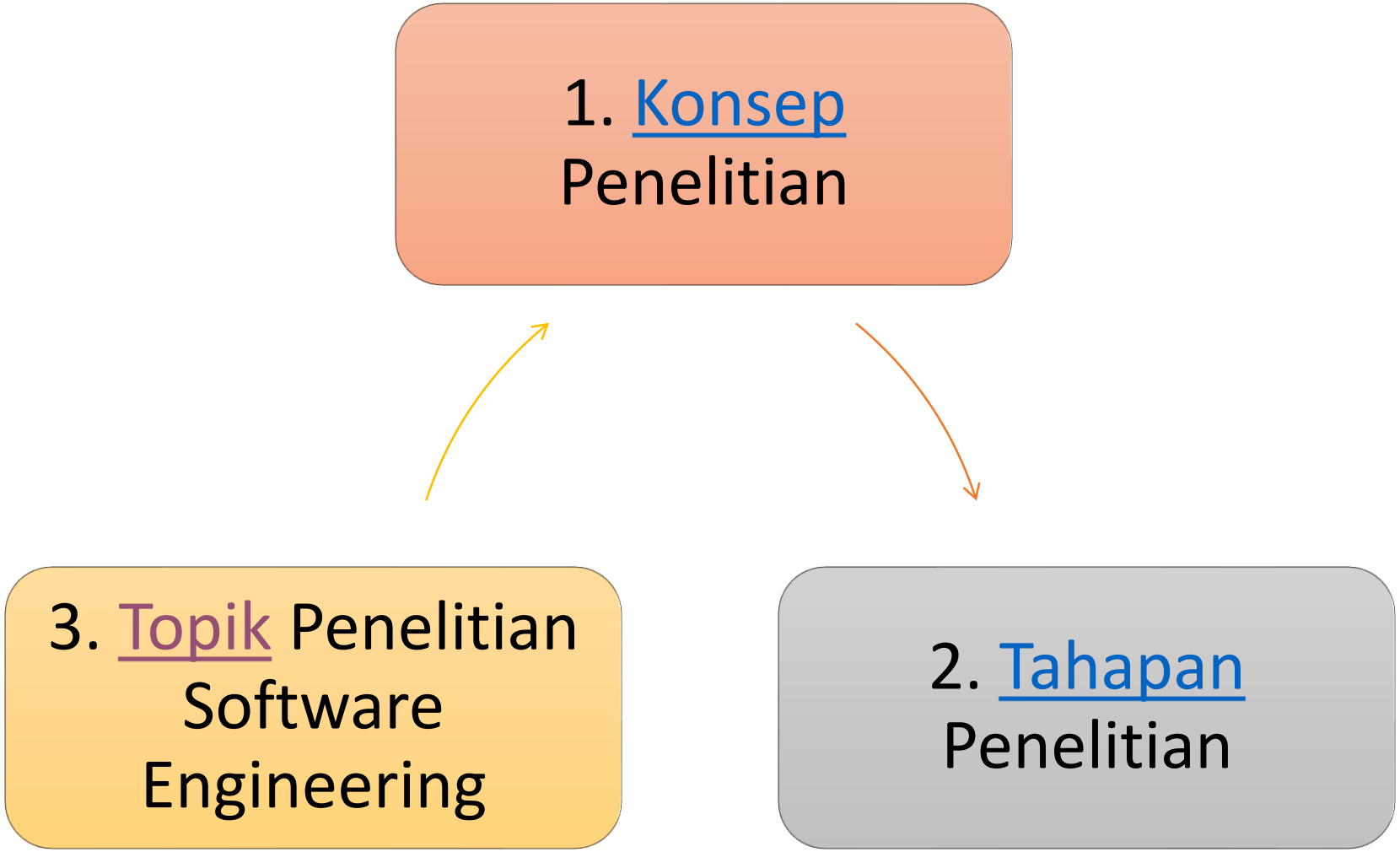
eluruh materi kuliah bisa diunduh dan
rse description, standard competency,

ed January 2015)
2015)
ed March 2015)

October 2013)

updated January 2015)
otation (updated January 2015)

Tiga Pokok Bahasan

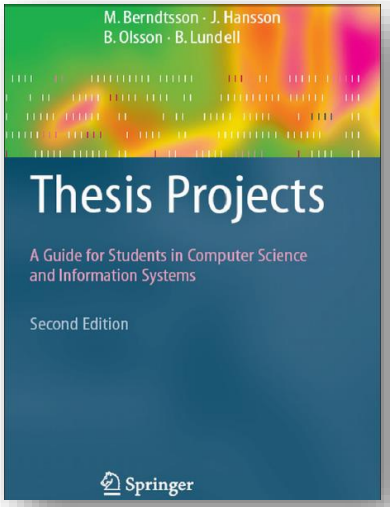


1. Konsep Penelitian

Mengapa Melakukan Penelitian?

- Berangkat dari adanya masalah penelitian
  - yang mungkin sudah diketahui metode pemecahannya
  - tapi belum diketahui metode pemecahan yang lebih baik

- Research (Inggris) dan recherche (Prancis)
  - re (kembali)
  - to search (mencari)



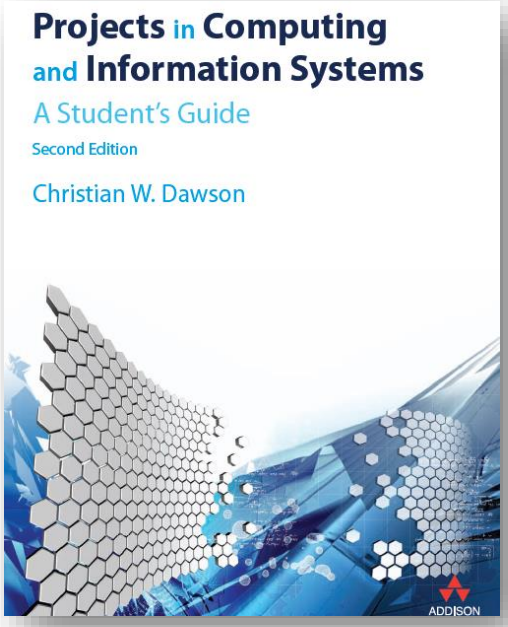
- The process of exploring the unknown, studying and learning new things, building new knowledge about things that no one has understood before
  (Berndtsson et al., 2008)

Apa Yang Dikejar di Penelitian?

Research is a considered activity, which aims to make an original contribution to knowledge

(contribution to the body of knowledge, in the research field of interest)

(Dawson, 2009)

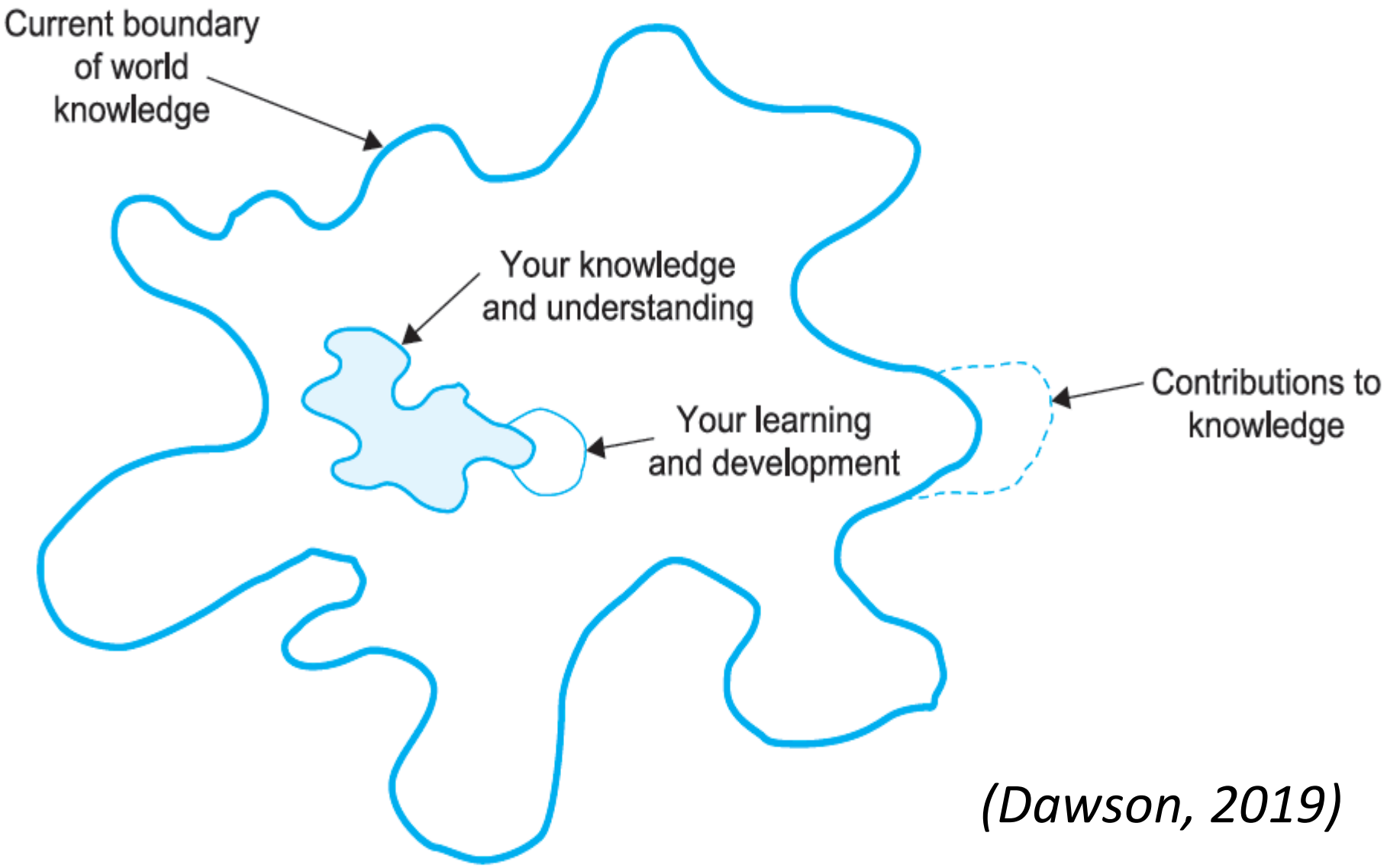


Bentuk Kontribusi ke Pengetahuan

Kegiatan penyelidikan dan investigasi terhadap suatu masalah yang dilakukan secara berulang-ulang dan sistematis, dengan tujuan untuk menemukan atau merevisi teori, metode, fakta, dan aplikasi

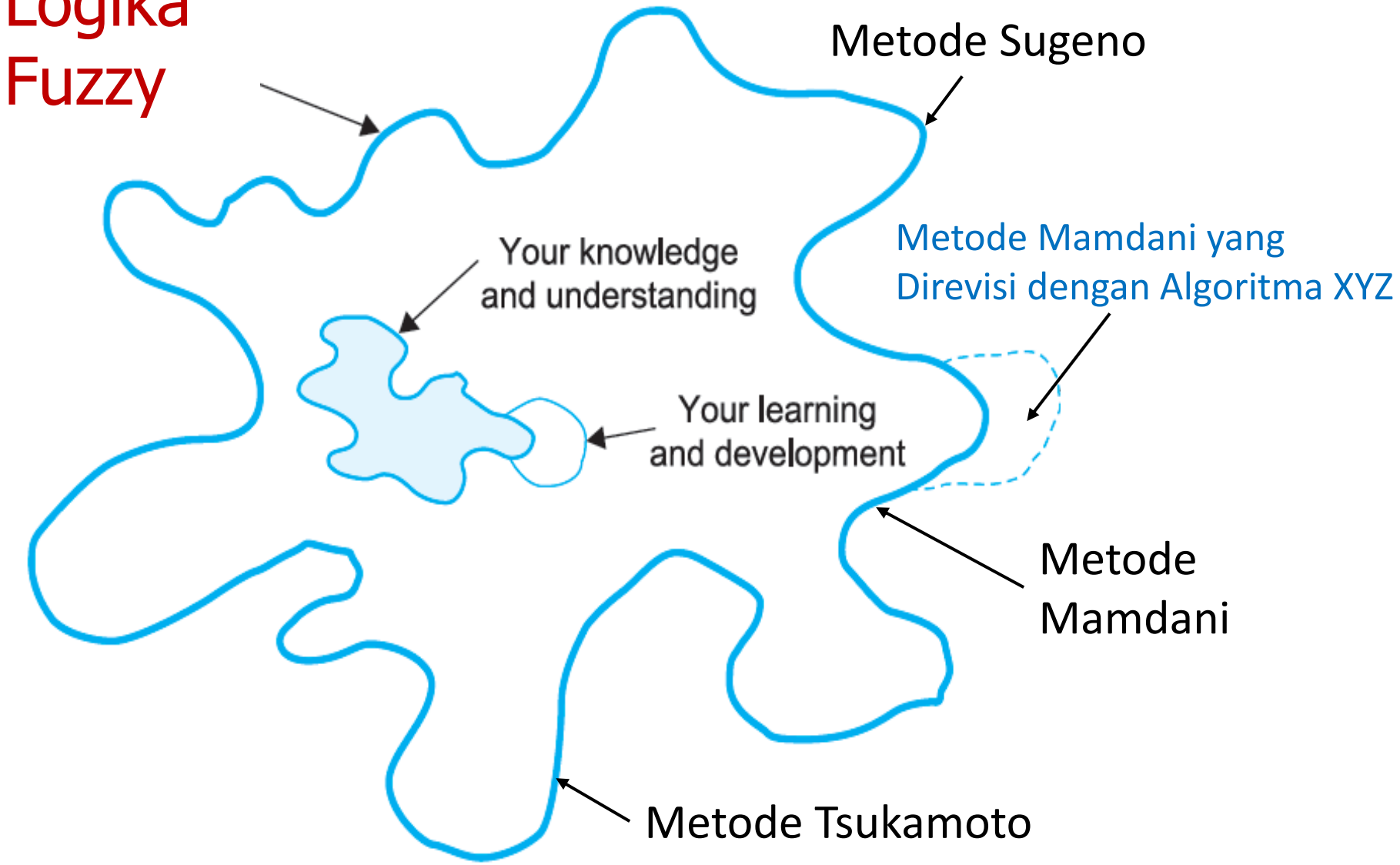
(Berndtsson et al., 2008)

Bentuk Kontribusi ke Pengetahuan

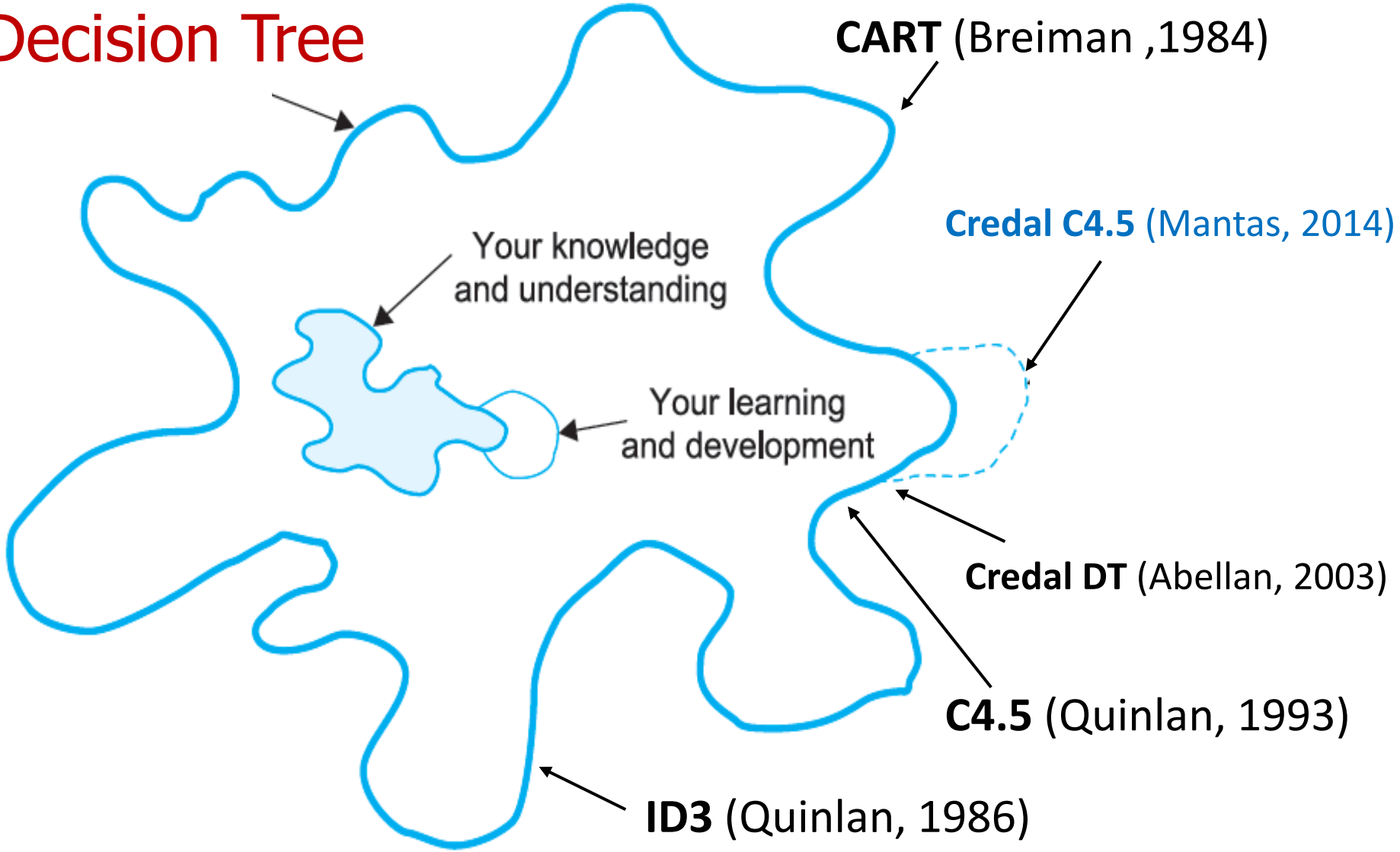


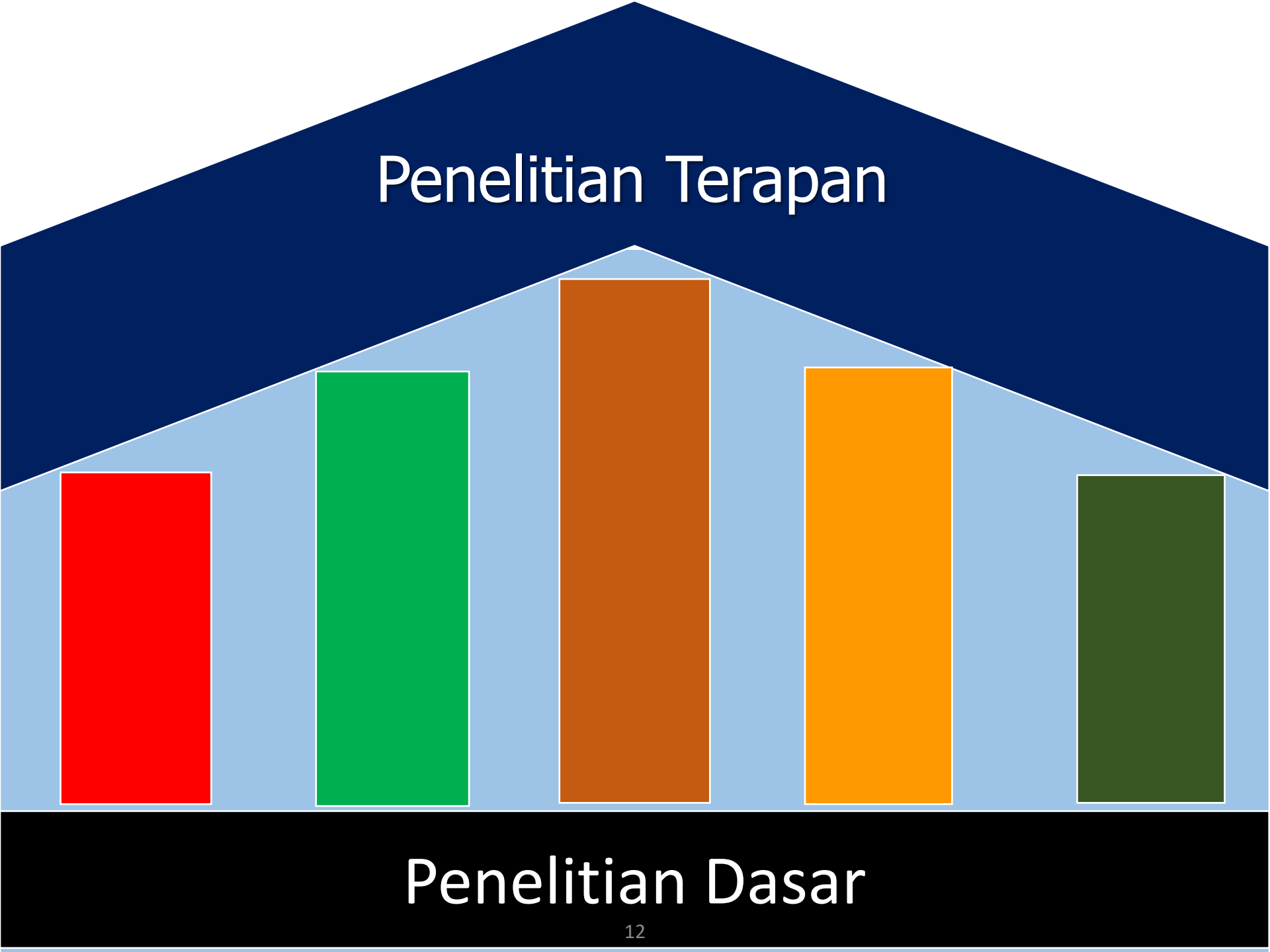
(Dawson, 2019)

Bentuk Kontribusi ke Pengetahuan

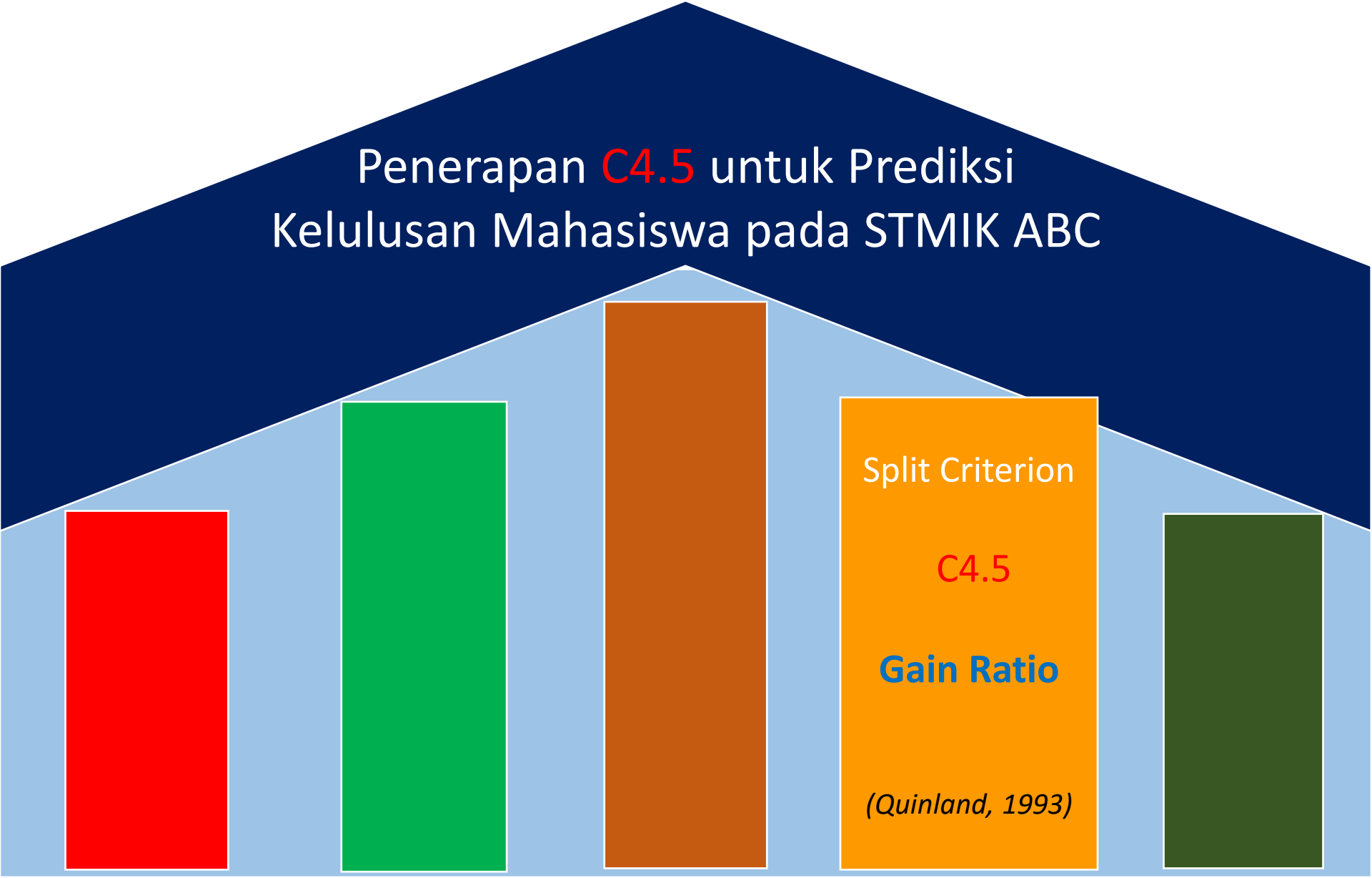


Bentuk Kontribusi ke Pengetahuan



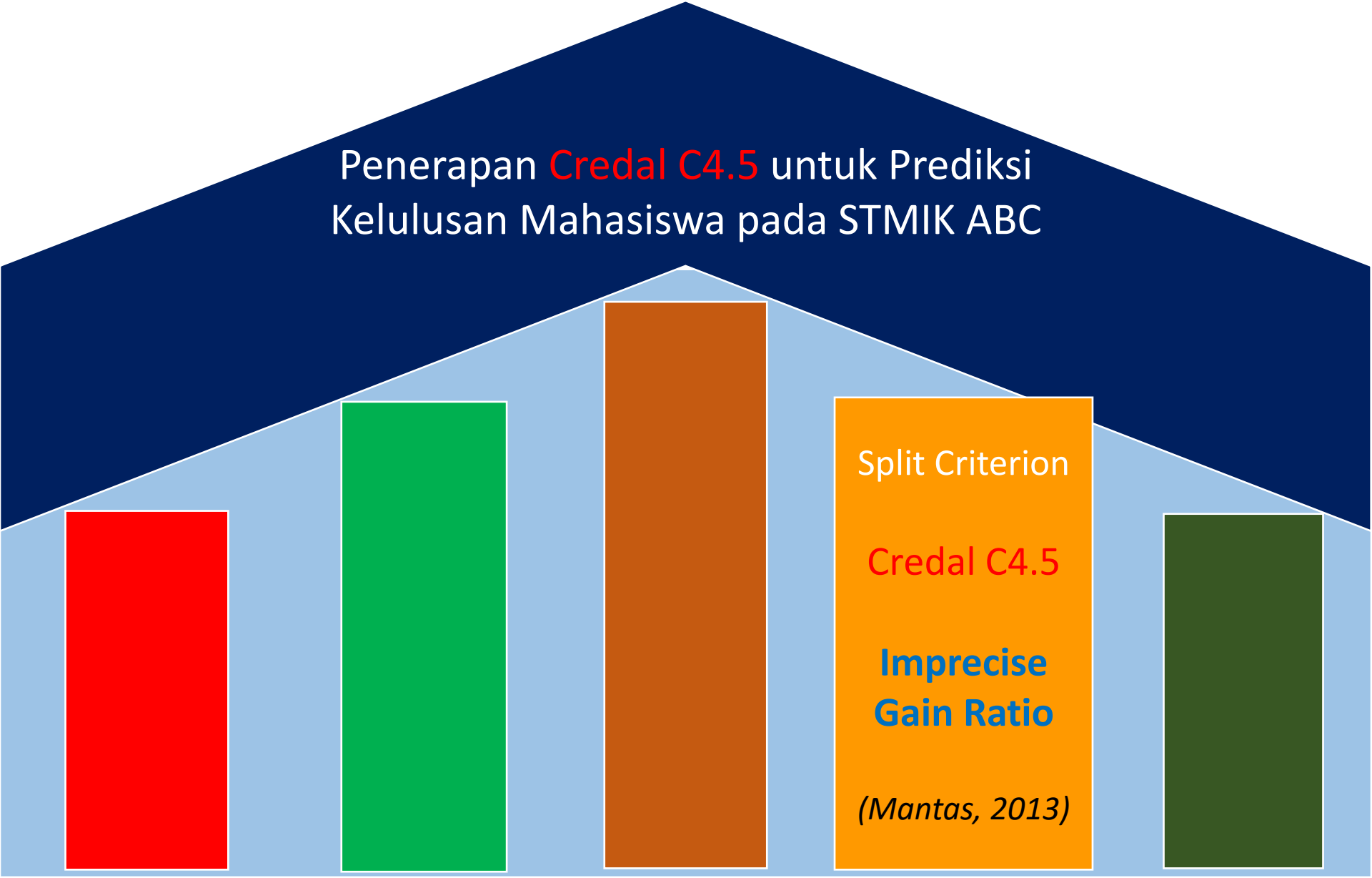

Penelitian Terapan
Penelitian Dasar

Penerapan C4.5 untuk Prediksi Kelulusan Mahasiswa pada STMIK ABC



Teori Gain *(Kullback & Leibler, 1951)*

Penerapan Credal C4.5 untuk Prediksi Kelulusan Mahasiswa pada STMIK ABC



Imprecise Probability Theory *(Walley, 1996)*

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Credal-C4.5: Decision tree based on imprecise probabilities to classify noisy data

Carlos J. Mantas, Joaquín Abellán *

Department of Computer Science & Artificial Intelligence, University of Granada, ETSI Informática, c/Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

Memperbaiki C4.5

Contents lists available at ScienceDirect

Information and Software Technology

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

Simplifying effort estimation based on Use Case Points ☆

M. Ochodek *, J. Nawrocki, K. Kwarciak

Poznan University of Technology, Institute of Computing Science, ul. Piotrowo 2, 60-965 Poznań, Poland

Memperbaiki Use Case Points

Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling

Shengxiang Yang, *Member, IEEE*, and Sadaf Naseem Jat

***Abstract*—The university course timetabling problem (UCTP) is a combinatorial optimization problem, in which a set of events has to be scheduled into time slots and located into suitable rooms. The design of course timetables for academic institutions is a very difficult task because it is an NP-hard problem. This paper investigates genetic algorithms (GAs) with a guided search strategy and local search (LS) techniques for the UCTP. The guided search strategy is used to create offspring into the population based on a data structure that stores information extracted from good individu-**

The research on timetabling problems has a long history of more than 40 years, starting with Gotlieb in 1962 [22]. Researchers have proposed various timetabling approaches by using graph coloring methods, constraint-based methods, population-based approaches (e.g., genetic algorithms (GAs), ant-colony optimization, and memetic algorithms), metaheuristic methods (e.g., tabu search (TS), simulated annealing (SA),

Memperbaiki Genetic Algorithms

Akademisi vs Technopreneur

Meja Indah

Meja Kuat

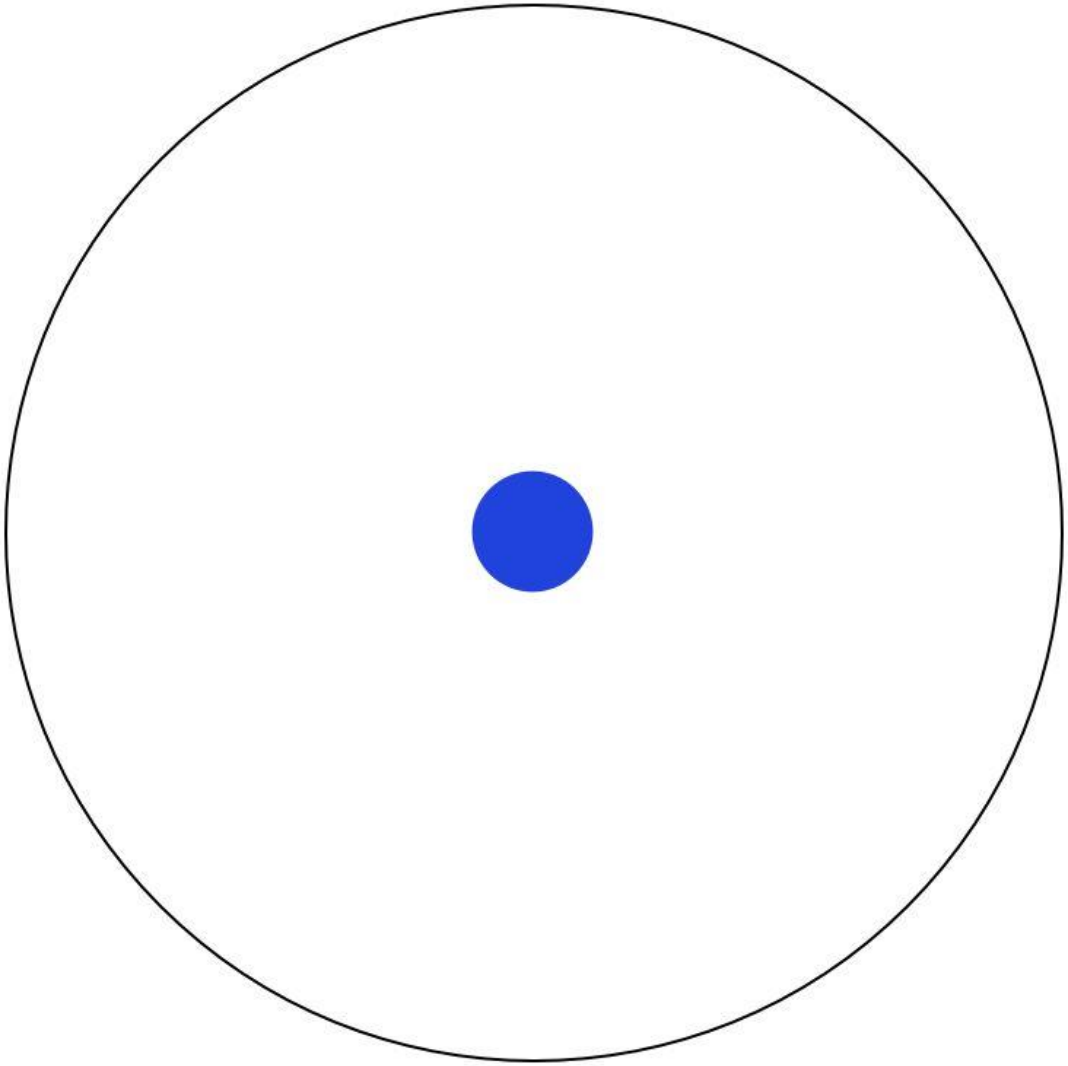
Meja Luas

- Technopreneur?
  1. Jual Produk
  2. Beri Nilai Tambah Produk
  3. Jadikan Aset, Jual Layanan
- Akademisi?
  - Pelajari, Preteli Komponen
  - Ciptakan Meja Baru yang Berbeda dengan 3 Meja Itu

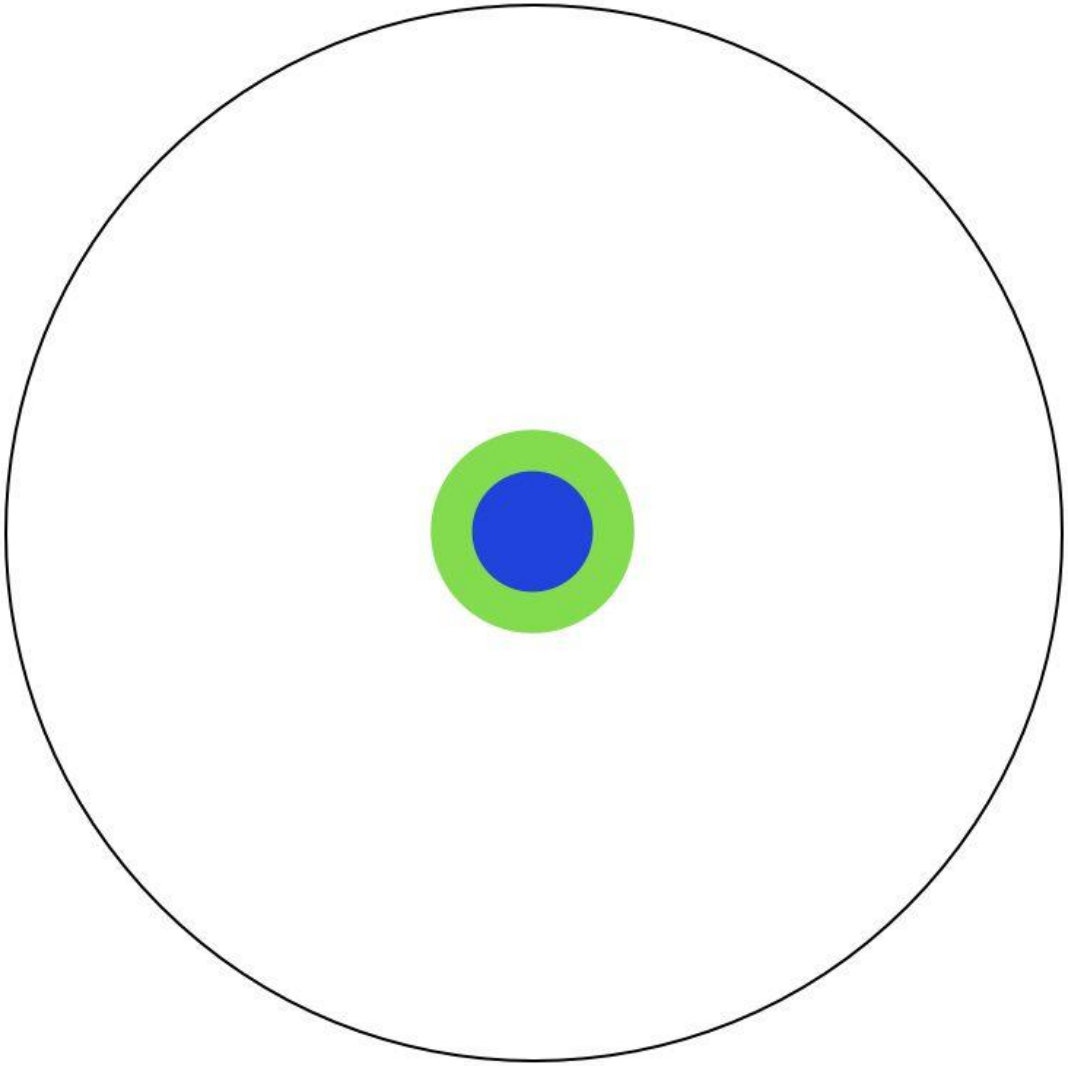
Penelitian yang Berkualitas Tinggi

Topik dan skalanya kecil, fokus, dalam, dan membawa pengaruh yang besar ke bidang penelitian kita

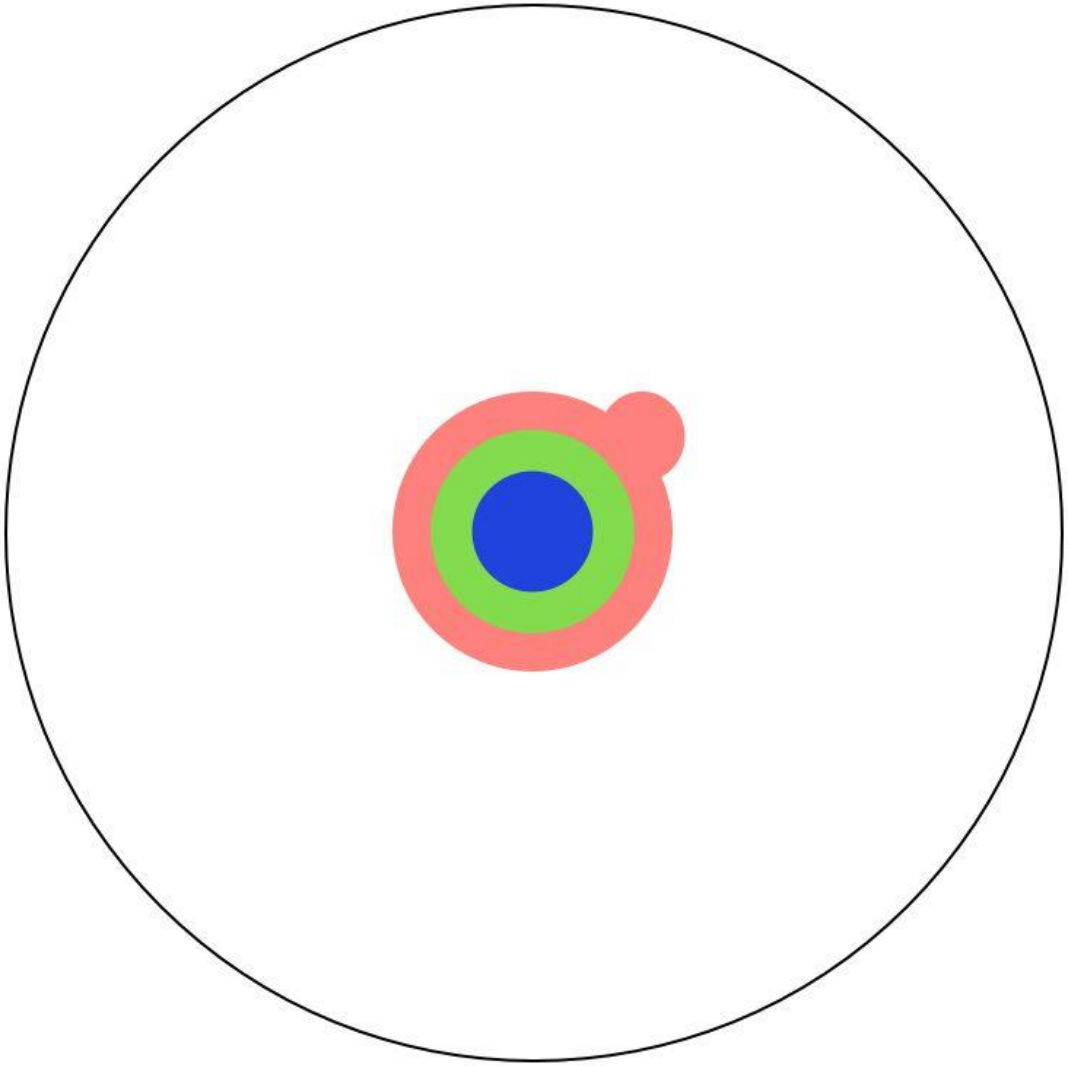
The Illustrated Guide to a Ph.D (Might, 2010)



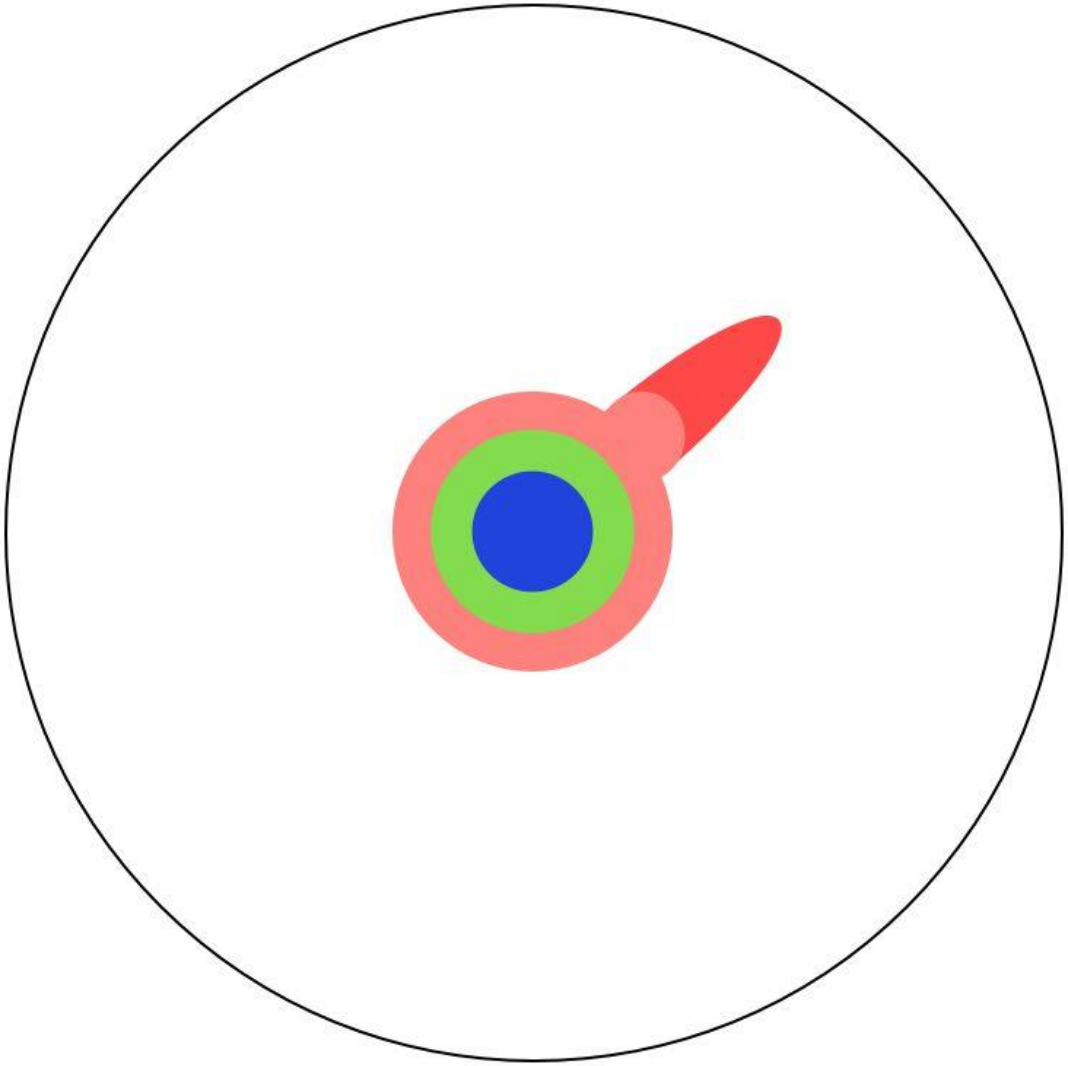
The Illustrated Guide to a Ph.D (Might, 2010)



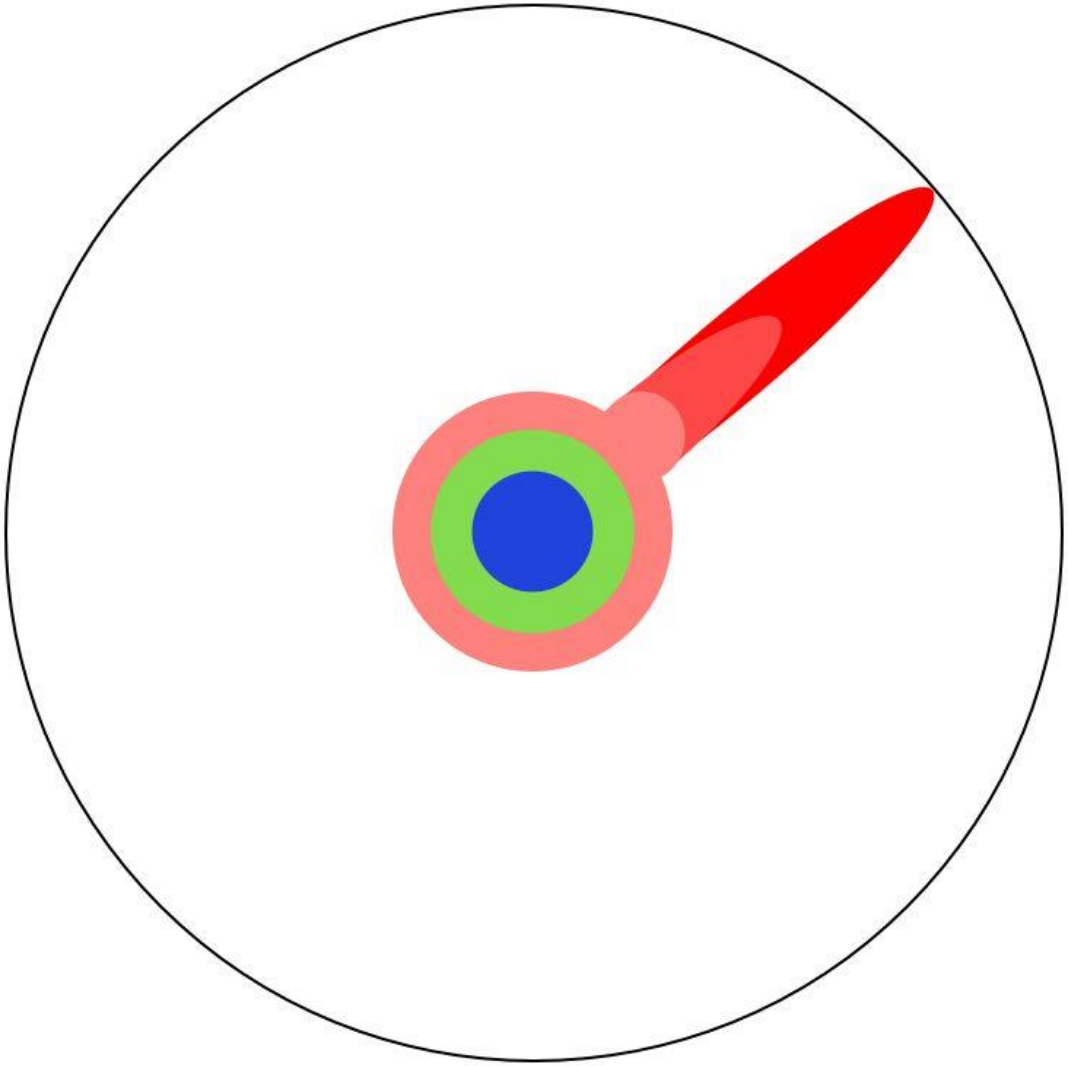
The Illustrated Guide to a Ph.D (Might, 2010)



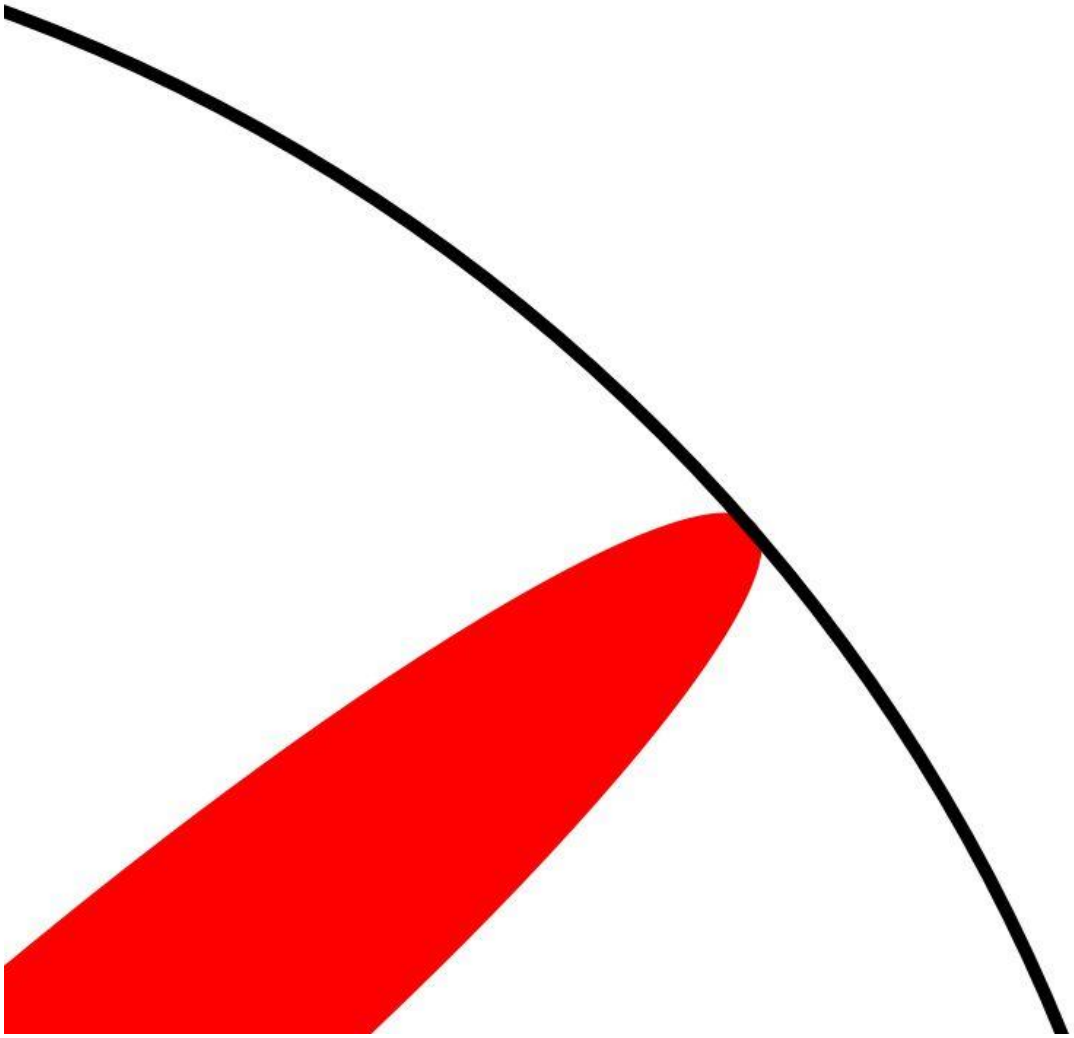
The Illustrated Guide to a Ph.D (Might, 2010)



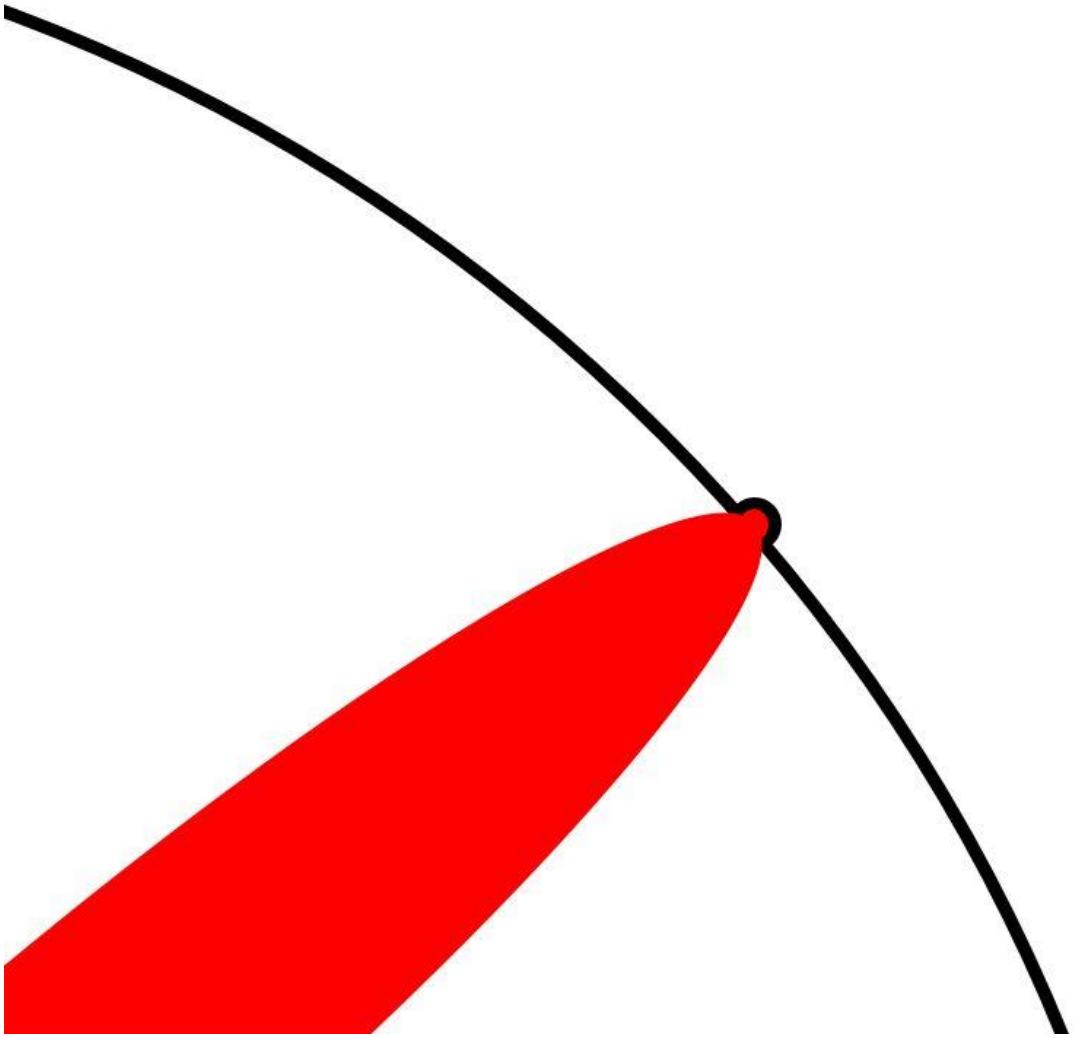
The Illustrated Guide to a Ph.D (Might, 2010)



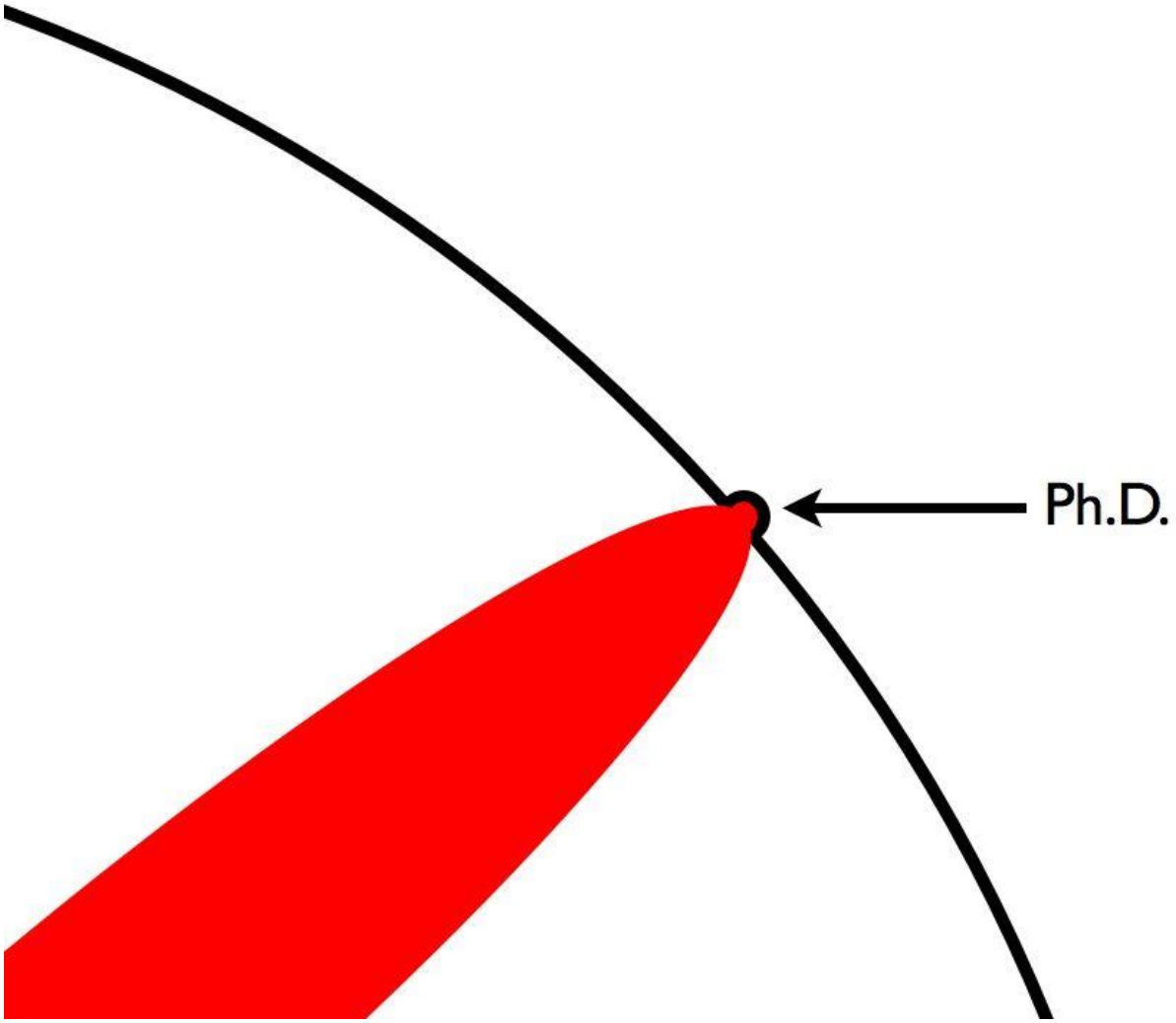
The Illustrated Guide to a Ph.D (Might, 2010)



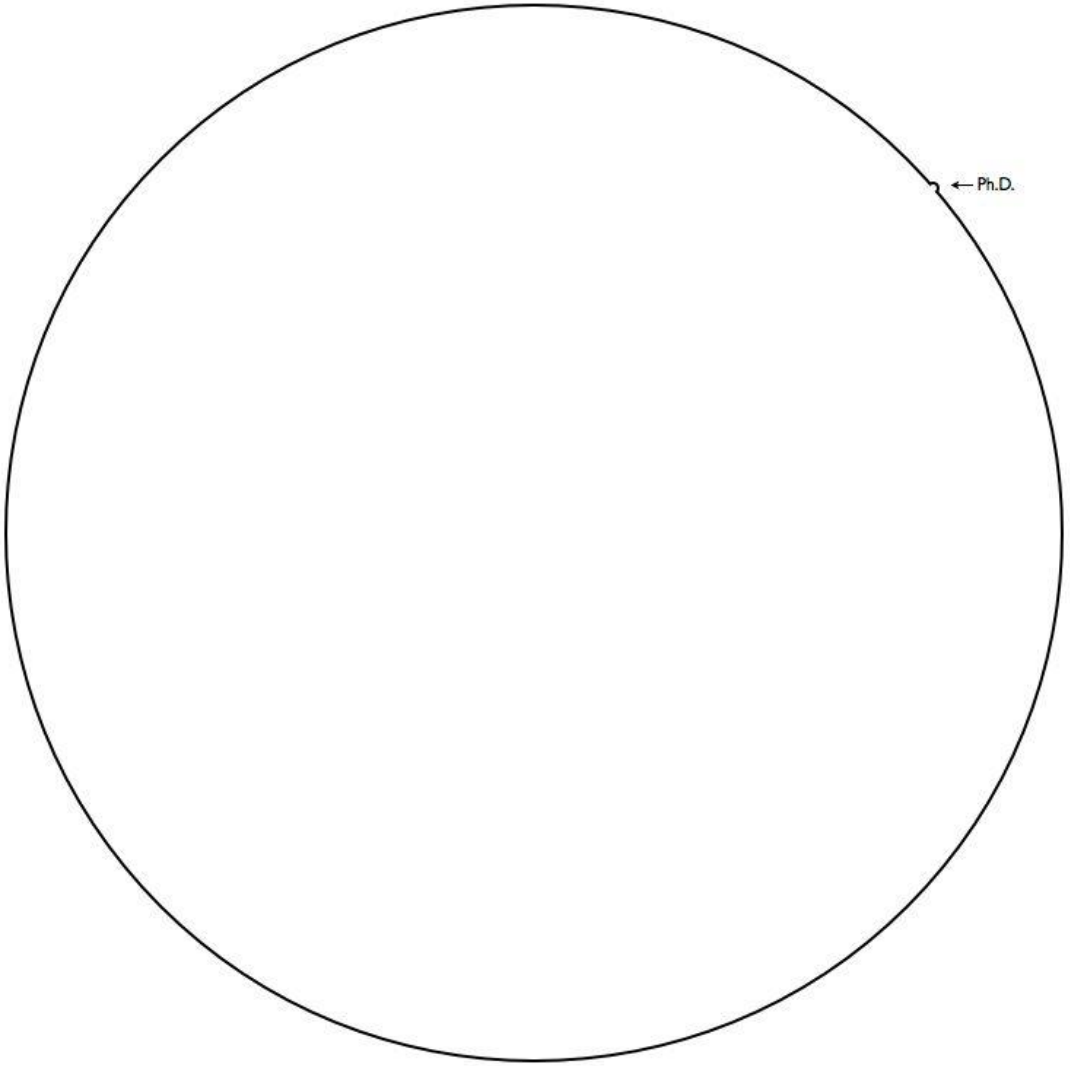
The Illustrated Guide to a Ph.D (Might, 2010)



The Illustrated Guide to a Ph.D (Might, 2010)



The Illustrated Guide to a Ph.D (Might, 2010)



Pengembangan Software vs Penelitian

- Membangun software bukanlah tujuan utama penelitian, hanya *testbed* untuk mempermudah kita dalam mengukur hasil penelitian
  - Tidak ada listing code, UML atau screenshot software di paper-paper journal (SCOPUS/WoS), kecuali penelitian tentang perbaikan paradigma pemrograman, analisis design, dsb

- Ketika pada penelitian kita mengusulkan perbaikan suatu algoritma (*proposed method*)
  - Bidang image processing, topik penelitian face recognition, memikirkan perbaikan metode/algoritma untuk pengenalan wajah dengan akurat/efisien
  - Bidang data mining, topik decision tree, memikirkan perbaikan algoritma decision tree sehingga bisa memprediksi (klasifikasi) dengan lebih akurat
  - Untuk mempermudah eksperimen dan evaluasi, kita menulis kode program (software) untuk menguji dan mengevaluasi performance dari algoritma yang kita usulkan

Topik Penelitian Bidang Software Engineering?

Penelitian bidang software engineering bukan penelitian tentang pengembangan software yang hasil akhirnya produk software, tapi penelitian untuk perbaikan metodologi pengembangan software

SOFTWARE LIFECYCLE

	1. SOFTWARE REQUIREMENTS	2. SOFTWARE DESIGN	3. SOFTWARE CONSTRUCTION	4. SOFTWARE TESTING	5. SOFTWARE MAINTENANCE
SWEBOK Knowledge Areas					
SFIA Competencies	• Requirements definition and management • Real-time/embedded systems development • Methods and tools • Testing • Configuration management • Safety engineering	• Software design • Systems design • Real-time/embedded systems development • Safety engineering	• Programming / software development • Real-time/embedded systems development • Systems integration and build • Testing	• Testing • Systems integration and build • Real-time/embedded systems development • Quality assurance	• Release and deployment • Application support

FOUNDATIONAL ACTIVITIES

	6. SOFTWARE CONFIGURATION	10. SOFTWARE QUALITY	9. SOFTWARE ENGINEERING MODELS
SWEBOK Knowledge Areas			
SFIA Competencies	• Configuration management	• Quality management • Quality assurance • Testing • Safety assessment • Conformance review	• Requirements definition and management • Systems design • Software design

SOFTWARE ENGINEERING MANAGEMENT AND GOVERNANCE

	7. SOFTWARE ENGINEERING MANAGEMENT	8. SOFTWARE ENGINEERING PROCESS	12. SOFTWARE ENGINEERING ECONOMICS	11. SOFTWARE ENGINEERING PROFESSIONAL PRACTICE
SWEBOK Knowledge Areas				
SFIA Competencies	• Systems development management • Project management	• Systems development management • Quality management • Measurement • Methods and tools • Organisational capability development	• Systems development management	• SFIA levels of responsibility

2. Tahapan Penelitian

Tahapan Penelitian Computing

Literature Review

1. Penentuan Bidang Penelitian (***Research Field***)
2. Penentuan Topik Penelitian (***Research Topic***)
3. Penentuan Masalah Penelitian (***Research Problem***)
4. Perangkuman Metode-Metode Yang Ada (***State-of-the-Art Methods***)
5. Penentuan Metode Yang Diusulkan (***Proposed Method***)
6. Evaluasi Metode Yang Diusulkan (***Evaluation***)
7. Penulisan Ilmiah dan Publikasi Hasil Penelitian (***Publications***)

**https://www.site.uottawa.ca/~bochmann/dsrg/how-to-do-good-research/*
**http://romisatriawahono.net/2013/01/23/tahapan-memulai-penelitian-untuk-mahasiswa-galau/*

Literature Review

Bingkai Penelitian

Selalu Dilakukan di Setiap Tahapan

Manfaat Literature Review

- Memperdalam pengetahuan tentang bidang yang diteliti (*Textbooks*)
- Mengetahui hasil penelitian yang berhubungan dan yang sudah pernah dilaksanakan (Related Research) (*Paper*)
- Mengetahui perkembangan ilmu pada bidang yang kita pilih (state-of-the-art) (*Paper*)

- Mencari dan memperjelas masalah penelitian (*Paper*)

Konsep Literature Review

- Literature Review is a critical and in-depth evaluation of previous research (Shuttleworth, 2009) (*https://explorable.com/what-is-a-literature-review*)

- A summary and synopsis of a particular area of research, allowing anybody reading the paper to establish the reasons for pursuing a particular research

- A good Literature Review evaluates quality and findings of previous research (State-of-the-Art Methods)

Metode Literature Review

- Types and Methods of Literature Review:
  1. Traditional Review
  2. Systematic Literature Review or Systematic Review
  3. Systematic Mapping Study (Scoping Study)
  4. Tertiary Study

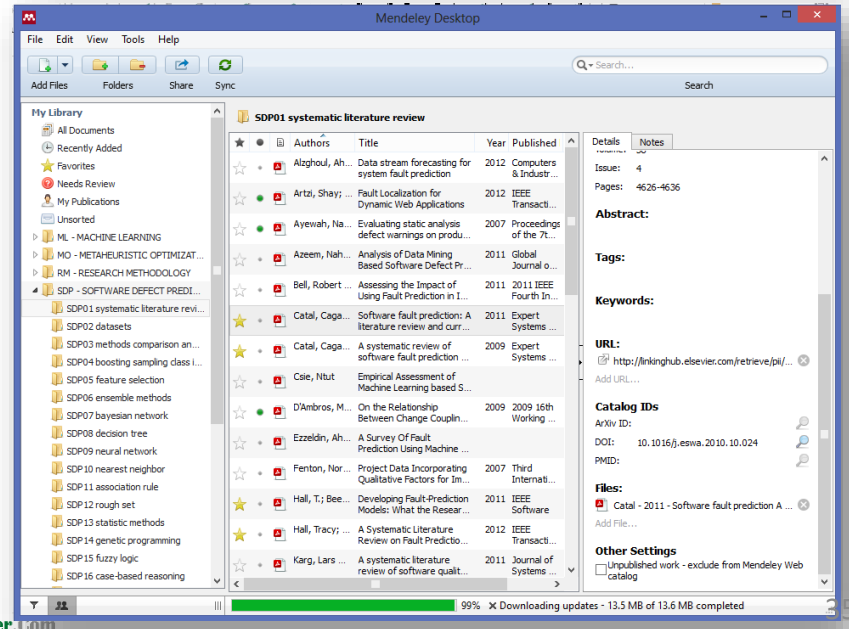
- SLR is now well established review method in the field of software engineering

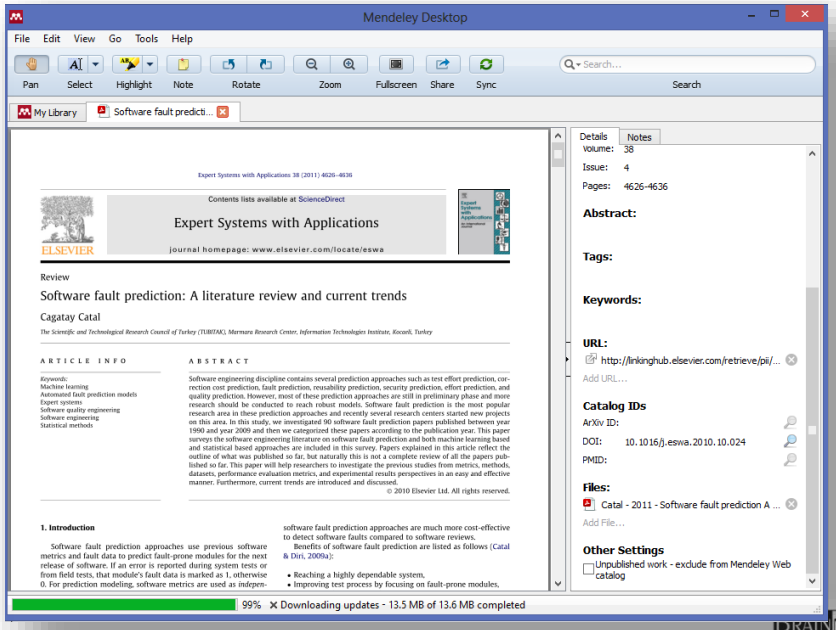
(Kitchenham & Charters, Guidelines in performing Systematic Literature Reviews in Software Engineering, EBSE Technical Report version 2.3, 2007)

Jumlah Literatur yang Harus Dibaca

- Adagium level pendidikan dan jumlah literatur yang harus dibaca untuk penyelesaian penelitian
  - **S1**: 20-70 paper
  - **S2**: 70-200 paper
  - **S3**: 200-700 paper

- Kepala jadi pusing, bukan karena kita banyak membaca, tapi karena yang kita baca memang "belum banyak"





Jenis Literatur Ilmiah

1. **Paper dari Journal**
2. Paper dari Book Chapter
3. Paper dari Conference (Proceedings)
4. Thesis dan Disertasi
5. Report (Laporan) dari Organisasi yang Terpercaya
6. Buku Textbook

** Prioritaskan paper yang terbit di journal yang terindeks oleh ISI dan SCOPUS, cek dengan http://scimagojr.com*

www.scimagojr.com/journalsearch.php

SJR SCImago Journal & Country Rank

EST MODUS IN REBUS
Horatio (Satire 1,1,106)

- Home
- Journal Rankings
- Journal Search
- Country Rankings
- Country Search
- Compare
- Map Generator
- Help
- About Us

Journal Search

Search query

Journal of Systems and Software | in | Journal Title | Search

Exact phrase

Please, select journal:

1. Journal of Systems and Software. United States.

Related product

SJR SCImago Journal & Country Rank

EST MODUS IN REBUS
Horatio (Satire 1,1,106)

Home
Journal Rankings
Journal Search
Country Rankings
Country Search
Compare
Map Generator
Help
About Us

Journal Search

Search query

in Journal Title Search

Exact phrase

Journal of Systems and Software

Country: United States

Subject Area: Computer Science

Subject Category:

Category	Quartile (Q1 means highest values and Q4 lowest values)														
	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
Hardware and Architecture	Q2	Q2	Q1	Q2	Q2	Q2	Q1	Q2	Q2	Q2	Q1	Q2	Q1	Q1	Q1
Information Systems	Q3	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2
Software	Q3	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2	Q2

Publisher: Elsevier Inc.. **Publication type:** Journals. **ISSN:** 01641212

Coverage: 1979-2014

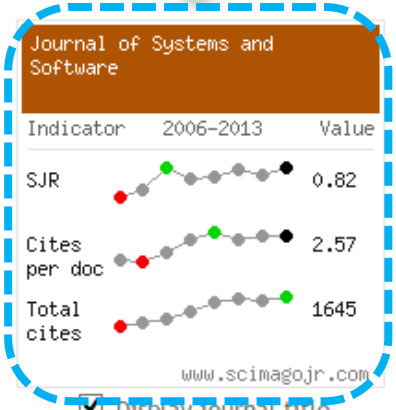
H Index: 60

Scope:

The Journal of Systems and Software publishes papers covering all aspects of programming methodology, software engineering, and related hardware-software-systems issues. [...]

Show full scope

Show this information in your own website



Display journal title

Tahapan Penelitian Computing

Literature Review

1. Penentuan Bidang Penelitian (***Research Field***)
2. Penentuan Topik Penelitian (***Research Topic***)
3. Penentuan Masalah Penelitian (***Research Problem***)
4. Perangkuman Metode-Metode Yang Ada (***State-of-the-Art Methods***)
5. Penentuan Metode Yang Diusulkan (***Proposed Method***)
6. Evaluasi Metode Yang Diusulkan (***Evaluation***)
7. Penulisan Ilmiah dan Publikasi Hasil Penelitian (***Publications***)

**https://www.site.uottawa.ca/~bochmann/dsrg/how-to-do-good-research/*
**http://romisatriawahono.net/2013/01/23/tahapan-memulai-penelitian-untuk-mahasiswa-galau/*

1. Penentuan Bidang Penelitian

- Ingat kembali seluruh mata kuliah yang sudah kita terima di perkuliahan
- Lakukan literature review untuk memperkuat penetapan bidang penelitian yang akan diambil
- Bidang penelitian disiplin ilmu computing:

Software Engineering	Data Mining
Image Processing	Computer Vision
Networking	Human Computer Interaction
Soft Computing	Information Retrieval
Bioinformatics	dsb

- Tentukan berdasarkan passion!
- Peneliti yang baik tidak akan mengubah bidang penelitian yang diambil, karena itu membentuk core competency dan track publikasi ilmiah yang dilakukan
- Contoh: Saya memilih bidang Software Engineering (SE)

2. Penentuan Topik Penelitian

1. Searching di ScienceDirect.Com, Springerlink, IEEE Explore, Google (Scholar):
   - research trends challenge topics on NAMA BIDANG
2. Untuk mempercepat pembelajaran, temukan survey paper berbentuk Tertiery Study (SLR dari SLR), karena isinya sudah merangkumkan satu bidang penelitian
3. Lanjutkan penentuan topik penelitian dengan menemukan suvey/review paper (SLR, SMS), karena survey/review paper yang masuk jurnal terindeks pasti membahas satu topik penelitian

- Contoh:
  - Dari paper-paper survey dan review tentang software engineering, saya tahu trend penelitian di bidang SE:
    1. Autonomic Computing or Self Adaptive Software
    2. Software Effort/Cost Estimation
    3. Software Defect Prediction
    4. Software Process Improvement
    5. Service Oriented Architecture
    6. etc
  - Saya mengambil topik penelitian: Software Defect Prediction

3. Penentuan Masalah Penelitian

- Searching di ScienceDirect.Com, Springerlink, IEEE Explore, Google (Scholar):
  - Survey review on NAMA TOPIK
  - Research problem challenge on NAMA TOPIK
- Dari "survey paper" yang ditemukan, kejar sampai dapat semua "technical paper" yang ada di daftar referensinya
- Dari puluhan/ratusan/ribuan paper yang didapat lakukan scanning, pilih paper journal yang terindeks SCOPUS/ISI, 3 tahun terakhir, dan peta kan masalah penelitian yang ada di paper-paper itu
  - Gunakan Mendeley untuk mempermudah pekerjaan kita dalam mengelola paper
- Pilih satu atau dua masalah penelitian yang kita anggap menarik dan menantang, dan jadikan itu masalah penelitian kita

Susun Research Problem dan Landasan

Masalah Penelitian	Landasan Literatur
Data set pada prediksi cacat software berdimensi tinggi, memiliki atribut yang bersifat noisy, dan classnya bersifat tidak seimbang, menyebabkan penurunan akurasi pada prediksi cacat software	There are noisy data points in the software defect data sets that can not be confidently assumed to be erroneous using such simple method *(Gray, Bowes, Davey, & Christianson, 2011)*
	The performances of software defect prediction improved when irrelevant and redundant attributes are removed *(Wang, Khoshgoftaar, & Napolitano, 2010)*
	The software defect prediction performance decreases significantly because the dataset contains noisy attributes *(Kim, Zhang, Wu, & Gong, 2011)*
	Software defect datasets have an imbalanced nature with very few defective modules compared to defect-free ones *(Tosun, Bener, Turhan, & Menzies, 2010)*
	Imbalance can lead to a model that is not practical in software defect prediction, because most instances will be predicted as non-defect prone *(Khoshgoftaar, Van Hulse, & Napolitano, 2011)*
	Software fault prediction data sets are often highly imbalanced *(Zhang & Zhang, 2007)*

Masalah Kehidupan vs Masalah Penelitian

- Penelitian dilakukan karena ada masalah penelitian
- Dimana masalah penelitian sendiri muncul karena ada latar belakang masalah penelitian
- Latar belakang masalah penelitian itu berangkatnya bisa dari masalah kehidupan (obyek penelitian)

Contoh Alur Latar Belakang Masalah Penelitian

- Nilai tukar uang adalah faktor penting pada perekonomian suatu negara. Nilai tukar uang perlu diprediksi supaya kebijakan perekonomian bisa diambil dengan lebih akurat dan efisien…
- Metode untuk prediksi nilai tukar yang saat ini digunakan adalah regresi linier, neural network dan support vector machine…
- Regresi linier memiliki kelebihan A dan kelemahan B…
- Neural network memiliki kelebihan C dan kelemahan D…
- Support vector machine memiliki kelebihan bisa mengatasi masalah B (pada regresi linier) dan D (pada neural network)… tapi memiliki kelemahan E

- Masalah penelitian pada penelitian di atas?
  - Kebijakan perekonomian negara?
  - Prediksi nilai tukar yang?
  - Metode apa yang seharusnya dipakai untuk prediksi nilai tukar?
- Masalah: Support vector machine memiliki kelebihan memecahkan masalah B dan D (argumentasi dipilih), tapi memiliki kelemahan E
- Tujuan: Menerapkan metode XYZ untuk memecahkan masalah E pada support vector machine

Contoh Alur Latar Belakang Masalah Penelitian

- Kemacetan lalu lintas di kota besar semakin meningkat
- Penyebab kemacetan adalah traffic light persimpangan jalan
- Traffic light yang ada adalah statis (tetap waktunya) sehingga tidak dapat menyelesaikan kondisi kepadatan kendaraan yang di berbagai waktu
- Traffic light harus didesain dinamis sesuai perubahan berbagai parameter
- Metode untuk menentukan waktu yang tepat secara dinamis dapat menggunakan AHP, ANP, Fuzzy Logic,
- AHP memiliki kelebihan A dan kelemahan B…
- ANP memiliki kelebihan C dan kelemahan D…
- Fuzzy logic memiliki kelebihan bisa mengatasi masalah B (pada AHP) dan D (pada ANP)… tapi memiliki kelemahan E

- Masalah penelitian pada penelitian di atas?
  - Bagaimana mengata emacetan lalu lintas?
  - Bagaimana mendes raffic light?
  - Metode apa yang sebaiknya dipakai untuk penentuan traffic light secara dinamis?
- Masalah: Fuzzy logic memiliki kelebihan memecahkan masalah B dan D (argumentasi dipilih), tapi memiliki kelemahan E
- Tujuan: Menerapkan metode XYZ untuk memecahkan masalah E pada fuzzy logic

Contoh Masalah Penelitian

- ***Ungu**: Obyek Data (Opsional, Bisa Data Publik)*
- ***Oranye**: Topik (Obyek Metode yang Diperbaiki)*
- ***Merah**: Masalah Penelitian*
- ***Hijau**: Metode Perbaikan yang Diusulkan*
- ***Biru**: Pengukuran Penelitian (Tidak Harus Masuk Judul)*

Penerapan Particle Swarm Optimization untuk Pemilihan Parameter Secara Otomatis pada Support Vector Machine untuk Prediksi Produksi Padi

Research Problem (RP)	Research Question (RQ)	Research Objective (RO)
SVM dapat memecahkan masalah 'over-fitting', lambatnya konvergensi, dan sedikitnya data training, akan tetapi memiliki kelemahan pada sulitnya pemilihan parameter SVM yang sesuai yang mengakibatkan akurasi tidak stabil	Seberapa meningkat akurasi metode SVM apabila PSO diterapkan pada proses pemilihan parameter?	Menerapkan PSO untuk pemilihan parameter yang sesuai pada SVM (C, lambda dan epsilon) , sehingga hasil prediksinya lebih akurat

Contoh Masalah Penelitian

- Masalah Penelitian (*Research Problem*):
  - Neural network terbukti memiliki performa bagus untuk menangani data besar seperti pada data prediksi harga saham, akan tetapi memiliki kelemahan pada pemilihan arsitektur jaringannya yang harus dilakukan secara trial error, sehingga tidak efisien dan mengakibatkan hasil prediksi kurang akurat
- Rumusan Masalah (*Research Question*):
  - Bagaimana peningkatan akurasi dan efisiensi neural network apabila pada pemilihan arsitektur jaringan diotomatisasi menggunakan algoritma genetika?
- Tujuan Penelitian (*Research Objective*):
  - Menerapkan algoritma genetika untuk mengotomatisasi pemilihan arsitektur jaringan pada neural nework sehingga lebih efisien dan hasil prediksi lebih akurat

Contoh Masalah Penelitian

- Research Problem (RP):
  - Algoritma K-Means merupakan algoritma clustering yang populer karena efisien dalam komputasi, akan tetapi memiliki kelemahan pada sulitnya penentuan K yang optimal dan komputasi yang tidak efisien bila menangani data besar (Zhao, 2010)
- Research Question (RQ):
  - Seberapa efisien algoritma Bee Colony bila digunakan untuk menentukan nilai K yang optimal pada K-Means?
- Research Objective (RO):
  - Menerapkan algoritma bee colony untuk menentukan nikai K yang optimal pada K-Means sehingga komputasi lebih efisien

4. Perangkuman Metode Yang Ada

- Lakukan literature review, pahami semua paper penelitian yang tujuannya memecahkan masalah yang sama dengan yang kita pilih
- Pahami metode/algoritma terkini yang mereka gunakan untuk memecahkan masalah penelitian mereka
  - Ini yang disebut dengan state-of-the-art method
- Dalam bidang computing, metode biasanya berupa algoritma yang secara sistematis, logis dan matematis menyelesaikan masalah

The State-of-the-Art Method

- The highest level of development, as of a device, technique, or scientific field, achieved at a particular time
- The level of development (as of a device, procedure, process, technique, or science) reached at any particular time usually as a result of modern methods (*Merriam Webster Dictionary*)
  - This machine is an example of state-of-the-art technology
  - The state of the art in this field is mostly related to the ABC technology
- A concept used in the process of assessing and asserting novelty and inventive step (*European Patent Convention (EPC)*)

State-of-the-Art Frameworks in Software Defect Prediction

Menzies Framework

(Menzies et al. 2007)

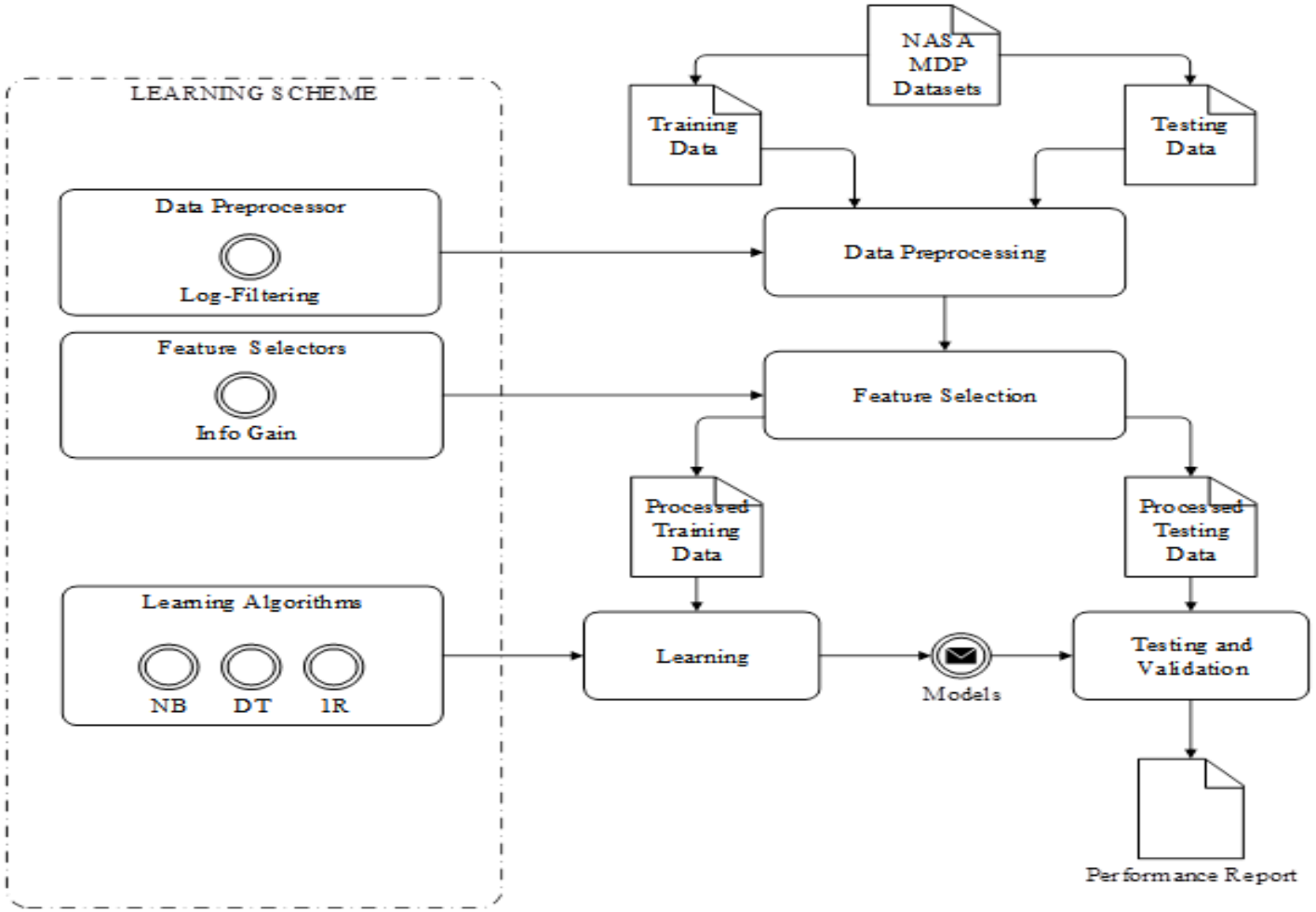
Lessmann Framework

(Lessmann et al. 2008)

Song Framework

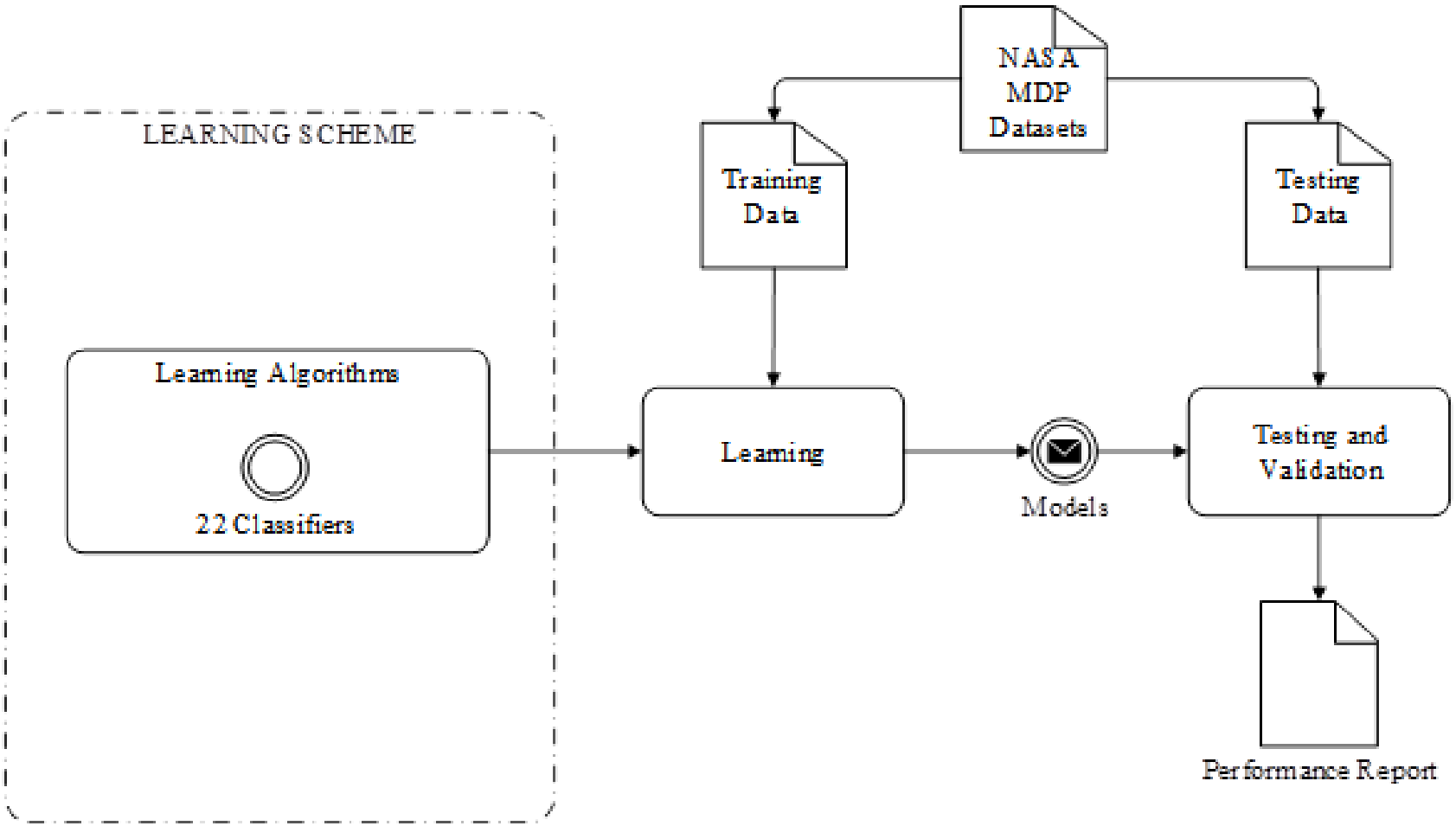
(Song et al. 2011)

Menzies Framework (Menzies et al. 2007)



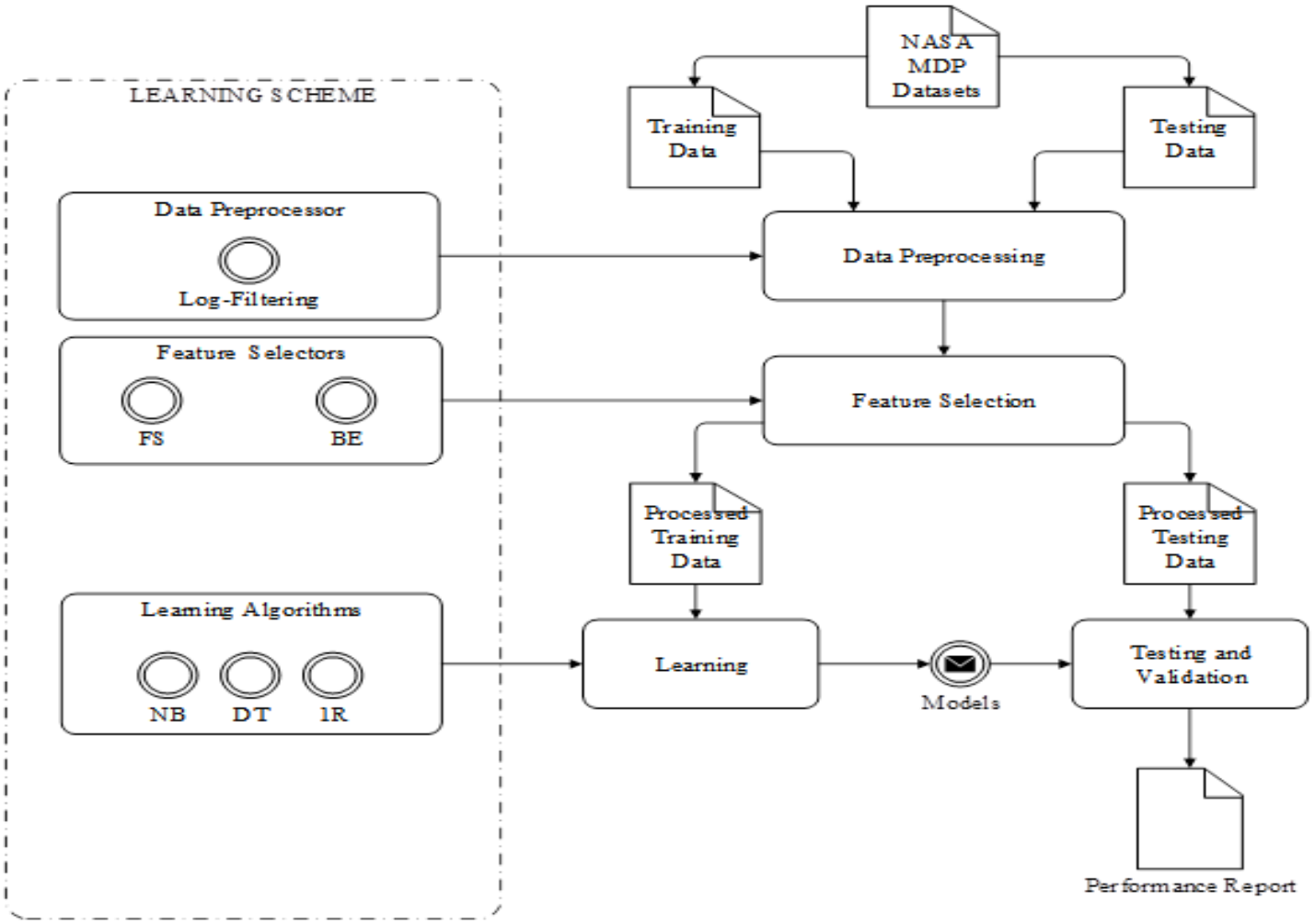
Framework	Dataset	Data Preprocessor	Feature Selectors	Meta-learning	Classifiers	Parameter Selectors	Validation Methods	Evaluation Methods
(Menzies et al. 2007)	NASA MDP	**Log Filtering**	**Info Gain**	-	**3 algorithms (DT, 1R, NB)**	-	10-Fold X Validation	ROC Curve (AUC)

Lessmann Framework (Lessmann et al. 2008)



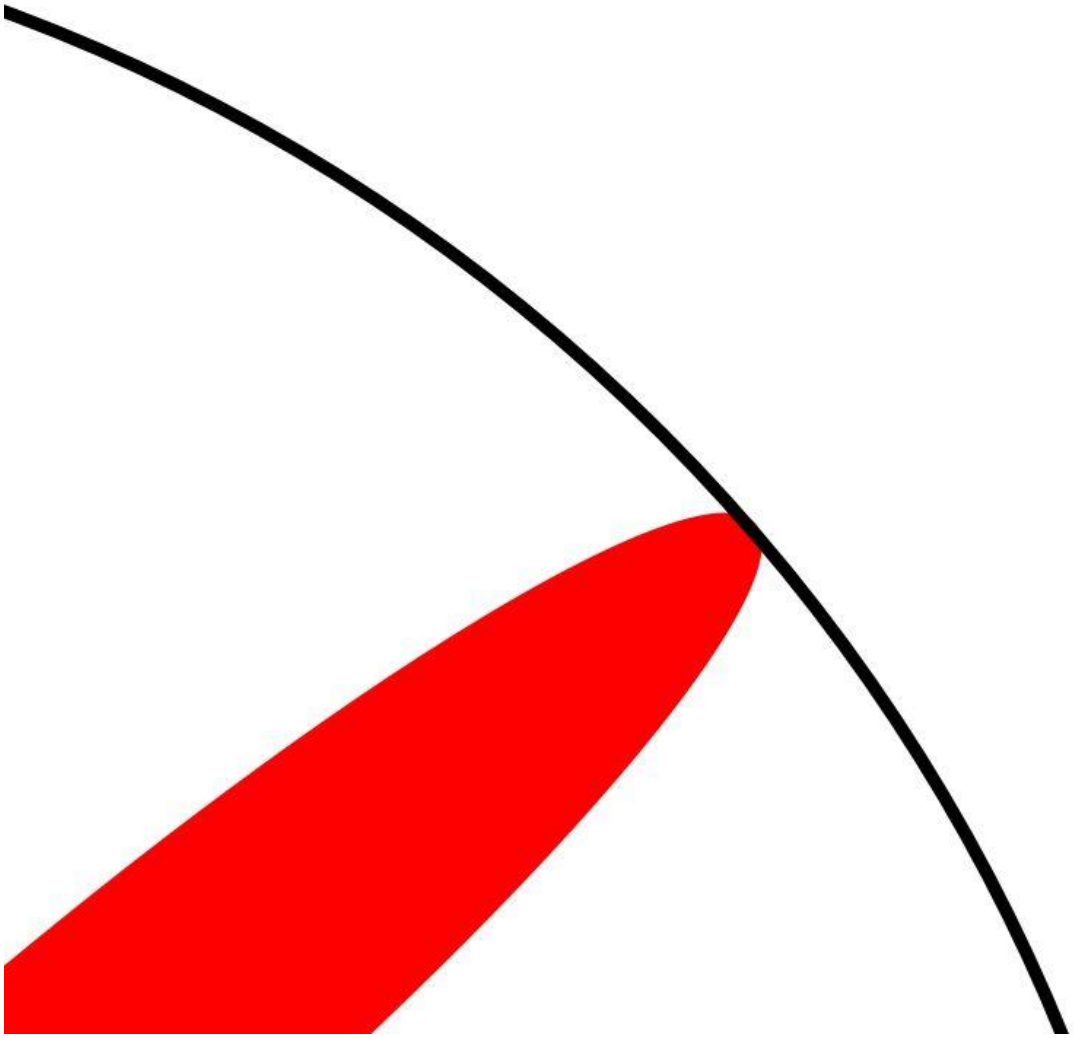
Framework	Dataset	Data Preprocessor	Feature Selectors	Meta-learning	Classifiers	Parameter Selectors	Validation Methods	Evaluation Methods
(Lessman et al. 2008)	NASA MDP	-	-	-	**22 algorithms**	-	10-Fold X Validation	ROC Curve (AUC)

Song Framework (Song et al. 2011)



Framework	Dataset	Data Preprocessor	Feature Selectors	Meta-learning	Classifiers	Parameter Selectors	Validation Methods	Evaluation Methods
(Song et al. 2011)	NASA MDP	**Log Filtering**	**FS, BE**	-	**3 algorithms (DT, 1R, NB)**	-	10-Fold X Validation	ROC Curve (AUC)

The Illustrated Guide to a Ph.D (Might, 2010)



5. Penentuan Metode Yang Diusulkan

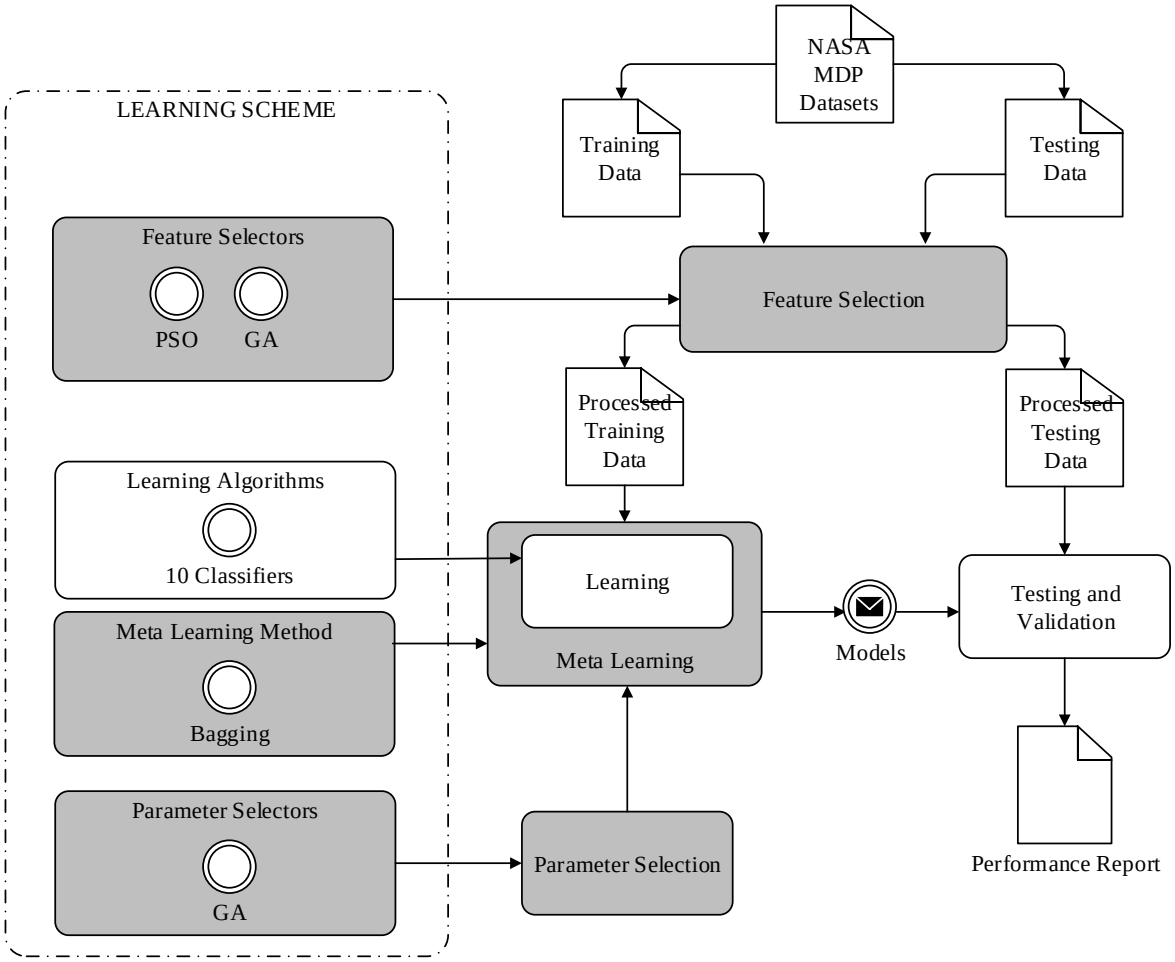
- Kita harus membangun dan mengusulkan suatu metode (*proposed method*), yg lebih baik bila dibandingkan dengan metode-metode yang ada saat ini
- Keunggulan metode yang kita usulkan harus dilandasi (*reference*), dibuktikan secara matematis dan empiris lewat hasil eksperimen dan perbandingan dengan metode yang ada
- Metode yang kita usulkan itu bisa saja dari *state-of-the-art methods*, kita kemudian "menambahkan" sesuatu (algoritma, koefisien, formula, dsb), yang akhirnya ketika kita bandingkan dengan metode original, metode kita lebih baik (lebih cepat, lebih akurat, lebih konsisten, dsb).
- "Penambahan" yang kita lakukan dan akhirnya membuat pemecahan masalah menjadi lebih baik itulah yang disebut dengan kontribusi ke pengetahuan (*contribution to knowledge*) (Dawson, 2009)

Research Gaps

- Dari hasil perangkuman metode yang ada (literature review), kita harus menemukan Research Gaps yang akan menjadi Kandidat Masalah Penelitian

- Contoh dari literature review yang saya lakukan, saya menemukan dua research gap:
  - Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models
  - Neural network and support vector machine have strong fault tolerance and strong ability of nonlinear dynamic processing of software fault data, but practicability of neural network and support vector machine are limited due to difficulty of selecting appropriate parameters

Proposed Framework



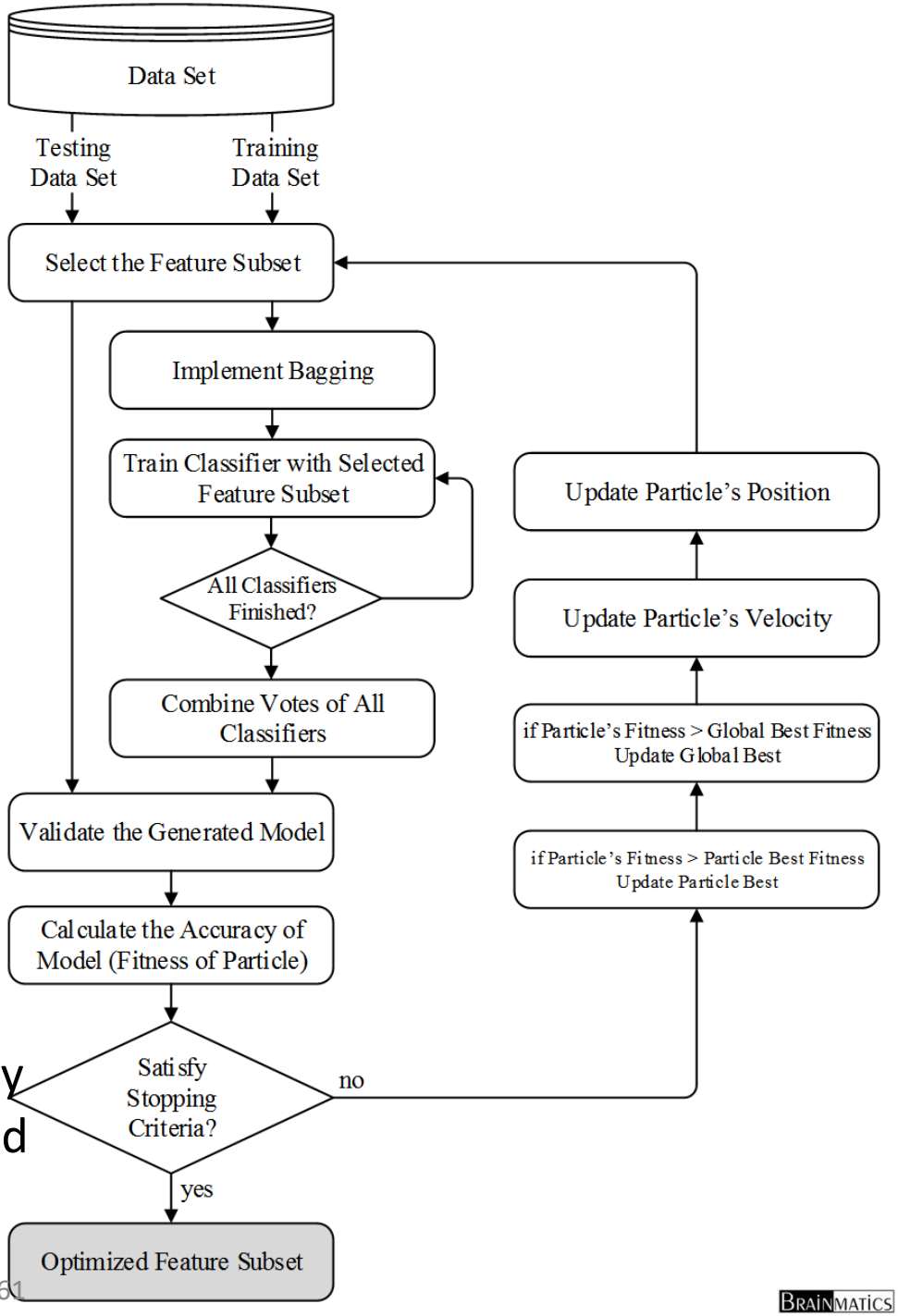
Framework	Dataset	Data Preprocessor	Feature Selectors	Meta-Learning	Classifiers	Parameter Selectors	Validation Methods	Evaluation Methods
(Menzies et al. 2007)	NASA MDP	Log Filtering	Info Gain		3 algorithm (DT, 1R, NB)	-	10-Fold X Validation	ROC Curve (AUC)
(Lessman et al. 2008)	NASA MDP	-	-		22 algorithm	-	10-Fold X Validation	ROC Curve (AUC)
(Song et al. 2011)	NASA MDP	Log Filtering	FS, BE		3 algorithm (DT, 1R, NB)	-	10-Fold X Validation	ROC Curve (AUC)
Proposed Framework	NASA MDP	-	**PSO, GA**	**Bagging**	**10 algorithms**	**GA**	10-Fold X Validation	ROC Curve (AUC)

Susun RP-RQ-RO

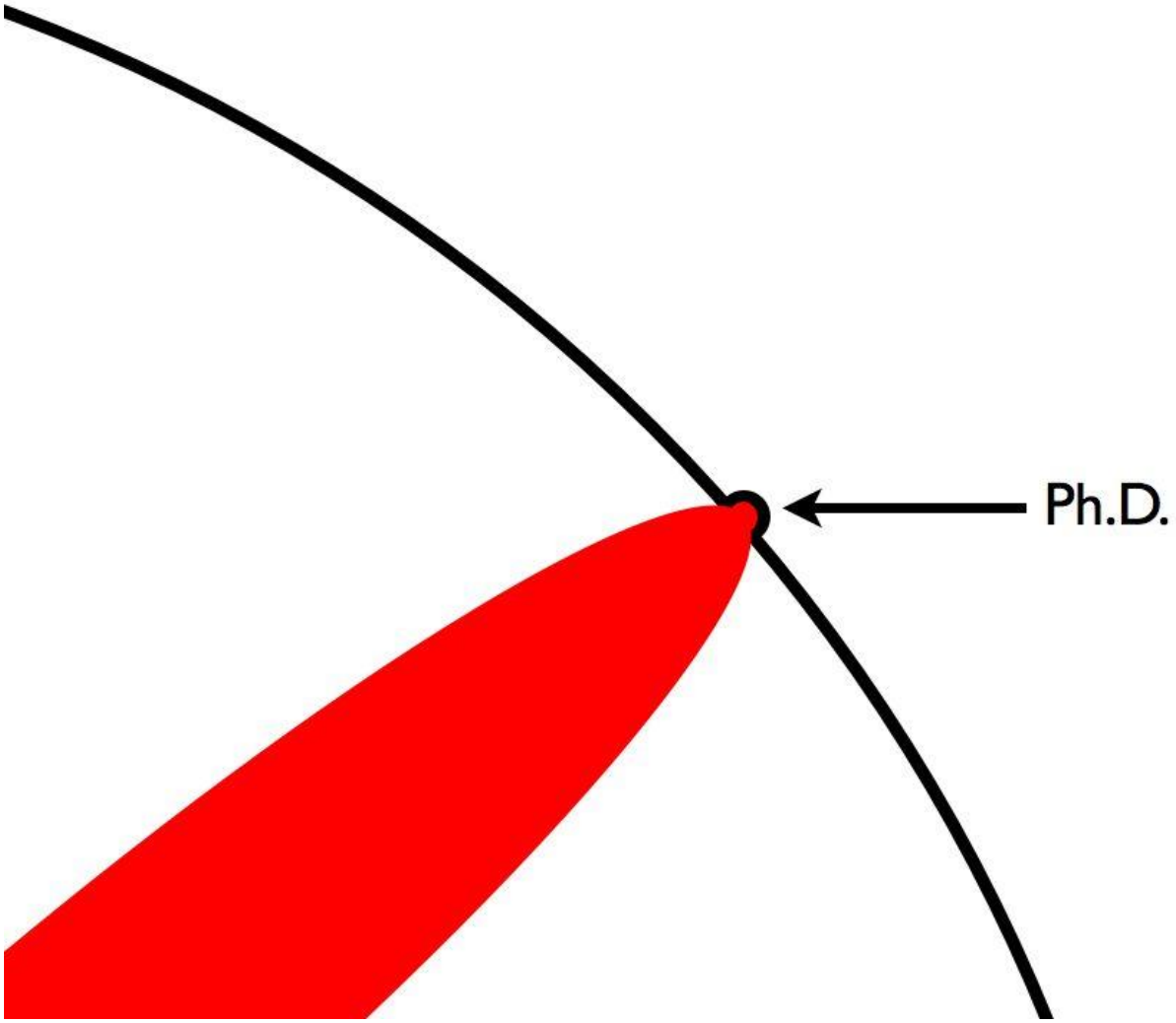
Research Problem (RP)	Research Question (RQ)	Research Objective (RO)
RP1. Data set pada prediksi cacat software berdimensi tinggi, memiliki atribut yang bersifat noisy, dan classnya bersifat tidak seimbang, menyebabkan penurunan akurasi pada prediksi cacat software	RQ1. Algoritma pemilihan fitur apa yang performanya terbaik untuk meyelesaikan masalah atribut yang noisy pada prediksi cacat software?	RO1. Mengidentifikasi algoritma pemilihan fitur apa yang memiliki performa terbaik apabila digunakan untuk menyelesaikan masalah atribut yang noisy pada prediksi cacat software
	RQ2. Algoritma meta learning apa yang performanya terbaik untuk menyelesaikan masalah class imbalance pada prediksi cacat software?	RO2. Mengidentifikasi algoritma meta learning apa yang memiliki performa terbaik apabila digunakan untuk menyelesaikan masalah class imbalance pada prediksi cacat software
	RQ3. Bagaimana pengaruh penggabungan algoritma pemilihan fitur dan metode meta learning terbaik pada peningkatan akurasi prediksi cacat software?	RO3. Mengembangkan algoritma baru yang menggabungkan algoritma pemilihan fitur dan meta learning terbaik untuk meningkatkan akurasi pada prediksi cacat software

A Hybrid Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction (PSOFS+B)

- Each particle represents a feature subset, which is a candidate solution
- Implement bagging technique and train the classifier on the larger training set based on the selected feature subset and the type of kernel
- If all classifiers are finished, combine votes of all classifiers
- Finally, measure validation accuracy on testing dataset via the generated model



The Illustrated Guide to a Ph.D (Might, 2010)



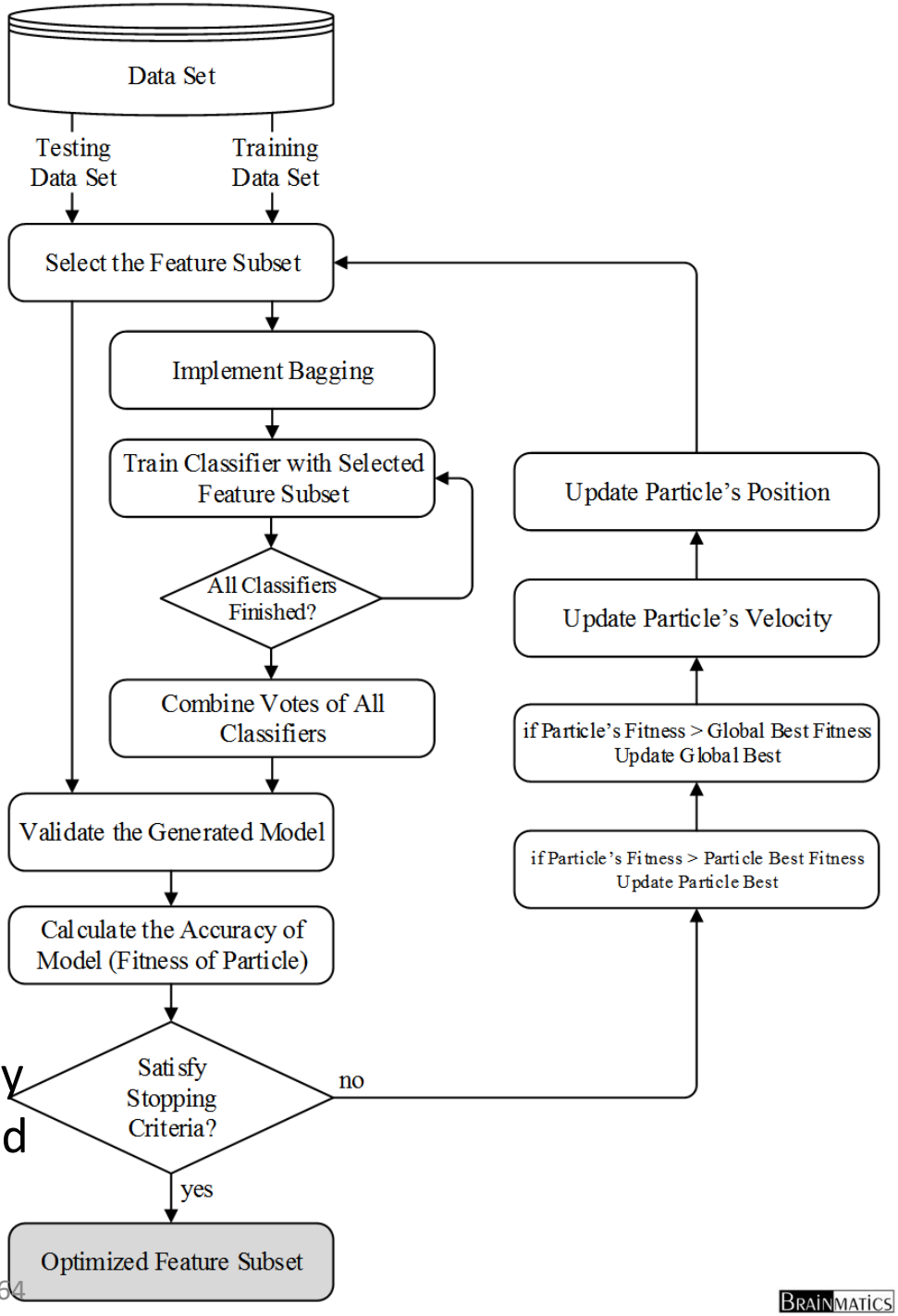
6. Evaluasi Metode Yang Diusulkan

- Metode yang diusulkan harus divalidasi dan dievaluasi dengan metode pengukuran standard dan disepakati para peneliti di bidang penelitian yang kita lakukan

- Pengukuran metode disesuaikan dengan desain masalah dan tujuan penelitian:
  - Rendahnya akurasi → meningkatnya akurasi
  - Rendahnya efisiensi → berkurangnya waktu
  - Rendahnya produktifitas → kenaikan produktifitas
  - Banyaknya noisy → berkurangnya noisy
  - Rendahnya pemahaman → meningkatnya pemahaman

A Hybrid Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction (PSOFS+B)

- Each particle represents a feature subset, which is a candidate solution
- Implement bagging technique and train the classifier on the larger training set based on the selected feature subset and the type of kernel
- If all classifiers are finished, combine votes of all classifiers
- Finally, measure validation accuracy on testing dataset via the generated model



Results: With PSOFS+B

Classifiers		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Statistical Classifier	LR	0.738	0.798	0.695	**0.78**	**0.751**	0.848	0.827	**0.816**	**0.897**
	LDA	0.469	**0.627**	**0.653**	**0.686**	**0.632**	**0.665**	0.571	**0.604**	**0.715**
	NB	**0.756**	**0.847**	**0.71**	0.732	**0.748**	**0.79**	**0.818**	**0.78**	**0.85**
Nearest Neighbor	k-NN	**0.632**	**0.675**	**0.578**	**0.606**	**0.648**	**0.547**	**0.594**	**0.679**	**0.738**
	K*	**0.681**	**0.792**	**0.66**	**0.725**	0.572	**0.822**	0.814	**0.809**	**0.878**
Neural Network	BP	0.7	**0.799**	**0.726**	**0.734**	**0.722**	**0.809**	0.89	**0.823**	**0.915**
Support Vector Machine	SVM	0.721	0.723	**0.67**	0.756	0.667	**0.792**	0.294	0.735	**0.903**
Decision Tree	C4.5	**0.682**	**0.606**	**0.592**	**0.648**	**0.615**	**0.732**	**0.732**	**0.78**	**0.769**
	CART	**0.611**	**0.679**	**0.787**	**0.679**	**0.682**	**0.831**	**0.794**	**0.845**	**0.912**
	RF	**0.62**	**0.604**	**0.557**	**0.533**	0.714	**0.686**	**0.899**	**0.759**	**0.558**

- Almost all classifiers that implemented PSOFS+B outperform the original method
- Proposed PSOFS+B method affected significantly on the performance of the class imbalance suffered classifiers

Without PSOFS+B vs With PSOFS+B

Classifiers		*P* value of t-Test	Result
Statistical Classifier	LR	0.323	Not Sig. ($P > 0.05$)
	LDA	**0.003**	**Sig. ($P < 0.05$)**
	NB	**0.007**	**Sig. ($P < 0.05$)**
Nearest Neighbor	k-NN	**0.00007**	**Sig. ($P < 0.05$)**
	K*	**0.001**	**Sig. ($P < 0.05$)**
Neural Network	BP	**0.03**	**Sig. ($P < 0.05$)**
Support Vector Machine	SVM	0.09	Not Sig. ($P > 0.05$)
Decision Tree	C4.5	**0.0002**	**Sig. ($P < 0.05$)**
	CART	**0.002**	**Sig. ($P < 0.05$)**
	RF	**0.01**	**Sig. ($P < 0.05$)**

- Although there are two classifiers that have no significant difference ($P > 0.05$), the results have indicated that those of remaining eight classifiers have significant difference ($P < 0.05$)
- The proposed PSOFS+B method makes an improvement in prediction performance for most classifiers

7. Penulisan Ilmiah dan Publikasi Hasil Penelitian

- Lakukan pendataan journal-journal yang ada di bidang kita, urutkan berdasarkan rangking SJR atau JIF
- Pilih target journal untuk tempat publikasi hasil penelitian kita
- Publikasikan hasil penelitian ke journal yang sesuai dengan kualitas kontribusi penelitian yang kita lakukan

If your research does not generate papers, it might just as well not have been done (Whitesides 2004)

No	Journal Publications	SJR	Q Category
1	IEEE Transactions on Software Engineering	3.39	Q1 in Software
2	Information Sciences	2.96	Q1 in Information Systems
3	IEEE Transactions on Systems, Man, and Cybernetics	2.76	Q1 in Artificial Intelligence
4	IEEE Transactions on Knowledge and Data Engineering	2.68	Q1 in Information Systems
5	Empirical Software Engineering	2.32	Q1 in Software
6	Information and Software Technology	1.95	Q1 in Information Systems
7	Automated Software Engineering	1.78	Q1 in Software
8	IEEE Transactions on Reliability	1.43	Q1 in Software
9	Expert Systems with Applications	1.36	Q2 in Computer Science
10	Journal of Systems and Software	1.09	Q2 in Software
11	Software Quality Journal	0.83	Q2 in Software
12	IET Software	0.55	Q2 in Software
13	Advanced Science Letters	0.24	Q3 in Computer Science
14	Journal of Software	0.23	Q3 in Software
15	International Journal of Software Engineering and Its Application	0.14	Q4 in Software

3. Topik Penelitian Software Engineering

Tahapan Penelitian Computing

*https://www.site.uottawa.ca/~bochmann/dsrg/how-to-do-good-research/
*http://romisatriawahono.net/2013/01/23/tahapan-memulai-penelitian-untuk-mahasiswa-galau/

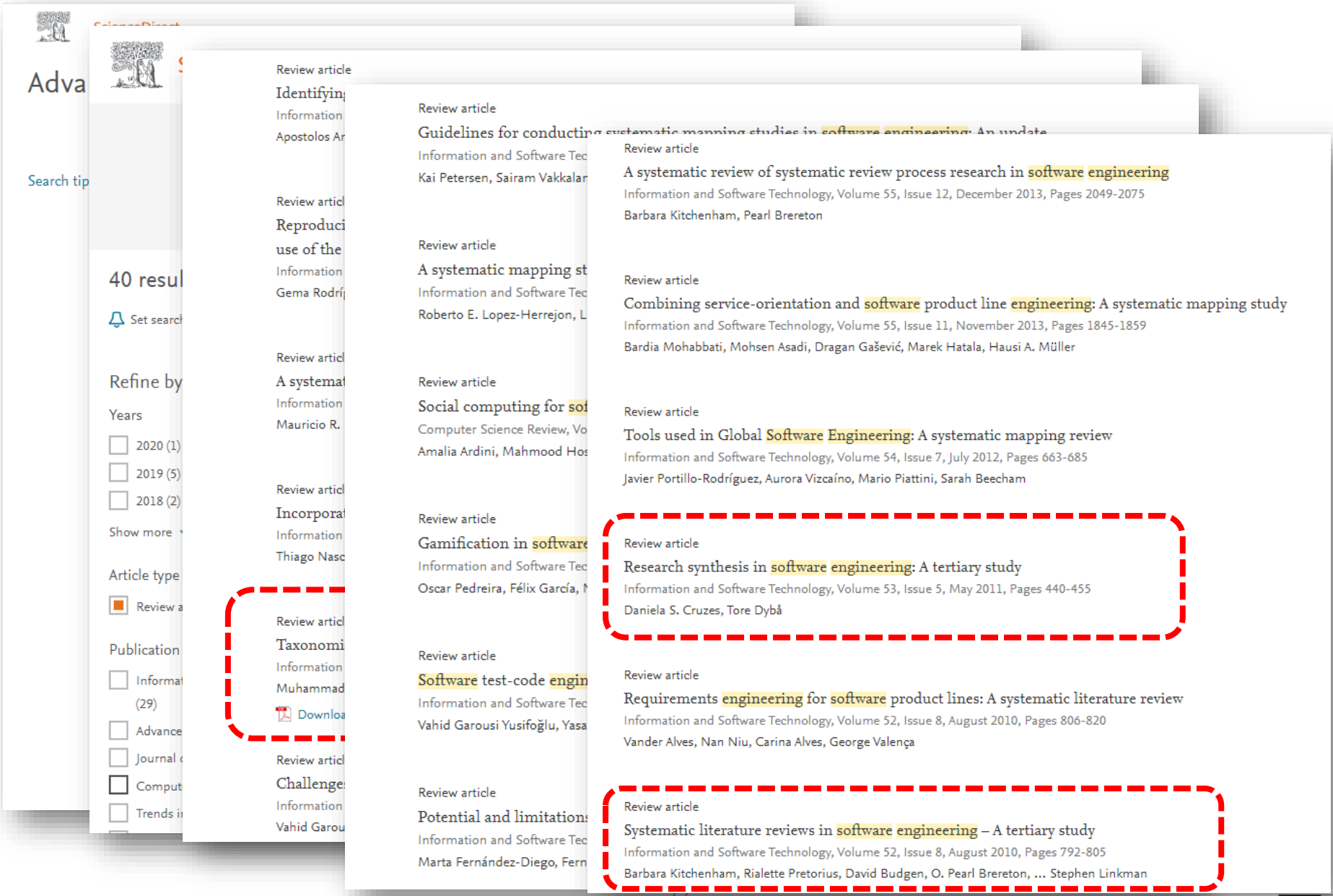
2. Penentuan Topik Penelitian

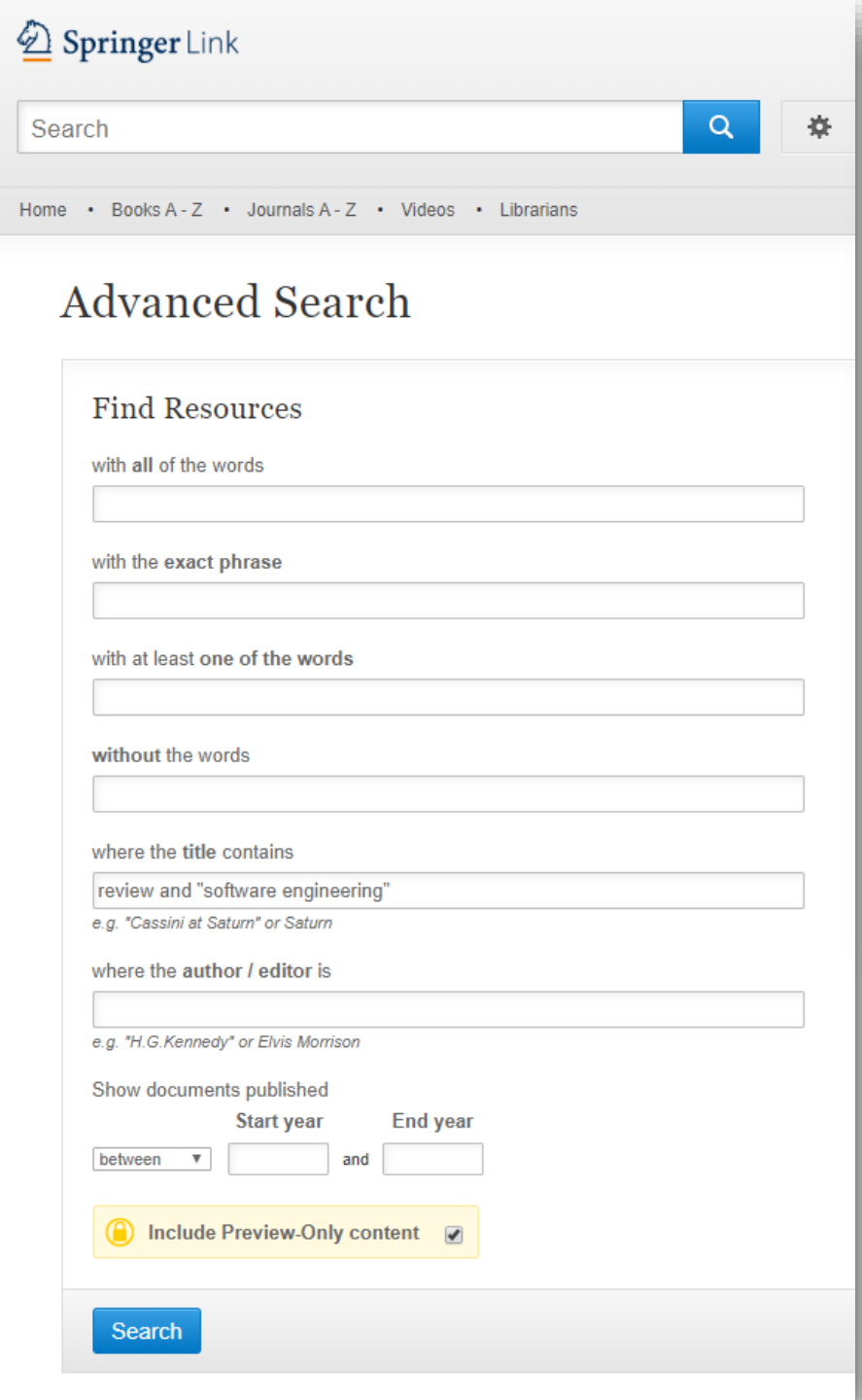
1. Searching di ScienceDirect.Com, Springerlink, IEEE Explore, Google (Scholar):
   - research trends challenge topics on NAMA BIDANG

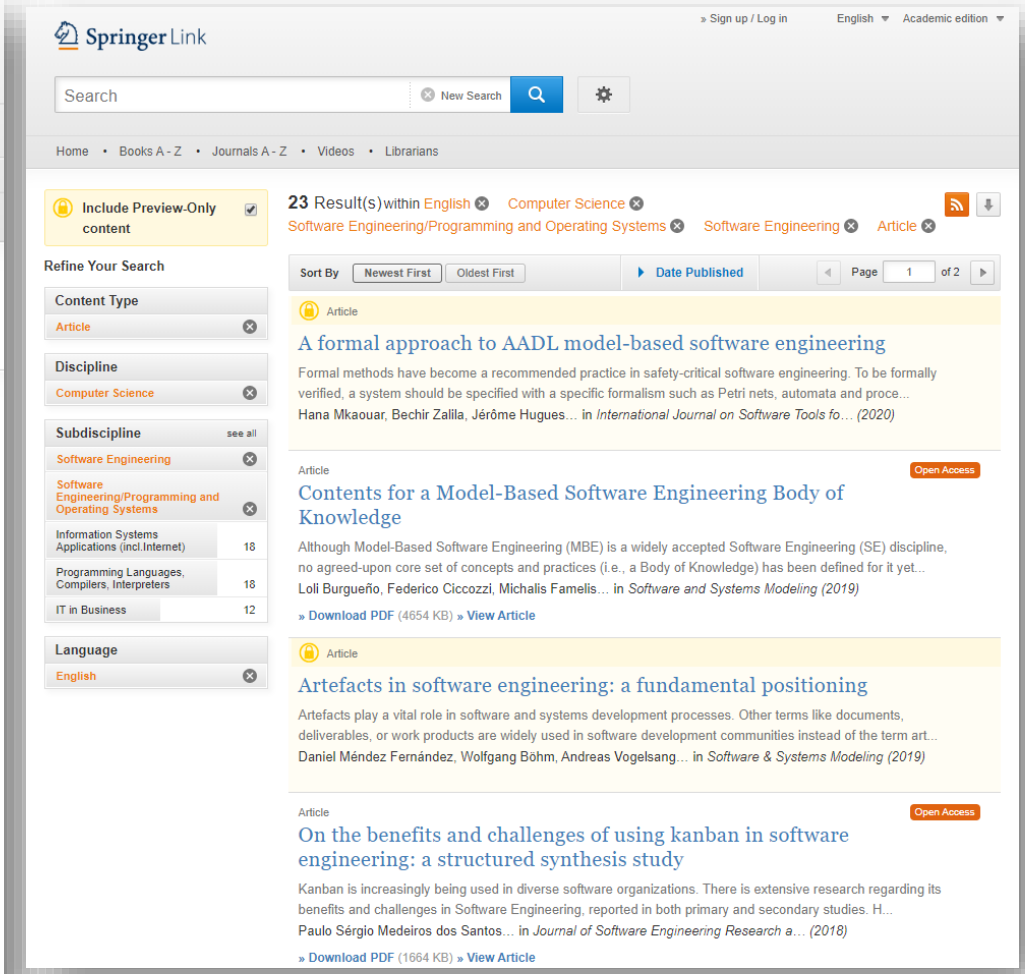
2. Untuk mempercepat pembelajaran, temukan survey paper berbentuk Tertiery Study (SLR dari SLR), karena isinya sudah merangkumkan satu bidang penelitian

3. Lanjutkan penentuan topik penelitian dengan menemukan suvey/review paper (SLR, SMS), karena survey/review paper yang masuk jurnal terindeks pasti membahas satu topik penelitian

1. Cari Tertiery Study di Bidang Software Engineering

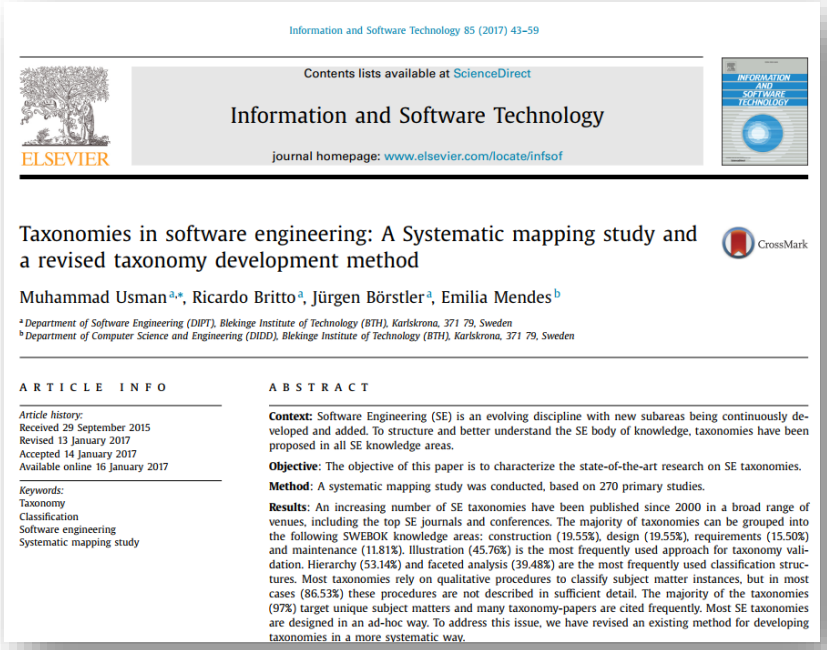






Taxonomies in Software Engineering *(Usman et al., 2017)*

- RQ1 – What taxonomy definitions and purposes are provided by publications on SE taxonomies?
- RQ2 – Which subject matters are classified in SE taxonomies?
- RQ3 – How is the utility of SE taxonomies demonstrated?
- RQ4 – How are SE taxonomies structured?
- RQ5 – To what extent are SE taxonomies used?
- RQ6 – How are SE taxonomies developed?



Information and Software Technology 85 (2017) 43–59

Contents lists available at ScienceDirect

Information and Software Technology

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

Taxonomies in software engineering: A Systematic mapping study and a revised taxonomy development method

Muhammad Usman[a,*], Ricardo Britto[a], Jürgen Börstler[a], Emilia Mendes[b]

[a] Department of Software Engineering (DIPT), Blekinge Institute of Technology (BTH), Karlskrona, 371 79, Sweden
[b] Department of Computer Science and Engineering (DIDD), Blekinge Institute of Technology (BTH), Karlskrona, 371 79, Sweden

ARTICLE INFO

Article history:
Received 29 September 2015
Revised 13 January 2017
Accepted 14 January 2017
Available online 16 January 2017

Keywords:
Taxonomy
Classification
Software engineering
Systematic mapping study

ABSTRACT

Context: Software Engineering (SE) is an evolving discipline with new subareas being continuously developed and added. To structure and better understand the SE body of knowledge, taxonomies have been proposed in all SE knowledge areas.

Objective: The objective of this paper is to characterize the state-of-the-art research on SE taxonomies.

Method: A systematic mapping study was conducted, based on 270 primary studies.

Results: An increasing number of SE taxonomies have been published since 2000 in a broad range of venues, including the top SE journals and conferences. The majority of taxonomies can be grouped into the following SWEBOK knowledge areas: construction (19.55%), design (19.55%), requirements (15.50%) and maintenance (11.81%). Illustration (45.76%) is the most frequently used approach for taxonomy validation. Hierarchy (53.14%) and faceted analysis (39.48%) are the most frequently used classification structures. Most taxonomies rely on qualitative procedures to classify subject matter instances, but in most cases (86.53%) these procedures are not described in sufficient detail. The majority of the taxonomies (97%) target unique subject matters and many taxonomy-papers are cited frequently. Most SE taxonomies are designed in an ad-hoc way. To address this issue, we have revised an existing method for developing taxonomies in a more systematic way.

Software Engineering Knowledge Areas:

1. Requirements – requirements engineering
2. Construction – software development
3. Design – software architecture
4. Management – software project management, software management
5. Process – software process, software life cycle
6. Models and methods – software model, software methods
7. Economics – software economics

SOFTWARE LIFECYCLE

	1. SOFTWARE REQUIREMENTS	2. SOFTWARE DESIGN	3. SOFTWARE CONSTRUCTION	4. SOFTWARE TESTING	5. SOFTWARE MAINTENANCE
SWEBOK Knowledge Areas					
SFIA Competencies	• Requirements definition and management • Real-time/embedded systems development • Methods and tools • Testing • Configuration management • Safety engineering	• Software design • Systems design • Real-time/embedded systems development • Safety engineering	• Programming / software development • Real-time/embedded systems development • Systems integration and build • Testing	• Testing • Systems integration and build • Real-time/embedded systems development • Quality assurance	• Release and deployment • Application support

FOUNDATIONAL ACTIVITIES

	6. SOFTWARE CONFIGURATION	10. SOFTWARE QUALITY	9. SOFTWARE ENGINEERING MODELS
SWEBOK Knowledge Areas			
SFIA Competencies	• Configuration management	• Quality management • Quality assurance • Testing • Safety assessment • Conformance review	• Requirements definition and management • Systems design • Software design

SOFTWARE ENGINEERING MANAGEMENT AND GOVERNANCE

	7. SOFTWARE ENGINEERING MANAGEMENT	8. SOFTWARE ENGINEERING PROCESS	12. SOFTWARE ENGINEERING ECONOMICS	11. SOFTWARE ENGINEERING PROFESSIONAL PRACTICE
SWEBOK Knowledge Areas				
SFIA Competencies	• Systems development management • Project management	• Systems development management • Quality management • Measurement • Methods and tools • Organisational capability development	• Systems development management	• SFIA levels of responsibility

Publications and Research Areas

Table 5
Publication venues with more than two taxonomy papers.

Publication venue	F
IEEE Intl. Conference on Software Maintenance (ICSM)	10
Intl. Conference on Requirements Engineering (RE)	6
Intl. Conference on Software Engineering (ICSE)	5
Hawaii Intl. Conference on Systems Sciences (HICSS)	4
Asia Pacific Software Engineering Conference (APSEC)	4
European Conference on Software Maintenance and Reengineering (CSMR)	4
Intl. Conference on Software Engineering and Knowledge Engineering (SEKE)	4
Intl. Symposium on Empirical Software Engineering and Measurement (ESEM)	4
Americas Conference on Information Systems (AMCIS)	3
Other Conferences	101
Conference papers total	**145**
IEEE Transactions on Software Engineering (TSE)	11
Information and Software Technology (IST)	9
ACM Computing Surveys (CSUR)	7
Journal of System and Software (JSS)	6
Journal of Software: Evolution and Process	5
IEEE Computer	5
Empirical Software Engineering (ESE)	4
IEEE Software	3
Communications of the ACM	3
Requirements Engineering	3
Other Journals	35
Journal papers total	**91**
Workshop papers total	**34**

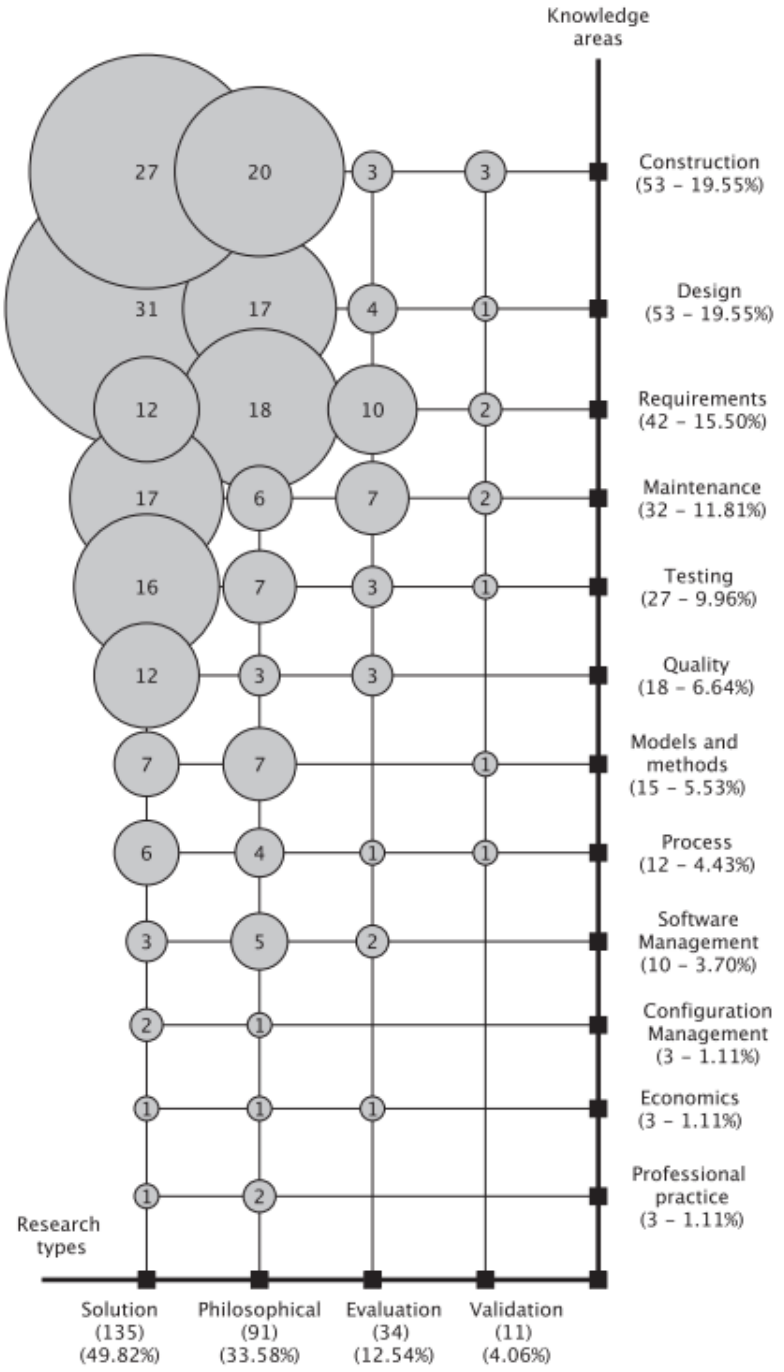


Fig. 8. Systematic map – knowledge area vs research type.

SLR in Software Engineering *(da Silva, 2011)*

3.1. Research questions

The five research questions (RQ) investigated in the SE were equivalent to the research questions used in the FE [19]. We performed minor adjustments and added subquestions as follows.

RQ1: How many SLRs were published between 1st January 2004 and 31st December 2009?
RQ1.1: How many SLRs were published between 1st January 2004 and 30th June 2008?
RQ1.2: How many SLRs were published between 1st July 2008 and 31st December 2009?

The subquestions of RQ1 investigate the development of SLRs in two separate periods. To answer RQ1.1, we used the results of OS/FE [18,19], whereas for RQ1.2, we performed the processes of search, selection, quality assessment, and data extraction defined in Sections 3.3–3.5. Similarly, we addressed the next questions considering the two time periods as we explicitly did for RQ1, searching for new evidence, combining with the results of the previous studies, and integrating all findings.

RQ2: What research topics are being addressed?
RQ3: Which individuals and organisations are most active in SLR-based research?

Information and Software Technology 53 (2011) 899–913

Contents lists available at ScienceDirect

Information and Software Technology

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

Six years of systematic literature reviews in software engineering: An updated tertiary study

Fabio Q.B. da Silva *, André L.M. Santos, Sérgio Soares, A. César C. França, Cleviton V.F. Monteiro, Felipe Farias Maciel

Centre for Informatics, UFPE, Cidade Universitária, 50.740-560 Recife, PE, Brazil

ARTICLE INFO

Article history:
Available online 23 April 2011

Keywords:
Systematic reviews
Mapping studies
Software engineering
Tertiary studies

ABSTRACT

Context: Since the introduction of evidence-based software engineering in 2004, systematic literature review (SLR) has been increasingly used as a method for conducting secondary studies in software engineering. Two tertiary studies, published in 2009 and 2010, identified and analysed 54 SLRs published in journals and conferences in the period between 1st January 2004 and 30th June 2008.
Objective: In this article, our goal was to extend and update the two previous tertiary studies to cover the period between 1st July 2008 and 31st December 2009. We analysed the quality, coverage of software engineering topics, and potential impact of published SLRs for education and practice.
Method: We performed automatic and manual searches for SLRs published in journals and conference proceedings, analysed the relevant studies, and compared and integrated our findings with the two previous tertiary studies.
Results: We found 67 new SLRs addressing 24 software engineering topics. Among these studies, 15 were considered relevant to the undergraduate educational curriculum, and 40 appeared of possible interest to practitioners. We found that the number of SLRs in software engineering is increasing, the overall quality of the studies is improving, and the number of researchers and research organisations worldwide that are

Table 1
Manual search sources.

ACM Computer Surveys
ACM Transactions on Software Engineering Methodologies
Communications of the ACM
Empirical Software Engineering Journal
Evaluation and Assessment of Software Engineering
IEE Proceedings Software (now IET Software)
IEEE Software
IEEE Transactions on Software Engineering
Software Practice and Experience
Information and Software Technology
Int. Conference on Software Engineering
Int. Symposium on Empirical Software Engineering and Measurement
Journal of Systems and Software

SLR in Software Engineering *(da Silva, 2011)*

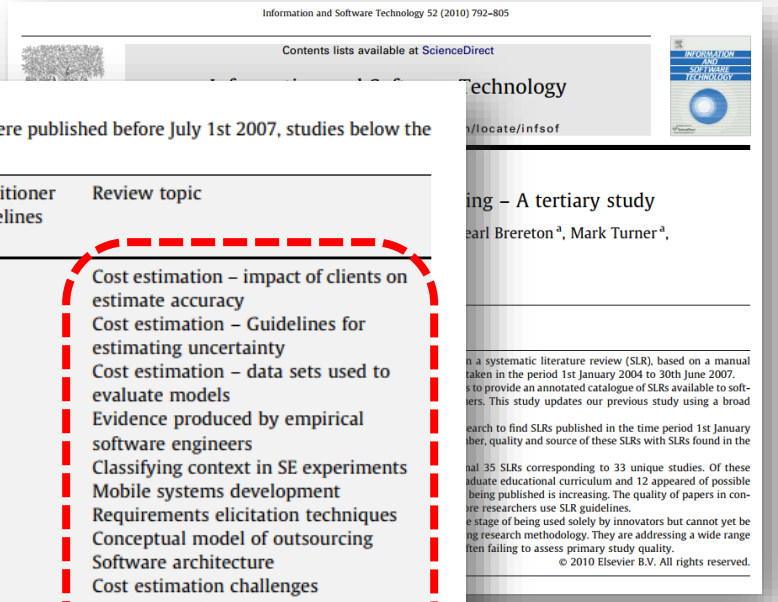
Table 7
Authors with three or more studies.

Authors	Country
Jørgensen	Norway
Guilherme Travassos	Brazil
Shepperd	UK
Tore Dyba	Norway
Muhammad Ali Babar	Ireland
Hannay	Norway
Sarah Beecham	UK
Sjøberg	Spain
Tony Gorschek	Sweden
Ambrosio Toval	Spain
Helen Sharp	UK
Hugh Robinson	UK
Juristo	Spain
Kampenes	Norway
Kitchenham	UK
Maya Daneva	The Netherlands
Moløkken-Østvold	Norway
Moreno	Spain
Nathan Baddoo	UK
Thelin	Sweden
Tracy Hall	UK

Table 2
Systematic literature reviews in software engineering between July 2008 and December 2009.

Study Ref. (N = 67)	Year	Quality score	Review type	Review focus	Review topic	Cited EBSE paper	Cited guidelines	Number primary studies	Practitioners guidelines	Paper type
[SE01]	2008	4	MS	SERT	Human Aspects	N	Y[d]	92	N	J
[SE02]	2008	4	SLR	RT	Knowledge Management	Y[a,b]	Y[d]	68	Y	J
[SE03]	2008	1.5	MS	RT	Research Topics in Software Engineering	N	N	691	N	J
[SE04]	2008	1	MS	SERT	Software Project Management	N	N	48	N	C
[SE05]	2008	4	MS	SERT	Agile Software Development	N	Y[f]	36	Y	J
[SE08]	2008	2	MS	SERT	Software Testing	N	Y[h]	14	Y	C
[SE09]	2008	2	MS	SERT	Requirements Engineering	N	Y[e]	240	N	C
[SE10]	2008	1	MS	SERT	Usability	N	Y[f]	51	Y	C
[SE11]	2008	2.5	MS	SERT	Software Process Improvement	N	Y[f]	50	Y	C
[SE12]	2008	1.5	MS	SERT	UML	N	Y[f]	33	N	C[i]
[SE13]	2008	1	SLR	RT	Distributed Software Development	N	N	12	N	C
[SE14]	2008	3	SLR	RQ	Usability	N	Y[e,h]	63	Y	J
[SE18]	2009	4	MS	SERT	Software Testing	N	Y[f]	35	N	J
[SE19]	2009	3.5	SLR	RT	Software Testing	Y[a,b]	Y[f,h]	64	N	J
[SE20]	2009	2.5	MS	SERT	Software Maintenance and Evolution	N	Y[e,f]	34	N	J
[SE21]	2009	2.5	MS	SERT	Requirements Engineering	N	Y[d]	58	N	C
[SE22]	2009	2	SLR	RQ	Agile Software Development	Y[a]	Y[d,f]	9	N	C
[SE23]	2009	2	MS	RQ	Design Patterns	Y[a]	Y[f]	4	N	C
[SE24]	2009	2	MS	SERT	Software Maintenance and Evolution	N	Y[d]	12	Y	C
[SE25]	2009	2	MS	SERT	Risk Management	N	Y[e,g]	80	N	J
[SE26]	2009	2	MS	SERT	Software Fault Prediction	N	N	74	N	J
[SE27]	2009	3	MS	SERT	Software Product Line	N	Y[f]	34	N	C
[SE28]	2009	3	MS	SERT	Software Product Line	Y[a]	Y[f]	97	N	C
[SE29]	2009	1.5	MS	SERT	Requirements Engineering	N	N	46	N	C[i]
[SE30]	2009	3	MS	SERT	Software Maintenance and Evolution	N	Y[d]	176	N	J
[SE32]	2009	1.5	SLR	RT	Empirical Research Methods	Y[a]	Y[d]	16	N	C[i]
[SE33]	2009	1.5	SLR	RQ	Software Security	N	Y[f]	64	N	C
[SE34]	2009	2.5	MS	SERT	Empirical Research Methods	N	N	8	N	J
[SE35]	2008	3	SLR	RQ	Software Testing	N	Y[f,h]	28	N	C
[SE36]	2009	3.5	MS	SERT	Human Aspects	Y[a]	Y[d]	92	N	J
[SE37]	2009	3	MA	RQ	Agile Software Development	Y[a]	Y[f]	18	Y	J
[SE38]	2009	3	MS	SERT	Context Aware Systems	N	N	237	N	J
[SE39]	2009	3.5	SLR	RQ	Software Maintenance and Evolution	N	Y[d]	18	Y	C
[SE40]	2009	4	MS	SERT	Distributed Software Development	N	Y[f]	20	Y	C
[SE42]	2009	2.5	SLR	RT	Requirements Engineering	N	Y[f]	97	Y	J
[SE43]	2009	2.5	MS	SERT	Distributed Software Development	N	Y[f]	78	Y	J
[SE44]	2009	2	MS	SERT	Distributed Software Development	N	Y[f]	98	Y	C
[SE45]	2009	2	MS	SERT	Distributed Software Development	N	Y[d]	122	Y	C
[SE46]	2009	4	MS	SERT	Software Product Line	N	Y[e]	89	N	J
[SE47]	2009	3	MS	SERT	Software Product Line	N	Y[d]	23	N	C

SLR in Software Engineering *(Kitchenham, 2010)*


Information and Software Technology 52 (2010) 792–805

Contents lists available at ScienceDirect

Technology

/locate/infsof

ing – A tertiary study

earl Brereton [a], Mark Turner [a],

a systematic literature review (SLR), based on a manual
taken in the period 1st January 2004 to 30th June 2007.
s to provide an annotated catalogue of SLRs available to soft-
ers. This study updates our previous study using a broad

earch to find SLRs published in the time period 1st January
ber, quality and source of these SLRs with SLRs found in the

al 35 SLRs corresponding to 33 unique studies. Of these
aduate educational curriculum and 12 appeared of possible
being published is increasing. The quality of papers in con-
re researchers use SLR guidelines.
e stage of being used solely by innovators but cannot yet be
g research methodology. They are addressing a wide range
ften failing to assess primary study quality.

Table 1
Additional software engineering SLRs published from 1st January 2004 to 30th June 2008 (studies above the double lines were published before July 1st 2007, studies below the double lines were published after June 30th 2007).

Study ref.	Review focus	Quality total score	Year	Cited EBSE paper	Cited guidelines	Paper type	Number primary studies	Practitioner guidelines	Review topic
[32]	RQ	2.5	2005	No	Yes	Conference	8	N	Cost estimation – impact of clients on estimate accuracy
[33]	RQ	2	2005	No	No	Journal	70	Y	Cost estimation – Guidelines for estimating uncertainty
[36]	RT	2.5	2005	No	Yes[a]	Workshop	50	N	Cost estimation – data sets used to evaluate models
[39]	RT	1.5	2005	No	No	Workshop	119	N	Evidence produced by empirical software engineers
[40]	RT	2	2005	No	Yes	Conference	13	N	Classifying context in SE experiments
[41]	SERT	2.5	2005	No	No	Conference	105	N	Mobile systems development
[34]	RQ	3.5	2006	Yes	Yes	Conference	26	Y	Requirements elicitation techniques
[37]	SERT	1.5	2006	No	No	Workshop	57	N	Conceptual model of outsourcing
[42]	SERT	1	2006	No	No	Technical report	750	N	Software architecture
[35]	SERT	1.5	2007	Yes	No	Workshop	653	N	Cost estimation challenges
[38]	SERT	1.5	2007	No	No	Journal	80	N	Approaches for mining software repositories in the context of evolution
[43]	SERT	1.5	2007	No	No	Book chapter (working conference)	4089	N	Requirements Engineering publications
[44]	RT	1.5	2007	No	No	Book chapter (workshop)	133	N	Evidence produced by empirical software engineers
[45]	SERT	2	2007	No	No	Book chapter (working conference)	155	N	Developing open source software – analysis of research
[11]	RT	2.5	2007	No	Yes	Journal	103	N	Empirical software engineering – effect size
[16]	SERT	2.5	2007	No	Yes	Conference	138	No	Software design – Object-oriented
[17]	RQ	4	2007	No	Yes	Conference	10	No	Cost estimation-local vs. global estimation models
[19]	RQ	3.5	2007	No	Yes	Book chapter (conference)	5	N	Software development process – tailoring and introduction of rational unified process
[20]	RQ	2.5	2007	No	Yes	Journal	11	N	Reuse – economic benefits
[21]	SERT	1	2007	No	No	Journal	137	N	Tool integration – a research agenda
[24]	SERT	3.5	2007	No	Yes	Conference	53	N	Web application development – design for accessibility
[10]	RQ	2.5	2008	No	Yes	Journal	103	N	Empirical software engineering – the value of laboratory experiments
[15]	SERT	2.5	2008	No	Yes	Conference	28	No	Re-engineering – multi-channel access
[18]	RQ	1.5	2008	No	No	Book chapter (conference)	21	N	Metrics – measurement programme

SLR in Software Engineering *(Kitchenham, 2010)*

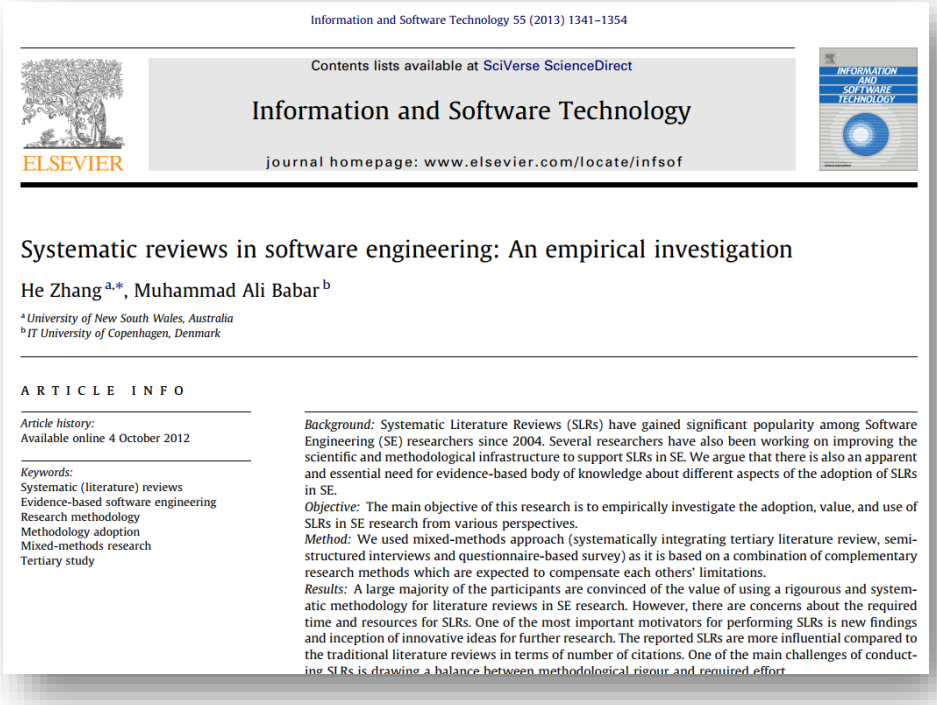
Table 5
Distribution of SLRs over the SE sections of the University curriculum guidelines.

Section	Number of sub-sections	Number of sub-topics	January 1st 2004 to June 30th 2007 manual search [12]		January 1st 2004 to June 30th 2007 additional papers (broad search)		July 1st 2007 to June 30th		Total SLRs	Total sub-topics
			SLRs	Num subs addressed	SLRs	Num subs addressed	SLRs	Num subs addressed		
Software modeling and analysis	7	41			[34]	1	[22,25]	2	3	3
Software design	7	37	[56]	0	[42]	0	[23,24]	0	4	0[a]
Software validation and verification	40	5	[61,65,66][b,c]	5			[31]	0	5	5
Software evolution	2	13								
Software process	2	14			[37]	1	[19]	1	2	2
Software quality	5	28	[57]	1				–	1	1
Software management	5	32	[58–60,62–64]	1	[32,33]	1	[17,18,48]	2	11	2
Computing essentials	4	41	[67]	1					1	1
Mathematical and engineering fundamentals	3	22					[22,48]	2	2	2
System and application specialties	42				[41]	1	[15]	1	2	1
Total		233	12	8	6	4	11[d]	8	29	17

[a] The papers addressed a general topic, not a specific technique.
[b] This paper addressed four sub-topics of the subsection testing.
[c] This paper addressed testing methods and inspection methods.
[d] Paper [48] addressed two topics.

SLR in Software Engineering (*Zhang, 2013*)

- RQ1. What is the value of SLR for SE? Why did (or did not) SE researchers do SLRs?
- RQ2. What SE topics have been addressed by what types of SLRs? What has the influence of SLRs been in SE research?
- RQ3. How did SE researchers perform SLRs (in terms of, for example, rigour and effort)?



Information and Software Technology 55 (2013) 1341–1354

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Systematic reviews in software engineering: An empirical investigation

He Zhang [a,*], Muhammad Ali Babar [b]

[a] University of New South Wales, Australia
[b] IT University of Copenhagen, Denmark

ARTICLE INFO

Article history:
Available online 4 October 2012

Keywords:
Systematic (literature) reviews
Evidence-based software engineering
Research methodology
Methodology adoption
Mixed-methods research
Tertiary study

Background: Systematic Literature Reviews (SLRs) have gained significant popularity among Software Engineering (SE) researchers since 2004. Several researchers have also been working on improving the scientific and methodological infrastructure to support SLRs in SE. We argue that there is also an apparent and essential need for evidence-based body of knowledge about different aspects of the adoption of SLRs in SE.
Objective: The main objective of this research is to empirically investigate the adoption, value, and use of SLRs in SE research from various perspectives.
Method: We used mixed-methods approach (systematically integrating tertiary literature review, semi-structured interviews and questionnaire-based survey) as it is based on a combination of complementary research methods which are expected to compensate each others' limitations.
Results: A large majority of the participants are convinced of the value of using a rigourous and systematic methodology for literature reviews in SE research. However, there are concerns about the required time and resources for SLRs. One of the most important motivators for performing SLRs is new findings and inception of innovative ideas for further research. The reported SLRs are more influential compared to the traditional literature reviews in terms of number of citations. One of the main challenges of conducting SLRs is drawing a balance between methodological rigour and required effort

SLR in Software Engineering (*Zhang, 2013*)

Table 4
Overview of SLR studies and publications in software engineering research by topic and year.

Rank	Review topics	2004	2005	2006	2007	2008	2009	2010	Sum
1	Global development					◊◊◊	◆[a]◊◊▽	◆[a] ◆[a]◊◊◊◊◊◊▽	13/16
1	Cost estimation	◆	◆◊◊▽	◆◊	◆◆◆[a]◊◊		◊▽		13/14
3	Requirement engineering			◊	◊	◊	◆◆◆◊◊[a]▽▽	◆◆◆	12/13
4	Empirical methods		◆◊▽	◆	◆◆▽	◆▽	◆▽		11/11
5	Agile development					◆	◆◊◊	◆◊◊◊▽▽	10/10
6	Inspection and testing	◆		◆	▽	◆[a]	◆◆[a]◊◊	◆◆◊	9/11
6	Software product lines					◊▽	◆◊◊◊◊◊	◆	9/9
8	Software process improvement				◆◊	◆◆◊	◊	◆◊	8/8
9	Software architecture			▽		▽	◆[a]▽	◆◊	5/6
10	Software process modelling			▽		◊	◊	◊◊[a]	4/5
10	Open source development						▽◊[a]	◆◊◊	4/5
10	Software measurement			◊		◆◊		◆	4/4
10	Program analysis						◆◆◊	◊	4/4
14	Model-based development					◊	◆[a]	◊▽	3/4
14	Tertiary study						◆	◆[a]◊▽	3/4
14	Software maintenance				◆	▽	◊		3/3
14	Software tools				◆▽		◆		3/3
14	Software security					◆◊		◆	3/3
14	Web engineering		◊		◊	▽			3/3
14	Software outsourcing						◆◊◊		3/3
14	Human-aspects (e.g., motivations)					◆	◆◆		3/3
22	Software design				◊		◊		2/2
22	Unified modelling language					▽	◆		2/2
22	Software evolution				◆			◊	2/2
22	Aspect-oriented programming						◊	◆	2/2
22	Business process			◊				◆	2/2
22	SE research in general	◆				◆			2/2
	Other topics		◊		◆	◆	◆	◆◆	6/6
Total	(*new/all SLR reports*)	3/3	9/9	9/9	18/19	25/26	44/50	40/44	**148/160**

◆ journal paper or book chapter, ◊ full conference paper, ▽ workshop or short paper.
[a] Update/extension of previous SLR report.

Research Synthesis in Software Engineering (Cruzes, 2011)

Information and Software Technology 53 (2011) 440–455

Contents lists available at ScienceDirect

oftware Technology

.elsevier.com/locate/infsof

ng: A tertiary study

ntrasting evidence from multiple studies is necessary to build knowledge and
e empirical support for a phenomenon. Therefore, research synthesis is at the
erprise in the software engineering discipline.
this article is to contribute to a better understanding of the challenges in syn-
ring research and their implications for the progress of research and practice.
journal articles and full proceedings papers from the inception of evidence-
was performed to assess the types and methods of research synthesis in sys-
e engineering.
the 49 reviews included in the study did not contain any synthesis. Of the
nthesis, two thirds performed a narrative or a thematic synthesis. Only a
nonstrated a robust, academic approach to research synthesis.
hat, despite the focus on systematic reviews, there is limited attention paid to
vare engineering. This trend needs to change and a repertoire of synthesis
tegral part of systematic reviews to increase their significance and utility for

Table 6
Main topic areas in SE systematic reviews.

Main topic area	Studies
Agile software development	S5, S12
Aspect-oriented programming	S45
Distributed software development	S15, S19, S47
Domain analysis	S22
Estimation models	S16, S21, S35, S43
Experimental methods in SE	S6, S17, S18, S38, S44
Global software engineering	S31
Knowledge management in SE	S3
Motivation in SE	S2, S28
Product lines software development	S20, S25, S33
Requirements engineering	S4, S14, S26, S29, S32, S48
Reuse	S36
Software design	S8, S23, S30, S39, S42
Computer games	S49
Software maintainability	S34
Software measurement	S9, S40
Software process	S27, S41
Technology acceptance model	S46
Testing	S1, S7
Theory use in SE	S10, S11
Web development	S13, S24, S37

Citation & Topics in Software Engineering (Garousi, 2016)

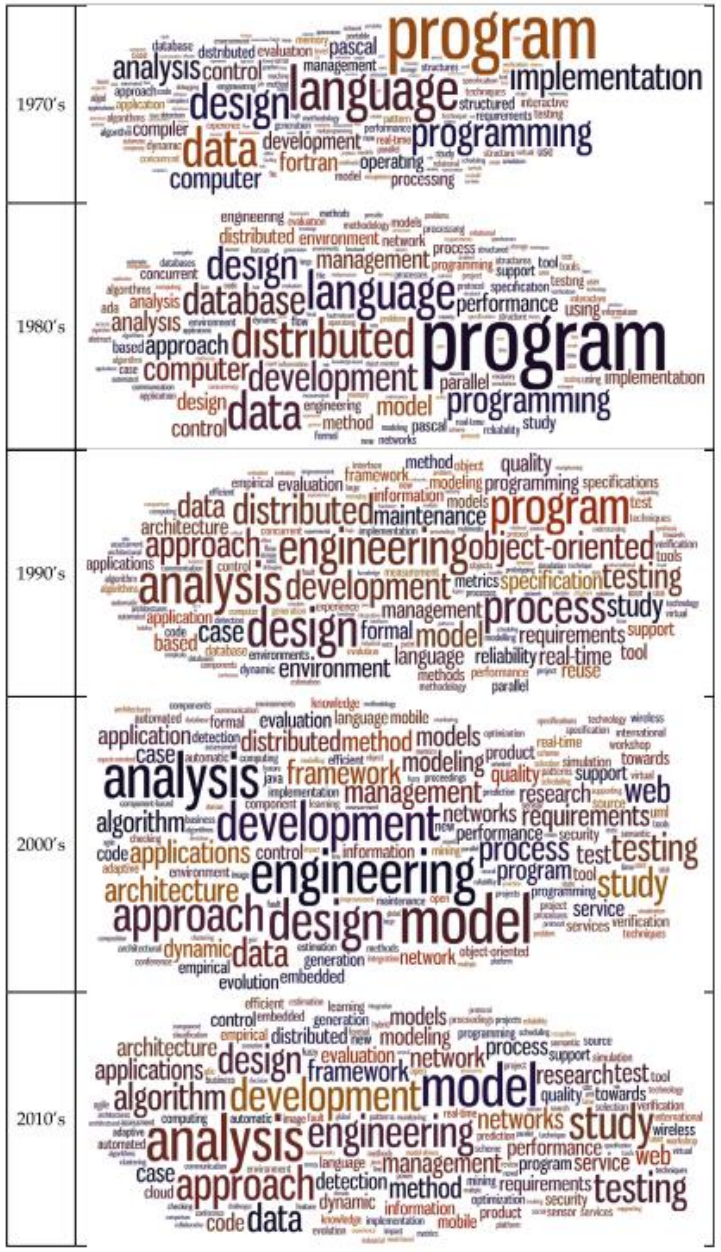


Fig. 10 – Focus areas of SE papers in each decade.

COMPUTER SCIENCE REVIEW 19 (2016) 56–77

Available online at www.sciencedirect.com

ScienceDirect

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

ELSEVIER

Computer Science Review

Survey

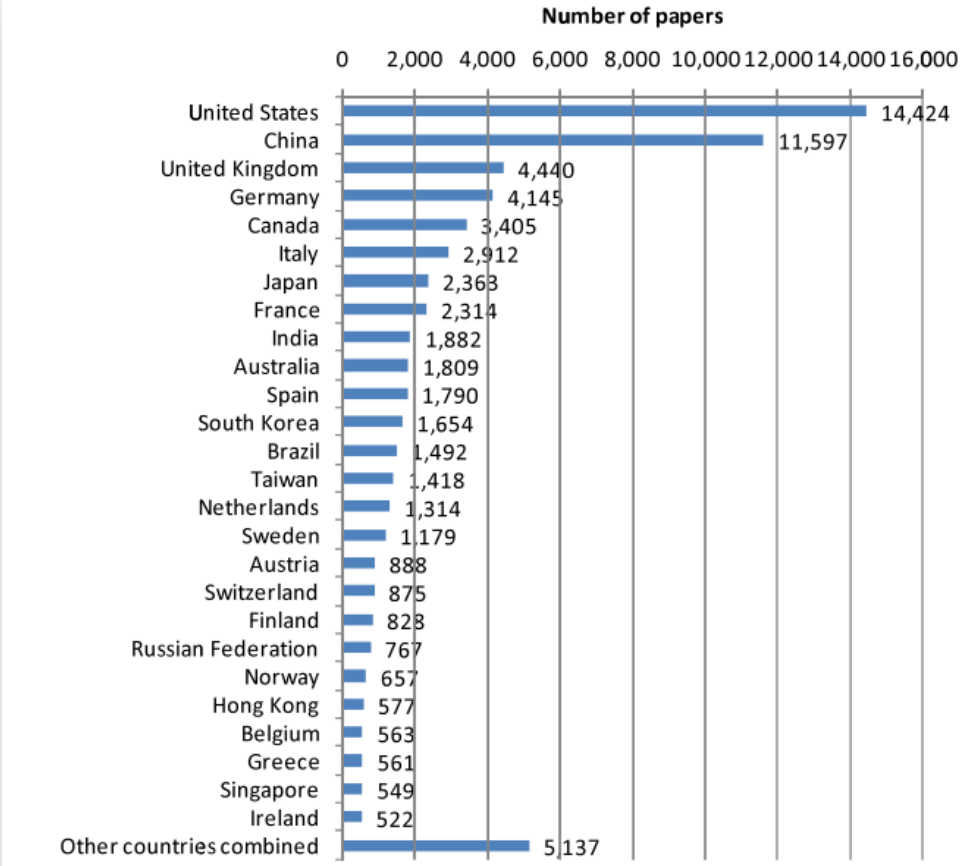


Fig. 13 – Ranking of the countries with more than 500 contributions.

10 Most Probable Terms in the Topics

Table 10 – Ten most probable terms in the topics with the highest number of papers.

Given topic name	Requirements engineering ($n = 3096$)	Database ($n = 2601$)	Software effort estimation ($n = 2601$)	Design patterns ($n = 2172$)	Incoherent topic ($n = 2155$)
1	"requirements"	"database"	"software"	"design"	"using"
2	"engineering"	"data"	"estimation"	"patterns"	"analysis"
3	"elicitation"	"processing"	"effort"	"implementation"	"time"
4	"nonfunctional"	"databases"	"cost"	"architectural"	"model"
5	"role"	"xml"	"measurement"	"pattern"	"real"
6	"tracing"	"relational"	"models"	"matching"	"behavior"
7	"goals"	"query"	"functional"	"style"	"functions"
8	"managing"	"objects"	"function"	"decisions"	"world"
9	"legal"	"efficient"	"size"	"principles"	"structure"
10	"goal-oriented"	"spatial"	"point"	"rationale"	"paper"

Table 11 – Ten most probable terms in the topics with the highest amount of citations per paper per year.

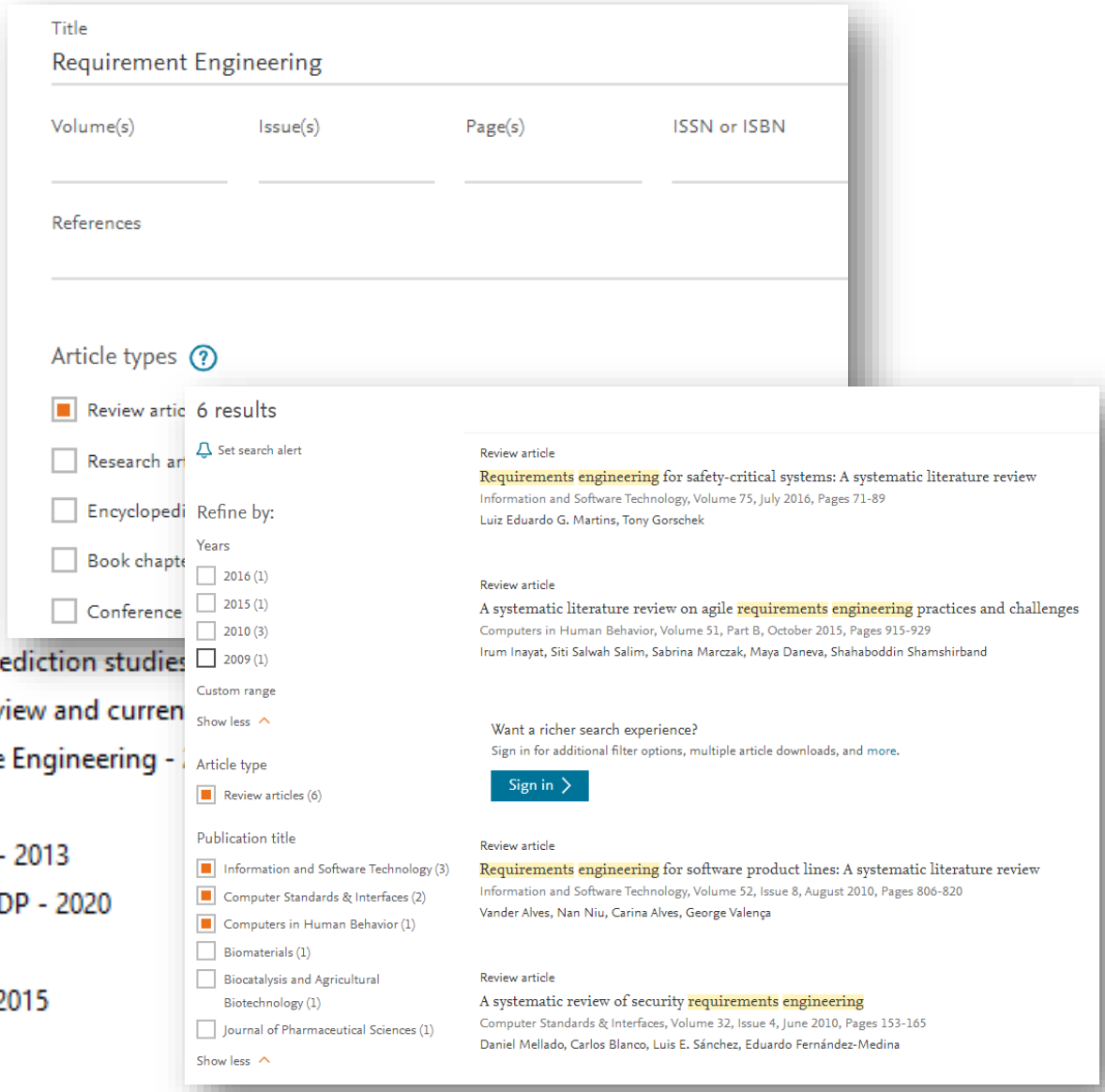
Given topic name	Model checking ($n = 686$)	Test generation (=923)	Source code ($n = 988$)	Automated testing ($n = 785$)	Systematic reviews ($n = 653$)
1	"model"	"test"	"code"	"testing"	"software"
2	"checking"	"generation"	"source"	"automated"	"systematic"
3	"transformation"	"automatic"	"open"	"model based"	"review"
4	"driven"	"coverage"	"projects"	"regression"	"challenges"
5	"transformations"	"automated"	"changes"	"mutation"	"survey"
6	"probabilistic"	"selection"	"usage"	"random"	"future"
7	"Markov"	"generating"	"documentation"	"strategies"	"issues"
8	"bounded"	"cases"	"API"	"GUI"	"mapping"
9	"graph"	"suite"	"detecting"	"conformance"	"results"
10	"properties"	"tests"	"clones"	"techniques"	"approaches"

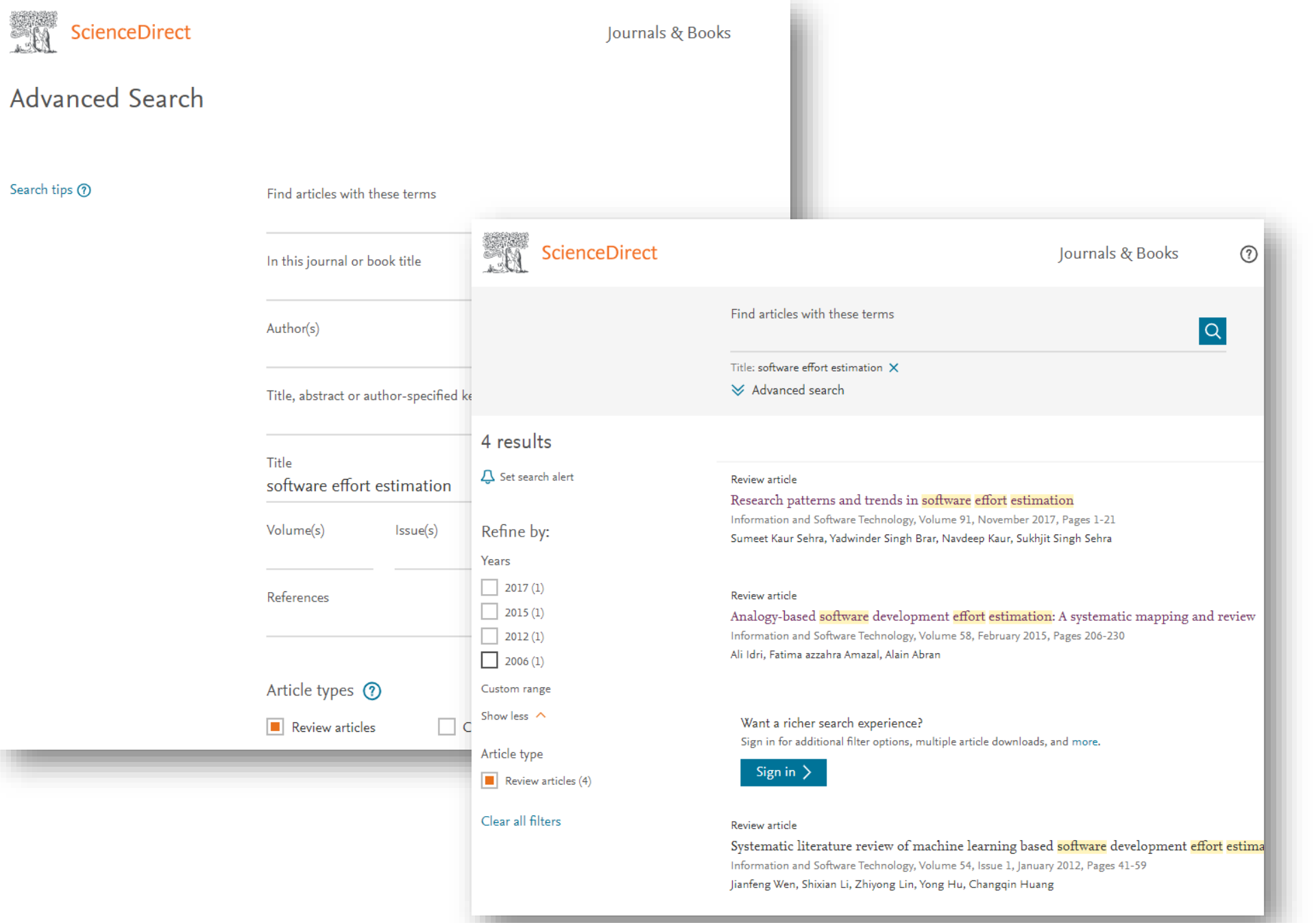
2. Cari SLR dari Topik Penelitian yang Dipilih

- Setelah kita paham beberapa topik penelitian di bidang software engineering dari *Tertiery Study (SLR dari SLR)*
- Langkah berikutnya, kita kumpulkan seluruh SLR dengan keyword topik seperti di paper *Tertiery Study (SLR dari SLR)*
- Lanjutkan dengan mengejar seluruh SLR dari topik yang kita akan angkat pada penelitian kita

Keyword harus masuk Title Paper, Pilih Review, dan Journal di Bidang Computing

- Global Software Engineering
- Requirement Engineering
- Self Adaptive Systems
- Service Oriented Architecture
- Software Architecture
- Software Construction
- Software Cost Effort Estimation
- Software Defect Prediction
  - Arisholm - fault prediction models - 2010
  - Catal - A systematic review of software fault prediction studies
  - Catal - Software fault prediction A literature review and curren
  - Hall - Fault Prediction Performance in Software Engineering -
  - mahmood - reproducibbility of SDP - 2018
  - Radjenovic - Software fault prediction metrics - 2013
  - Shepperd - SLR of Unsupervised Learning for SDP - 2020
  - Strate - Software Defect Reporting - 2013
  - Wahono - SLR of Software Defect Prediction - 2015





Are You Sure This is a Topic?

A Systematic Literature Review on Fault Prediction Performanc in Software Engineering

Tracy Hall, Sarah Beecham, David Bowes, David Gray, and Stev

Abstract—*Background:* The accurate prediction of where faults are likely to occur in code can help direct te improve the quality of software. *Objective:* We investigate how the context of models, the independent va modeling techniques applied influence the performance of fault prediction models. *Method:* We used a sys identify 208 fault prediction studies published from January 2000 to December 2010. We synthesize the results of 36 studies which report sufficient contextual and methodological information according to the crit *Results:* The models that perform well tend to be based on simple modeling techniques such as Naive Ba Combinations of independent variables have been used by models that perform well. Feature selection h combinations when models are performing particularly well. *Conclusion:* The methodology used to build mo to predictive performance. Although there are a set of fault prediction studies in which confidence is possibl that use a reliable methodology and which report their context, methodology, and performance comprehe

Index Terms—Systematic literature review, software fault prediction

1 Introduction

This Systematic Literature Review (SLR) aims to identify and analyze the models used to predict faults in source code in 208 studies published between January 2000 and December 2010. Our analysis investigates how model performance is affected by the context in which the model was developed, the independent variables used in the model, and the technique on which the model was built. Our results enable researchers to develop prediction models based on best knowledge and practice across many previous studies. Our results also help practitioners to make effective decisions on prediction models most suited to their context.

Fault[1] prediction modeling is an important area of research and the subject of many previous studies. These studies typically produce fault prediction models which allow software engineers to focus development activities on fault-prone code, thereby improving software quality and

making better use of resources models published are complex date comprehensive picture o prediction exists. Two previou been performed in [1] and [2] these reviews in the following

- *Timeframes.* Our review because it includes stu 2010. Fenton and Neil c software fault predictio Catal and Diri's [2] rev between 1990 and 2007.
- *Systematic approach.* W Charters [3] original and rigorous procedures for conducting systematic reviews. Catal and Diri did not report on how they sourced their studies, stating that they adapted Jørgensen and Shepperd's [4]

1. Introduction

Global Software Engineering (GSE) has become a growing area of research, apart from being an expanding trend in the Information Technology (IT) industry [3]. GSE requires software tools (management tools, development tools, etc.) to support the special characteristics that this environment has, and which have principally come about as a result of the distance factor (temporal, geographic and socio-cultural distance) [4].

Modern software development, such as globally dispersed teams, creates specific challenges and risks (in spite of the benefits that can be obtained) for the software industry, which need to be considered [5]. In fact, developing software systems through collaboration with other partners and in different geographical locations is a great challenge for organizations [6,7].

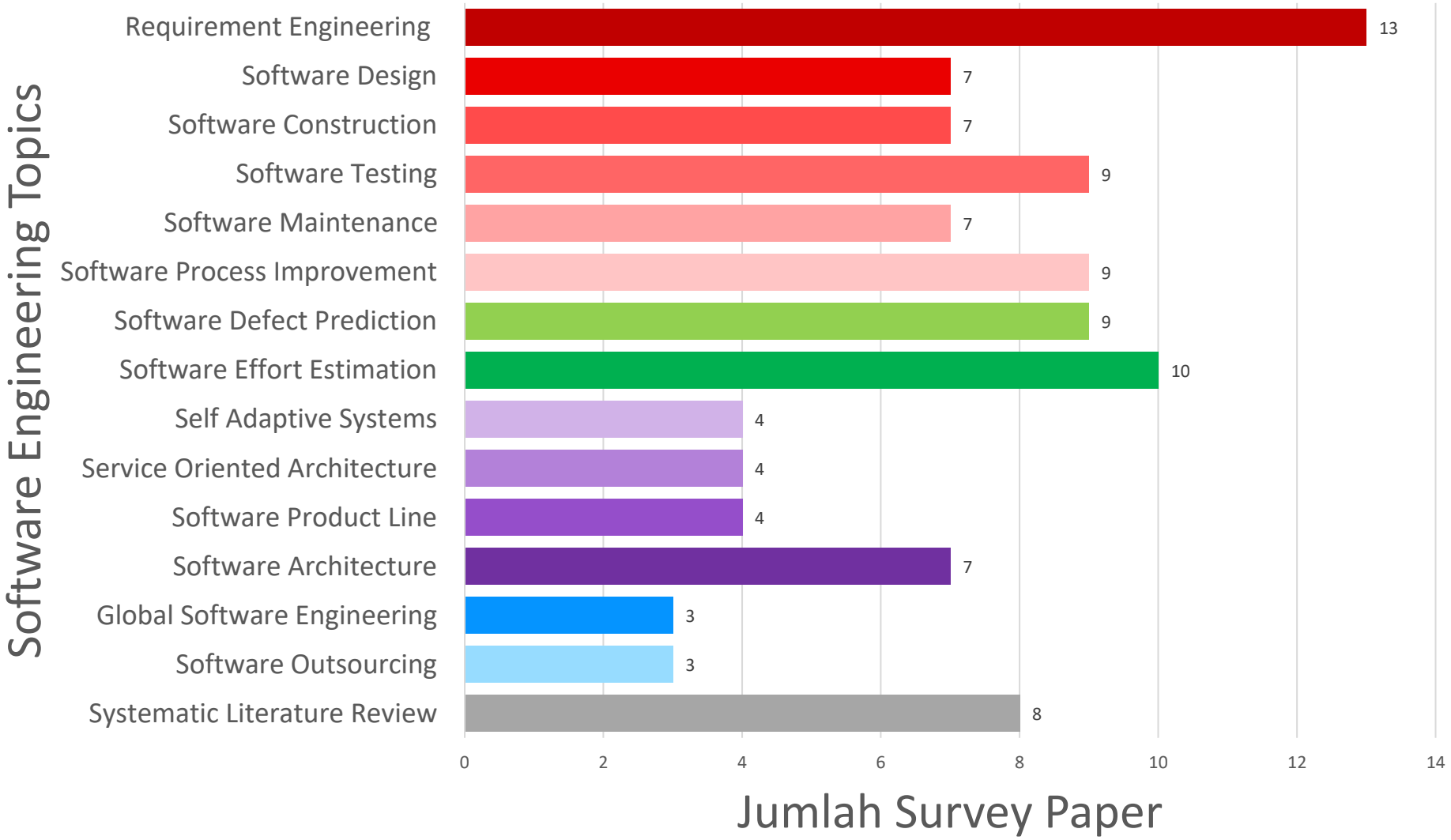
Software tools for GSE should therefore help to alleviate problems such as: (a) *Geographic Dispersion*, which sometimes causes a loss of synchronous communication or team interactions, since the sites are in different time zones; (b) *Control and Coordination Breakdown*, owing to the difficulties created by a distributed environment; (c) *Loss of Communication*; this is the case in this type of environment, if we consider that the richest communication medium is face-to-face communication; (d) *Loss of Team Spirit* and trust among team members [8] and (e) *Cultural Differences* which occur when people from different cultures work together in a global environment [9].

Software Effort = Software Cost?

1. Introduction

Software development effort estimation (SDEE) is the process of predicting the effort required to *develop* a software system. Generally speaking, SDEE can be considered as a sub-domain of *software effort estimation*, which includes the predictions of not only software development effort but also software maintenance effort. The terminology *software cost estimation* is often used interchangeably with *software effort estimation* in research community, though *effort* just forms the major part of software *cost*. Estimating development effort accurately in the early stage of software life cycle plays a crucial role in effective project management. Since 1980s, many estimation methods have been proposed in SDEE domain. Jørgensen and Shepperd [1] conducted a review which identifies up to 11 estimation methods used for SDEE. Among them, regression-based method is found to dominate, and the use of expert judgment and analogy method is increasing. In recent years, machine learning (ML) based method has been receiving increasing attention in SDEE research. Some researchers [2–4] even view ML based method as one of the three major categories of estimation methods (the other two are expert judgment and algorithmic model). Boehm and Sullivan [5] considered the learning-oriented technique as one of the six categories of software effort estimation techniques. Zhang and Tsai [6] summarized the applications of various ML techniques in SDEE domain.

Software Engineering Research Trends



Resources: Survey Papers from ScienceDirect, SpringerLink, and IEEE Explore (2011-2020)

Research Topics	Description
Global Software Engineering	Metode dan teknik pengembangan dan pelayanan software dengan environment dan sumber daya tersebar di berbagai negara
Requirement Engineering	Metode dan teknik pengumpulan kebutuhan dalam proses pengembangan software
Self Adaptive Systems	Software yang berkarakter autonomous dan bisa memperbaiki diri sendiri
Software Architecture	Metode dan teknik pengembangan arsitektur software untuk mengurangi kompleksitas: arsitektur model-view-controller, enterprise architecture, etc
Service Oriented Architecture	Metode dan teknik pengembangan dan pelayanan software sebagai sebuah service (software as a services (SaaS) serta proses deliverynya ke pengguna
Software Construction	Metode dan teknik konstuksi software, termasuk: programming paradigm, code programming, refactoring, clone detection, code convention, etc
Software Cost Estimation	Estimasi effort atau cost (berapa orang dan bulan) dari pengembangan software, termasuk: function points, use case points, atau dengan metode machine learning
Software Defect Prediction	Prediksi bug software dengan menggunakan pendekatan machine learning
Software Design	Metode dan teknik perancangan software, termasuk: design pattern, modelling language, forward and reverse engineering, model driven development, etc
Software Maintenance	Metode dan teknik perawatan software setelah software dikembangkan
Software Outsourcing	Metode dan teknik outsourcing dan offshoring pengembangan serta pelayanan software, termasuk: strategi dan parameter dalam pemilihan vendor, etc
Software Process Improvement	Perbaikan proses, siklus, metodologi, dan pengukuran maturity model dari proses pengembangan software
Software Product Line	Metode dan teknik pengembangan dan pengklasifikasian software produk yang memiliki kesamaan karakter dan tujuannya
Software Testing	Metode dan teknik pengujian software untuk berbagai jenis pengujian dan platform
Systematic Literature Review	Penelitian survey yang membahas satu topik penelitian bidang software engineering

Kiat Memilih Topik Penelitian

- Pilih topik bukan karena pekerjaan kita sekarang, tapi karena topiknya menarik (ada passion) dan secara penelitian dapat kita lakukan (tidak mission impossible)
- Usahakan cari penelitian yang membuat kita bisa konsentrasi penuh ke method improvement, tidak harus pontang-panting menjelaskan tentang obyek organisasi, mencari dataset, dsb
- Pilih topik yang dataset sudah tersedia secara public, jadi tidak perlu kita repot mencari dataset untuk eksperimen kita
- Pilih topik yang mudah secara pengukuran penelitian dan bila memungkinkan pengukuran cukup dengan komputer
  - Penelitian requirement engineering, termasuk yang rumit pengukuran penelitiannya, melibatkan manusia dan organisasi sebagai obyek
- Pilih topik sesuai kapasitas dan kapabilitas
  - Kita tidak mungkin penelitian tentang software process improvement apabila tidak tersedia organisasi sebagai testbed yang menerapkan metodologi yang kita kembangkan
- Pilih topik yang memungkinkan kita lakukan dengan laptop kita yang kita miliki sekarang, kecuali kita mendapatkan grant research besar yang memungkinkan pembelian infrastruktur penelitian
  - Penelitian global software engineering, software outsourcing, product line, relative agak perlu biaya lebih besar dan kompleks

Terima Kasih

Romi Satria Wahono
romi@romisatriawahono.net
http://romisatriawahono.net
08118228331

Reference

- Abbott, M., & McKinney, J. (2013). Understanding and Applying Research Design. John Wiley & Sons, Inc.
- Berndtsson, M., Hansson, J., & Olsson, B. (2008). Thesis Projects: a Guide for Students in Computer Science and Information Systems (2nd ed.). London: Springer-Verlag
- Blaxter, L., Hughes, C., & Tight, M. (2006). How to Research (3rd ed.). Open University Press
- Blessing, L. T. M., & Chakrabarti, A. (2009). DRM, a Design Research Methodology. Springer-Verlag London
- Cohen, L., Manion, L., & Morrison, K. (2005). Research Methods in Education (5th ed.). Taylor & Francis Group.
- Dawson, C. W. (2009). Projects in Computing and Information Systems A Student's Guide (2nd ed.). Pearson Education Limited
- Jonker, J., & Pennink, B. (2010). The Essence of Research Methodology. Springer-Verlag Berlin Heidelberg
- Lichtfouse, E. (2013). Scientific Writing for Impact Factor Journals. Nova Science Publishers, Inc.

Reference

- Kothari, C. (2004). Research Methodology: Methods and Techniques. New Age International
- Might, M. (2010). The Illustrated Guide to a Ph.D. Matt.might.net. Retrieved from *http://matt.might.net/articles/phd-school-in-pictures/*
- Marczyk, G., DeMatteo, D., & Fertinger, D. (2005). Essentials of Research Design and Methodology. John Wiley & Sons, Inc.
- Rea, L. M., & Parker, R. A. (2014). Designing and Conducting Survey Research: A Comprehensive Guide (4th ed.). John Wiley & Sons, Inc.
- Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, Inc.
- Sahu, P. K. (2013). Research Methodology: A Guide for Researchers In Agricultural Science, Social Science and Other Related Fields. Springer.
- Veit, R., Gould, C., & Gould, K. (2013). Writing, Reading, and Research (9th ed.). Cengage Learning.

Reference