5 KIAT S3

Lulus Tepat Waktu

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

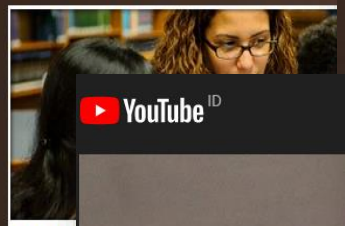
Romi Satria Wahono

- 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

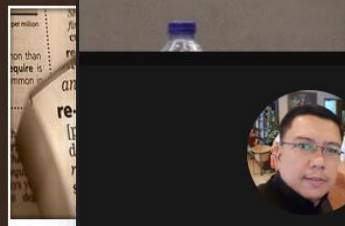
Lecture Notes in Software Engineering, Computing Research and Technopreneurship

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)

Serial Metodologi Penelitian (*Youtube Channel: Romi Satria Wahono*)

Research Methodology

5 videos • 35 views • Updated 2 days ago

Romi Satria Wahono

SUBSCRIBED

1 

10 Mitos Kesalahan Penelitian Computing yang Membuat Mahasiswa dan Dosen Gelisah

Romi Satria Wahono

2 

Research Methodology untuk Mahasiswa dan Dosen Galau

Romi Satria Wahono

3 

Systematic Literature Review (SLR): Jadi Nggak Galau Lagi Nulis Bab 2 Skripsi, Tesis dan Disertasi

Romi Satria Wahono

4 

Software Engineering Research Trends untuk Nolong Mahasiswa, Dosen & Peneliti Milih Topik

Romi Satria Wahono

5 

Kuliah 20 Menit tentang Metodologi Penelitian

Romi Satria Wahono

5 KIAT S3

Lulus Tepat Waktu

1. Pilih **Topik yang Tepat**, Lakukan Penelitian dan **Publikasi** Sebelum Masuk Program S3, Mulai Jalin **Komunikasi** dengan Calon Supervisor

2. Buat **Proposal Lengkap** Termasuk Systematic Literature Review (**SLR**) yang Memuat State-of-the-art Problems, Methods, Dataset (Anggap Draft Awal Bab 1 dan 2 Disertasi), Prioritaskan **Fulltime S3** Bila Mungkin

3. Masuk Program S3, Target Semester 1 Rapikan dan **Konversi SLR menjadi Paper** untuk Publikasi, Ikuti Perkuliahan dengan Aktif di Kelas, dan Jalin **Komunikasi Cerdas** dan Intensif dengan Supervisor

4. Eksekusi Proposal, Mulai dari RQ1-RQ2, Lakukan Eksperimen dengan Target **Publikasi ke Journal** Q3 atau Q4, Lakukan **Perbaikan Eksperimen dan Paper berdasarkan Hasil Review** Submission Paper

5. Lanjutkan Eksperimen ke RQ berikutnya (RQ3-RQ4-RQ*n*), **Ulangi Siklus Eksperimen dan Publikasi** Hingga Persyaratan Kelulusan Terpenuhi, **Disertasi adalah Kumpulan dari Publikasi** Penelitian

KIAT 1

Pilih Topik yang Tepat, Lakukan Penelitian dan Publikasi Sebelum Masuk Program S3, Mulai Jalin Komunikasi dengan Calon Supervisor

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 Tahapan Penelitian

- 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*)

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

Jenis Paper Ilmiah

1. Technical Paper
   - Paper yang isinya adalah hasil penelitian dan eksperimen yang dilakukan seorang peneliti
   - Penilaian kualitas technical paper dari kontribusi ke pengetahuan

2. Survey Paper
   - Paper yang isinya adalah review dan survey tentang topik/tema suatu penelitian, biasanya jumlah penelitian yang direview mencapai ratusan atau ribuan
   - Rujukan dan panduan penting bagi peneliti yang baru memulai penelitian untuk memahami suatu topik/tema penelitian secara komprehensif

Jenis Paper Survey

1. Traditional Review
2. Systematic Mapping Study (Scoping Study)
3. Systematic Literature Review or Systematic Review
   - Review komprehensif tentang satu topik penelitian
4. Tertiary Study (SLR of SLR)
   - Review komprehensif tentang berbagai topik penelitian yang ada di suatu bidang penelitian

1. Traditional Review

- Provides an overview of the research findings on particular topics
- Advantages: produce insightful, valid syntheses of the research literature if conducted by the expert
- Disadvantages: vulnerable to unintentional and intentional bias in the selection, interpretation and organization of content
- Examples:
  - Liao et al., Intrusion Detection System: A Comprehensive Review, Journal of Network and Computer Applications, 36(2013)
  - Galar et al., A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), Vol. 42, No. 4, July 2012
  - Cagatay Catal, Software fault prediction: A literature review and current trends, Expert Systems with Applications 38 (2011)

2. Systematic Mapping Study

- Suitable for a very broad topic
- Identify clusters of evidence (making classification)
- Direct the focus of future SLRs
- To identify areas for future primary studies
- Examples:
  - Neto et al., A systematic mapping study of software product lines testing, Information and Software Technology Vol. 53, Issue 5, May 2011
  - Elberzhager et al., Reducing test effort: A systematic mapping study on existing approaches, Information and Software Technology 54 (2012)

3. Systematic Literature Review (SLR)

- The purpose of a systematic literature reviews is to provide as complete a list as possible of all the published studies relating to a particular subject area
- A process of identifying, assessing, and interpreting all available research evidence, to provide answers for a particular research question
- A form of secondary study that uses a well-defined methodology
- SLRs are well established in other disciplines, particularly medicine. They integrate an individual clinical expertise and facilitate access to the outcomes of the research

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

3. Systematic Literature Review (SLR)

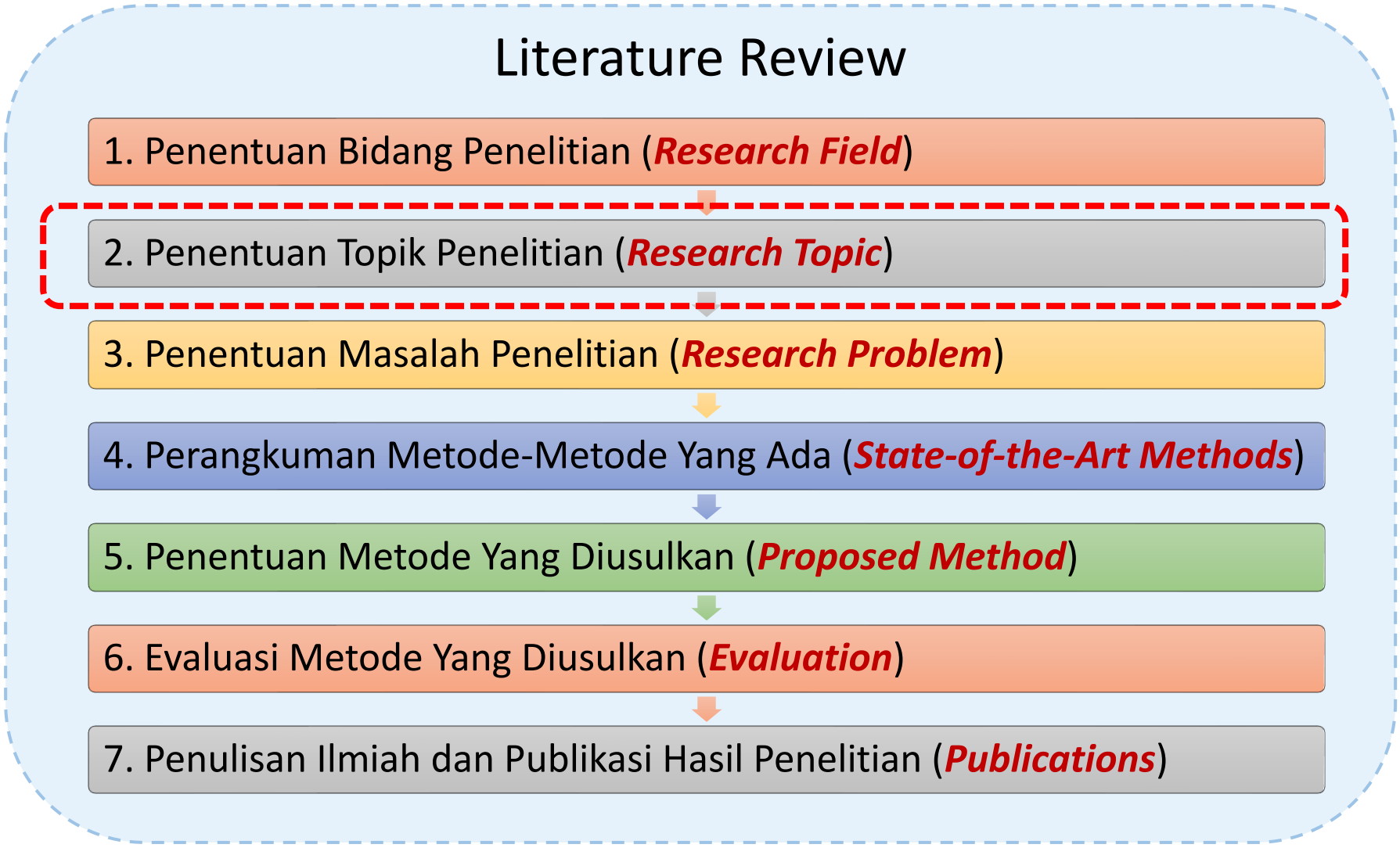
Examples of SLR:

- Hall et al., A Systematic Literature Review on Fault Prediction Performance in Software Engineering, IEEE Transaction on Software Engineering, Vol. 38, No. 6, 2012
- Romi Satria Wahono, A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks, Journal of Software Engineering, Vol. 1, No. 1, April 2015
- Matthias Galster, Danny Weyns, Dan Tofan, Bartosz Michalik, and Paris Avgeriou, Variability in Software Systems: A Systematic Literature Review, IEEE Transactions on Software Engineering, Vol 40, No 3, 2014

4. Tertiary study

- Is a SLR of SLRs
- To answer a more wider question
- Uses the same method as in SLR
- Potentially less resource intensive
- Examples:
  - Kitchenham et al., Systematic literature reviews in software engineering – A tertiary study, Information and Software Technology 52 (2010)
  - Cruzes et al., Research synthesis in software engineering: A tertiary study, Information and Software Technology 53 (2011)

Tahapan Penelitian Computing

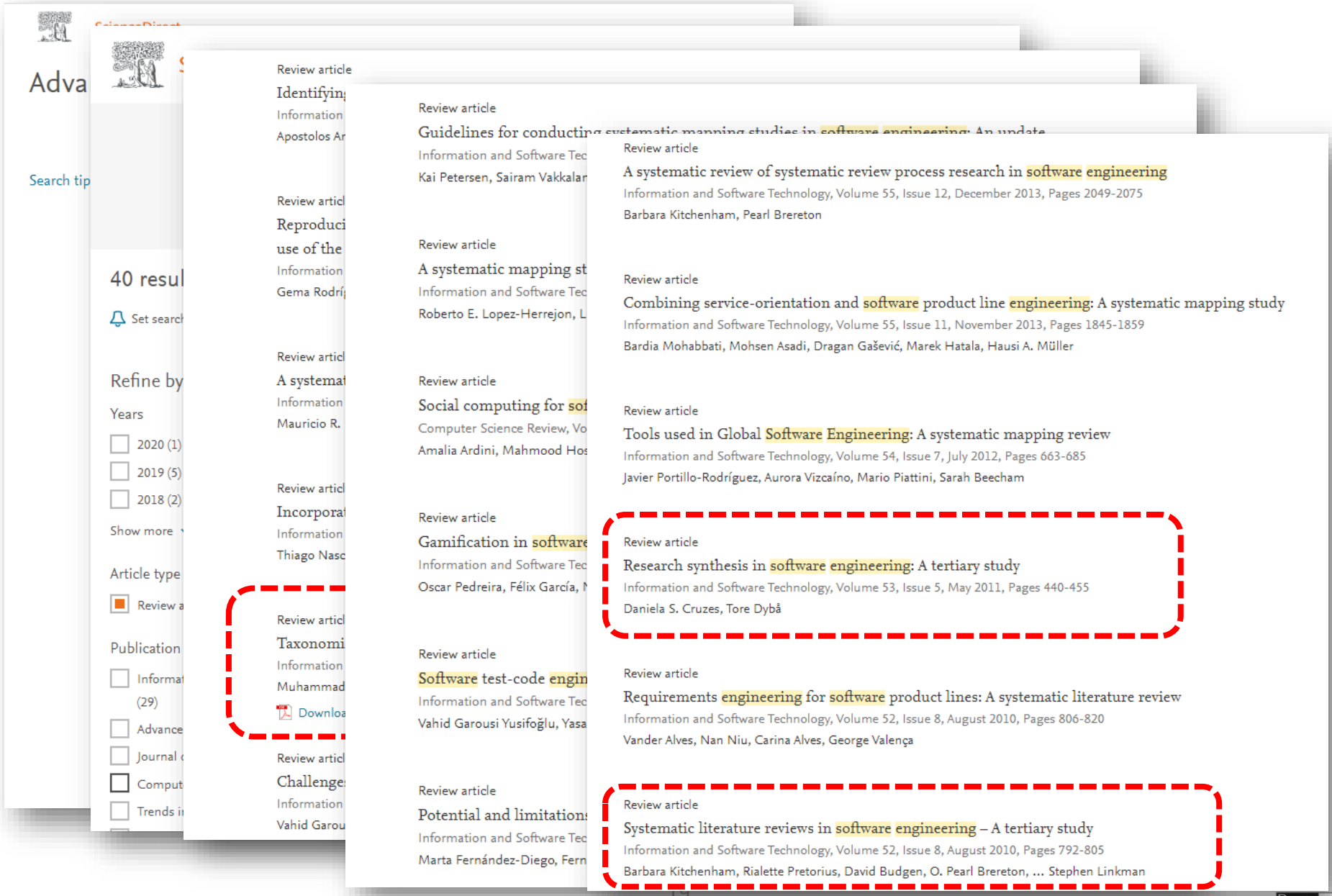


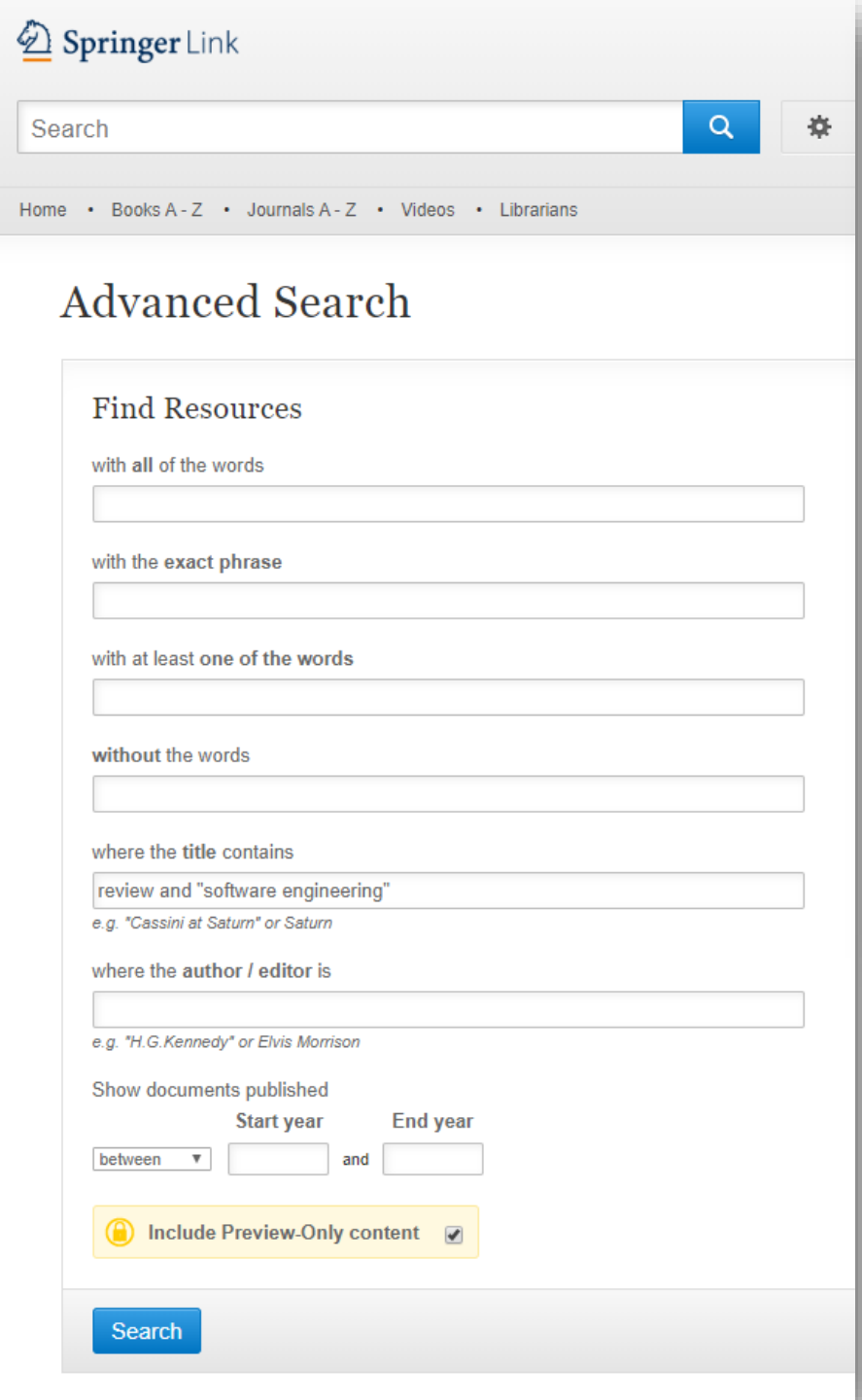
*https://www.site.uottawa.ca/~bochmann/dsrg/how-to-do-good-research/
*http://romisatriawahono.net/2013/01/23/tahapan-memulai-penelitian-untuk-mahasiswa-galau/

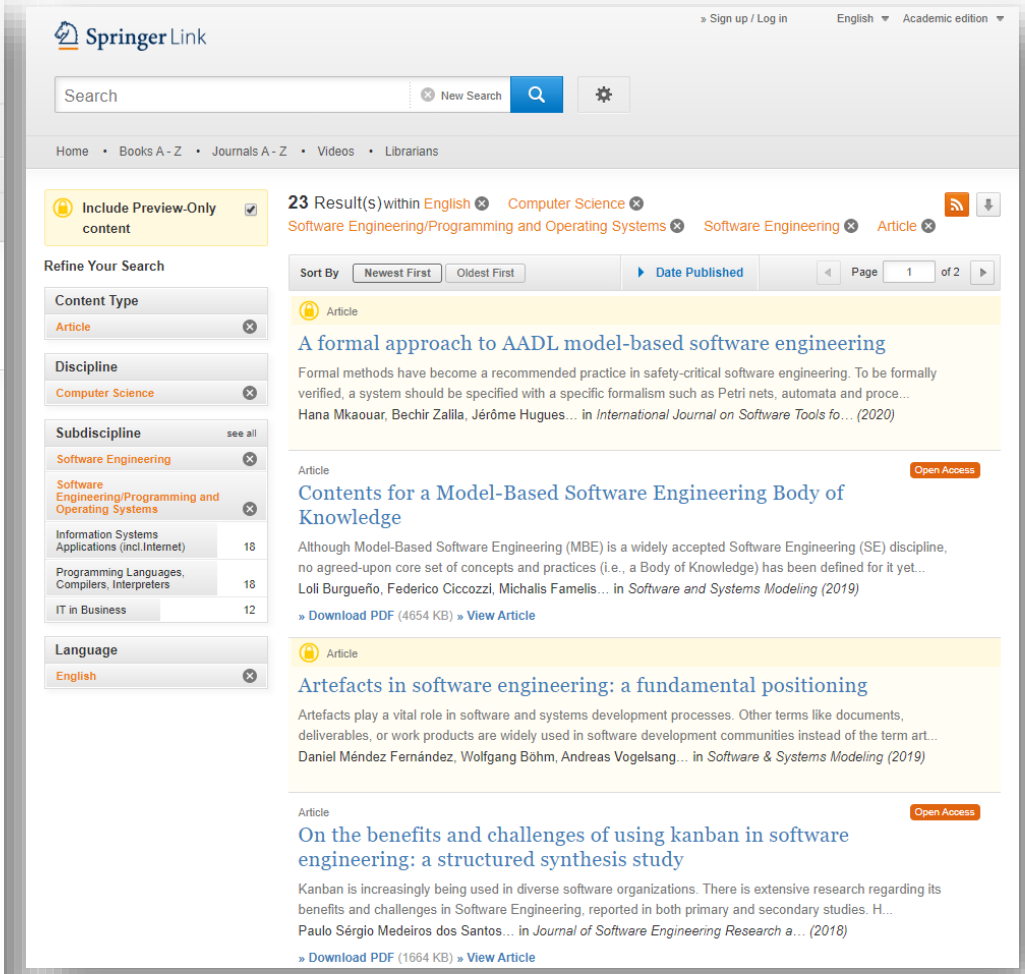
2. Penentuan Topik Penelitian

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

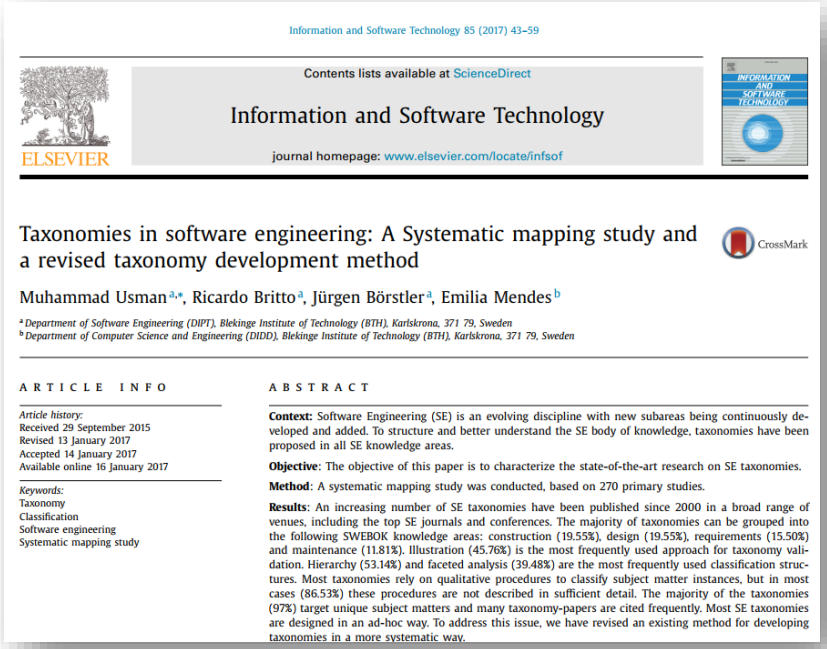






Taxonomies in Software Engineering *(Usman et al., 2017)*

- 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

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**

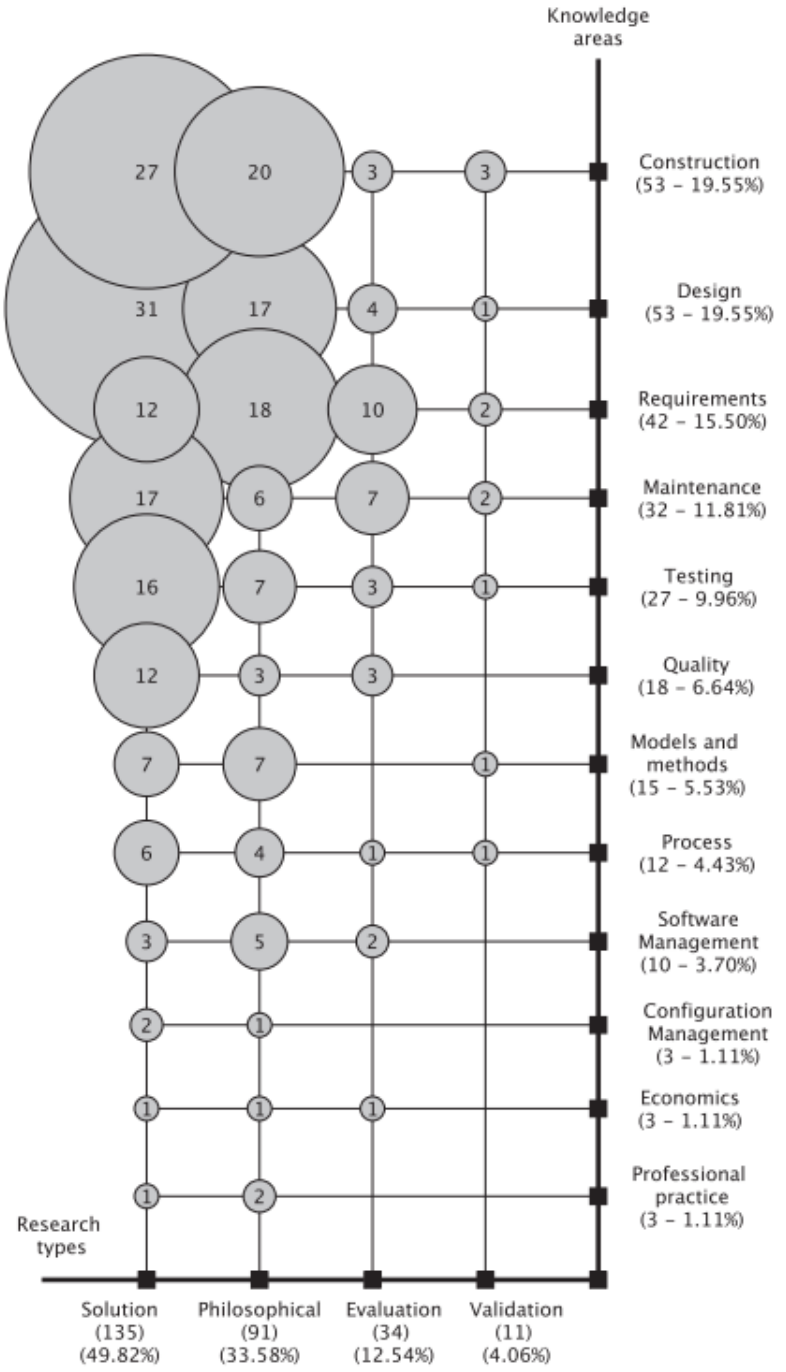


Fig. 8. Systematic map – knowledge area vs research type.

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)*

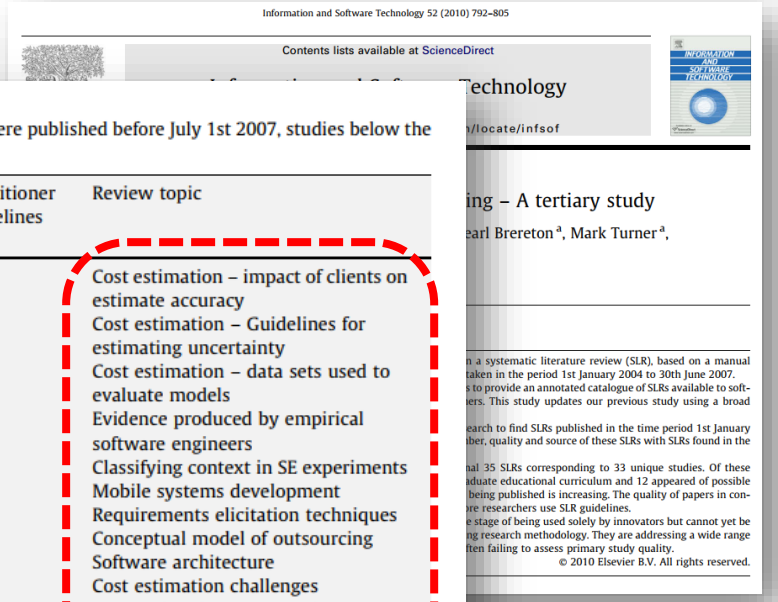
Table 7
Authors with three or more studies.

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)*

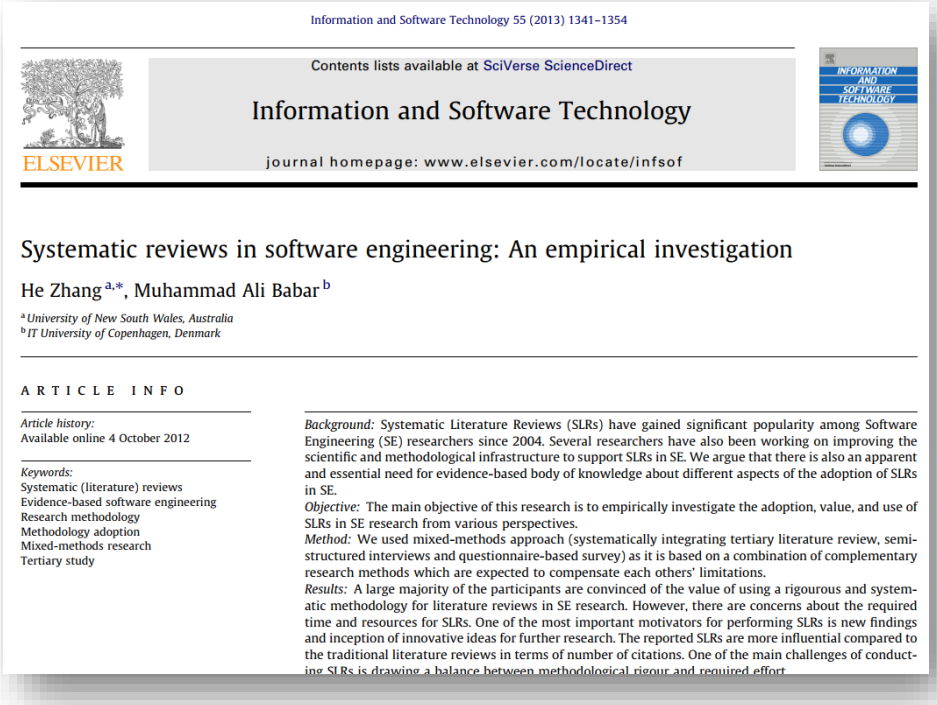
Table 5
Distribution of SLRs over the SE sections of the University curriculum guidelines.

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)

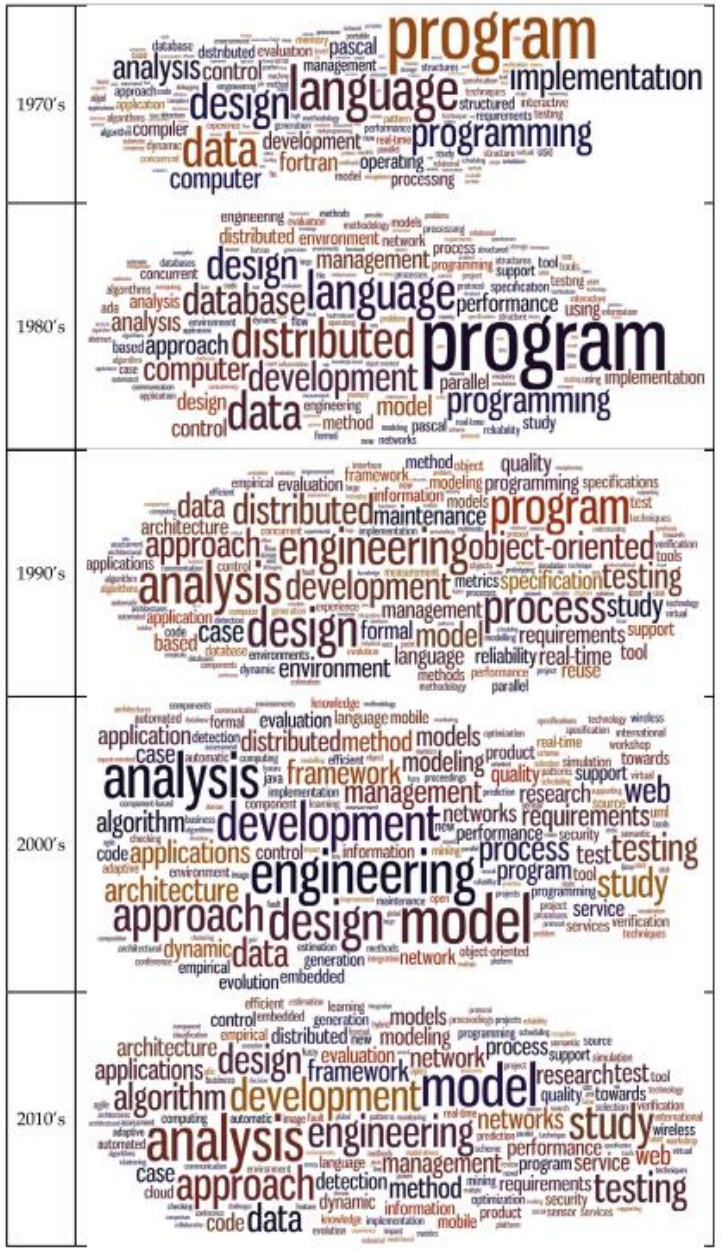


Fig. 10 – Focus areas of SE papers in each decade.

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

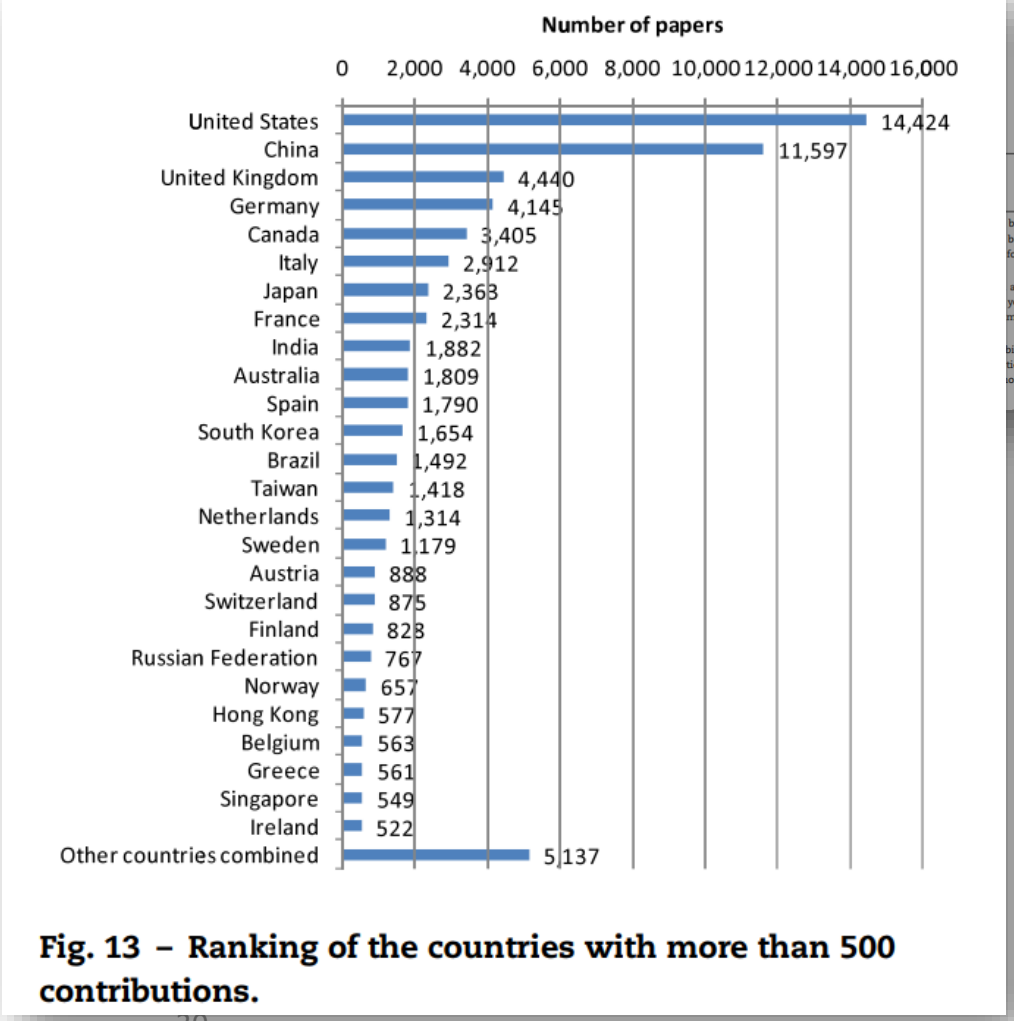


Fig. 13 – Ranking of the countries with more than 500 contributions.

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.

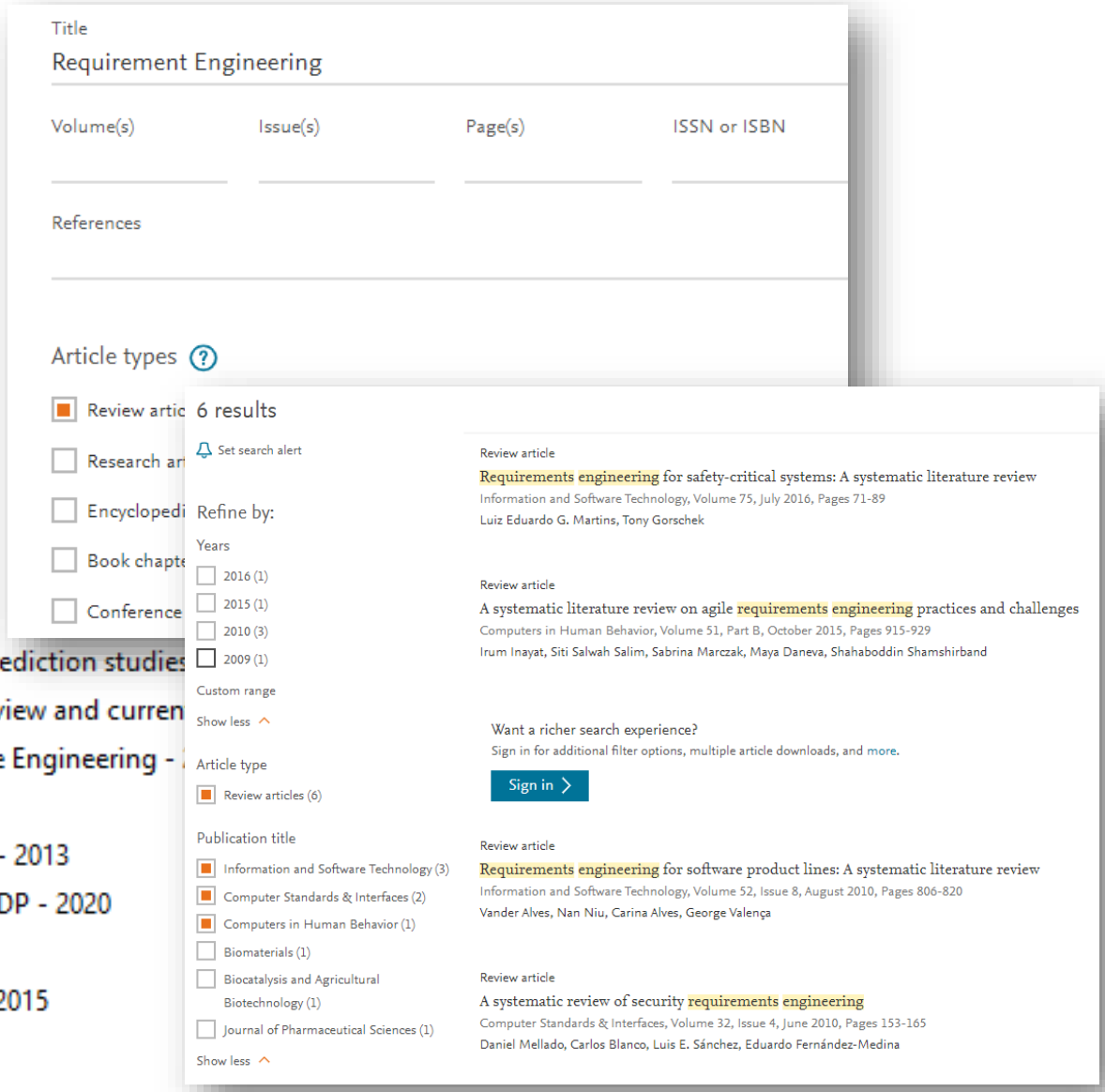
Given topic name	Model checking ($n = 686$)	Test generation (=923)	Source code ($n = 988$)	Automated testing ($n = 785$)	Systematic reviews ($n = 653$)
1	"model"	"test"	"code"	"testing"	"software"
2	"checking"	"generation"	"source"	"automated"	"systematic"
3	"transformation"	"automatic"	"open"	"model 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"

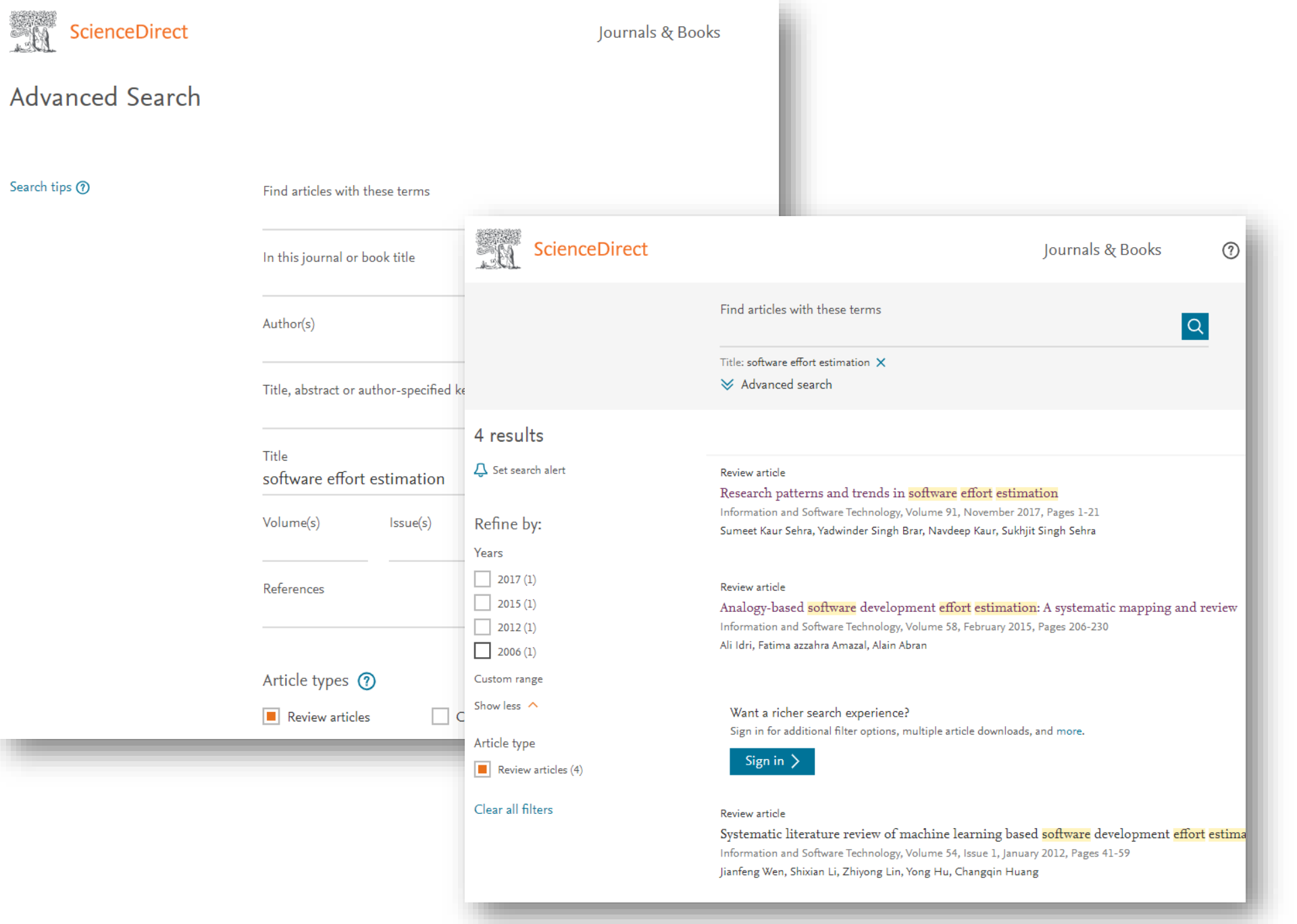
2. Cari SLR dari Topik Penelitian yang Dipilih

1. Setelah kita paham beberapa topik penelitian di bidang software engineering dari *Tertiery Study (SLR dari SLR)*
2. Langkah berikutnya, kita kumpulkan seluruh SLR dengan keyword topik seperti di paper *Tertiery Study (SLR dari SLR)*
3. Lanjutkan dengan mengejar seluruh SLR dari topik yang kita akan angkat pada penelitian kita
   - Sambil mengkonfirmasi apakah klaim dari peneliti di langkah 2 di atas itu benar dan valid

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]

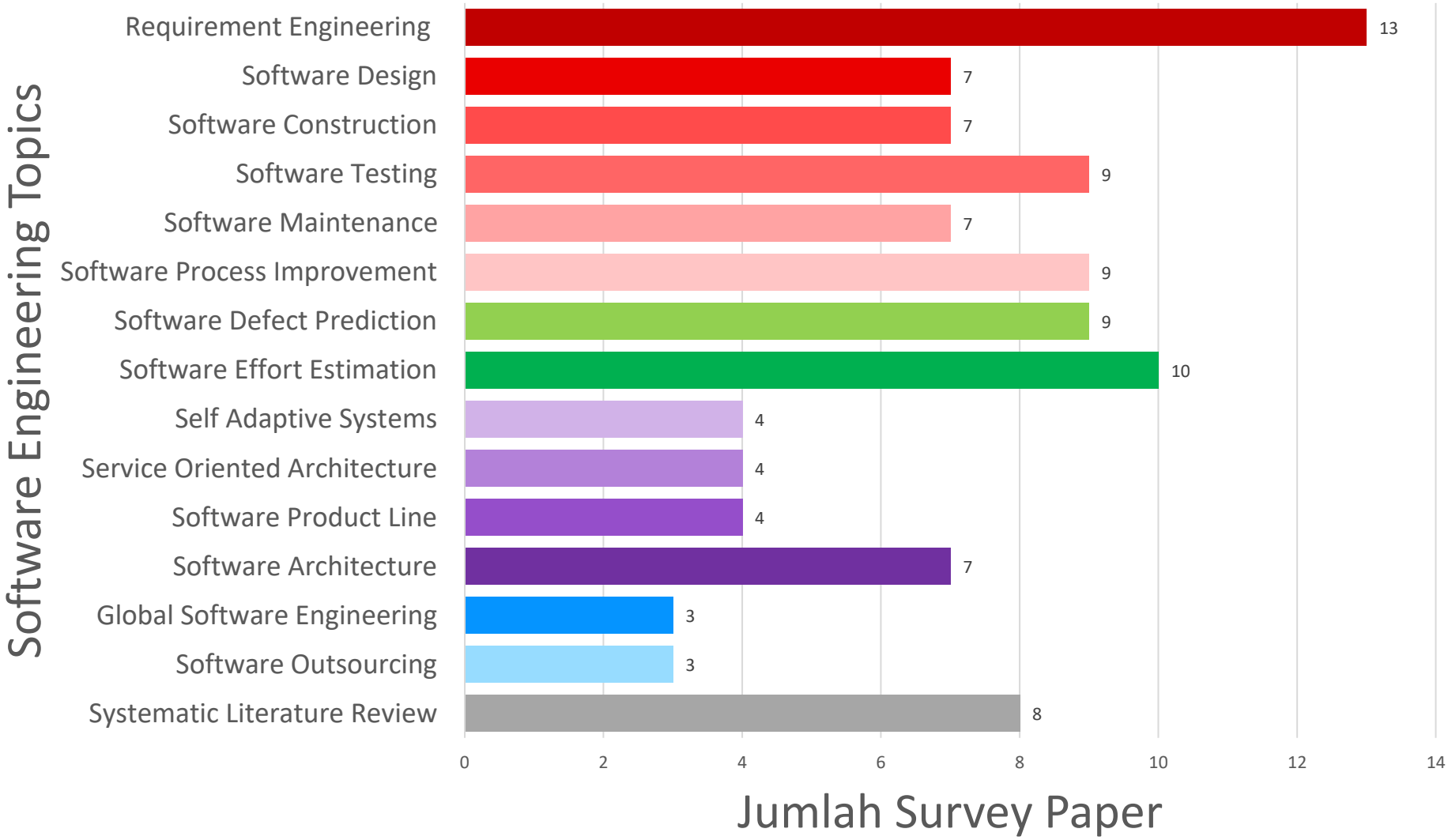
1. Introduction

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 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

Kiat Sebelum Masuk Program S3

- Usahakan topik penelitian linier dengan S2, kalau S2 nya ternyata "sekedar lulus", lakukan *taubatan nasuha*, perbaiki niat untuk tidak memperbaiki diri, dan lakukan pemilihan topik penelitian yang benar
- Lakukan penelitian di topik tersebut sampai siklus terakhir penelitian (publikasi penelitian) dilakukan, paling tidak publikasi ke journal terindeks scopus Q4 atau Q3.
  - Pengalaman publikasi di journal terindeks akan menjadi dukungan signikan ketika menjalani proses ujian masuk program S3
- Pahami persyaratan untuk lulus S3, berapa jumlah publikasi di journal terindeks yg disyaratkan harus diketahui, supaya langkah kita dalam penelitian dan publikasi bisa sistematis
- Lakukan pendekatan ke calon supervisor, lihat dan amati bidang garapan dan kualitas publikasi penelitian dari calon supervisor
  - Kualitas publikasi supervisor termasuk salah satu parameter penentu kita bisa lulus tepat waktu atau tidak

KIAT 2

Buat Proposal Lengkap Termasuk Systematic Literature Review (SLR) yang Memuat *State-of-the-art* Problems, Methods, Dataset (Kandidat Draft Awal Bab 1 dan 2 Disertasi), Prioritaskan Fulltime S3 Bila Mungkin

Mengapa Menggunakan Metode Systematic Literature Review (SLR)?

- SLR is now well-established review method in the field of software engineering
- Banyak journal terindeks yang menerima paper survey dengan menggunakan metode SLR
- SLR yang kita reformat menjadi paper, dan kita kirim ke journal, akan menjadi tabungan publikasi penelitian kita untuk program S3

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

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)

Tahapan SLR

1. Formulate the review's research question
2. Develop the review's protocol

PLANNING

1. Identify the relevant literature
2. Perform selection of primary studies
3. Perform data extraction
4. Assess studies' quality
5. Conduct synthesis of evidence

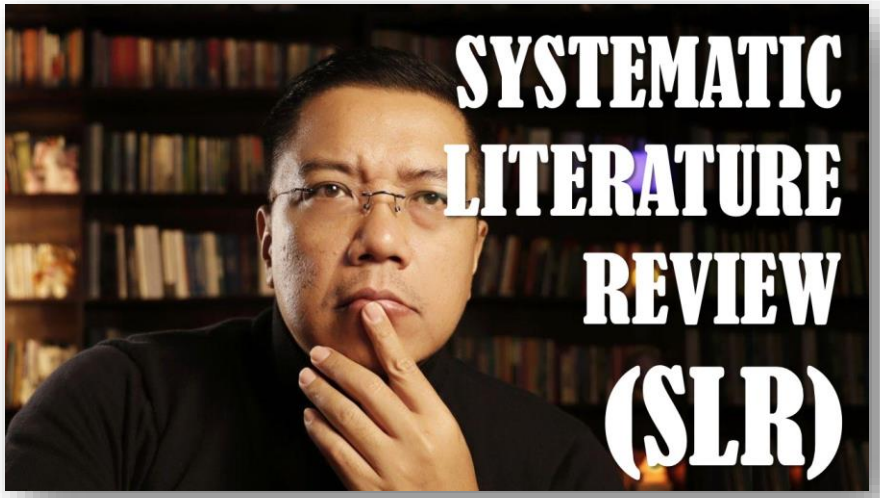
CONDUCTING

1. Write up the SLR report/paper
2. Choose the Right Journal

REPORTING

Sistematika Penulisan SLR

1. Introduction
   - General introduction about the research
   - State the purpose of the review
   - Emphasize the reason(s) why the RQ is important
   - State the significance of the review work and how the project contributes to the body of knowledge of the field
2. Main Body
   1. Review method – briefly describe steps taken to conduct the review
   2. Results – findings from the review
   3. Discussion – implication of review for research & practice
3. Conclusions



Contoh dan Studi Kasus SLR

Romi Satria Wahono, A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks, Journal of Software Engineering, Vol. 1, No. 1, pp. 1-16, April 2015

https://romisatriawahono.net/2016/05/15/systematic-literature-review-pengantar-tahapan-dan-studi-kasus/

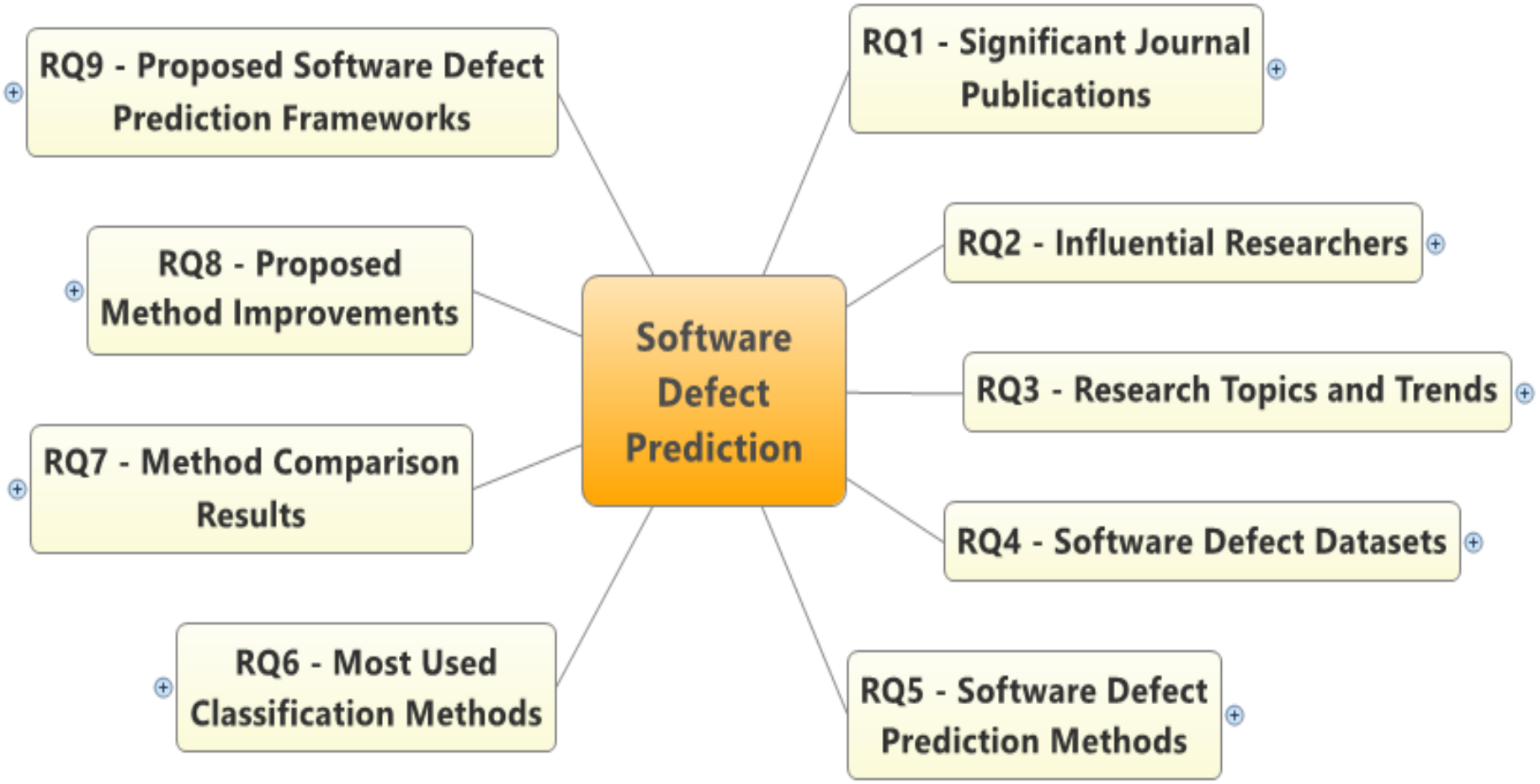
PICOC

Population	Software, software application, software system, information system
Intervention	Software defect prediction, fault prediction, error-prone, detection, classification, estimation, models, methods, techniques, datasets
Comparison	n/a
Outcomes	Prediction accuracy of software defect, successful defect prediction methods
Context	Studies in industry and academia, small and large data sets

Research Question (RQ)

ID	Research Question
RQ1	Which journal is the most significant software defect prediction journal?
RQ2	Who are the most active and influential researchers in the software defect prediction field?
RQ3	What kind of research topics are selected by researchers in the software defect prediction field?
RQ4	What kind of datasets are the most used for software defect prediction?
RQ5	What kind of methods are used for software defect prediction?
RQ6	What kind of methods are used most often for software defect prediction?
RQ7	Which method performs best when used for software defect prediction?
RQ8	What kind of method improvements are proposed for software defect prediction?
RQ9	What kind of frameworks are proposed for software defect prediction?

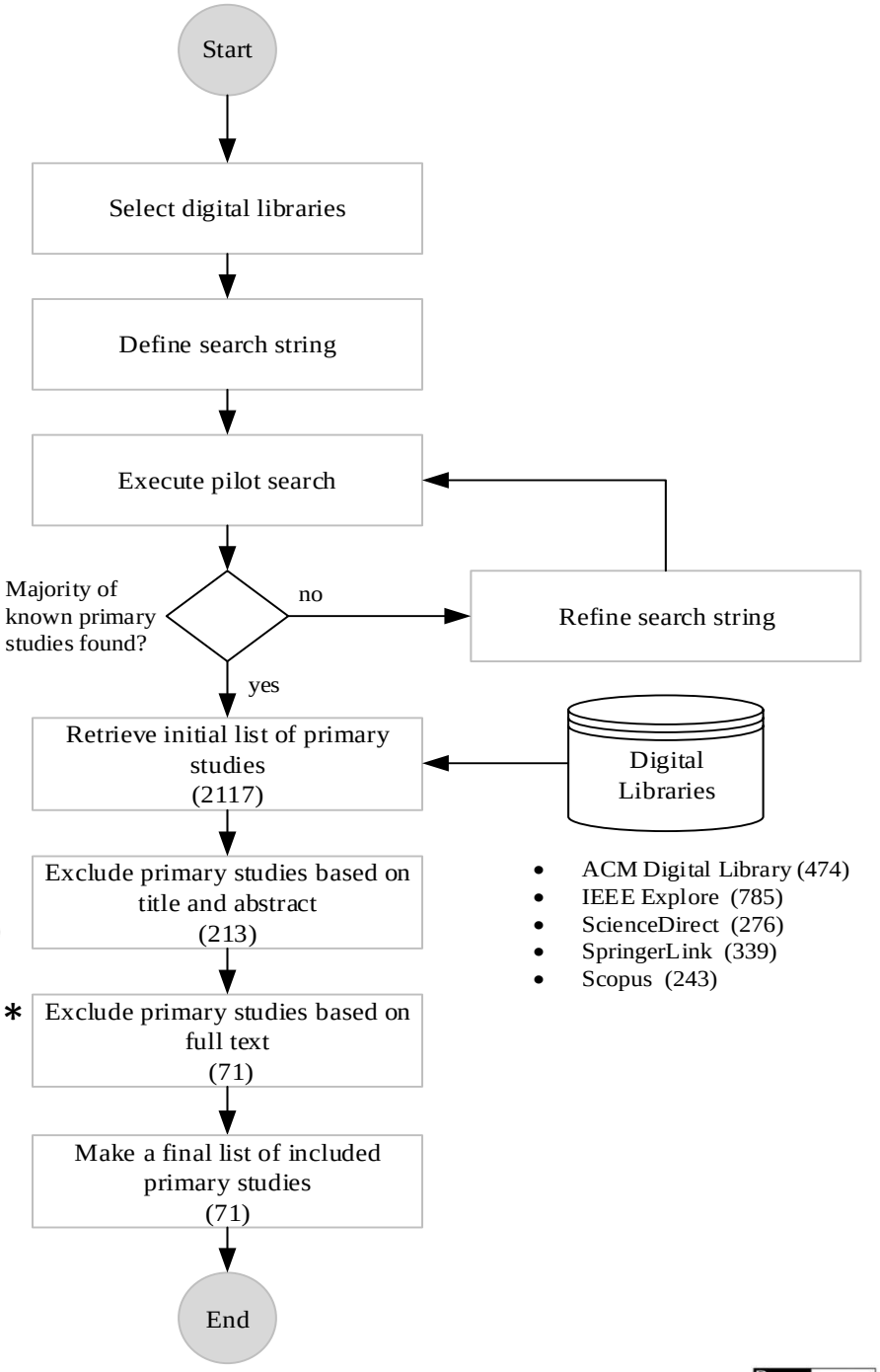
Research Question (RQ)



Studies Selection Strategy

- Publication Year:
  - ✓2000-2013
- Publication Type:
  - ✓Journal
  - ✓Conference Proceedings
- Search String:

  software
  AND
  (fault* OR defect* OR quality OR error-prone)
  AND
  (predict* OR prone* OR probability OR assess* OR detect* OR estimat* OR classificat*)
- Selected Studies:
  - ✓71



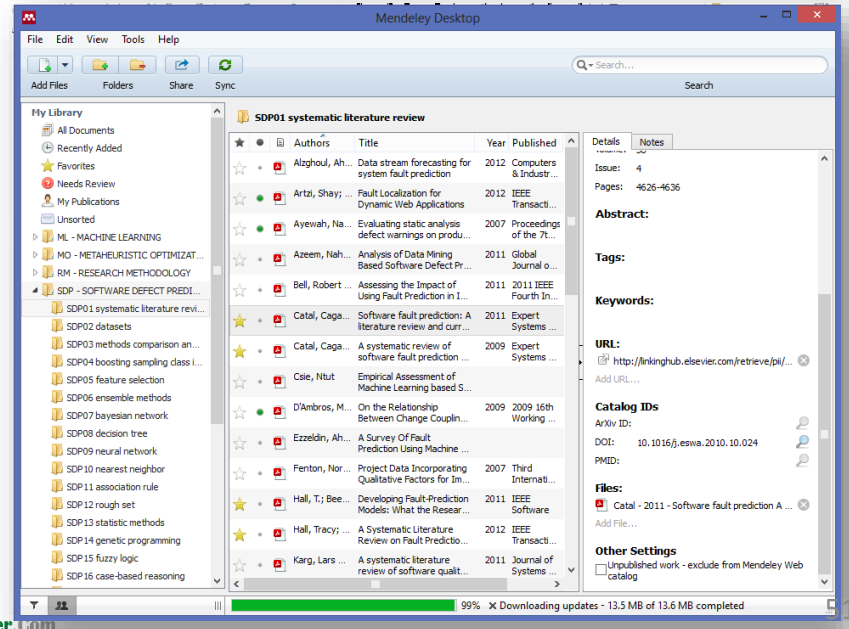
Inclusion and Exclusion Criteria

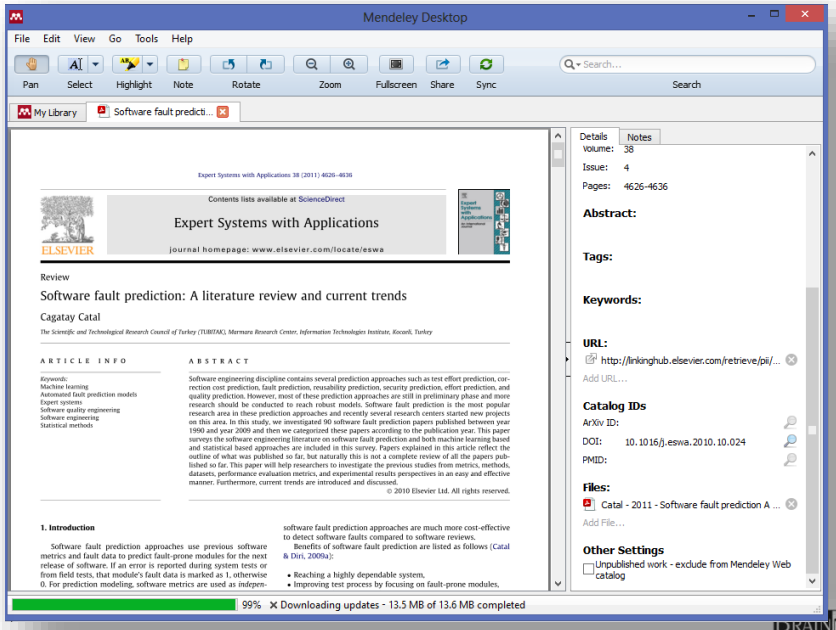
Inclusion Criteria	Studies in academic and industry using large and small scale data sets
	Studies discussing and comparing modeling performance in the area of software defect prediction
	For studies that have both the conference and journal versions, only the journal version will be included
	For duplicate publications of the same study, only the most complete and newest one will be included
Exclusion Criteria	Studies without a strong validation or including experimental results of software defect prediction
	Studies discussing defect prediction datasets, methods, frameworks in a context other than software defect prediction
	Studies not written in English

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"

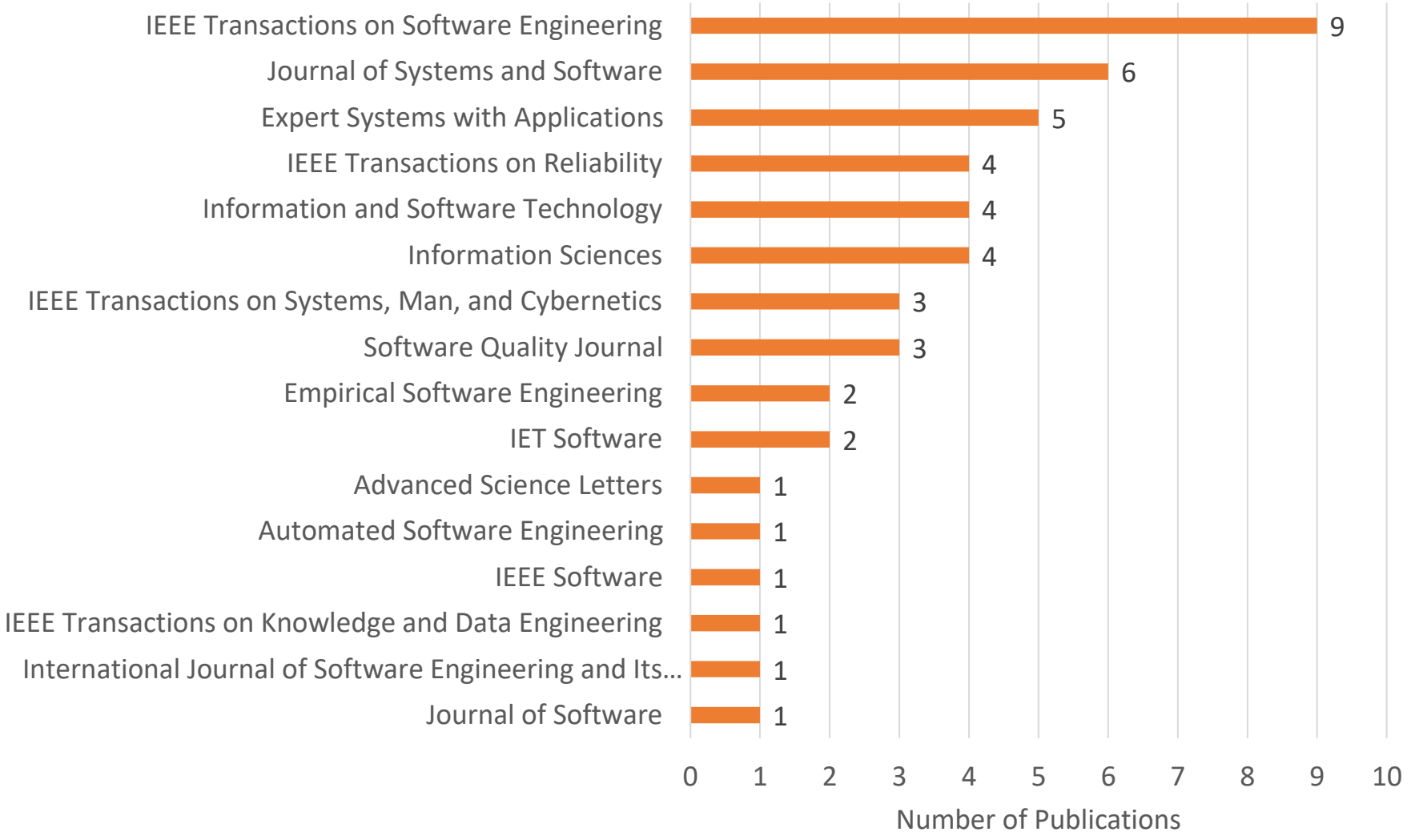




Result

Romi Satria Wahono, A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks, Journal of Software Engineering, Vol. 1, No. 1, pp. 1-16, April 2015

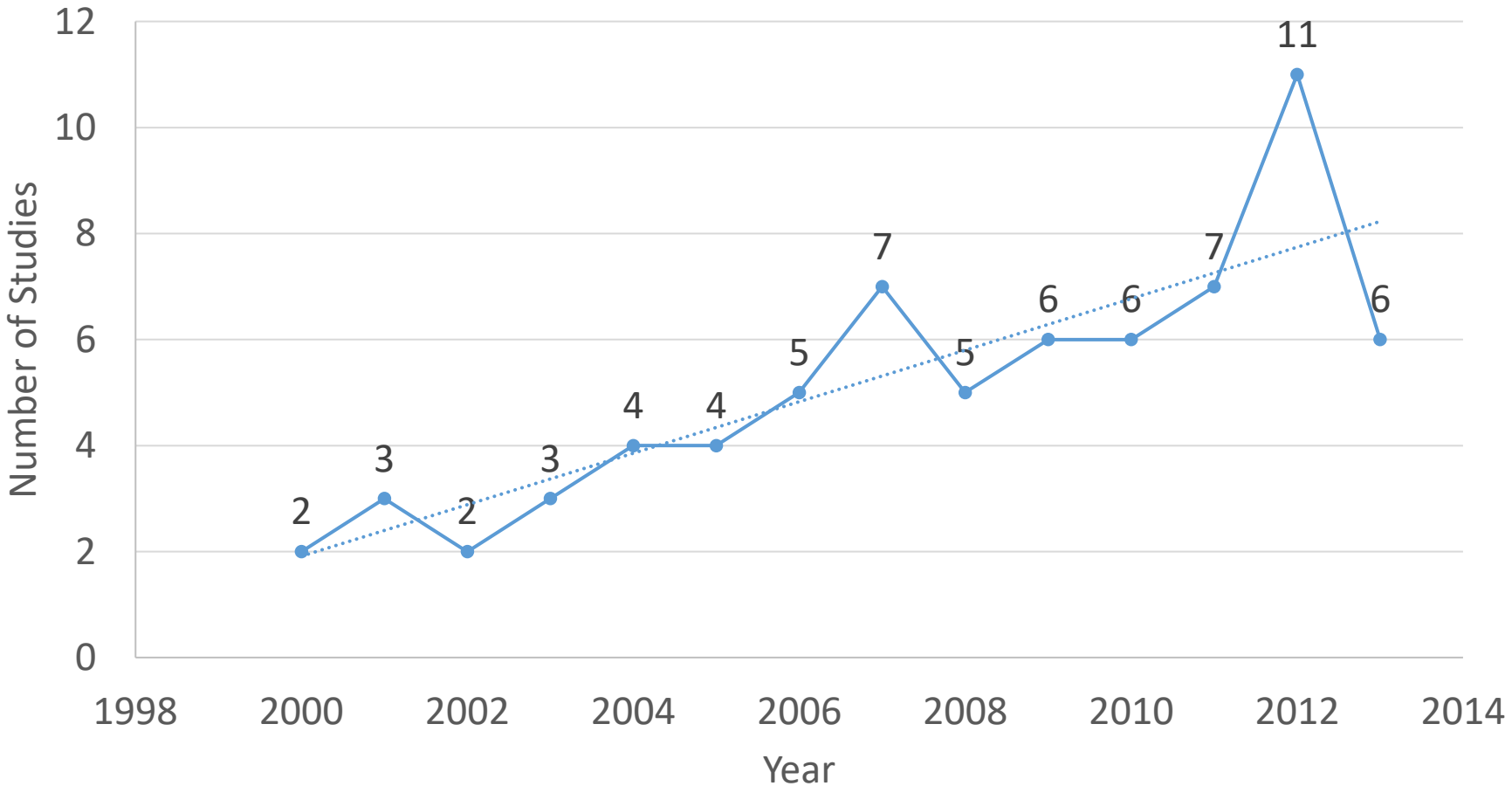
RQ1: Significant Journal Publications



Journal Quality Level of Selected Studies

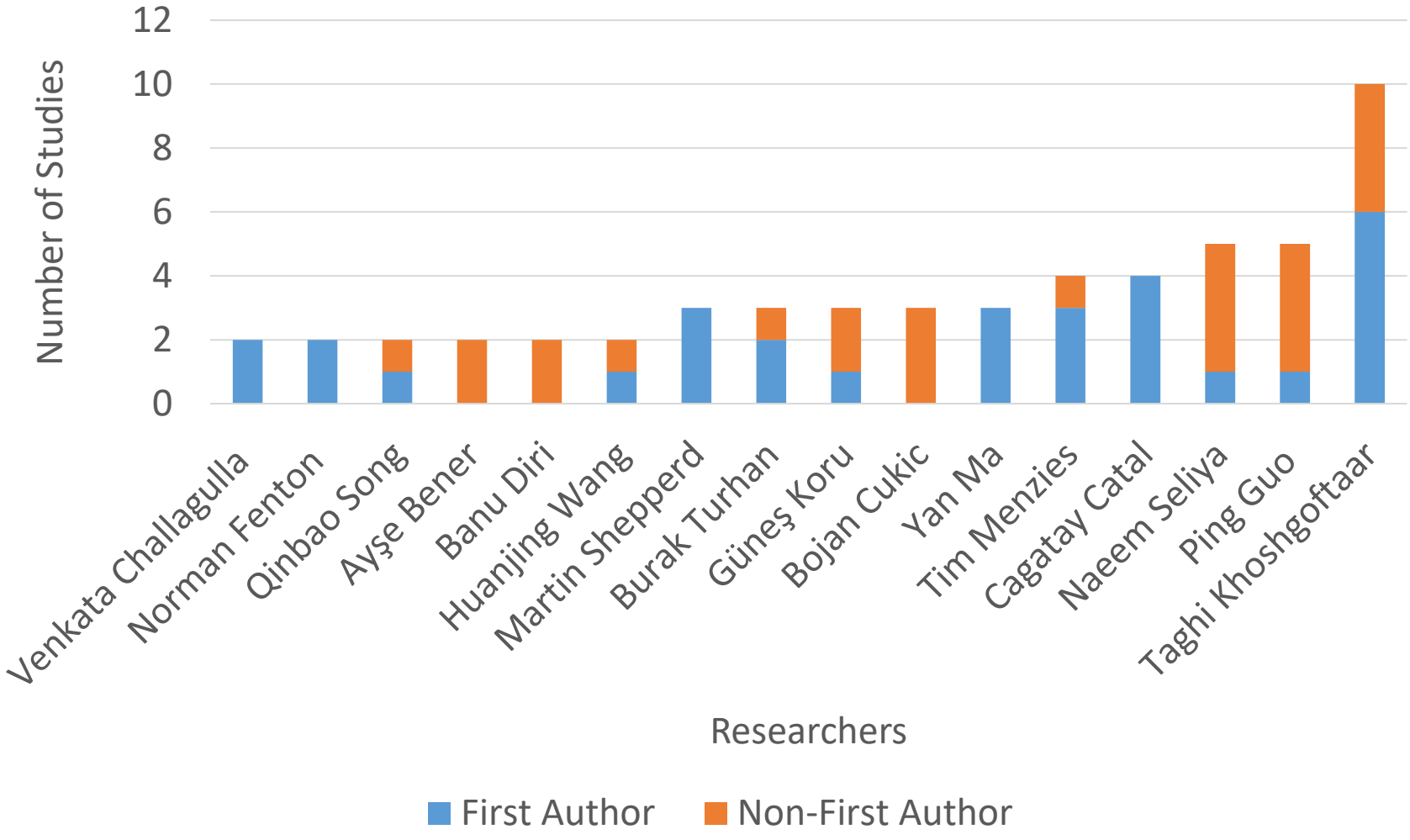
No	Journal Publications	SJR	Q Category
1	IEEE Transactions on Software Engineering	3.39	Q1 in Software
2	Information Sciences	2.96	Q1 in Information Systems
3	IEEE Transactions on Systems, Man, and Cybernetics	2.76	Q1 in Artificial Intelligence
4	IEEE Transactions on Knowledge and Data Engineering	2.68	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

Distribution of Selected Studies by Year



- The interest in software defect prediction has changed over time
- Software defect prediction research is still very much relevant to this day

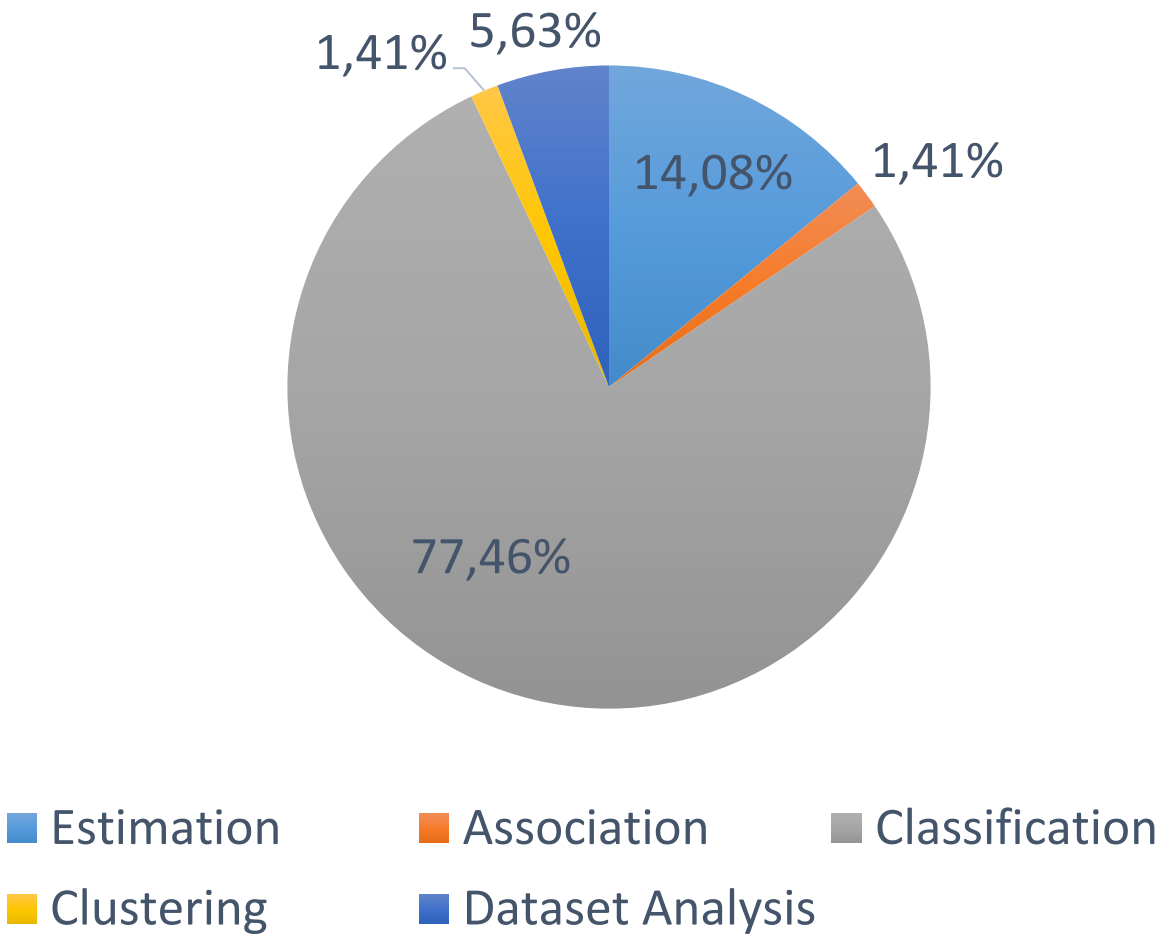
RQ2: Influential Researchers



RQ3: Research Topics and Trends

1. Estimating the number of defects remaining in software systems using estimation algorithm (Estimation)
2. Discovering defect associations using association rule algorithm (Association)
3. Classifying the defect-proneness of software modules, typically into two classes, defect-prone and not defect-prone, using classification algorithm (Classification)
4. Clustering the software defect based on object using clustering algorithm (Clustering)
5. Analyzing and pre-processing the software defect datasets (Dataset Analysis)

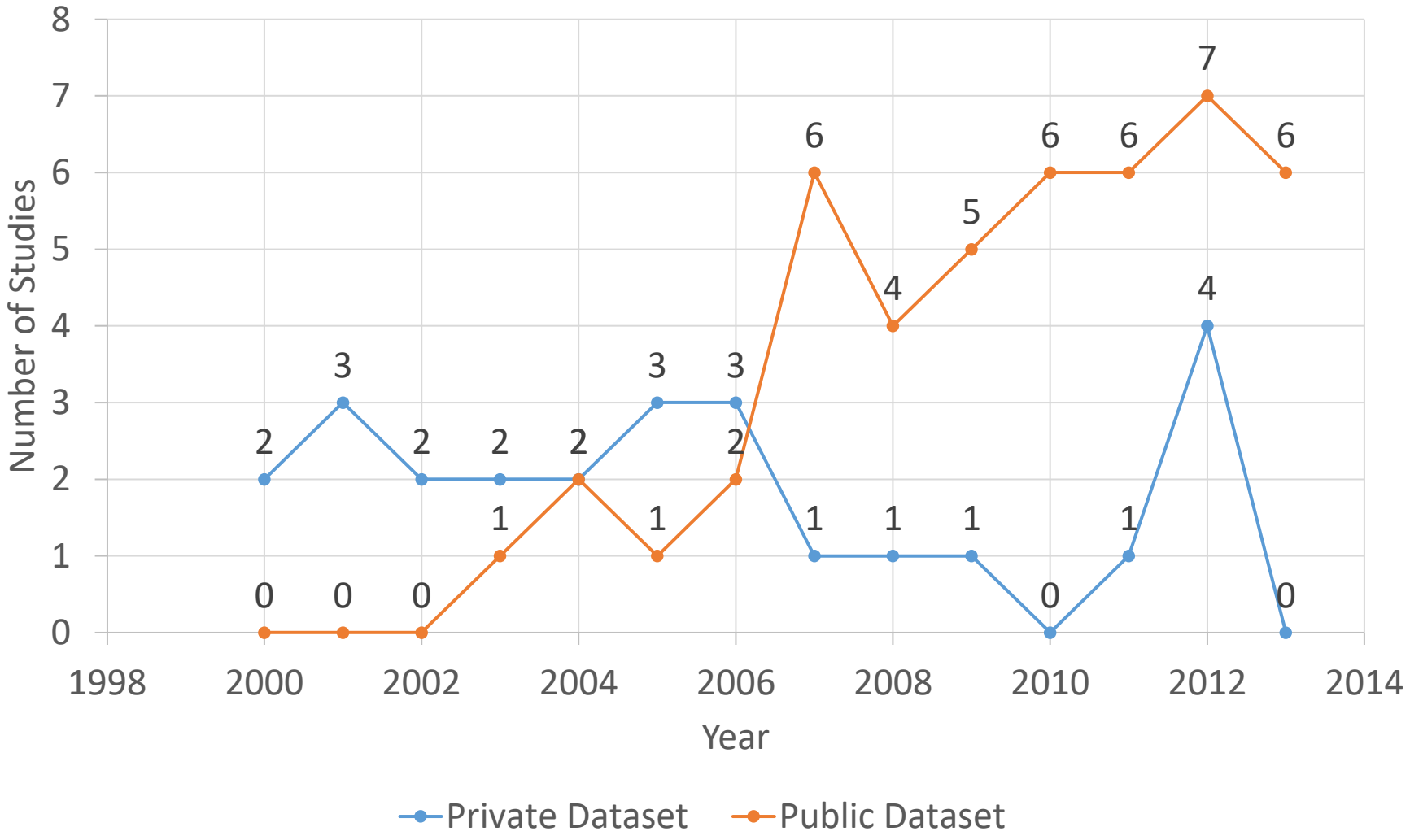
Distribution of Research Topics and Trends



Example Distribution of Research Topics and Trends

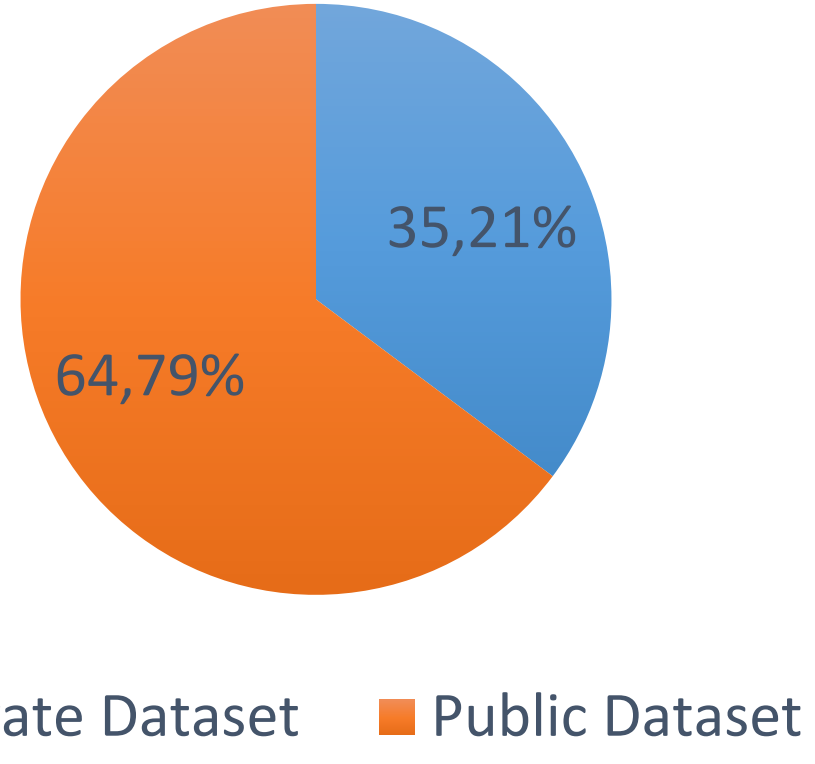
Year	Primary Studies	Publications	Datasets	Topics
2008	(Lessmann et al., 2008)	IEEE Transactions on Software Engineering	Public	Classification
	(Bibi et al., 2008)	Expert Systems with Applications	Private	Estimation
	(Gondra, 2008)	Journal of Systems and Software	Public	Classification
	(Vandecruys et al., 2008)	Journal of Systems and Software	Public	Classification
	(Elish and Elish 2008)	Journal of Systems and Software	Public	Classification
2012	(Gray et al., 2012)	IET Software	Public	Dataset Analysis
	(Ying Ma, Luo, Zeng, & Chen, 2012)	Information and Software Technology	Public	Classification
	(Benaddy and Wakrim 2012)	International Journal of Software Engineering	Private	Estimation
	(Y. Peng, Wang, & Wang, 2012)	Information Sciences	Public	Classification
	(Zhang and Chang 2012)	International Conference on Natural Computation	Private	Estimation
	(Bishnu and Bhattacherjee 2012)	IEEE Transactions on Knowledge and Data Engineering	Private	Clustering
	(Sun, Song, & Zhu, 2012)	IEEE Transactions on Systems, Man, and Cybernetics	Public	Classification
	(Pelayo and Dick 2012)	IEEE Transactions on Reliability	Public	Classification
	(Jin, Jin, & Ye, 2012)	IET Software	Public	Classification
	(Cao, Qin, & Feng, 2012)	Advanced Science Letters	Public	Classification
2013	(Park et al., 2013)	Information Sciences	Public	Classification
	(Dejaeger, Verbraken, & Baesens, 2013)	IEEE Transactions on Software Engineering	Public	Classification
	(Shepperd, Song, Sun, & Mair, 2013)	IEEE Transactions on Software Engineering	Public	Dataset Analysis
	(Wang and Yao 2013)	IEEE Transactions on Reliability	Public	Classification
	(Peters, Menzies, Gong, & Zhang, 2013)	IEEE Transactions on Software Engineering	Public	Dataset Analysis
	(Radjenović et al., 2013)	Information and Software Technology	Public	Dataset Analysis

RQ4: Software Defect Datasets



Distribution of Software Defect Datasets

- The use of public data sets makes the research repeatable, refutable, and verifiable (Catal & Diri 2009a)
- Since 2005 more public datasets were used
- NASA MDP repository have been developed in 2005 and researchers started to be aware regarding the use of public datasets



NASA MDP Dataset

Dataset	Project Description	Language	Number of Modules	Number of *fp* Modules	Faulty Percentage
CM1	Spacecraft instrument	C	505	48	12.21%
KC1	Storage management for ground data	C++	1571	319	15.51%
KC3	Storage management for ground data	Java	458	42	18%
MC2	Video guidance system	C	127	44	34.65%
MW1	Zero gravity experiment related to combustion	C	403	31	10.23%
PC1	Flight software from an earth orbiting satellite	C	1059	76	8.04%
PC2	Dynamic simulator for attitude control systems	C	4505	23	1.01%
PC3	Flight software for earth orbiting satellite	C	1511	160	12.44%
PC4	Flight software for earth orbiting satellite	C	1347	178	12.72%

Code Attributes		Symbols	Description
LOC counts	LOC_total		The total number of lines for a given module
	LOC_blank		The number of blank lines in a module
	LOC_code_and_comment	NCSLOC	The number of lines which contain both code and comment in a module
	LOC_comments		The number of lines of comments in a module
	LOC_executable		The number of lines of executable code for a module
	number_of_lines		Number of lines in a module
Halstead	content	μ	The halstead length content of a module $\mu = \mu_1 + \mu_2$
	difficulty	D	The halstead difficulty metric of a module D = 1/L
	effort	E	The halstead effort metric of a module E = V/L
	error_est	B	The halstead error estimate metric of a module $B = E^{2/3}/1000$
	length	N	The halstead length metric of a module $N = N_1+N_2$
	level	L	The halstead level metric of a module $L = (2^* \mu_2)/ \mu_1^*N_2$
	prog_time	T	The halstead programming time metric of a module T = E/18
	volume	V	The halstead volume metric of a module $V = N^*\log_2(\mu_1+ \mu_2)$
	num_operands	N_1	The number of operands contained in a module
	num_operators	N_2	The number of operators contained in a module
	num_unique_operands	μ_1	The number of unique operands contained in a module
	num_unique_operators	μ_2	The number of unique operators contained in a module
McCabe	cyclomatic_complexity	v(G)	The cyclomatic complexity of a module v(G) = e – n +2
	cyclomatic_density		v(G) / NCSLOC
	design_complexity	iv(G)	The design complexity of a module
	essential_complexity	ev(G)	The essential complexity of a module
Misc.	branch_count		Branch count metrics
	call_pairs		Number of calls to functions in a module
	condition_count		Number of conditionals in a given module
	decision_count		Number of decision points in a module
	decision_density		condition_count / decision_count
	edge_count		Number of edges found in a given module from one module to another
	essential_density		Essential density is calculated as: (ev(G)-1)/(v(G)-1)
	parameter_count		Number of parameters to a given module
	maintenance_severity		Maintenance Severity is calculated as: ev(G)/v(G)
	modified_condition_count		The effect of a condition affect a decision outcome by varying that condition only
	multiple_condition_count		Number of multiple conditions within a module
	global_data_complexity	gdv(G)	the ratio of cyclomatic complexity of a module's structure to its parameter count
	global_data_density		Global Data density is calculated as: gdv(G)/v(G)
	normalized_cyclo_cmplx		v(G) / numbe_of_lines
	percent_comments		Percentage of the code that is comments
	node_count		Number of nodes found in a given module

Code Attributes		NASA MDP Dataset								
		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
LOC counts	LOC_total	√	√	√	√	√	√	√	√	√
	LOC_blank	√	√	√	√	√	√		√	√
	LOC_code_and_comment	√	√	√	√	√	√	√	√	√
	LOC_comments	√	√	√	√	√	√	√	√	√
	LOC_executable	√	√	√	√	√	√	√	√	√
	number_of_lines	√		√	√	√	√	√	√	√
Halstead	content	√	√	√	√	√	√	√	√	√
	difficulty	√	√	√	√	√	√	√	√	√
	effort	√	√	√	√	√	√	√	√	√
	error_est	√	√	√	√	√	√	√	√	√
	length	√	√	√	√	√	√	√	√	√
	level	√	√	√	√	√	√	√	√	√
	prog_time	√	√	√	√	√	√	√	√	√
	volume	√	√	√	√	√	√	√	√	√
	num_operands	√	√	√	√	√	√	√	√	√
	num_operators	√	√	√	√	√	√	√	√	√
	num_unique_operands	√	√	√	√	√	√	√	√	√
	num_unique_operators	√	√	√	√	√	√	√	√	√
McCabe	cyclomatic_complexity	√	√	√	√	√	√	√	√	√
	cyclomatic_density	√		√	√	√	√	√	√	√
	design_complexity	√	√	√	√	√	√	√	√	√
	essential_complexity	√	√	√	√	√	√	√	√	√
Misc.	branch_count	√	√	√	√	√	√	√	√	√
	call_pairs	√		√	√	√	√	√	√	√
	condition_count	√		√	√	√	√	√	√	√
	decision_count	√		√	√	√	√	√	√	√
	decision_density	√		√	√	√	√	√	√	√
	edge_count	√		√	√	√	√	√	√	√
	essential_density	√		√	√	√	√	√	√	√
	parameter_count	√		√	√	√	√	√	√	√
	maintenance_severity	√		√	√	√	√	√	√	√
	modified_condition_count	√		√	√	√	√	√	√	√
	multiple_condition_count	√		√	√	√	√	√	√	√
	global_data_complexity			√	√					
	global_data_density			√	√					
	normalized_cyclo_complx	√		√	√	√	√	√	√	√
	percent_comments	√		√	√	√	√	√	√	√
	node_count	√		√	√	√	√	√	√	√
Programming Language		C	C++	Java	C	C	C	C	C	C
Number of Code Attributes		37	21	39	39	37	37	36	37	37
Number of Modules		344	2096	200	127	264	759	1585	1125	1399
Number of fp Modules		42	325	36	44	27	61	16	140	178

Code Attribute

```
1.  void main()
2.  {
3.          //This is a sample code

4.          //Declare variables
5.          int a, b, c;

6.          // Initialize variables
7.          a=2;
8.          b=5;

9.          //Find the sum and display c if greater
    than zero
10.         c=sum(a,b);
11.         if c < 0
12.           printf("%d\n", a);
13.         return;
14. }


15. int sum(int a, int b)
16. {
17.         // Returns the sum of two numbers
18.         return a+b;
19. }
```

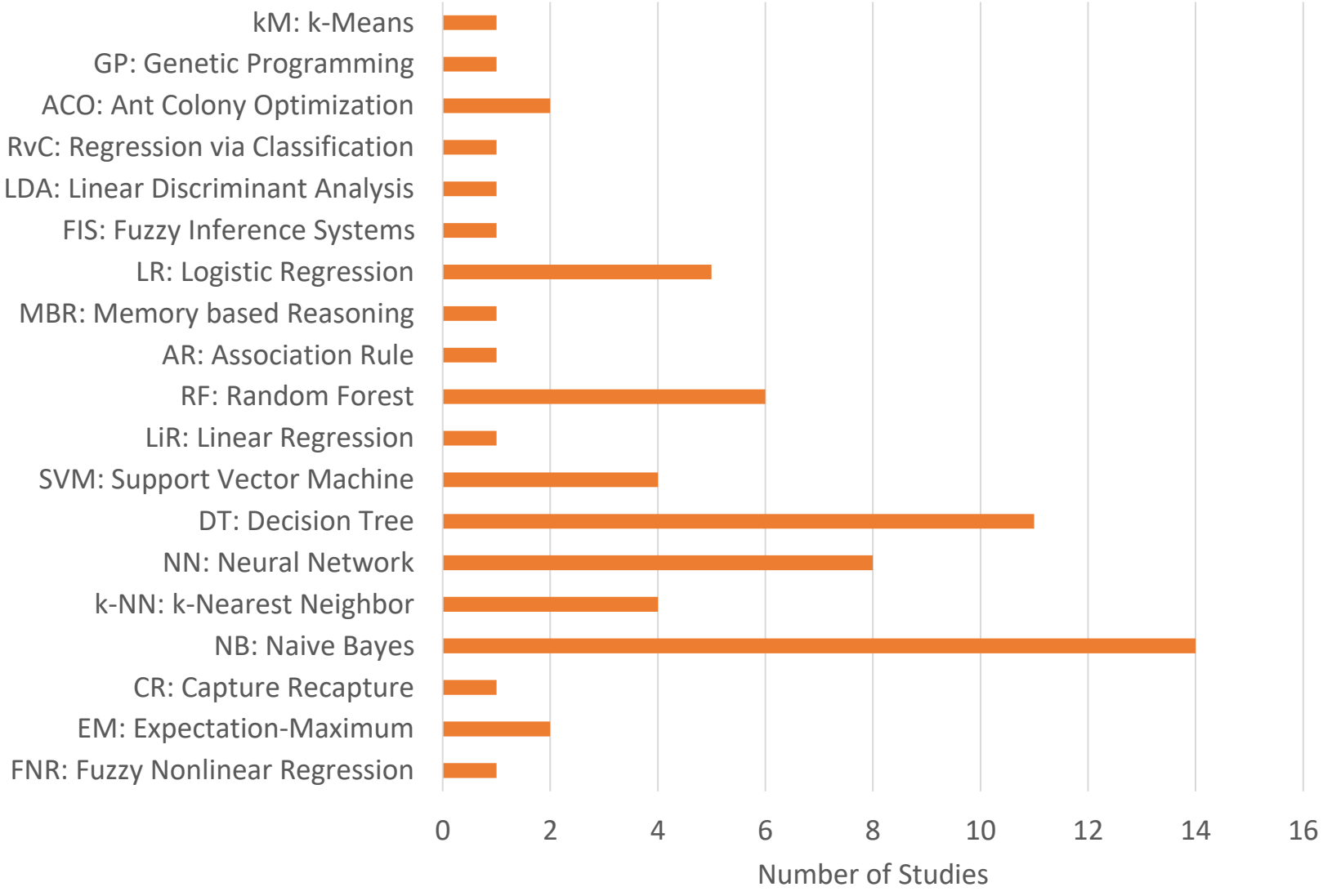
c > 0

c

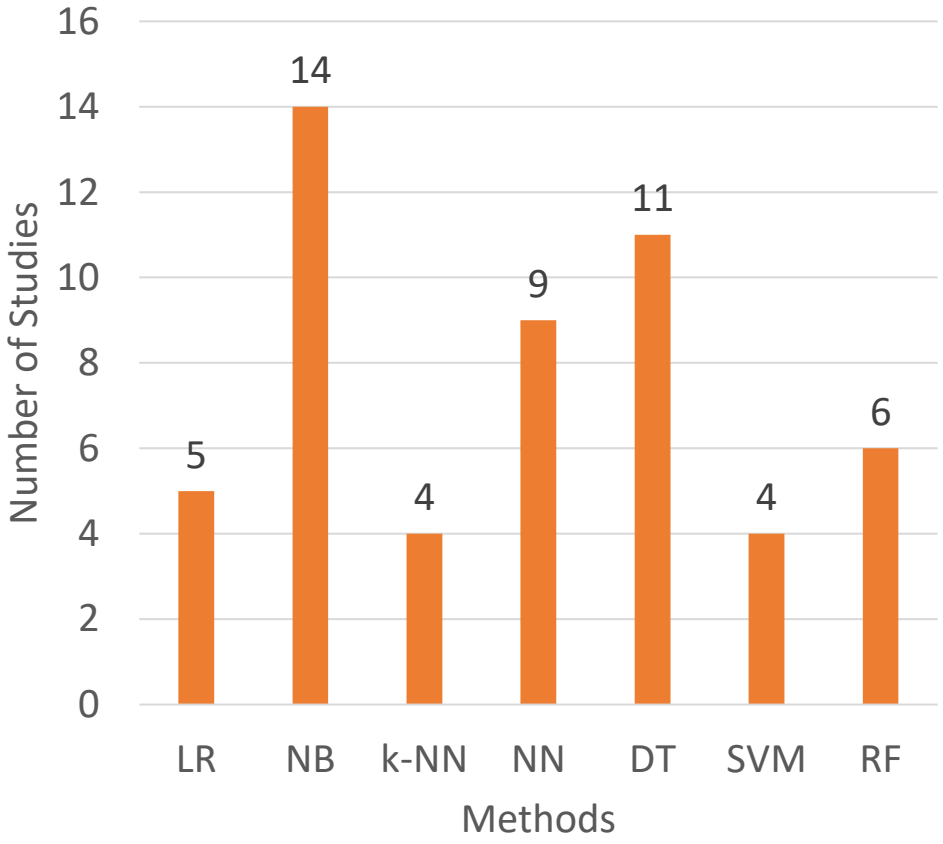
Module	LOC	LOCC	V	CC	**Error**
main()	16	4	5	2	**2**
sum()	5	1	3	1	**0**

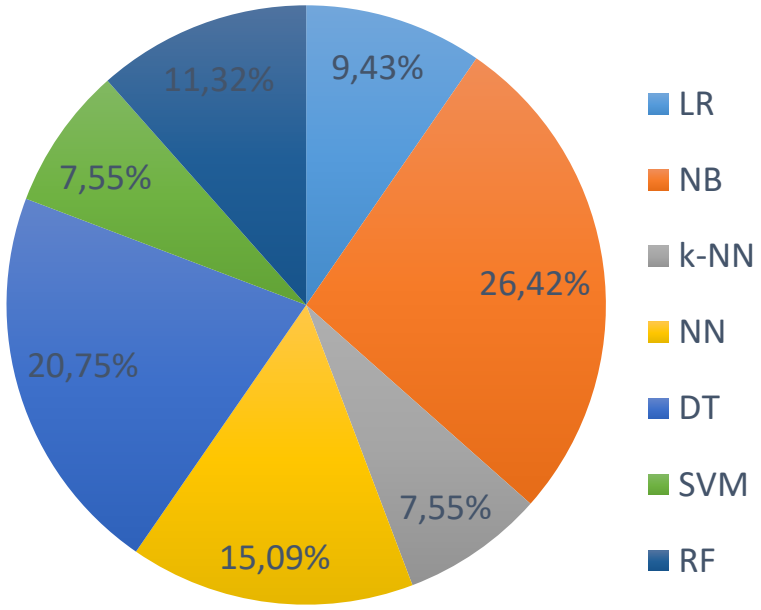
LOC: Line of Code
LOCC: Line of commented Code
V: Number of unique operands&operators
CC: Cyclometric Complexity

RQ5: Software Defect Prediction Methods



RQ6: Most Used Software Defect Prediction Methods





RQ7: Method Comparison Results

- The comparisons and benchmarking result of the defect prediction using machine learning classifiers indicate that:
  - ✓Poor accuracy level is dominant (Lessmann et al. 2008)
  - ✓No significant performance differences could be detected (Lessmann et al. 2008)
  - ✓No particular classifiers that performs the best for all the data sets (Song et al. 2011) (Hall et al. 2012)

- The accurate and reliable classification algorithms to build a better prediction model is an open issue in software defect prediction

RQ8: Method Improvement Efforts

- Researchers proposed some techniques for improving the accuracy of classifiers for software defect prediction
- Recent proposed techniques try to increase the prediction accuracy of a generated model:
  - ✓ By modifying and ensembling some machine learning methods (Mısırlı et al. 2011) (Tosun et al. 2008)
  - ✓ By using boosting algorithm (Zheng 2010) (Jiang et al. 2011)
  - ✓ by adding feature selection (Gayatri et al. 2010) (Khoshgoftaar & Gao, 2009) (Song et al. 2011)
  - ✓ By using parameter selection for some classifiers (Peng & Wang 2010) (Lin et al. 2008) (Guo et al. 2008)
- While considerable works have been done separately, limited research can be found on investigating them all together

RQ9: Existing Frameworks

Three frameworks have been highly cited and influential in software defect prediction field

Menzies Framework

(Menzies et al. 2007)

Lessmann Framework

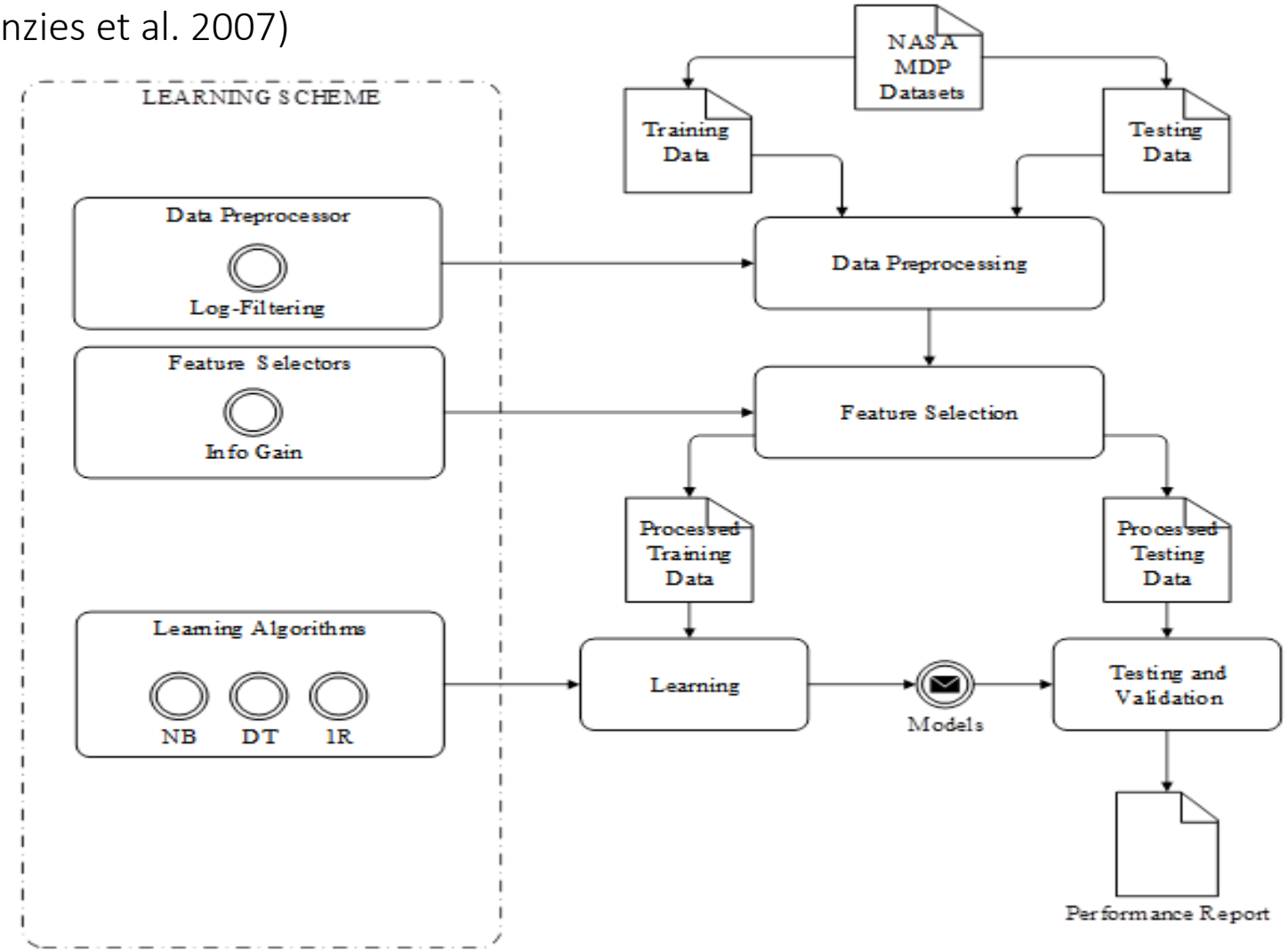
(Lessmann et al. 2008)

Song Framework

(Song et al. 2011)

Menzies Framework

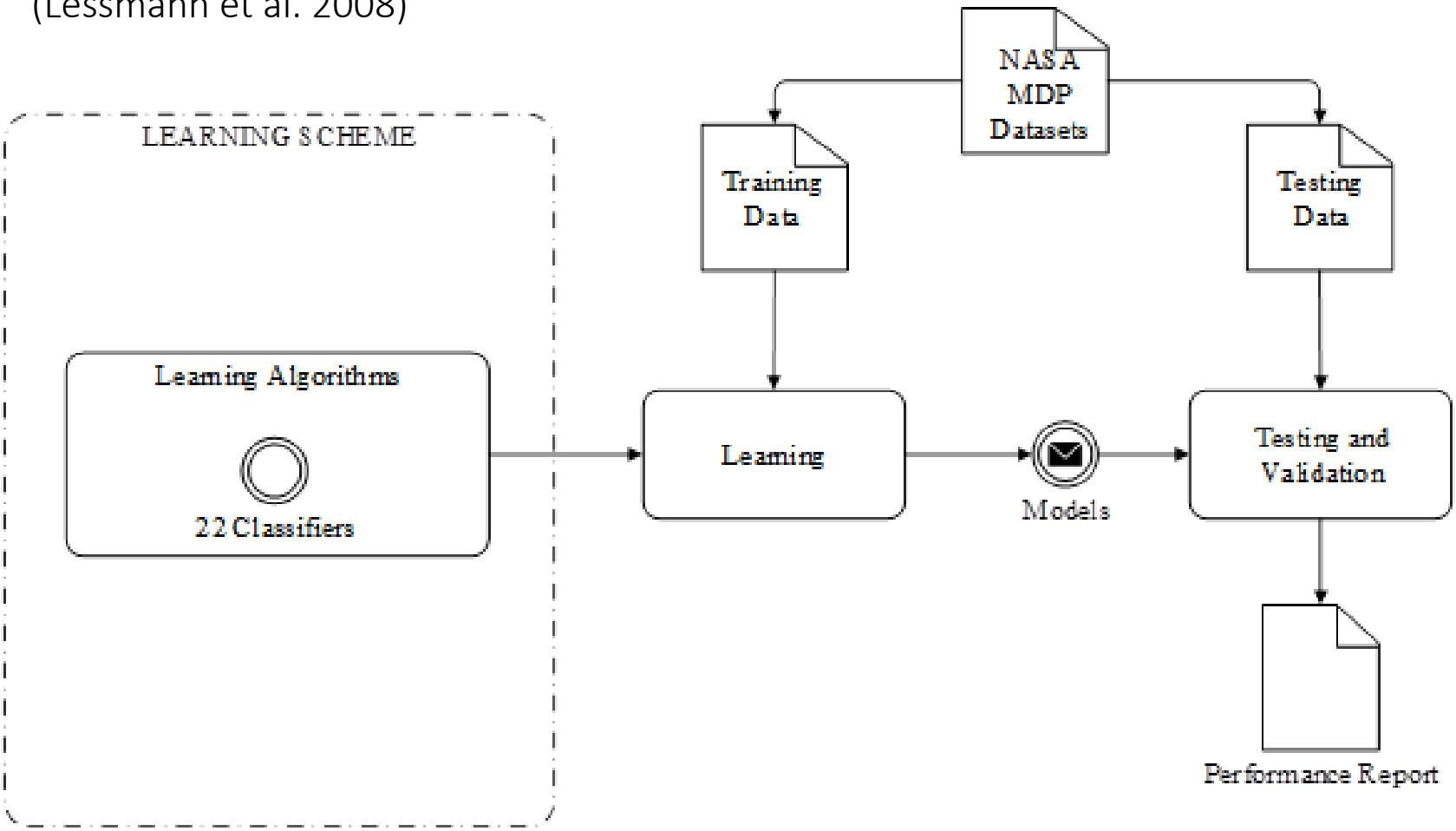
(Menzies et al. 2007)



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

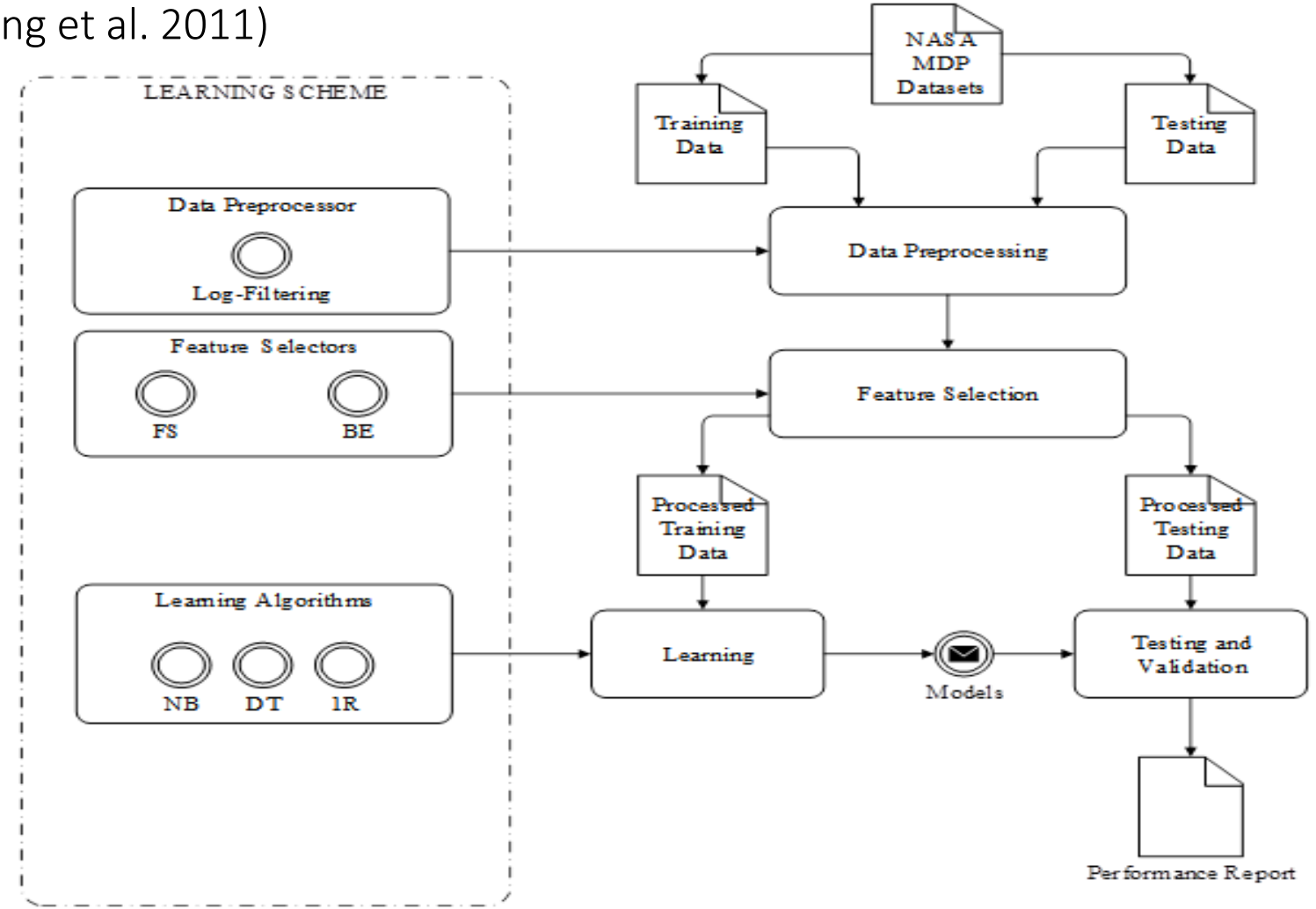
(Lessmann et al. 2008)



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)

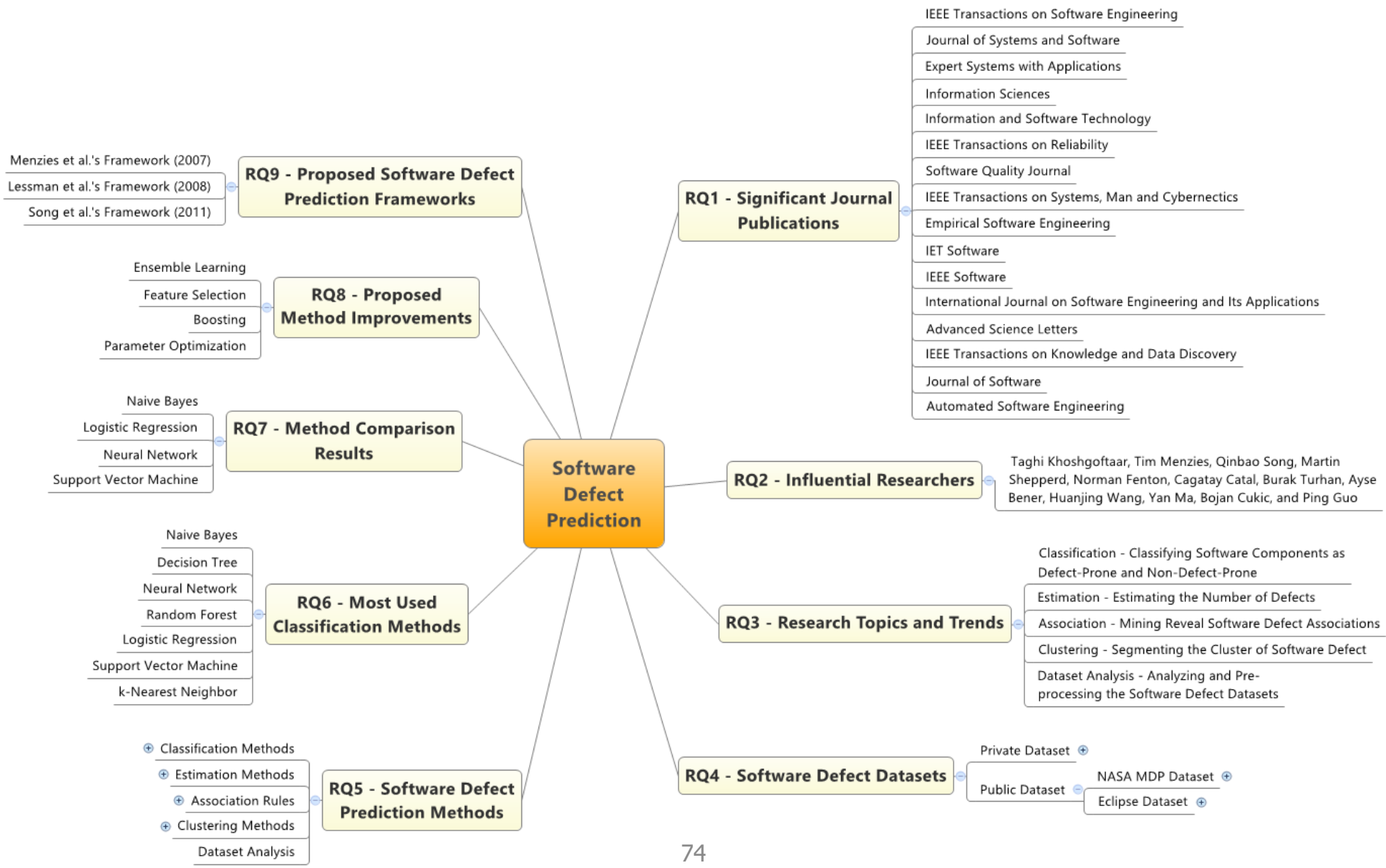
Song Framework

(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)

Mind Map of the SLR Results



SLR Melahirkan Research Gaps

Dari Hasil SLR, Kita Menemukan Research Gaps yang Menjadi Kandidat Masalah Penelitian yang Kita Angkat pada Penelitian Kita

Gap Analysis of Framework

1. The comparisons and benchmarking result of the defect prediction using machine learning classifiers indicate that:
   - Poor accuracy level is dominant (Lessmann et al. 2008)
   - No significant performance differences could be detected (Lessmann et al. 2008)
   - No particular classifiers that performs the best for all the data sets (Song et al. 2011) (Hall et al. 2012)
2. Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models
3. 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

Gap Research akan Menjadi Kandidat Masalah Penelitian

- Masalah penelitian adalah alasan utama mengapa penelitian harus dilakukan
- Reviewer jurnal internasional menjadikan “masalah penelitian“sebagai parameter utama proses review
- Masalah penelitian harus objective (tidak subjective), dan harus dibuktikan secara logis dan valid bahwa masalah itu benar-benar masalah
- Supaya logis dan valid, perlu dilakukan objektifikasi masalah, dengan cara melandasi masalah penelitian dengan literature terbaru
- Dimana munculnya di paper:
  - Abstract
  - Introduction

Alur Terbentuknya 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:

Penerapan XYZ untuk Masalah E pada SVM untuk Prediksi Nilai Tukar Uang

- 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 uang?
  - 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:

Penerapan XYZ untuk E pada Fuzzy Logic untuk Pengaturan Lampu Lalu Lintas Dinamis

- 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[X]emacetan lalu lintas?
  - Bagaimana mendes[X]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

Akademisi vs Technopreneur

Meja Indah

Meja Kuat

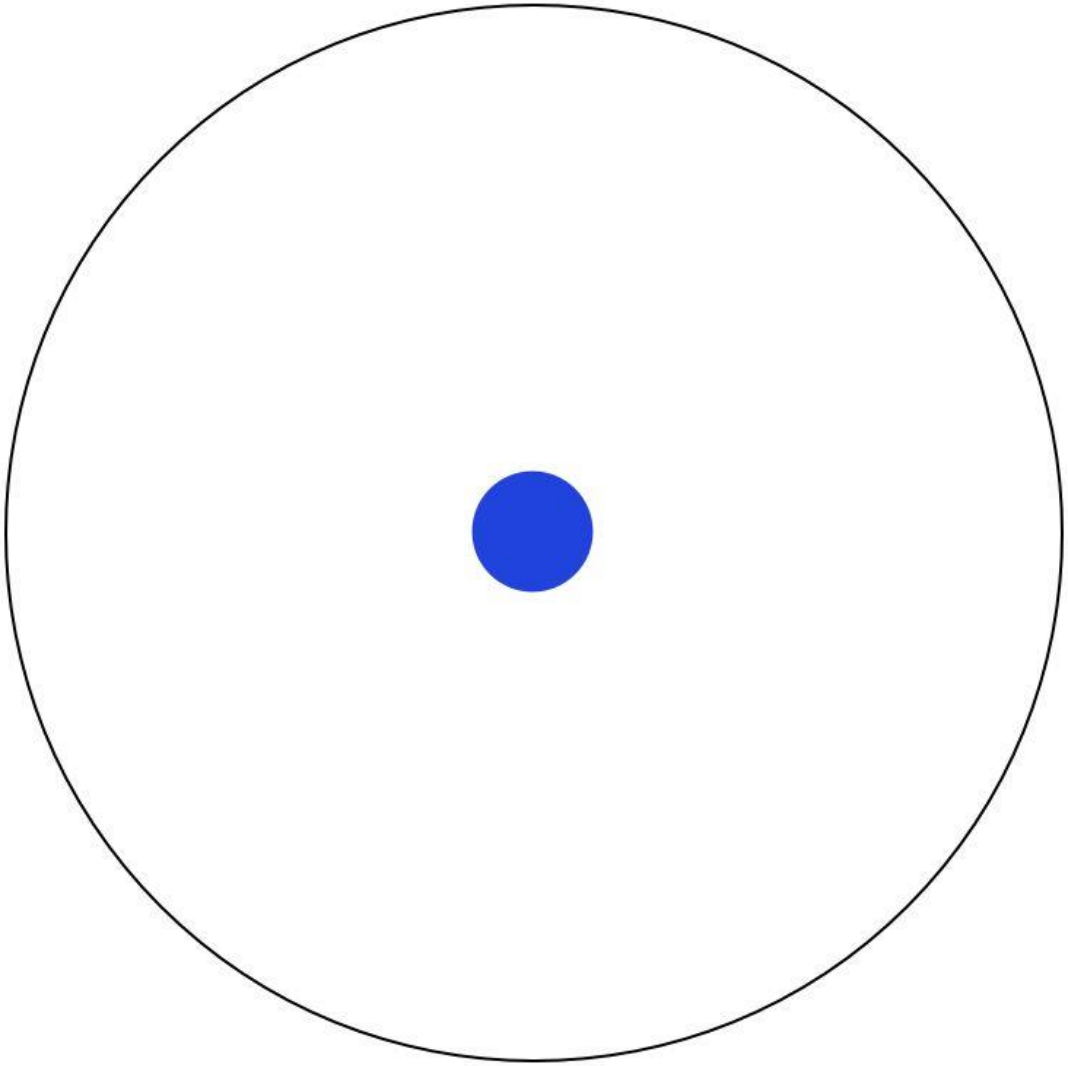
Meja Luas

- 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

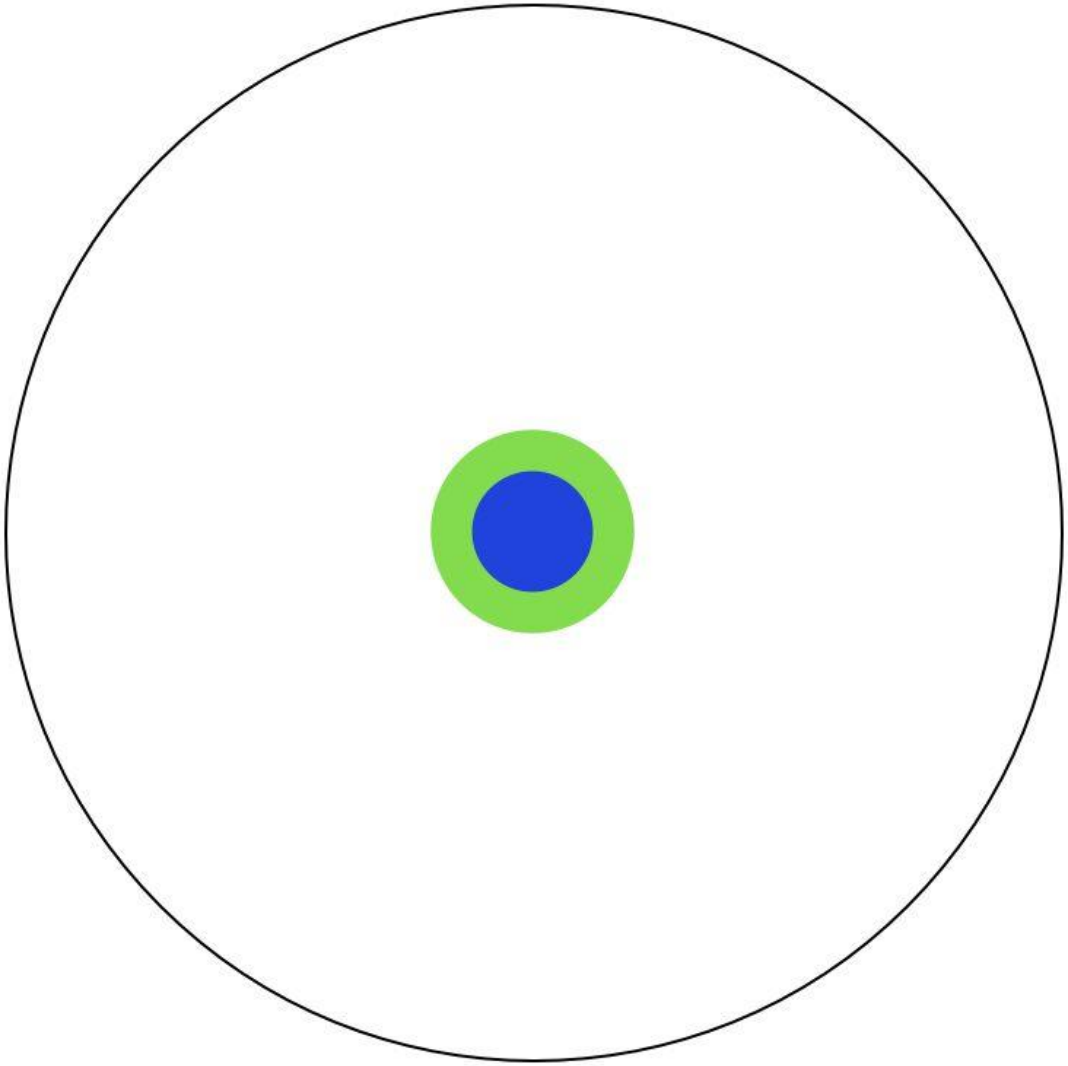
Penelitian yang Berkualitas Tinggi

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

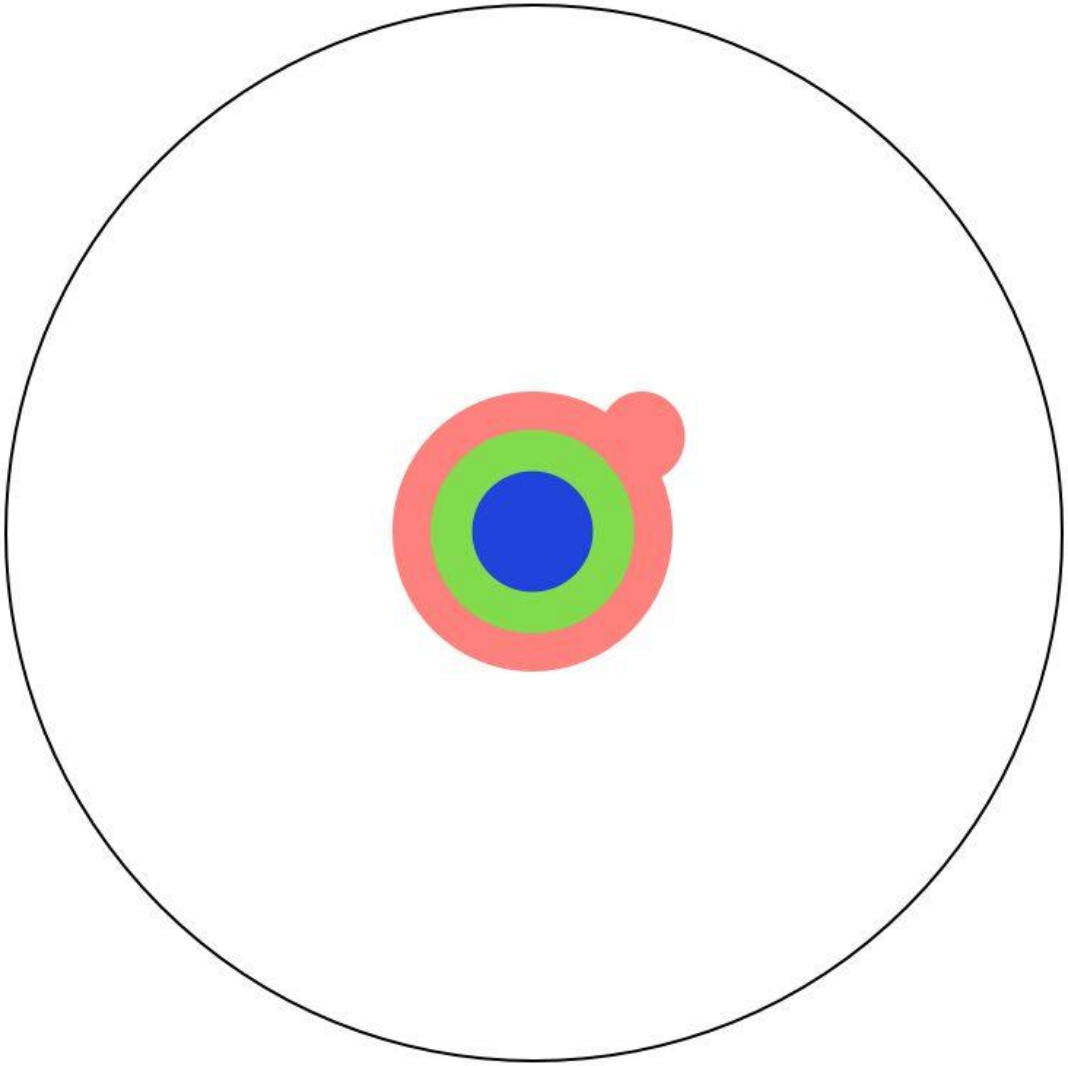
The Illustrated Guide to a Ph.D (Might, 2010)



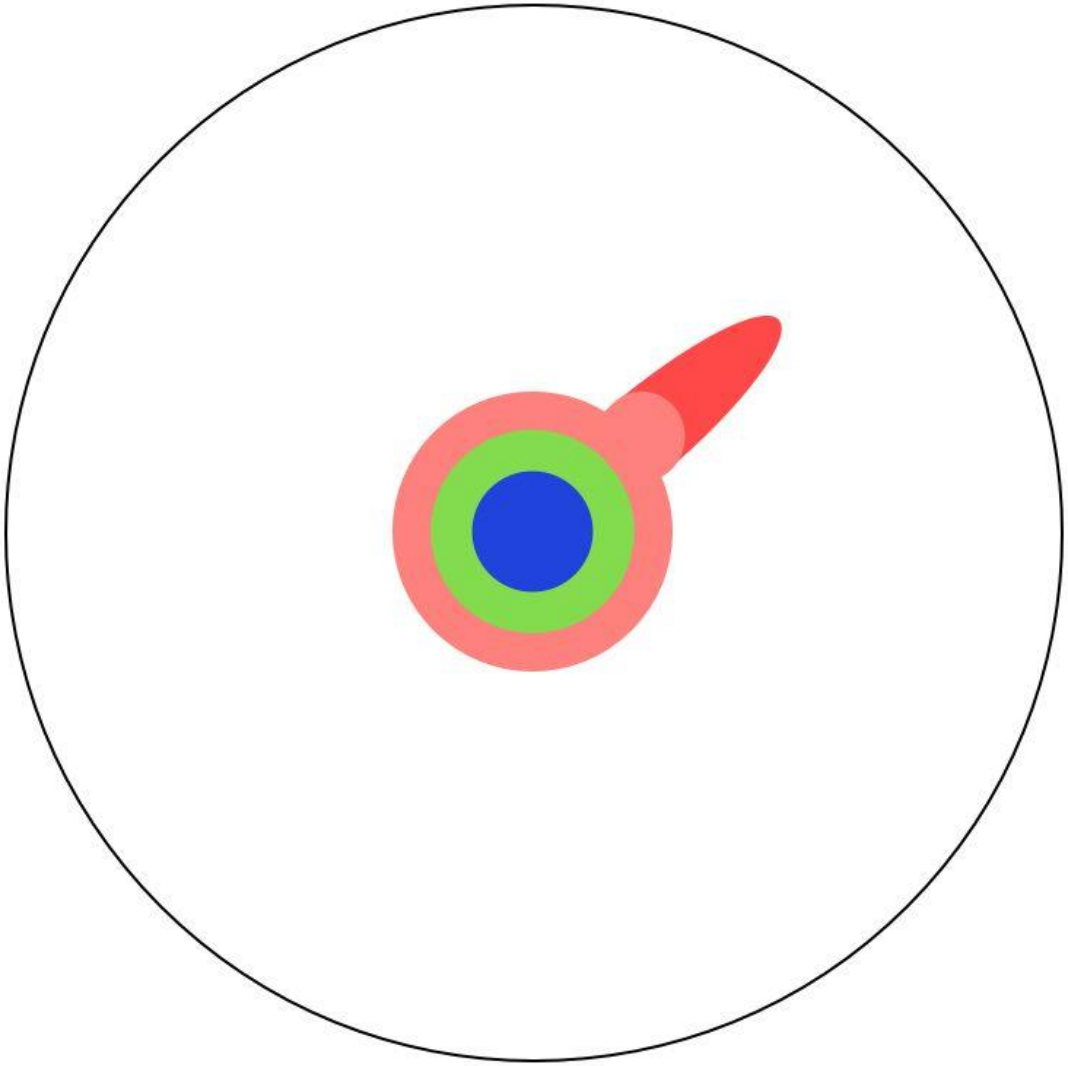
The Illustrated Guide to a Ph.D (Might, 2010)



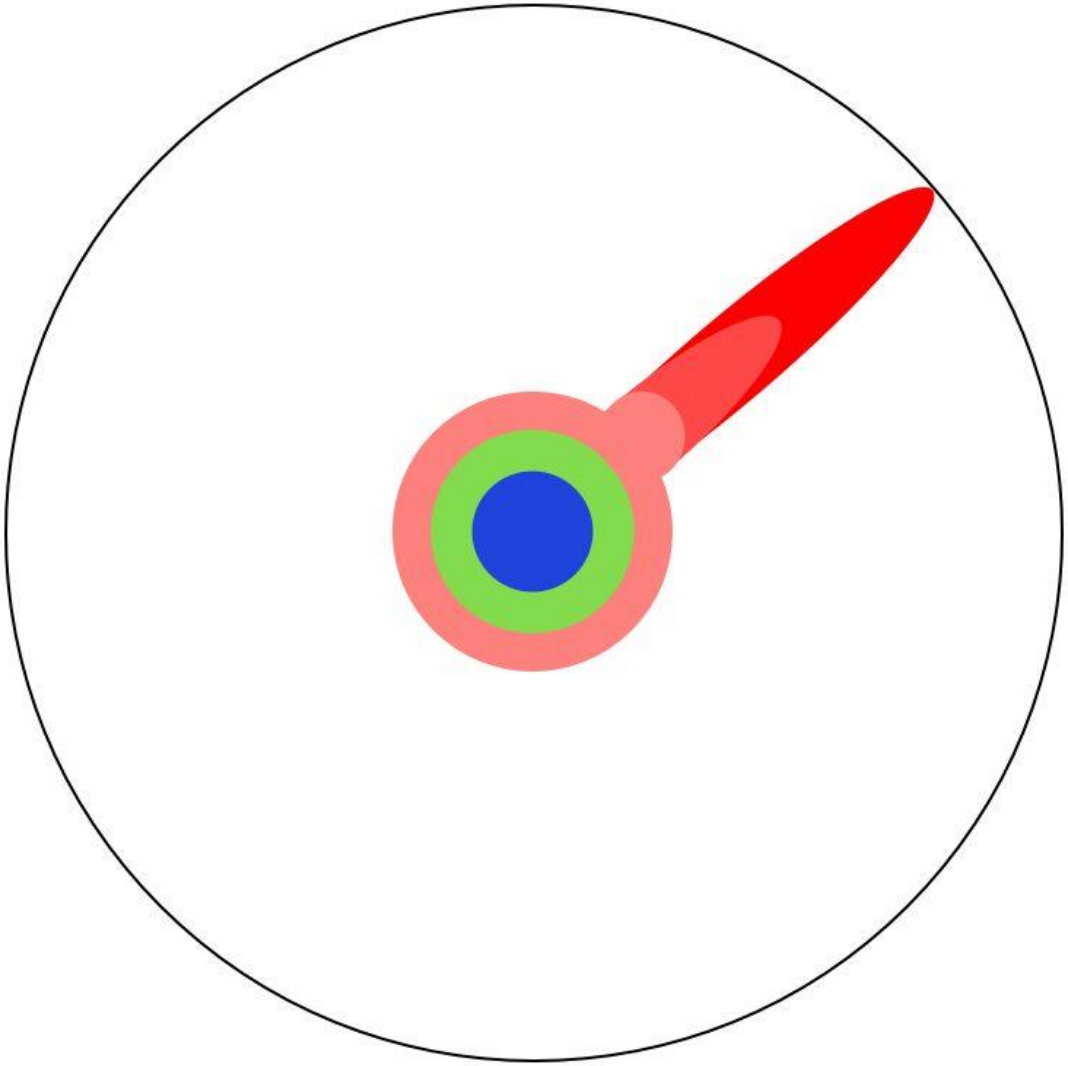
The Illustrated Guide to a Ph.D (Might, 2010)



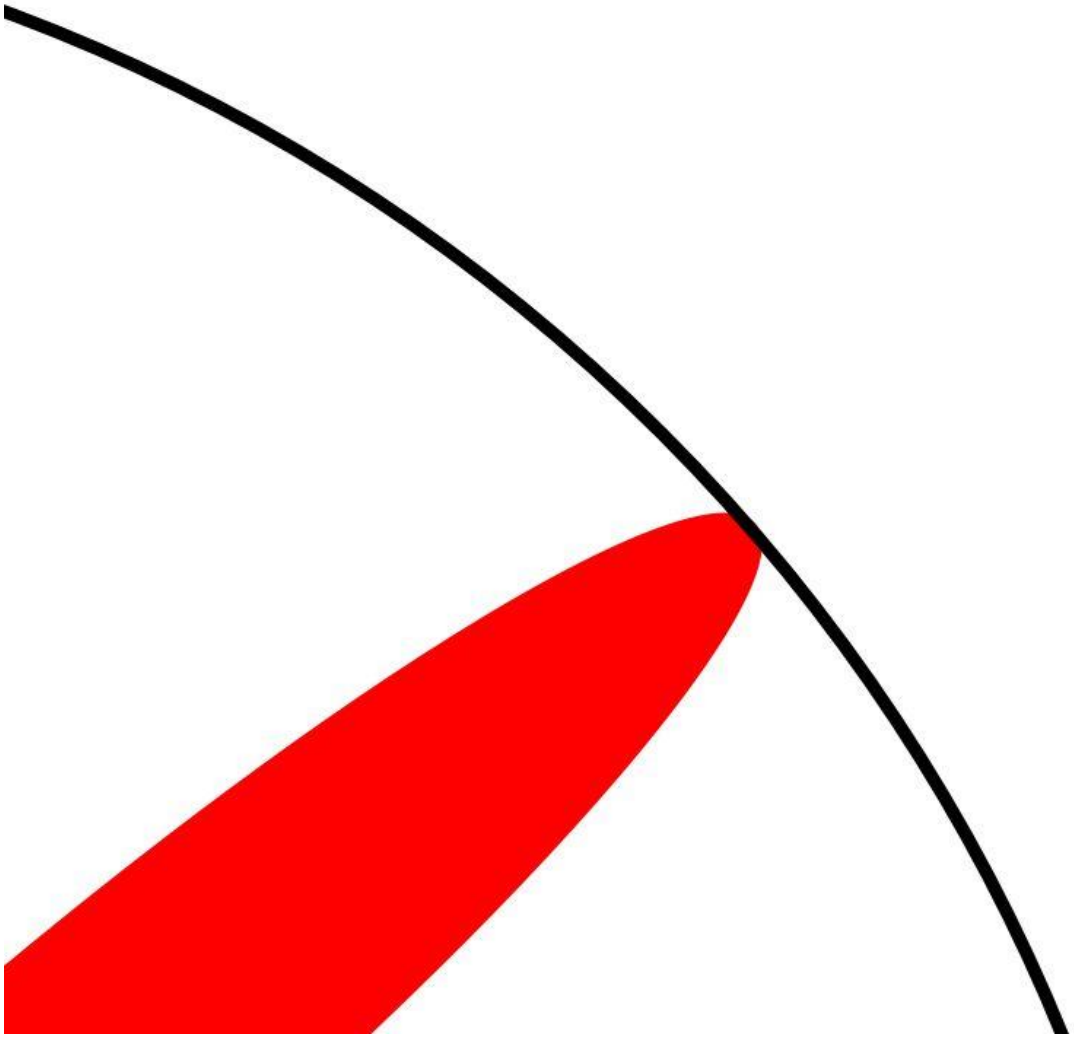
The Illustrated Guide to a Ph.D (Might, 2010)



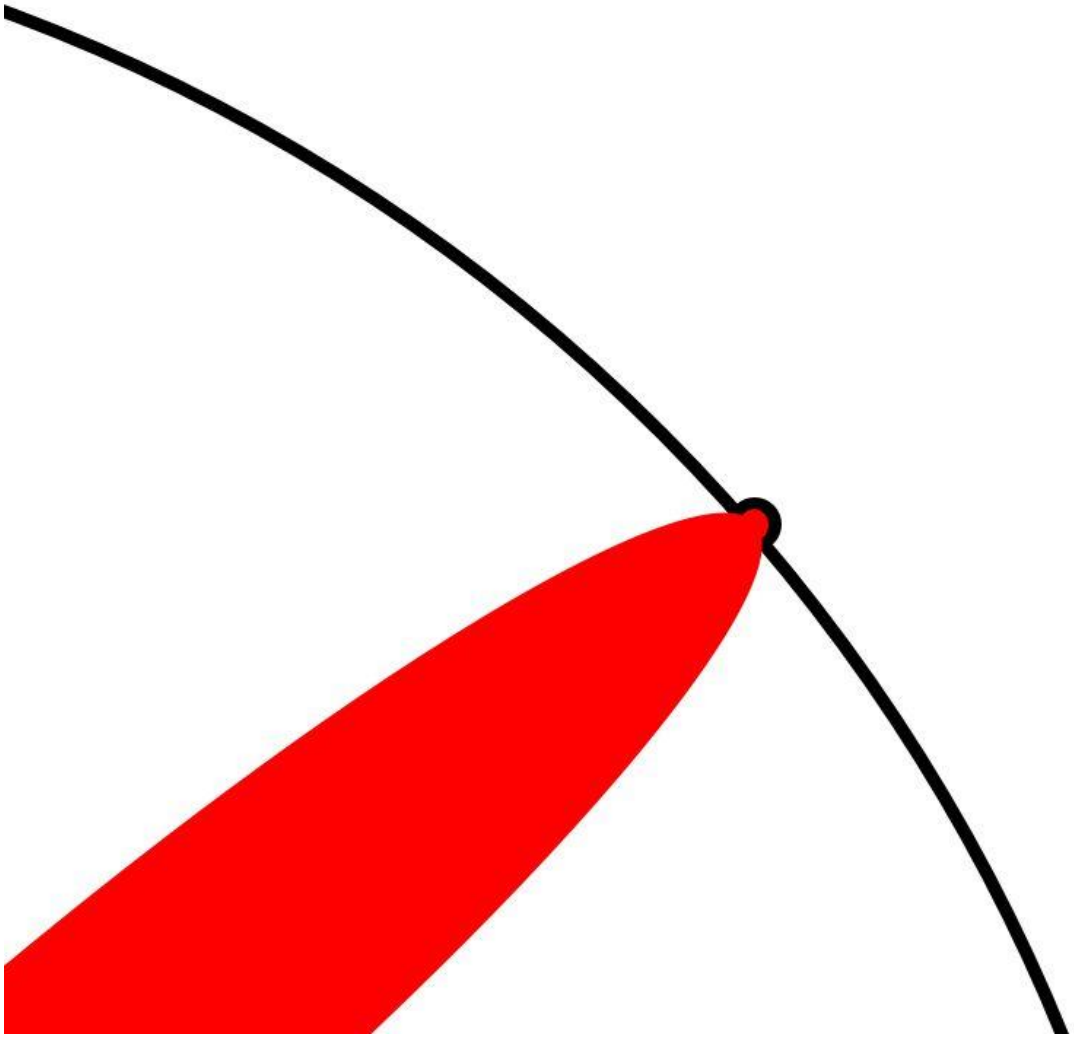
The Illustrated Guide to a Ph.D (Might, 2010)



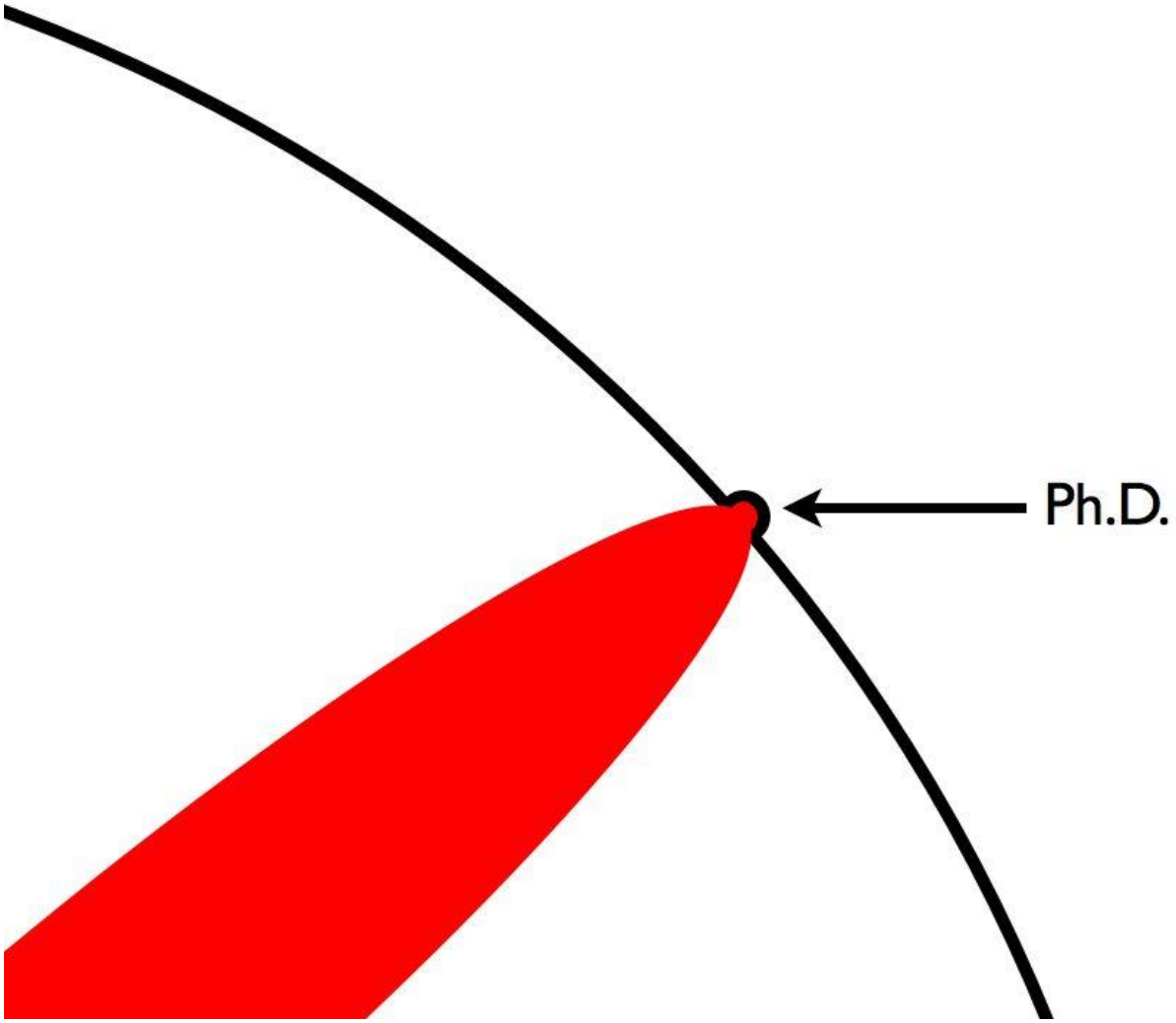
The Illustrated Guide to a Ph.D (Might, 2010)



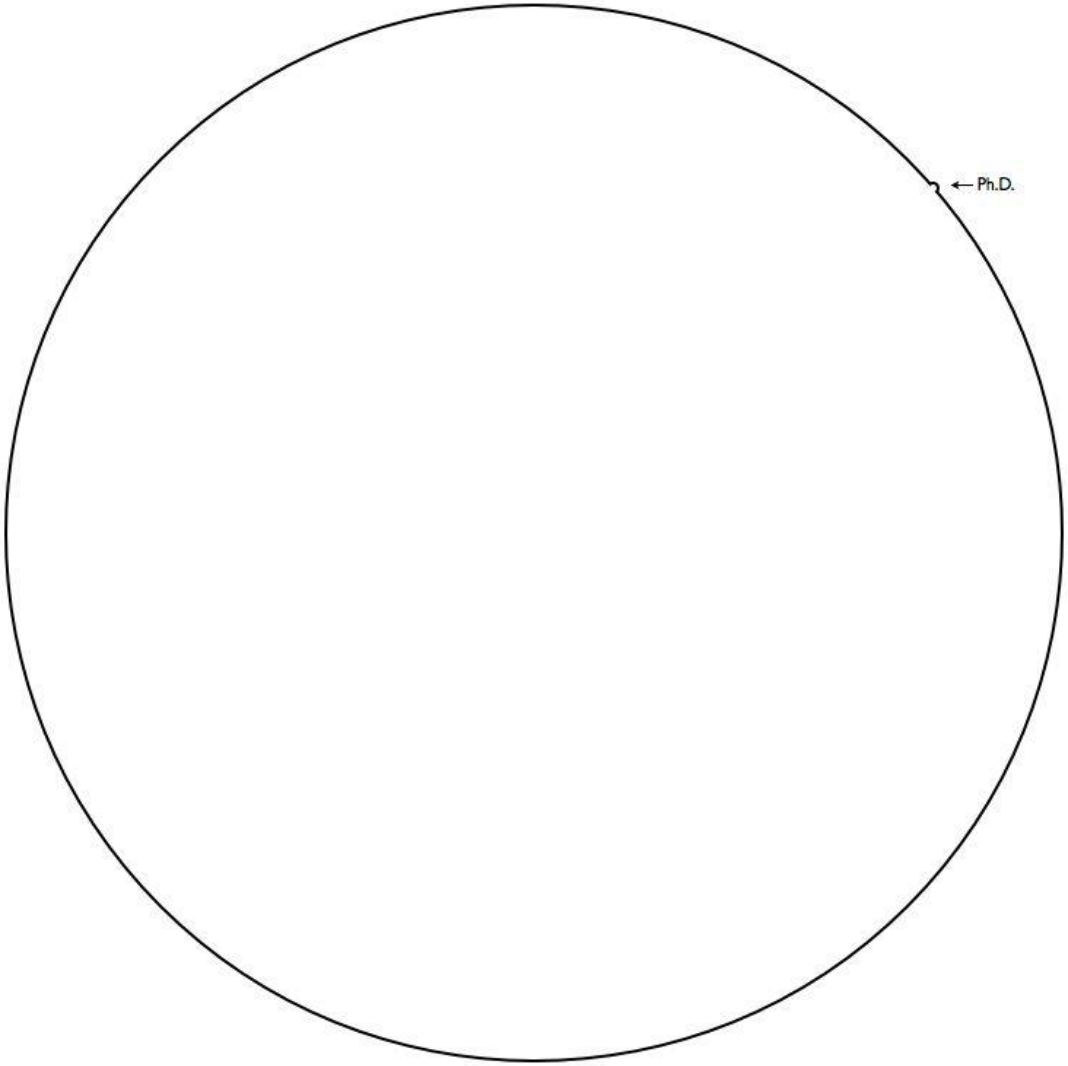
The Illustrated Guide to a Ph.D (Might, 2010)



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



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

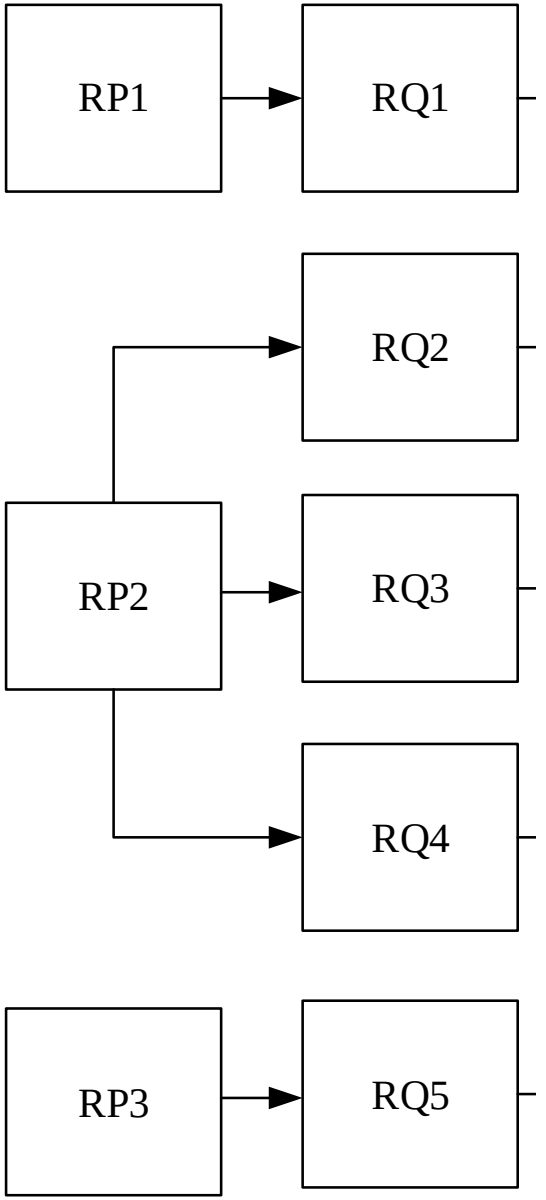


Research Problems (RP)

RP1 While many studies on software defect prediction report the comparative performance of the classification algorithms used, but there is no strong consensus on which classifiers perform best when individual studies are looked separately

RP2 Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models

RP3 Neural network has strong fault tolerance and strong ability of nonlinear dynamic processing of software fault data, but practicability of neural network is limited due to difficulty of selecting appropriate parameters



Masalah Penelitian dan Landasannya

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)*

Research Questions 1 (RQ1)

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP1	While many studies on software defect prediction report the comparative performance of the modelling techniques they have used, no clear consensus on which classifier perform best emerges when individual studies are looked at separately	RQ1	**Which machine learning classification algorithms perform best when used in software defect prediction?**	RO1	To identify and determine the best machine learning classification algorithms when used in software defect prediction

Research Questions 2-4 (RQ2-RQ4)

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP2	Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models	RQ2	**How does the integration between genetic algorithm based feature selection and bagging technique affect the accuracy of software defect prediction?**	RO2	To develop a hybrid genetic algorithm based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ3	**How does the integration between particle swarm optimization based feature selection and bagging technique affect the accuracy of software defect prediction?**	RO3	To develop a hybrid particle swarm optimization based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ4	**Which metaheuristic optimization techniques perform best when used in feature selection of software defect prediction?**	RO4	To identify the best metaheuristic optimization techniques when used in feature selection of software defect prediction

Research Questions 5 (RQ5)

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP3	Neural network has strong fault tolerance and strong ability of nonlinear dynamic processing of software fault data, but practicability of neural network is limited due to difficulty of selecting appropriate parameters	RQ5	**How does the integration between genetic algorithm based neural network parameter selection and bagging technique affect the accuracy of software defect prediction?**	RO5	To develop a hybrid genetic algorithm based neural network parameter selection and bagging technique for improving the accuracy of software defect prediction

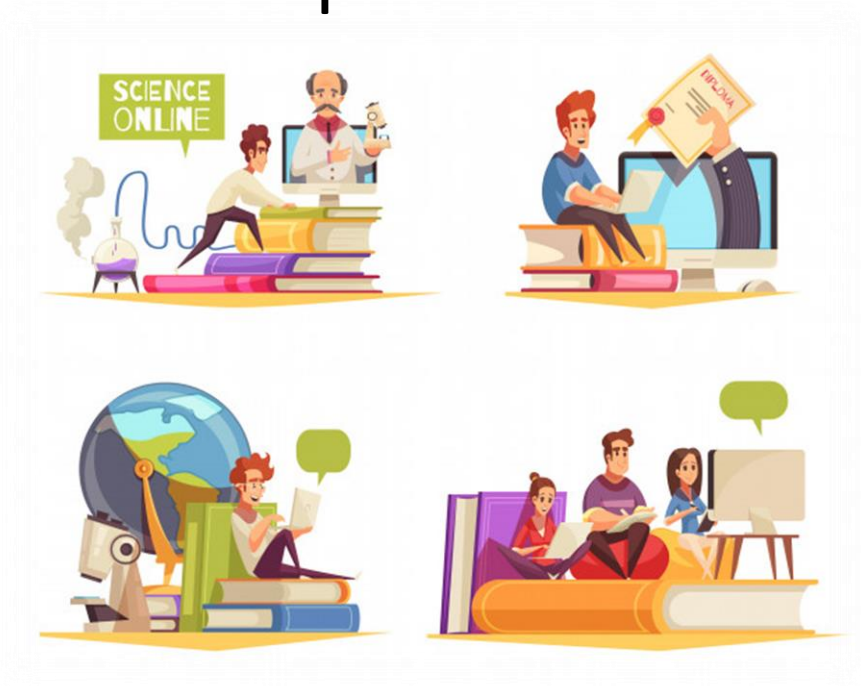
Proposal Penelitian S3

1. Dari research gaps yang ditemukan ketika menyusun SLR, tentukan RP-RQ-RO, narasikan secara komprehensif dalam bentuk latar belakang penelitian, dan jadilah itu bagian pertama dari proposal
2. Bagian kedua proposal adalah SLR yang sudah kita susun

Proposal yang komprehensif berisi point 1 dan 2 (kandidat Bab 1 dan 2 disertasi), beserta pengalaman publikasi di journal terindeks, akan menjadi proposal yang sempurna ketika kita publikasikan di ujian masuk program S3

KIAT 3

Masuk Program S3, Target Semester 1 Rapikan dan Konversi SLR menjadi Paper untuk Publikasi, Ikuti Perkuliahan dengan Rajin, dan Jalin Komunikasi Cerdas dan Intensif dengan Supervisor

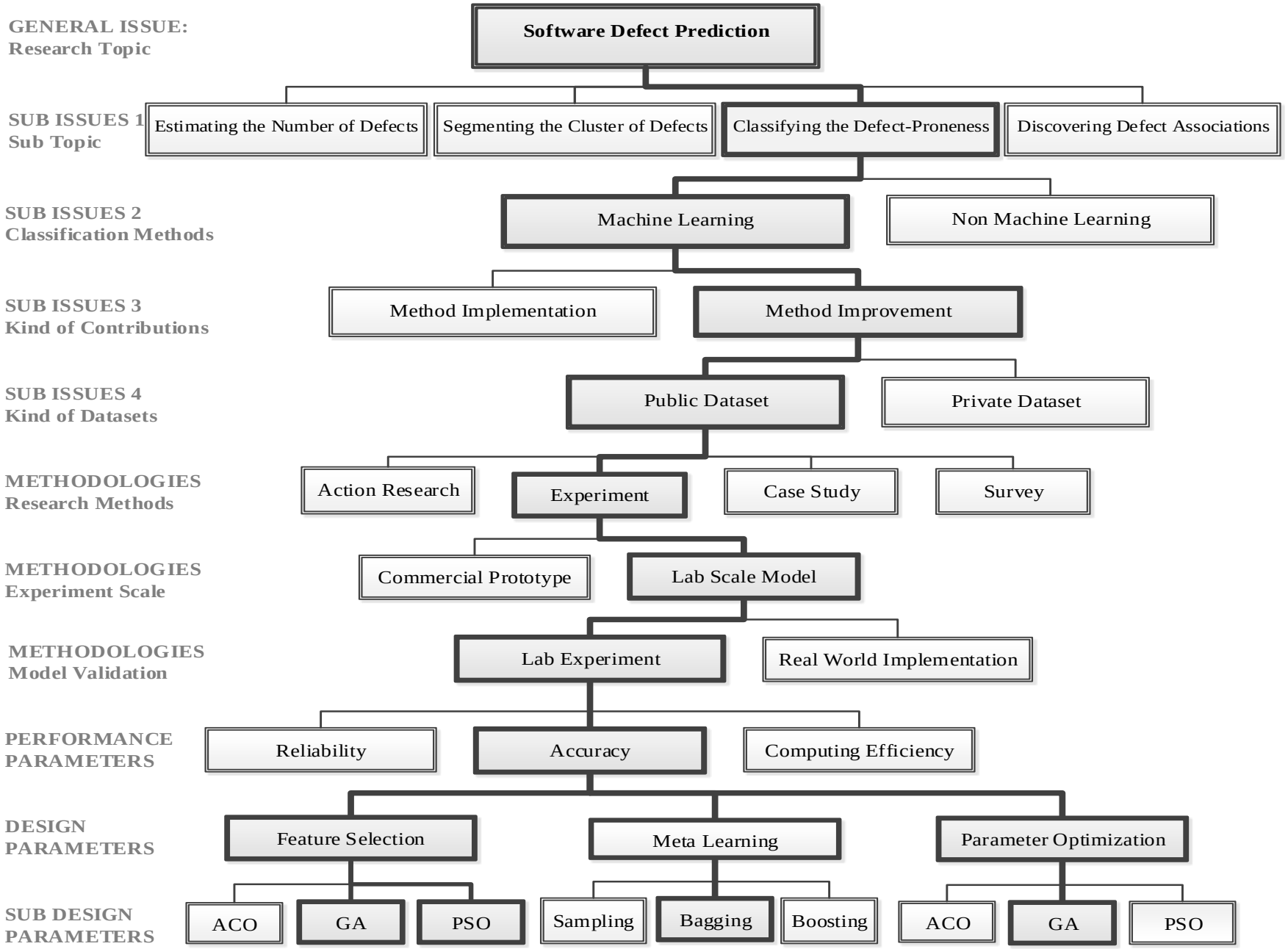


Target Penting Semester 1 Program S3

- Target Semester 1 program Ph.D adalah memperbaiki SLR supaya bisa menjadi paper yang layak untuk disubmit ke journal
  - Dan supervisor harus diyakinkan bahwa kita bisa bergerak cepat, karena sudah kita siapkan semua sebelum masuk program S3
- Ikuti mata kuliah wajib dengan baik, lakukan pendekatan dan perkenalan dengan supervisor dan dosen-dosen yang mengajar kita
  - Aktif di kelas, tunjukkan bahwa kita datang ke kelas dengan "kepala penuh isi", bukan kepala kosong yang siap dicekoki apapun oleh dosen
- Please note that, kebenaran yang diajarkan oleh dosen pada program graduate (pasca sarjana), dan yang ditulis di paper-paper itu relatif terhadap ruang dan waktu, bukan kebenaran mutlak
  - Konsep pada penelitian adalah APAPUN BOLEH KITA LAKUKAN ASAL ADA LANDASAN
  - Pertanyaan paling bodoh dari mahasiswa adalah, "Apakah ini boleh jadi topik atau masalah penelitian?"
- Perkuat metodologi penelitian, banyak baca buku metodologi penelitian supaya kita bisa berargumentasi dengan solid, logis dan sistematis
- Persiapkan infrastruktur untuk eksperimen lebih intensif di semester 2
- Pertajam proposal dan positioning dari penelitian

SCOPE OF STUDY

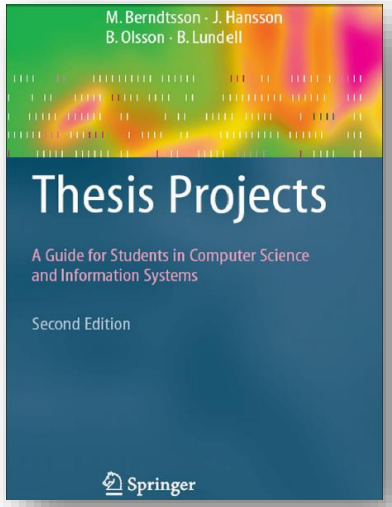
Software Defect Prediction Framework based on Hybrid Metaheuristic Optimization Methods



Perkuat Konsep dan Metodologi Penelitan

- 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)



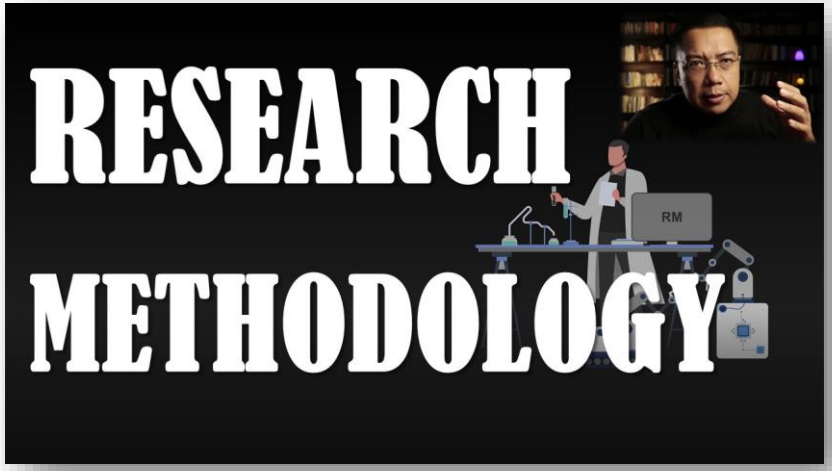
- The process of exploring the unknown, studying and learning new things, building new knowledge about things that no one has understood before
  (Berndtsson et al., 2008)

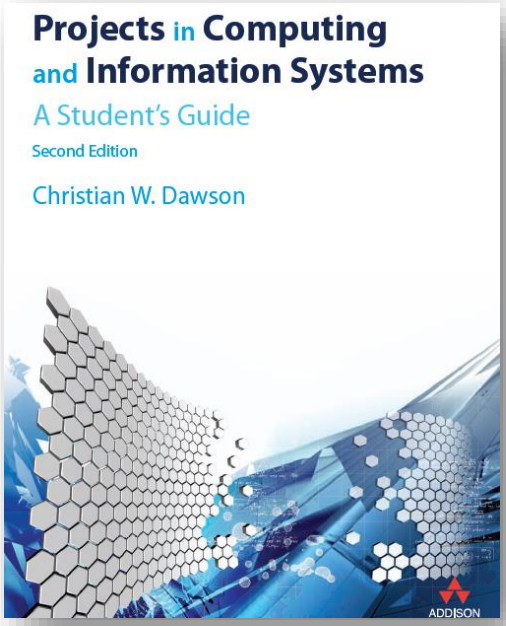
Pahami Apa Yang Dikejar di Penelitian?

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)



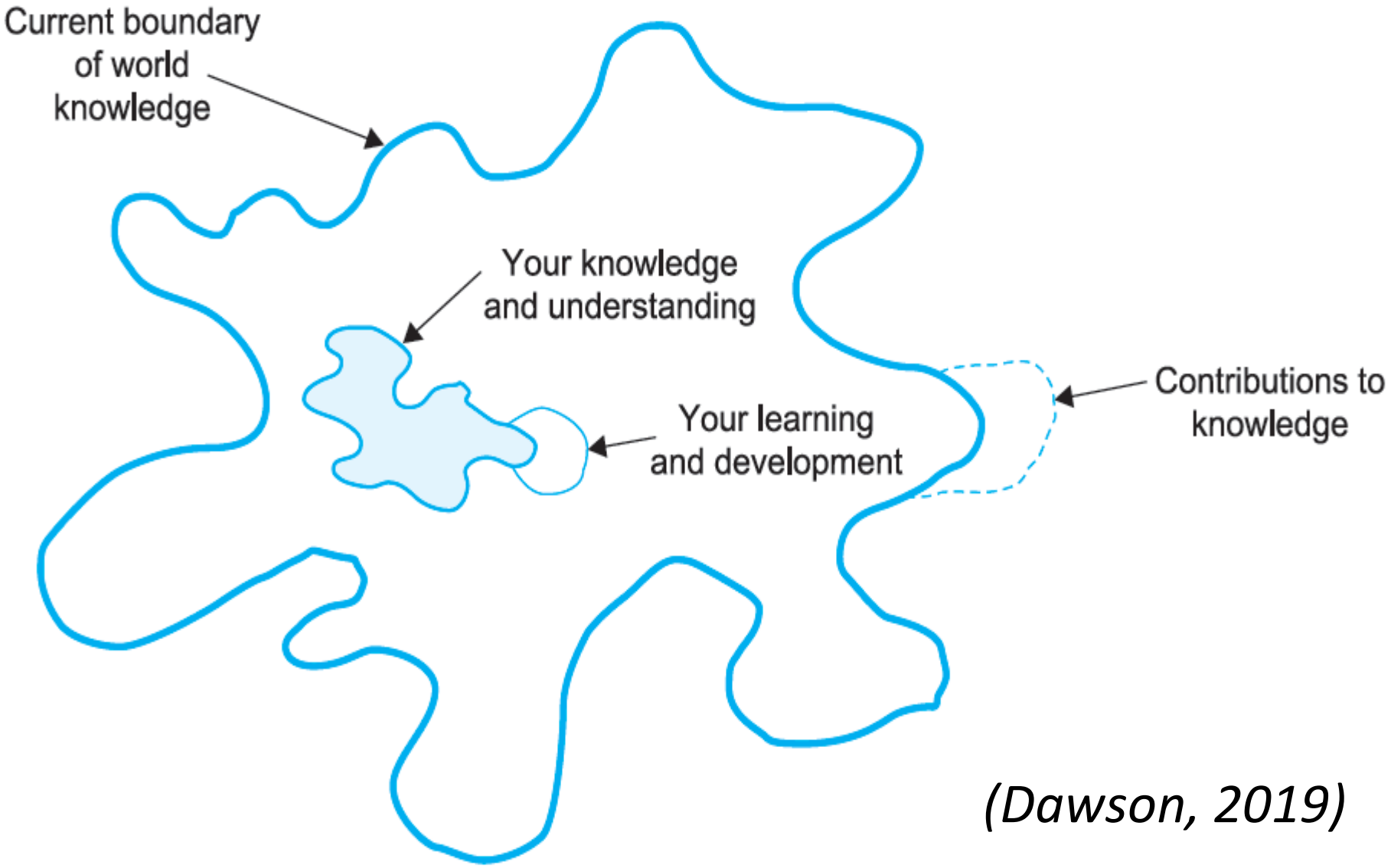


Pahami Bentuk Kontribusi ke Pengetahuan

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

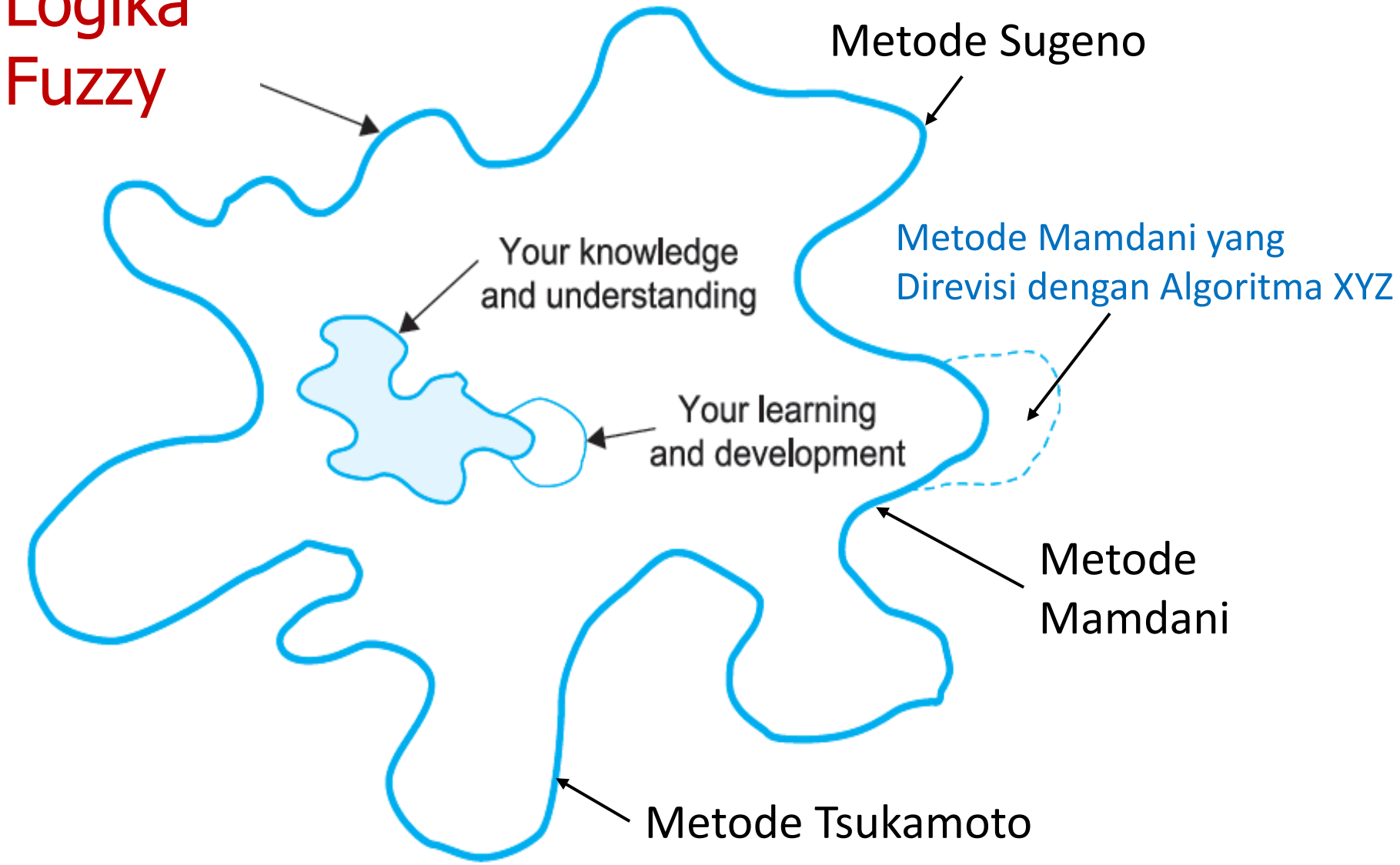
(Berndtsson et al., 2008)

Bentuk Kontribusi ke Pengetahuan

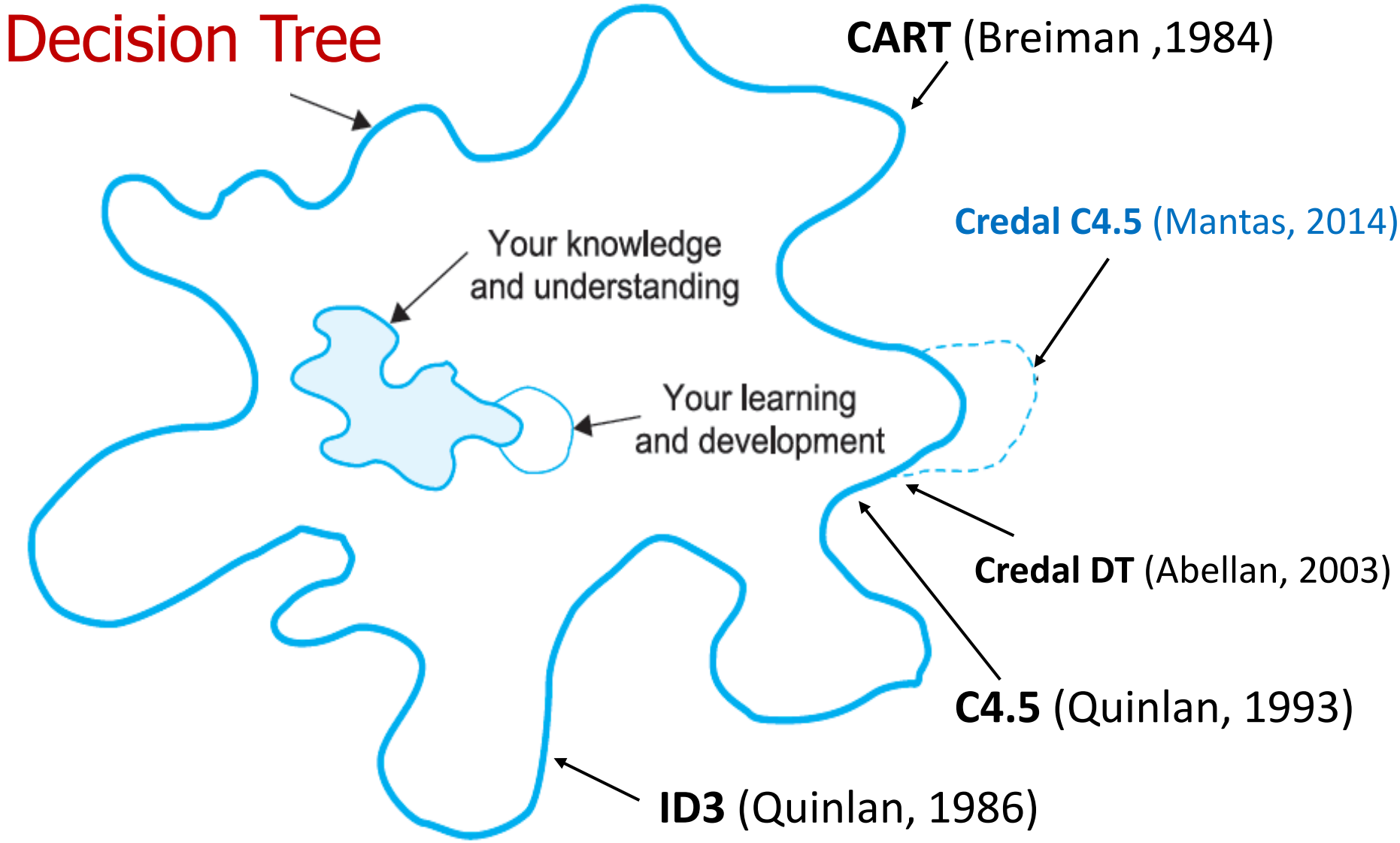


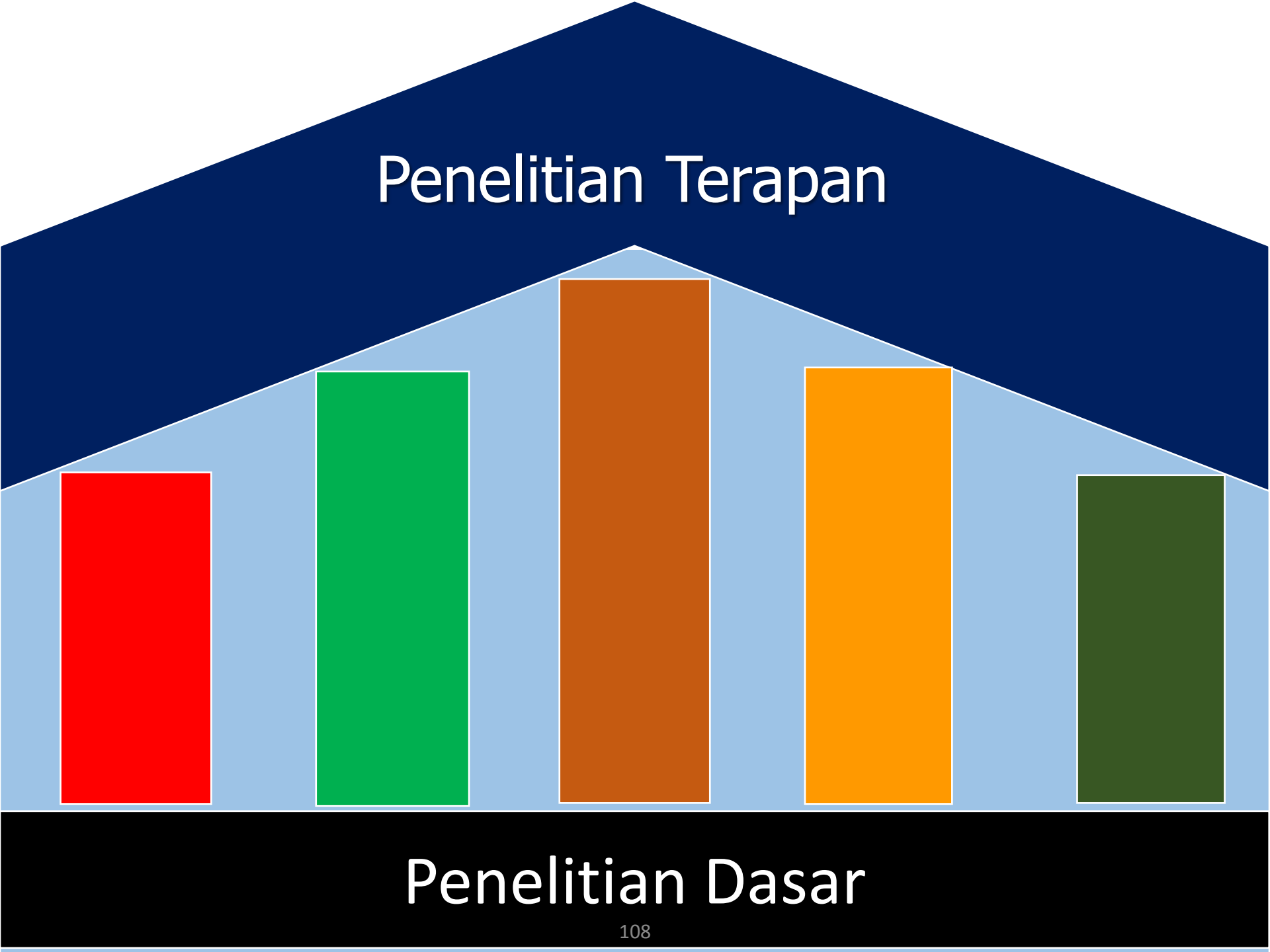
(Dawson, 2019)

Bentuk Kontribusi ke Pengetahuan

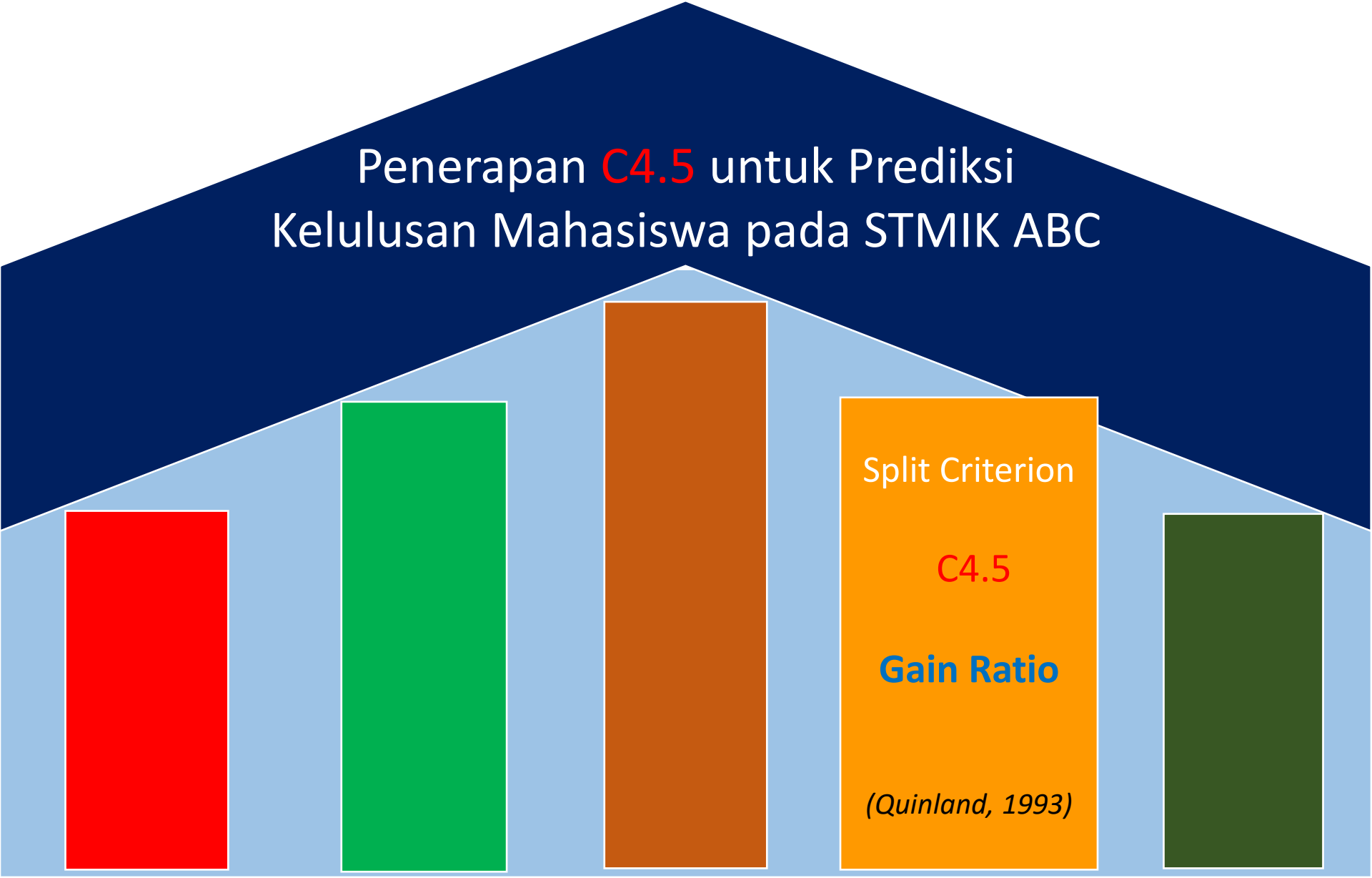


Bentuk Kontribusi ke Pengetahuan

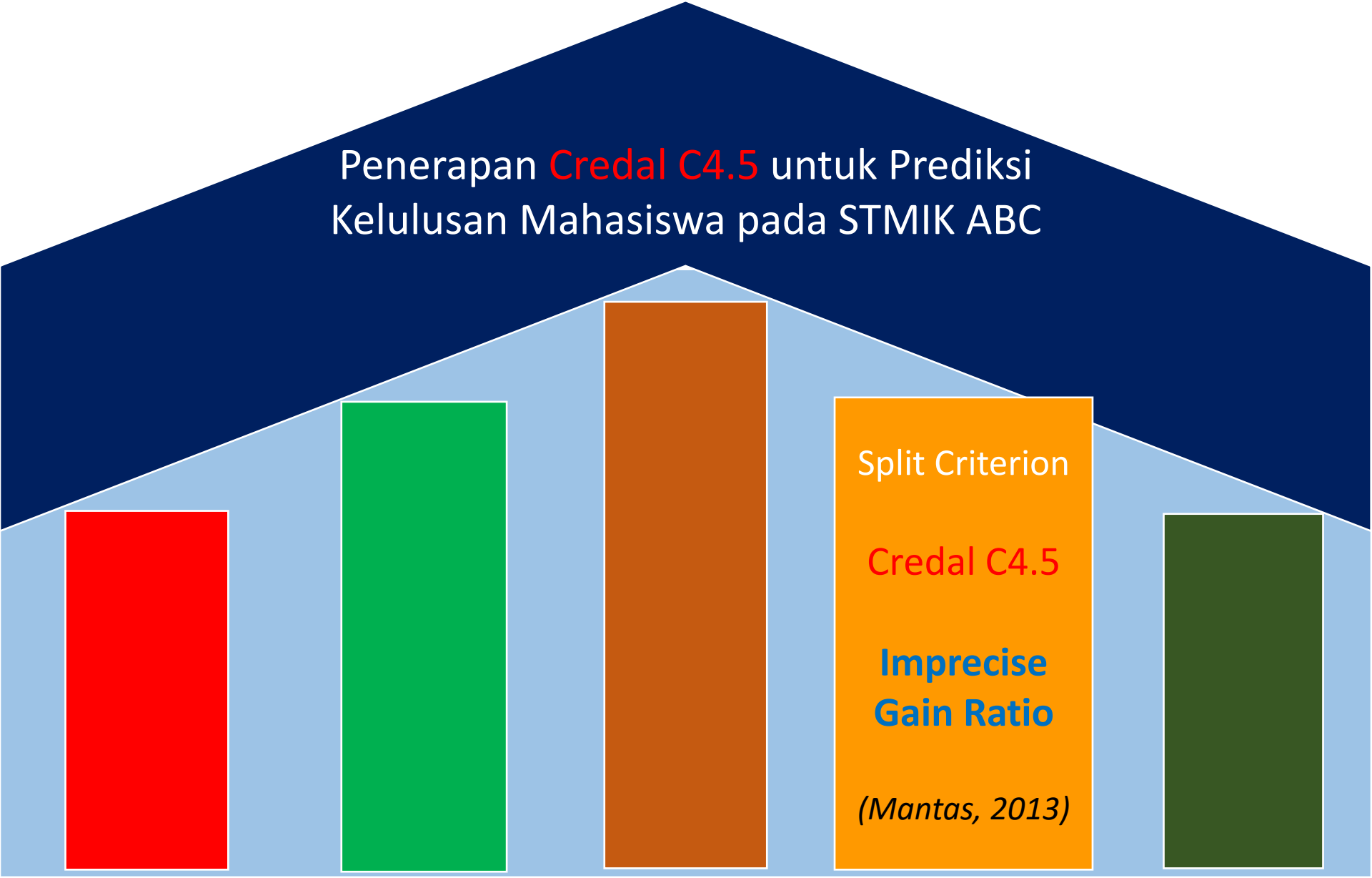



Penelitian Terapan
Penelitian Dasar

Penerapan C4.5 untuk Prediksi Kelulusan Mahasiswa pada STMIK ABC



Teori Gain *(Kullback & Leibler, 1951)*



Imprecise Probability Theory *(Walley, 1996)*

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Memperbaiki C4.5

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

Contents lists available at ScienceDirect

Information and Software Technology

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

Memperbaiki Use Case Points

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 Genetic Algorithms

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),

KIAT 4

Eksekusi Proposal dengan Tenang di Semester 2, Mulai dari RQ1-RQ2, Lakukan Eksperimen dengan Target Publikasi ke Journal Q3 atau Q4, Lakukan Perbaikan Eksperimen dan Paper berdasarkan Hasil Review Submission Paper

Target Penting Semester 2 Program S3

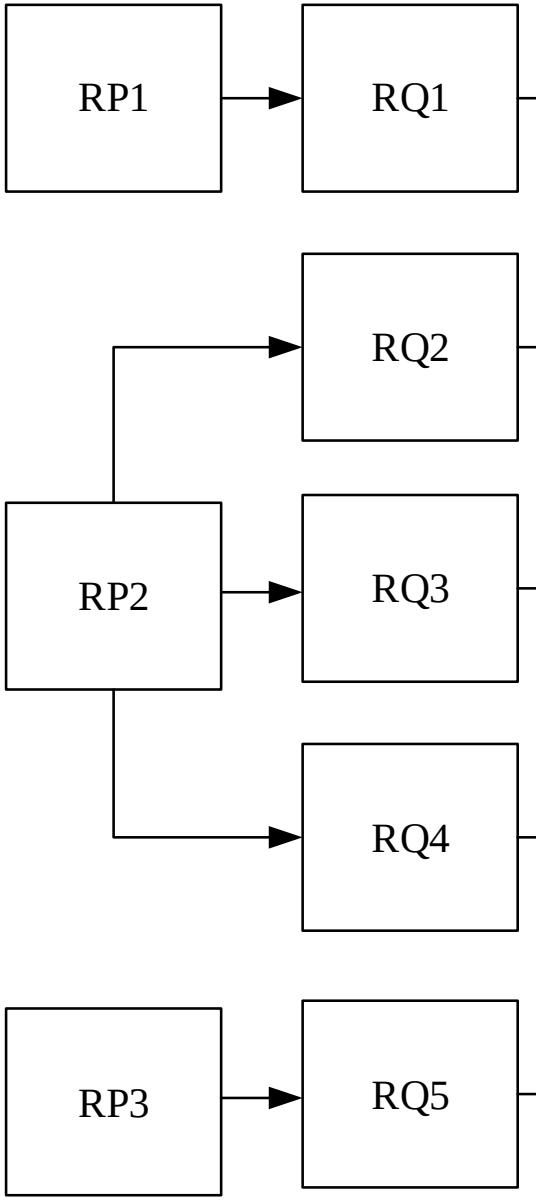
- Target Semester 2 program Ph.D adalah adanya hasil eksperimen dan draft untuk technical papernya, meskipun tingkat kontribusi ke pengetahuan belum signifikan
- Apabila semester 2 belum ada sama sekali eksperimen ataupun draft paper untuk publikasi, maknanya bahwa tidak akan mungkin S3 bisa lulus dalam 3 tahun
- Ketika ada rejection dari journal tempat kita submit, segera pelajari hasil review, lakukan perbaikan dan kirimkan kembali paper ke journal lain dengan Quartile lebih rendah
  - Journal-journal sudah kita listing levelnya pada SLR yang kita buat

Research Problems (RP)

RP1 While many studies on software defect prediction report the comparative performance of the classification algorithms used, but there is no strong consensus on which classifiers perform best when individual studies are looked separately

RP2 Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models

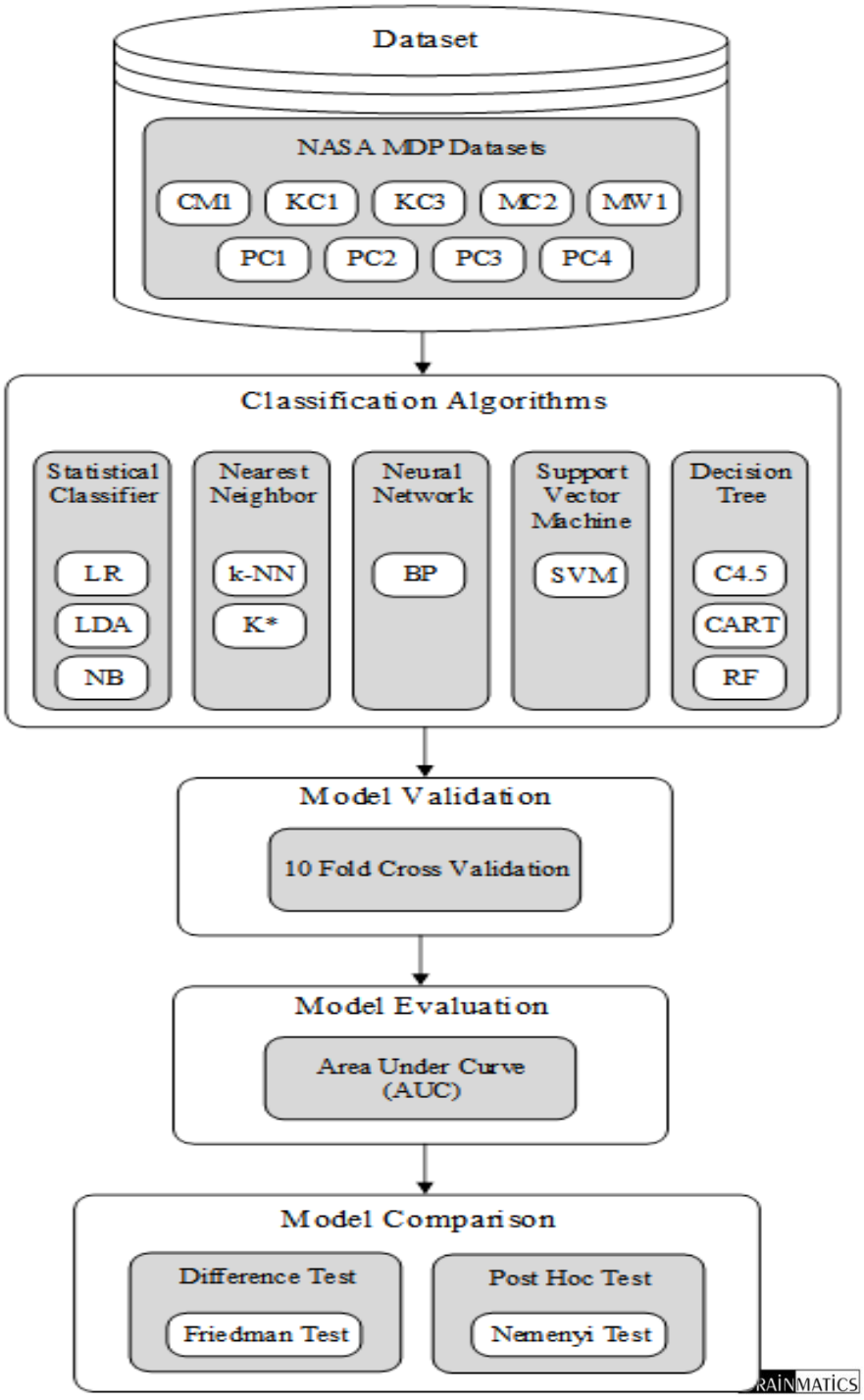
RP3 Neural network has strong fault tolerance and strong ability of nonlinear dynamic processing of software fault data, but practicability of neural network is limited due to difficulty of selecting appropriate parameters



Research Result on RQ1

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP1	While many studies on software defect prediction report the comparative performance of the modelling techniques they have used, no clear consensus on which classifier perform best emerges when individual studies are looked at separately	RQ1	Which machine learning classification algorithms perform best when used in software defect prediction?	RO1	To identify and determine the best machine learning classification algorithms when used in software defect prediction

A Comparison Framework of Classification Models for Software Defect Prediction (CF SDP)



AUC and Friedman Test Results

	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4	*M*	*R*
LR	**0.763**	**0.801**	**0.713**	**0.766**	0.726	**0.852**	0.849	**0.81**	0.894	**0.797**	**1.44**
LDA	0.471	0.536	0.447	0.503	0.58	0.454	0.577	0.524	0.61	0.522	8.33
NB	0.734	0.786	0.67	0.739	0.732	0.781	0.811	0.756	0.838	0.761	3
k-NN	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	8.778
K*	0.6	0.678	0.562	0.585	0.63	0.652	0.754	0.697	0.76	0.658	5.33
BP	0.713	0.791	0.647	0.71	0.625	0.784	0.918	0.79	0.883	0.762	3.22
SVM	0.753	0.752	0.642	0.761	0.714	0.79	0.534	0.75	**0.899**	0.733	3.33
C4.5	0.565	0.515	0.497	0.455	0.543	0.601	0.493	0.715	0.723	0.567	7.78
CART	0.604	0.648	0.637	0.482	0.656	0.574	0.491	0.68	0.623	0.599	6.89
RF	0.573	0.485	0.477	0.525	**0.74**	0.618	0.649	0.678	0.2	0.549	6.89

- LR is dominant in most datasets
- *R* rank: LR has the highest rank, followed by NB, BP, and SVM
- *M* results: no excellent or good models, and a few fair models

AUC	Meaning	Symbol
0.90 - 1.00	excellent classification	
0.80 - 0.90	good classification	
0.70 - 0.80	fair classification	
0.60 - 0.70	poor classification	
< 0.60	failure	

P-value of Nemenyi Post Hoc Test

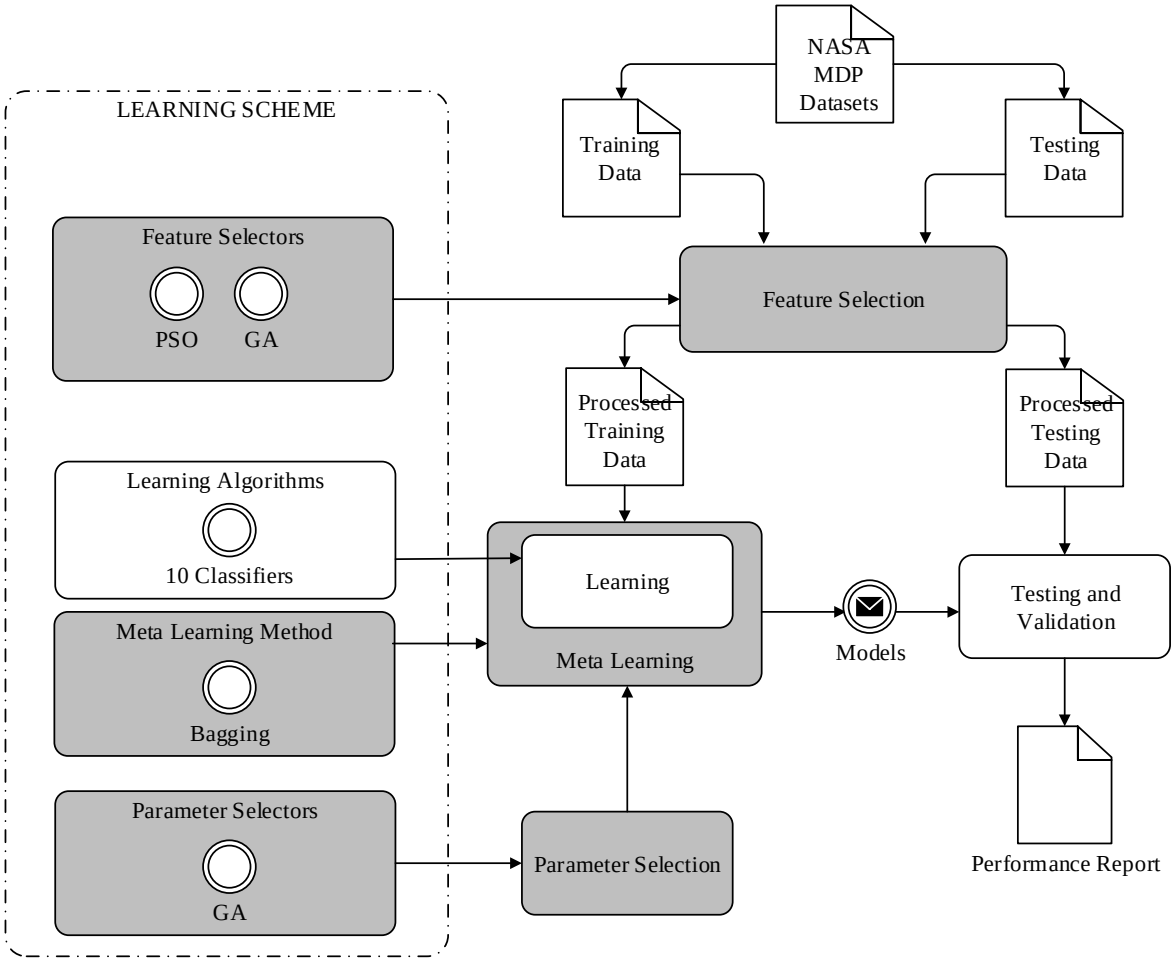
	LR	LDA	NB	k-NN	K*	BP	SVM	C4.5	CART	RF
LR	1	**0.0001**	0.986	**0.0001**	0.164	0.965	0.949	**0.000**	**0.005**	**0.005**
LDA	**0.0001**	1	**0.007**	1.000	0.526	**0.013**	**0.017**	1.000	0.992	0.992
NB	0.986	**0.007**	1	**0.002**	0.831	1.000	1.000	**0.028**	0.164	0.164
k-NN	**0.0001**	1.000	**0.002**	1	0.318	**0.004**	**0.005**	1.000	0.949	0.949
K*	0.164	0.526	0.831	0.318	1	0.901	0.927	0.789	0.986	0.986
BP	0.965	**0.013**	1.000	**0.004**	0.901	1	1.000	**0.046**	0.232	0.232
SVM	0.949	**0.017**	1.000	**0.005**	0.927	1.000	1	0.058	0.273	0.273
C4.5	**0.000**	1.000	**0.028**	1.000	0.789	**0.046**	0.058	1	1.000	1.000
CART	**0.005**	0.992	0.164	0.949	0.986	0.232	0.273	1.000	1	1.000
RF	**0.005**	0.992	0.164	0.949	0.986	0.232	0.273	1.000	1.000	1

- If *P value < 0.05* (boldfaced print), it indicate that there is significant different between two classifiers
- Based on significant difference results, there is no significant difference between LR, NB, BP, and SVM models

Research Publication on RQ1

1. Romi Satria Wahono, Nanna Suryana Herman and Sabrina Ahmad, A Comparison Framework of Classification Models for Software Defect Prediction, Proceedings of the 2014 International Conference on Internet Services Technology and Information Engineering (ISTIE 2014), Bali, Indonesia, May 2014
2. Romi Satria Wahono, Nanna Suryana Herman and Sabrina Ahmad, A Comparison Framework of Classification Models for Software Defect Prediction, Advanced Science Letters, Vol. 20, No. 8, August 2014 (SCOPUS SJR: 0.240)

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)

Research Result on RQ2

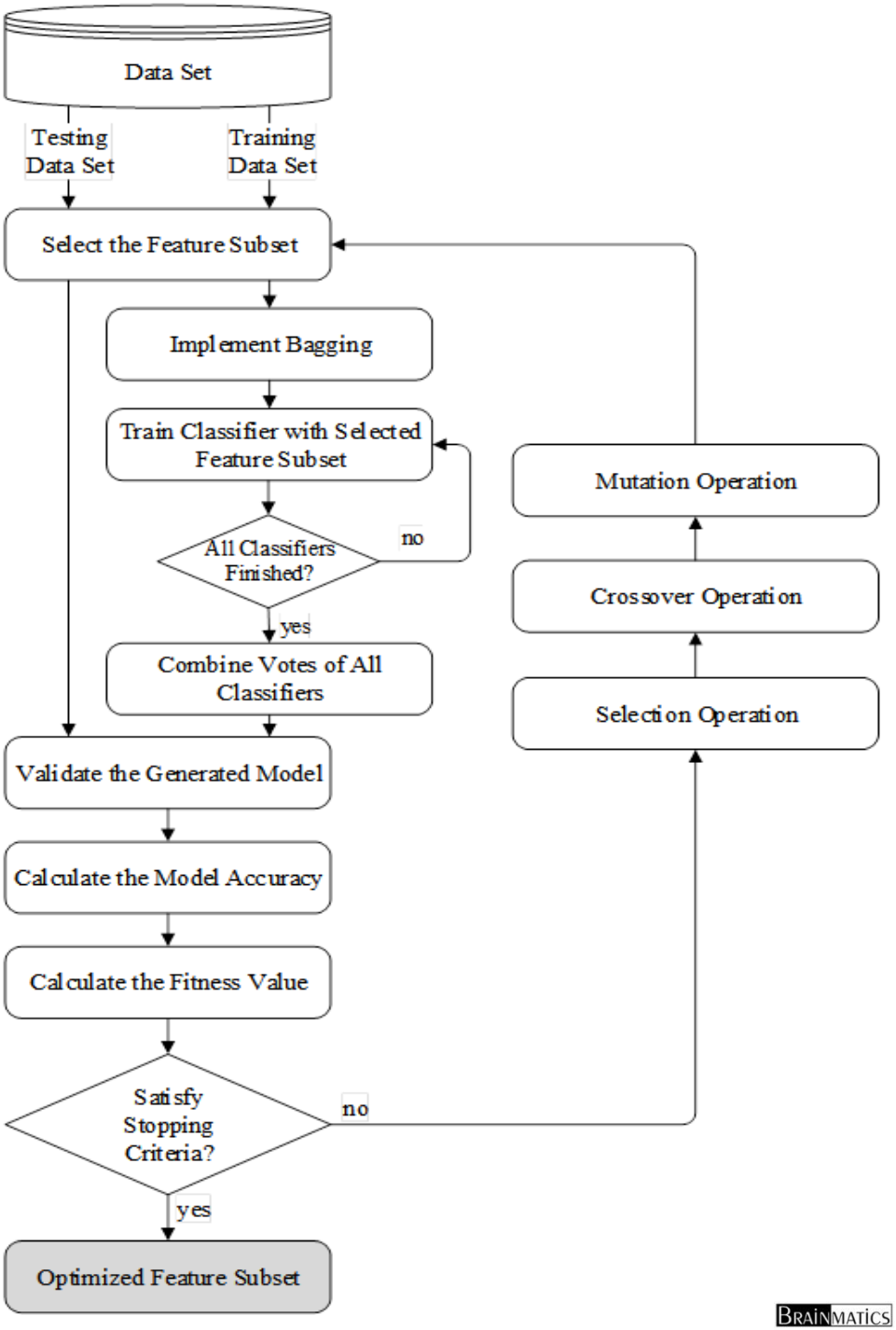
Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP2	Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models	RQ2	How does the integration between genetic algorithm based feature selection and bagging technique affect the accuracy of software defect prediction?	RO2	To develop a hybrid genetic algorithm based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ3	How does the integration between particle swarm optimization based feature selection and bagging technique affect the accuracy of software defect prediction?	RO3	To develop a hybrid particle swarm optimization based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ4	Which metaheuristic optimization techniques perform best when used in feature selection of software defect prediction?	RO4	To identify the best metaheuristic optimization techniques when used in feature selection of software defect prediction

A Hybrid Genetic Algorithm based Feature Selection and Bagging Technique (GAFS+B)

- Every chromosome is evaluated by the fitness function Equation

$$fitness = W_A \times A + W_F \times \left(P + \left(\sum_{i=1}^{n_f} C_i \times F_i \right) \right)^{-1}$$

- Where
  - A: classification accuracy
  - F_i: feature value
  - W_A: weight of classification accuracy
  - W_F: feature weight
  - C_i: feature cost
- When ending condition is satisfied, the operation ends, otherwise, continue with the next genetic operation



Results: Without GAFS+B

Classifiers		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Statistical Classifier	**LR**	0.763	0.801	0.713	0.766	0.726	0.852	0.849	0.81	0.894
	LDA	0.471	0.536	0.447	0.503	0.58	0.454	0.577	0.524	0.61
	NB	0.734	0.786	0.67	0.739	0.732	0.781	0.811	0.756	0.838
Nearest Neighbor	**k-NN**	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	K*	0.6	0.678	0.562	0.585	0.63	0.652	0.754	0.697	0.76
Neural Network	**BP**	0.713	0.791	0.647	0.71	0.625	0.784	0.918	0.79	0.883
Support Vector Machine	**SVM**	0.753	0.752	0.642	0.761	0.714	0.79	0.534	0.75	0.899
Decision Tree	**C4.5**	0.565	0.515	0.497	0.455	0.543	0.601	0.493	0.715	0.723
	CART	0.604	0.648	0.637	0.482	0.656	0.574	0.491	0.68	0.623
	RF	0.573	0.485	0.477	0.525	0.74	0.618	0.649	0.678	0.2

Results: With GAFS+B

Classifiers		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Statistical Classifier	LR	0.753	0.795	0.691	0.761	**0.742**	0.852	0.822	0.813	**0.901**
	LDA	**0.592**	**0.627**	**0.635**	**0.64**	**0.674**	**0.637**	**0.607**	**0.635**	**0.715**
	NB	0.702	**0.79**	**0.677**	0.739	0.724	**0.799**	0.805	**0.78**	**0.861**
Nearest Neighbor	k-NN	**0.666**	**0.689**	**0.67**	**0.783**	**0.656**	**0.734**	**0.554**	**0.649**	**0.732**
	K*	**0.71**	**0.822**	0.503	**0.718**	**0.68**	**0.876**	**0.877**	**0.816**	**0.893**
Neural Network	BP	**0.744**	**0.797**	**0.707**	**0.835**	**0.689**	**0.829**	0.905	**0.799**	**0.921**
Support Vector Machine	SVM	0.667	**0.767**	0.572	**0.747**	**0.659**	0.774	0.139	0.476	**0.879**
Decision Tree	C4.5	**0.64**	**0.618**	**0.658**	**0.732**	**0.695**	**0.758**	**0.642**	**0.73**	**0.844**
	CART	**0.674**	**0.818**	**0.754**	**0.709**	**0.703**	**0.819**	**0.832**	**0.842**	**0.9**
	RF	**0.706**	**0.584**	**0.605**	0.483	0.735	**0.696**	**0.901**	**0.734**	**0.601**

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

Without GAFS+B vs With GAFS+B

Classifiers		*P* value of t-Test	Result
Statistical Classifier	**LR**	0.156	Not Sig. ($\alpha > 0.05$)
	LDA	**0.00004**	**Sig. ($\alpha < 0.05$)**
	NB	0.294	Not Sig. ($\alpha > 0.05$)
Nearest Neighbor	**k-NN**	**0.00002**	**Sig. ($\alpha < 0.05$)**
	K*	**0.001**	**Sig. ($\alpha < 0.05$)**
Neural Network	**BP**	**0.008**	**Sig. ($\alpha < 0.05$)**
Support Vector Machine	**SVM**	**0.03**	**Sig. ($\alpha < 0.05$)**
Decision Tree	**C4.5**	**0.0002**	**Sig. ($\alpha < 0.05$)**
	CART	**0.0002**	**Sig. ($\alpha < 0.05$)**
	RF	**0.01**	**Sig. ($\alpha < 0.05$)**

- Although there are two classifiers (LR and NB) that have no significant difference (*P value* > 0.05), the remaining eight classifiers (LDA, k-NN, K*, BP, SVM, C4.5, CART and RF) have significant difference (*P value* < 0.05)
- The proposed GAFS+B method makes an improvement in prediction performance for most classifiers

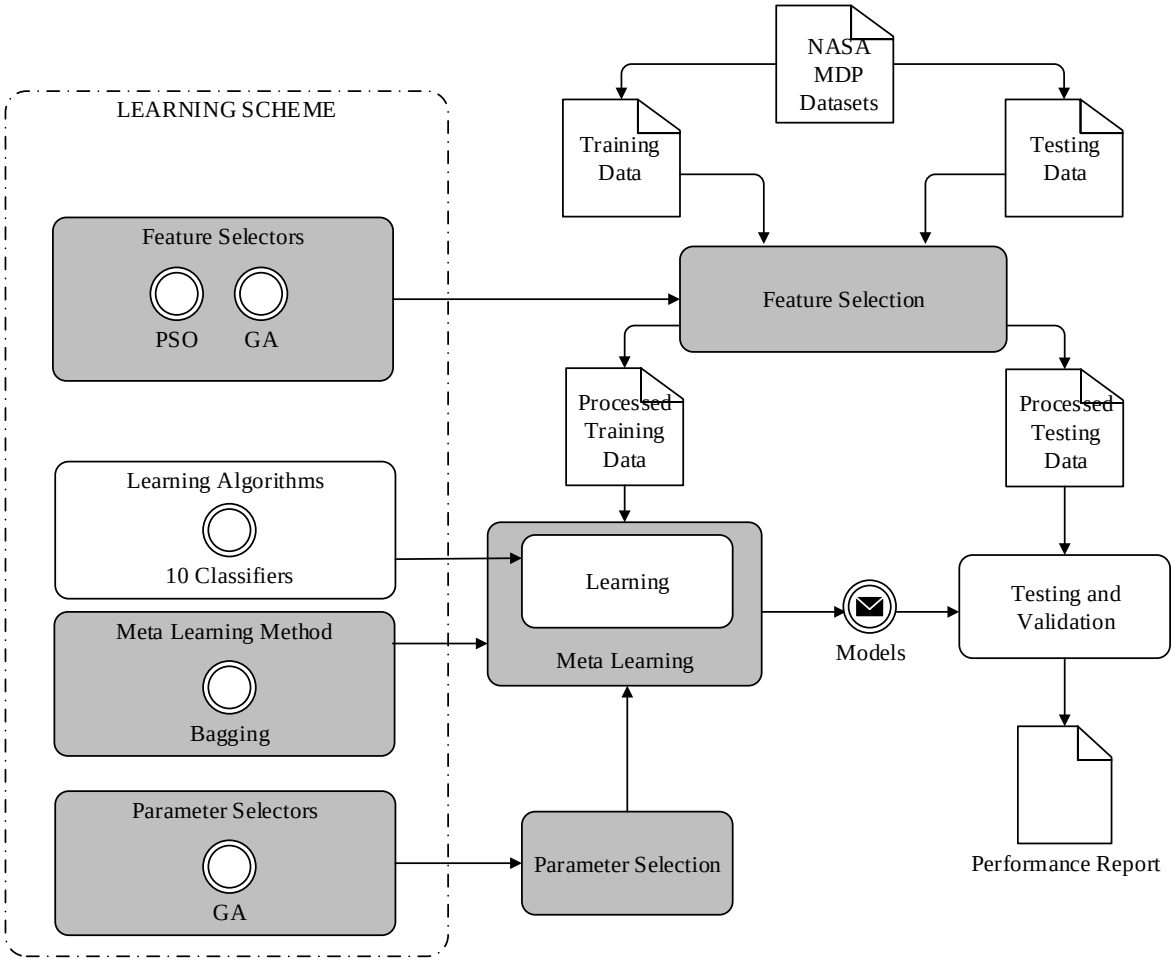
Research Publication on RQ2

1. Romi Satria Wahono and Nanna Suryana Herman, *Genetic Feature Selection for Software Defect Prediction*, Proceedings of the 2013 International Conference on Internet Services Technology and Information Engineering (ISTIE 2013), Bogor, Indonesia, May 2013
2. Romi Satria Wahono and Nanna Suryana Herman, *Genetic Feature Selection for Software Defect Prediction*, Advanced Science Letters, Volume 20, Number 1, January 2014 , pp. 239-244
(SCOPUS SJR: 0.240)

KIAT 5

Lanjutkan Eksperimen ke RQ berikutnya (RQ3-RQ4-RQ*n*), Ulangi Siklus Eksperimen dan Publikasi Hingga Persyaratan Kelulusan Terpenuhi, Disertasi adalah Kumpulan dari Publikasi Penelitian

Proposed Framework



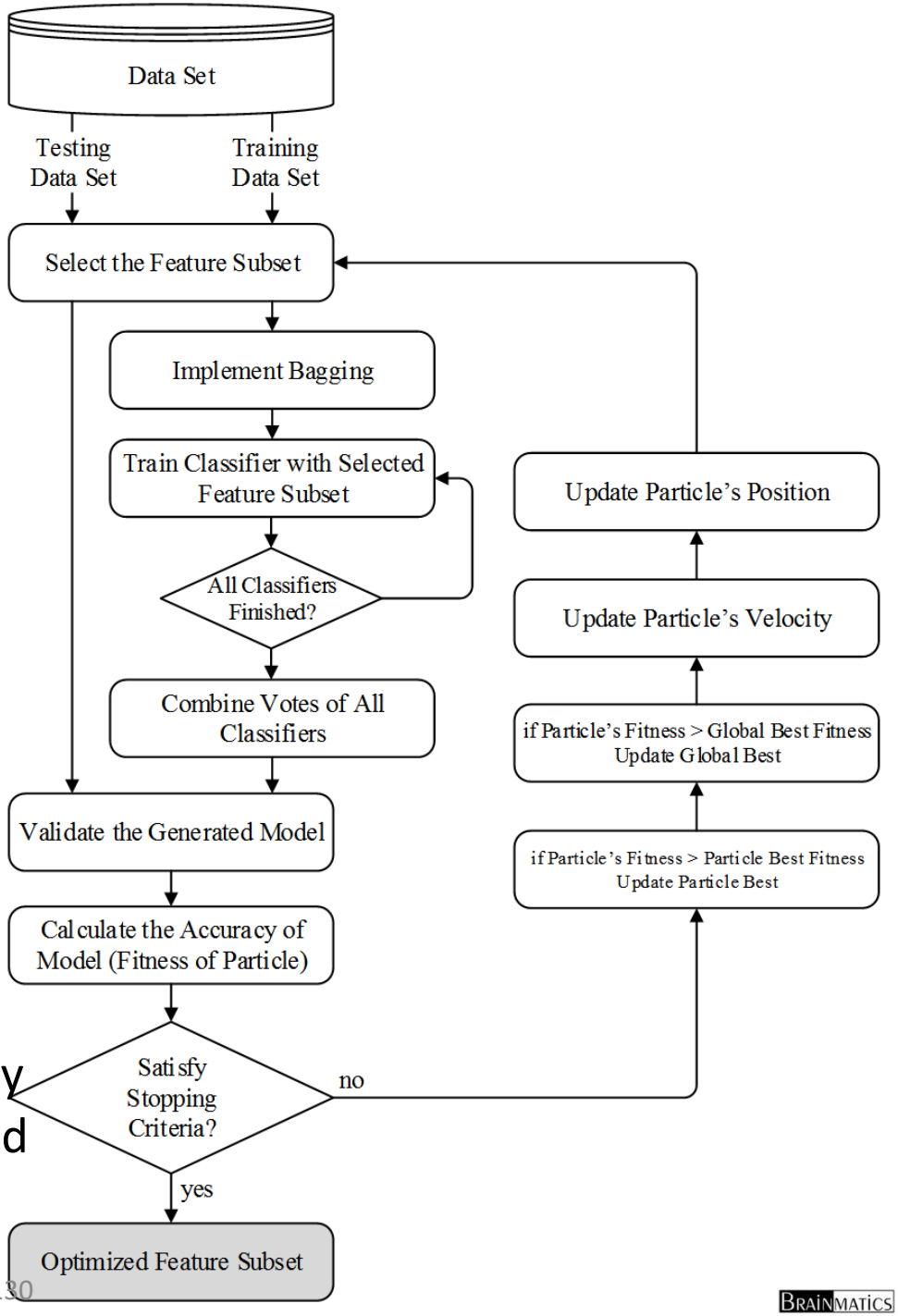
Framework	Dataset	Data Preprocessor	Feature Selectors	Meta-Learning	Classifiers	Parameter Selectors	Validation Methods	Evaluation Methods
(Menzies et al. 2007)	NASA MDP	Log Filtering	Info Gain		3 algorithm (DT, 1R, NB)	-	10-Fold X Validation	ROC Curve (AUC)
(Lessman et al. 2008)	NASA MDP	-	-		22 algorithm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

Research Result on RQ3

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP2	**Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models**	RQ2	How does the integration between genetic algorithm based feature selection and bagging technique affect the accuracy of software defect prediction?	RO2	To develop a hybrid genetic algorithm based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ3	**How does the integration between particle swarm optimization based feature selection and bagging technique affect the accuracy of software defect prediction?**	**RO3**	**To develop a hybrid particle swarm optimization based feature selection and bagging technique for improving the accuracy of software defect prediction**
		RQ4	Which metaheuristic optimization techniques perform best when used in feature selection of software defect prediction?	RO4	To identify the best metaheuristic optimization techniques when used in feature selection of software defect prediction

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

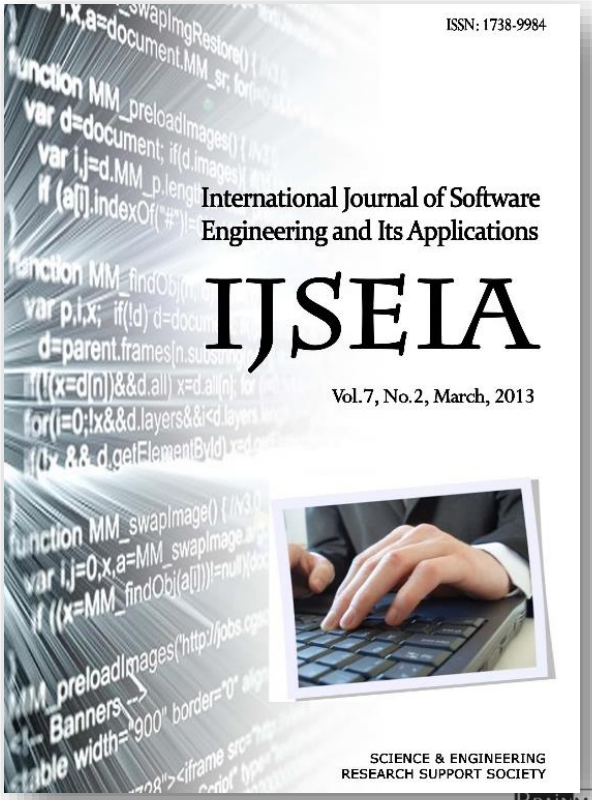
Without PSOFS+B vs With PSOFS+B

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

Research Publication on RQ3

Romi Satria Wahono and Nanna Suryana, *Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction*, International Journal of Software Engineering and Its Applications, Vol 7, No 5, September 2013



Research Result on RQ4

Research Problems (RP)		Research Questions (RQ)		Research Objectives (RO)	
RP2	**Noisy attribute predictors and imbalanced class distribution of software defect datasets result in inaccuracy of classification models**	RQ2	How does the integration between genetic algorithm based feature selection and bagging technique affect the accuracy of software defect prediction?	RO2	To develop a hybrid genetic algorithm based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ3	How does the integration between particle swarm optimization based feature selection and bagging technique affect the accuracy of software defect prediction?	RO3	To develop a hybrid particle swarm optimization based feature selection and bagging technique for improving the accuracy of software defect prediction
		RQ4	**Which metaheuristic optimization techniques perform best when used in feature selection of software defect prediction?**	**RO4**	**To identify the best metaheuristic optimization techniques when used in feature selection of software defect prediction**

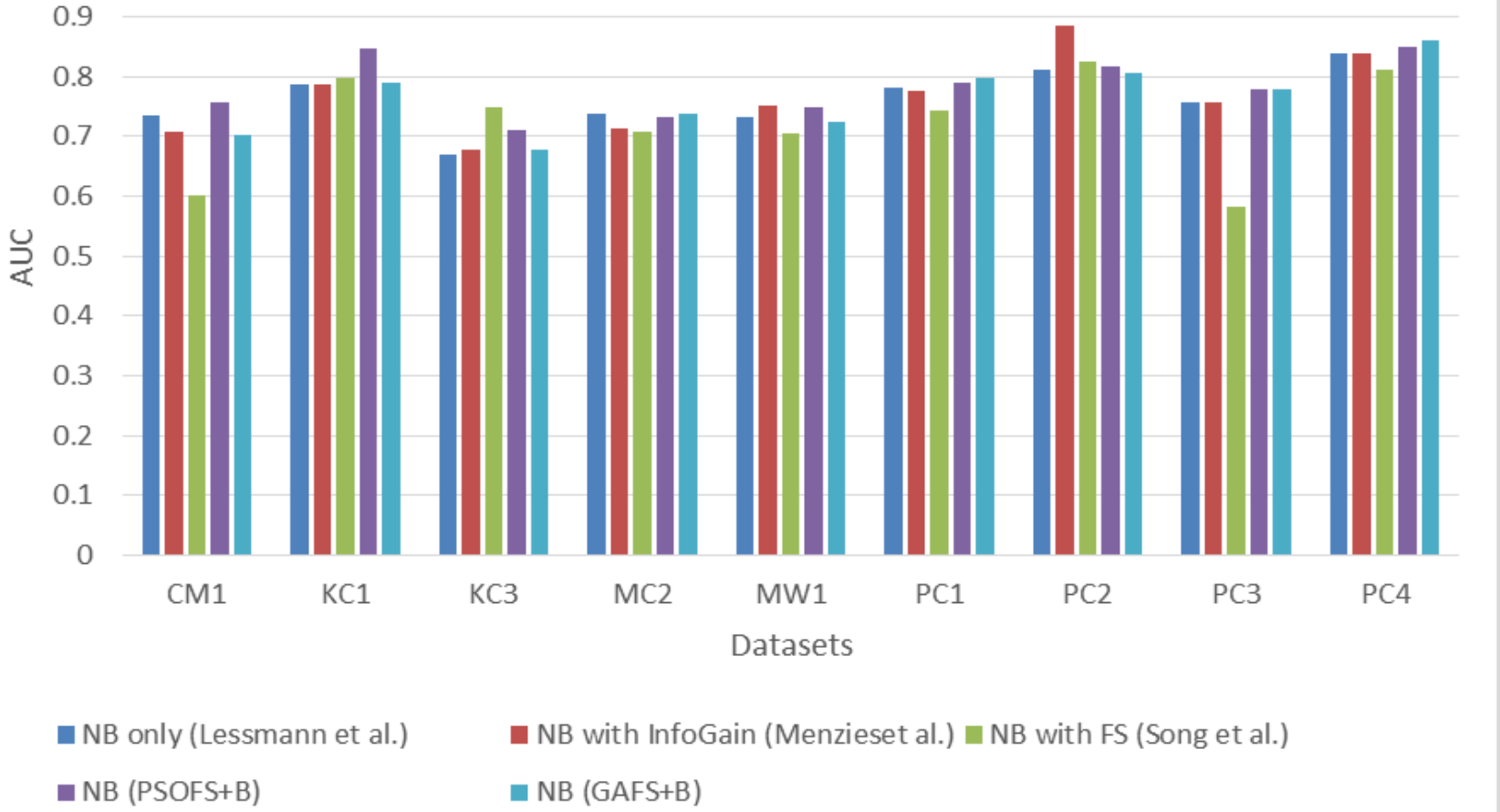
GAFS+B vs PSOFS+B

Classifiers		*P* value of t-Test	Result
Statistical Classifier	**LR**	0.25	Not Sig. (α > 0.05)
	LDA	0.19	Not Sig. (α > 0.05)
	NB	0.044	**Sig. (α < 0.05)**
Nearest Neighbor	**k-NN**	0.063	Not Sig. (α > 0.05)
	K*	0.268	Not Sig. (α > 0.05)
Neural Network	**BP**	0.203	Not Sig. (α > 0.05)
Support Vector Machine	**SVM**	0.003	**Sig. (α < 0.05)**
Decision Tree	**C4.5**	0.3	Not Sig. (α > 0.05)
	CART	0.216	Not Sig. (α > 0.05)
	RF	0.088	Not Sig. (α > 0.05)

- Although there are two classifier that have significant difference (*P* < 0.05) (NB and SVM), the results have indicated that those of remaining eight classifiers have no significant difference (P > 0.05)
- There is no significant difference between PSO and GA when used as feature selection for most classifiers

Proposed Methods vs Other Methods

	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB only (Lessmann et al.)	0.734	0.786	0.67	**0.739**	0.732	0.781	0.811	0.756	0.838
NB with InfoGain (Menzies et al.)	0.708	0.786	0.677	0.712	**0.752**	0.775	**0.885**	0.756	0.84
NB with FS (Song et al.)	0.601	0.799	**0.749**	0.707	0.704	0.742	0.824	0.583	0.812
NB (PSOFS+B)	**0.756**	**0.847**	0.71	0.732	0.748	0.79	0.818	**0.78**	0.85
NB (GAFS+B)	0.702	0.79	0.677	**0.739**	0.724	**0.799**	0.805	**0.78**	**0.861**

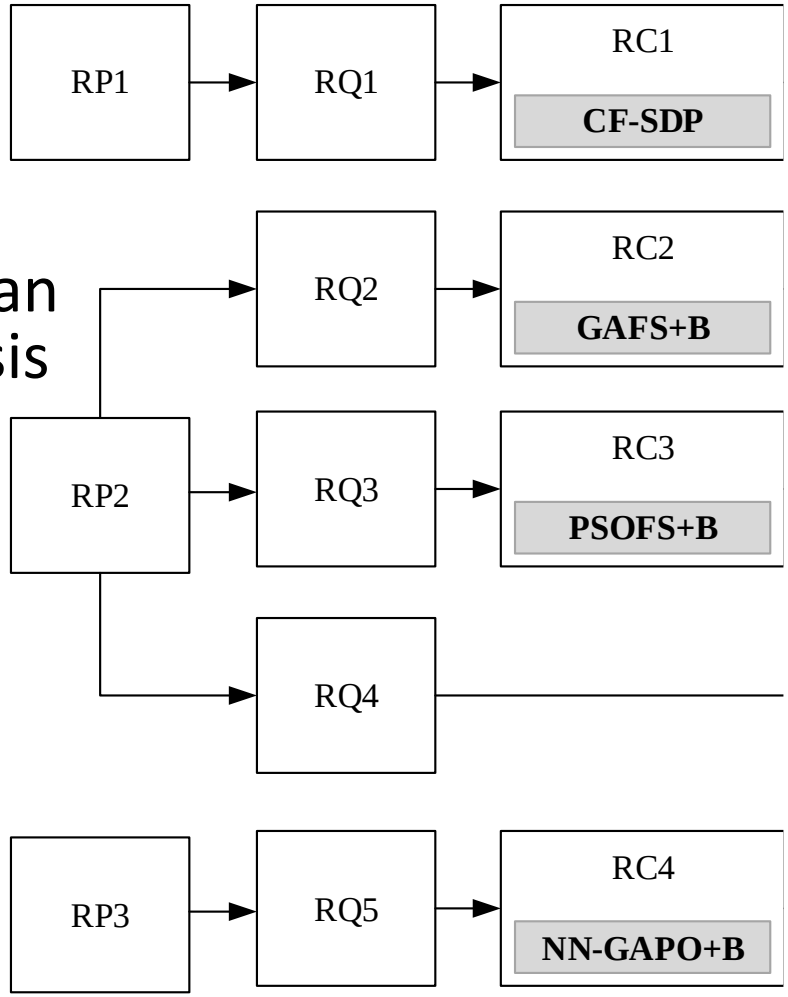


Research Publication on RQ4

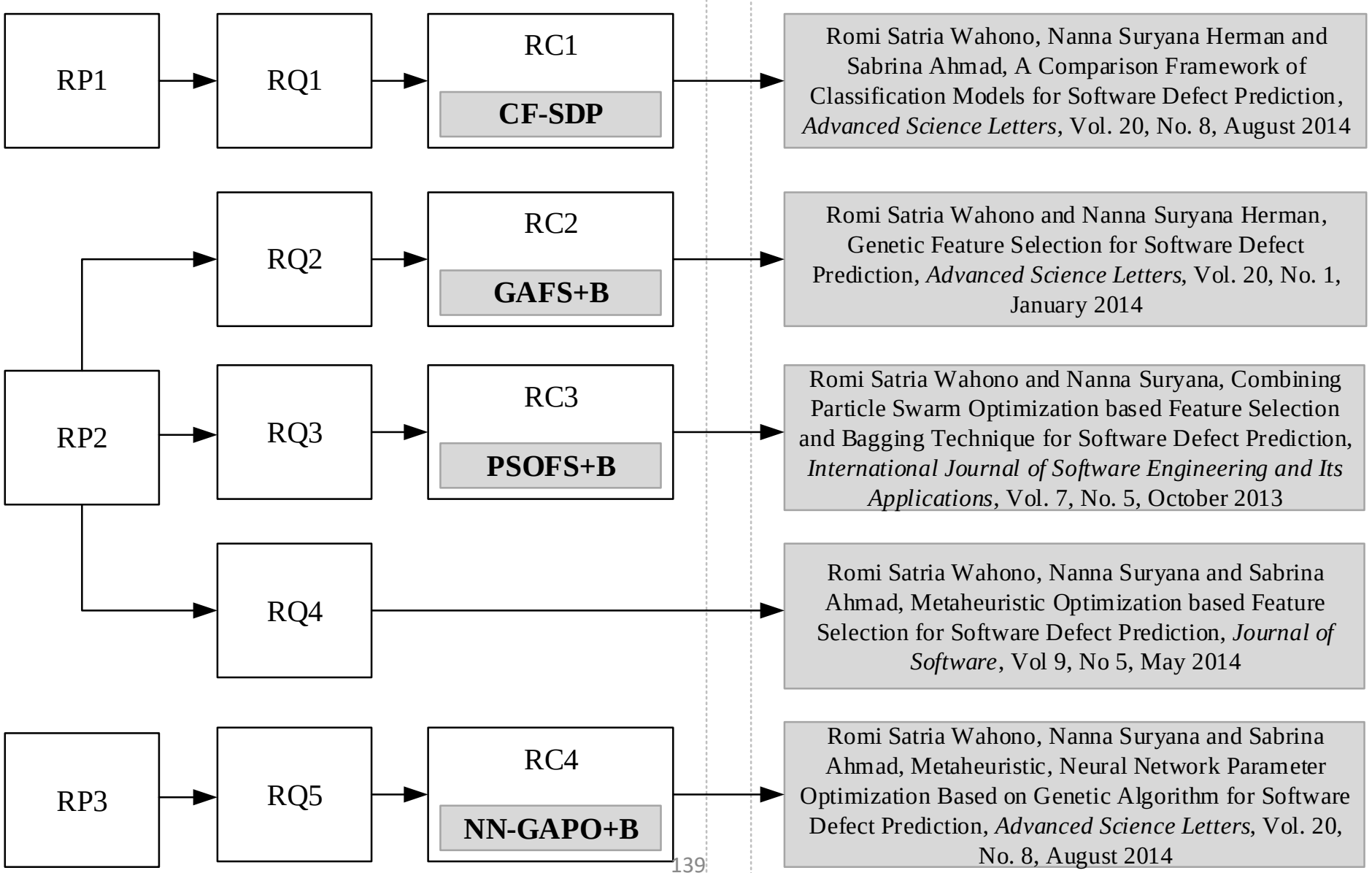
Romi Satria Wahono, Nanna Suryana and Sabrina Ahmad, *Metaheuristic Optimization based Feature Selection for Software Defect Prediction*, Journal of Software, Vol. 9, No. 5, May 2014
(SCOPUS SJR: 0.260)

Pola RP – RQ – RC pada Penelitian

- Research Problem (RP) atau masalah penelitian adalah alasan kita melakukan penelitian
- Satu RP bisa coba dipecahkan dengan banyak cara/metode/solusi/hipotesis (Research Question (RQ))
- Satu RQ akan membentuk satu kontribusi ke pengetahuan (Research Contribution (RC))
- Satu RC akan menjadi satu paper publikasi



Software Defect Prediction Framework based on Hybrid Metaheuristic Optimization Methods



Disertasi?

Susunan paper-paper publikasi yang sudah kita lakukan

TABLE OF CONTENTS

Perbaikan dari Proposal dan Perubahan Hasil Penelitian

Systematic Literature Review (SLR)

Metodologi Penelitian
(Cara dan Tahapan Melakukan Penelitian)

Paper 1 (RC1: CF-SDP)

Paper 2 (RC2: GAFS+B)

Paper 3 (RC3: PSOFS+B)

Paper 5 (RC5: NNGAPO+B)

1. Pilih **Topik yang Tepat**, Lakukan Penelitian dan **Publikasi** Sebelum Masuk Program S3, Mulai Jalin **Komunikasi** dengan Calon Supervisor

2. Buat **Proposal Lengkap** Termasuk Systematic Literature Review (**SLR**) yang Memuat State-of-the-art Problems, Methods, Dataset (Anggap Draft Awal Bab 1 dan 2 Disertasi), Prioritaskan **Fulltime S3** Bila Mungkin

3. Masuk Program S3, Target Semester 1 Rapikan dan **Konversi SLR menjadi Paper** untuk Publikasi, Ikuti Perkuliahan dengan Aktif di Kelas, dan Jalin **Komunikasi Cerdas** dan Intensif dengan Supervisor

4. Eksekusi Proposal, Mulai dari RQ1-RQ2, Lakukan Eksperimen dengan Target **Publikasi ke Journal** Q3 atau Q4, Lakukan **Perbaikan Eksperimen dan Paper berdasarkan Hasil Review** Submission Paper

5. Lanjutkan Eksperimen ke RQ berikutnya (RQ3-RQ4-RQ*n*), **Ulangi Siklus Eksperimen dan Publikasi** Hingga Persyaratan Kelulusan Terpenuhi, **Disertasi adalah Kumpulan dari Publikasi** Penelitian

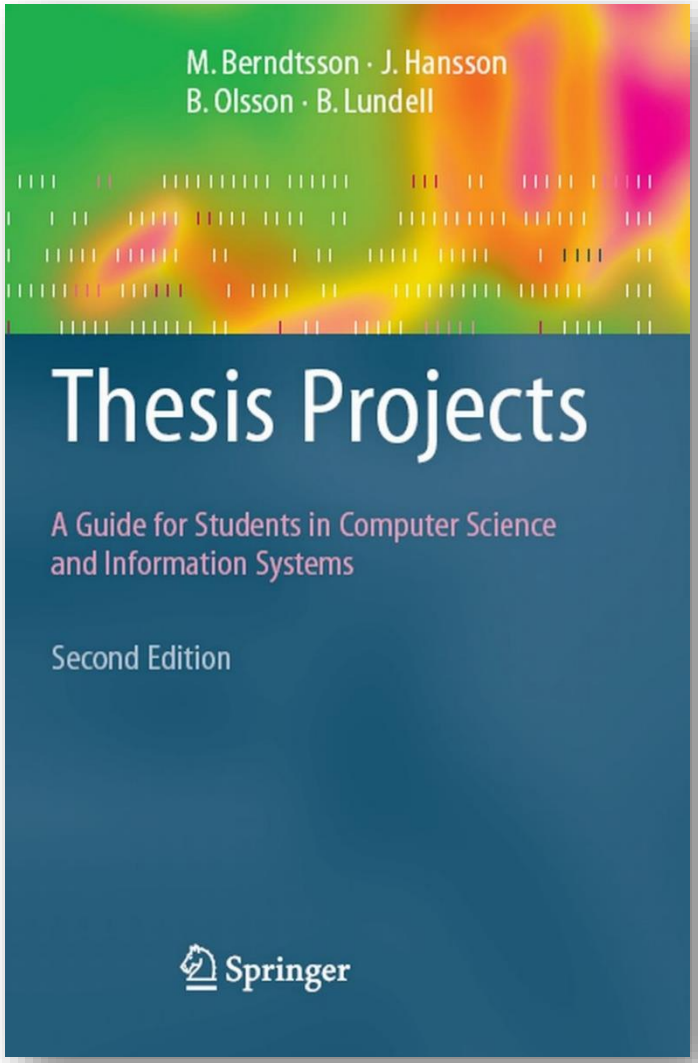
5 KIAT S3

Lulus Tepat Waktu

Terima Kasih

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

Reference



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.