Metrics for Software Process Simulation Modeling

Bohan Liu, He Zhang, Liming Dong, Zhiqi Wang, Shanshan Li

Abstract—**Background**: Software Process Simulation (SPS) has become an effective tool for software process management and improvement. However, its adoption in industry is less than what the research community expected due to the burden of measurement cost and the high demand for domain knowledge. The difficulty of extracting appropriate metrics with real data from process enactment is one of the great challenges.
Objective: We aim to provide evidence-based support of the process metrics for software process (simulation) modeling.
Method: A systematic literature review was performed by extending our previous review series to draw a comprehensive understanding of the metrics for process modeling following a meta-model of ontology of metrics in SPS.
Results: We identified 145 process modeling studies that collectively involve 2130 metrics and classified them using the coding technique. Two diagrams which illustrate the high frequency causal relationships used between metrics are proposed in terms of two hierarchical levels of modeling purposes. The specific metrics of different paradigms are compared, and the main difference is that Discrete-Event Simulation (DES) and Agent-Based Simulation (ABS) can provide more detailed simulations from the perspective of development activities and individual developers whilst System Dynamics (SD) tends to use the mean value as an alternative. We revisited the data issues encountered in SPS data preparing phases, as well as identified the corresponding strategies.
Conclusion: The results of this study provide process modelers with an evidence-based reference of the identification and the use of metrics in SPS modeling, and further contribute to the development of the body of knowledge on software metrics in the context of process modeling. Furthermore, this study is not limited to process simulation but can be extended to software process modeling, in general. Taking simulation metrics as standards and references can further motivate and guide software developers to improve the collection, governance, and application of process data in practice.

Index Terms—software metric, software process model, process simulation, systematic literature review

✦

1 INTRODUCTION

SOFTWARE process models are built to gain insights into software processes so that we can predict, modify or control them [1]. Software process model can be either a static (descriptive) model or a dynamic (simulation) model whose behavior changes over time. The simulation requires more information and knowledge, but it can simulate the real world in more detail. It has been widely claimed and accepted that SPS is an effective tool in support of software process management and improvement. Since Abdel-Hamid and Madnick [2] introduced Software Process Simulation (SPS) to Software Engineering (SE) in the 1980s, there have been a large number of studies published in the community, including quite a few industrial cases. Ahmed et al. [3] conducted a survey to investigate the state-of-practice of simulation practice in SPS, nearly half of the respondents (8/17) are from industry. Furthermore, researchers have applied the SPS technique within the integration of the capacity maturity model (CMMI) for process optimization [4, 5, 6], the SPS was recognized as the key to achieving levels 4 and 5 [7]. In addition to CMMI, Mishra et al. [8] built a system dynamics model to understand the global software development of the Indian software

industry. The system dynamics technology is also applied to choose the best gate timing strategy in new product development projects [Van 17][1].

Zhang et.al [9] highlight benefits of SPS for various purposes such as prediction, process investigation, technology evaluation, and risk management. To achieve the modeling purposes, an SPS model may involve a number of (sometimes even hundreds of) metrics. The identification of the metrics and their relationships needed in a specific process model is a challenging task, particularly for novice modelers, and the collection of the quality data on these metrics is even effort-consuming.

The panel of domain experts in SPS indicates that a prerequisite for building SPS models in the industry is that companies are able to analyze their information needs [10]. It accounts for most of the cost to identify the appropriate metrics and gather the corresponding data. The analysis of information, as well as the identification of metrics, requires considerable knowledge and skills. Hence, the software process community encourages the development of the knowledge base and the model library with the common set of process metrics as the key component to unleash reusability [10]. There are a number of SPS studies indicated the problems on metrics, some of the evidence are presented as follows:

"For some of the relevant data it is hardly possible to determine

---

• B. Liu, H. Zhang, L. Dong, Z. Wang, and S. Li are with State Key Laboratory of Novel Software Technology, Software Institute, Nanjing University, Nanjing, Jiangsu, China.
E-mail: bohanliu@nju.edu.cn, hezhang@nju.edu.cn, lmdongmg@gmail.com, 502022320013@smail.nju.edu.cn, lss@nju.edu.cn

---

1. We use a distinct citation format to distinguish the reviewed studies from other references.

the necessary information in real-life projects ... we are elaborating approaches to take such human attributes into account, which are not directly observable, and to consider them in the quantitative logic of the model." [Neu 03].

"Obtaining the quantitative data is another difficulty regardless of developing the simulation model." [Park 07]

The challenging task on modeling metrics can be twofold: 1) identifying key metrics from real-process based on the domain knowledge and the data available; 2) collecting and mining the required data for measurement. Knowing what metrics were used in SPS modeling is a prerequisite for studying these two challenges. To the best of our knowledge, no secondary study that investigates metrics is dedicated to SPS modeling yet, although many papers and books [11, 12, 13, 14, 15, 16, 17] have been published on the topic of software metrics over the past decades. It motivated us to create an evidence-based view of the metrics adopted in SPS models to contribute to the development of the body of knowledge on software metrics in the context of process modeling.

Therefore, the objective of this study and its follow-ups are to relieve the high burden and cost of SPS modeling by systematically identifying the modeling metrics of the exemplar process models and the experiences extracted from the relevant literature available. Although this research takes the metrics in SPS modeling as the research object, it is also applicable to general software process modeling as well as general software process measurement. The process of building a simulation model usually consists of the following steps. In the process of building an executable simulation model, a static model is often an essential intermediate product. From an evaluation point of view, we require that the descriptive model is semantically correct, while we need to assess the appropriateness and fidelity of the simulation model, as its output is a distribution of values for a specific project [18, 19]. From a modeling perspective, the building of descriptive model is to abstract, collect and transform fragments of the real world based on the incomplete knowledge we have gained so far [20]. Simulation models require detailed understanding of the processes they simulate, as well as the reliable data for their initial construction. For example, a set of variables needs to be specified that represents a continuously differentiable function of time [Zhan 06a, Zhan 09]. Hence, SPS modeling associates with a higher standard of metric than static process modeling. The metrics used in SPS models also apply to static models in most cases. Moreover, it is able to examine which metrics are needed to describe the real world driven by modeling.

To achieve this, we developed a meta-model of the ontology of metrics for modeling as the guidance of the research. Following the meta-model, we conducted a Systematic Literature Review (SLR) to aggregate, classify, and synthesize the metrics used in the SPS models. As a result, 145 studies that report SPS models are identified from the pool of SPS related papers until 2021. From the included studies, we extracted 2130 metrics. Although software metrics classification schemes have been proposed in the software measurement area [21, 22], they are not adaptive to SPS modeling since the focus of SPS and software measurement researches are different.

Under the guidance of the meta-model, we investigated metrics and their directly related elements from four research questions. We developed a new classification framework based on the extracted metrics and referred to the high level of categories (entities and attributes) suggested by the study [22] (RQ1. metrics). Causal relationships between metrics in SPS models were discussed. We studied the considerations of metrics in terms of modeling purposes and paradigms of SPS models respectively (RQ2. causal relationships between metrics). Causal relationship diagrams that illustrate similarities and differences of modeling models at the cognitive level and models at tactical & strategic levels are proposed and differences of metrics used in different paradigms are discussed (RQ3. selection of metrics). We provide a mapping on the relationships between data issues for measurement and solution strategies (RQ4. data for metrics).

This study contributes to both the research community and practitioners in industry.

- It addresses the first challenge (identification of key metrics) in modeling; meanwhile, it provides researchers the necessary foundation for conducting research to solve the second challenge (data acquisition for measurement).
- This work serves as a knowledge base, enabling practitioners and researchers to gain a comprehensive understanding of the metrics and their relationships involved in the SPS modeling.
- The classification framework and the considerations of metrics from modeling purposes, paradigms, and data issues provide a reference for practitioners to reuse existing knowledge; at the same time, it provides a clue to which metrics and causal relationships researchers need to focus on.
- The second challenge is discussed based on evidence as a set of data issues, coping strategies, and available data sources for hard-to-get metrics are identified.

Note that we use a distinct citation format (author's surname & year, e.g., [Pfah 01]) to distinguish the reviewed studies from other references. The complete list of references of the included studies is shown in the APPENDIX.

2 RELATED WORK

This section introduces the process simulation and the software metrics that have been studied for decades in the SE community.

2.1 Software Process Simulation

Kellner et al. [1] offered an overview of the SPS area to answer three fundamental questions, i.e. *why*, *what*, and *how*. They identified the reasons for conducting an SPS study, defined the scope of SPS models, and discussed the relationships among purposes, scope, and metrics. They also provided a framework to support decisions about simulation approaches, techniques, and required metrics.

Zhang et al. [23, 24, 25] conducted a series of SLRs on SPS modeling. They identify ten purposes for SPS research that are classified into three levels and two dimensions of the scope of the model [23]. Several modeling paradigms and

simulation tools are summarized in [23, 25]. They indicate five trends of process modeling based on the findings [24]. The impact of SPS research on practice was also reported in another study [26] with an impact roadmap that traces the successful SPS industrial application cases to their origins.

The impact of SPS has gradually expanded and it has become more and more mature in the last decade. The adoption of SPS in SE education is studied in [27], which confirms that education is an important application area of SPS with continuous research interests and shows that the SPS game appears more attractive to educators than the other forms of SPS. Integration strategies and recommendations for constructing hybrid SPS models are developed since software processes become more and more complicated [28]. The Verification and Validation (V&V) take a critical role in securing the quality of SPS models, a mapping of quality aspects for V&V and the possible V&V methods is presented in [29].

However, there are still debates over the impact and usefulness of SPS research. França et al. [30] present a quasi-systematic review of 108 studies to investigate the reliability of SPS studies in SE. As a result, they identified a few problems and indicated that SPS studies lack the necessary information for replication. Ali et al. [31] aggregate all the points of view on the usefulness of SPS but find that no conclusion on these conflicting claims can be made based on the secondary studies. They conducted an SLR and evaluated 87 SPS studies. The results show that there is still a lack of conclusive evidence on it, as a few studies report the cost of developing an SPS model. Pfahl [32] also argues that it still lacks the evidence that SPS has become an accepted and regularly used tool for software project managers and its high cost is the main reason. On the other hand, the panel of domain experts on SPS collectively offered a different perspective that the impacts of SPS on practice cannot be ignored compared to many other SE technologies, although its high cost is still a major barrier against its wide application, and indicated that the consequences of waiving simulation should be considered whilst simulation is regarded as a cost saver rather than as a cost driver in other engineering disciplines [10].

2.2 Software (Process) Metrics

Software metrics play a crucial role in quantitative software engineering and have been researched for many years from different perspectives. We discuss the previous secondary studies of software metrics and present the overview in Table 1.

Gómez et al. [11] performed a systematic literature review whose objective is to answer the questions of how, when and what to measure. They adopted the classification of concepts defined in the Software Measurement Ontology proposed by [33] which aims to contribute to the harmonization of the different software measurement proposals and standards, providing a coherent set of common concepts used in software measurement.

Bellini et al. [12] conducted an SLR to investigate five key conceptual and methodological issues, i.e. how to apply measurement theory to software, how to frame software metrics, how to develop metrics, how to collect core measures and how to analyze measures. They adopted Fenton's

classification framework [22]. They finally provide methods for collecting and analyzing the measurements and suggest that attention is increasingly being paid to multidimensional metrics.

Kitchenham et al. [13] developed a preliminary mapping of software metrics studies that focus on identifying influential studies from 2000 to 2005 on software metrics. They conclude that empirical studies are of major importance to the software metrics research community. Although software metrics have been studied by a considerable number of researchers in various dimensions, the empirical methodology adopted needs to be refined.

Abilio et al. [14] presented an SLR to identify metrics associated with software maintainability and proposed for feature-oriented and aspect-oriented technologies from 11 primary papers. The metrics are classified according to the software attribute they measure, which ranges from architectural, parallel development, debugging, to quality attributes.

Kupiainen et al. [15] undertook an SLR on using metrics in industrial Lean and Agile software development. The authors classify the metric based on Fenton's classification framework [22]. They identify the degree of influence and popularity of metrics in agile development and present a mapping between metrics and agile principles.

Nuñez-Varela et al. [16] conducted a mapping study to investigate the trend of research on code metrics. They classified code metrics into four categories, including object-oriented programming, aspect-oriented programming, feature-oriented programming, and procedural programming.

Meidan et al. [17] conducted a mapping study to create a classification scheme of studies on the measurement of software development process in terms of source type, publication year, research type, contribution type, proposal type, validation type, entity/abstraction and study context. They identified 13 process attributes, 3 developer attributes, 4 project attributes, and 2 organization attributes.

This study have different research scopes, and we focus more on process metrics, which can help SPS researchers understand using metrics in software process (simulation) modeling comprehensively.

3 ONTOLOGY

To systematically study metrics for SPS modeling, we proposed an ontology to illustrate the relationship between metrics and simulation modeling, as well as their related concepts. Our research revolves around this ontology.

Olsina and Martín [34] proposed an ontology for software metrics and indicators based on different software-related ISO standards and research articles. To adapt to SPS, we proposed an adjusted meta-model of the ontology referred to the metrics section of their ontology and introduced the concept of SPS modeling as presented in Fig. 1. The descriptions of the concepts and relationships presented in the meta-model are shown in Table 2 and 3 respectively.

All models are simplified representations (abstractions) of the real world, as well as a conceptual model is the abstraction of a simulation model. All simulation modeling involves conceptual modeling [35]. Conceptual modeling

TABLE 1: An overview of the comparison between this work and previous studies on software metrics

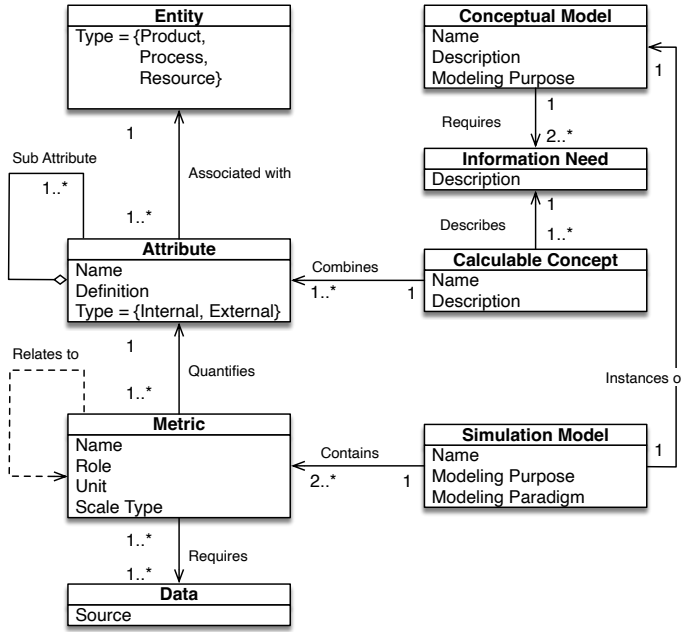| Studies	Year	Metrics' Classification	Research Focus
Gómez et al. [11]	2006	Software Measurement Ontology [33]	What, when, and how to measure
Bellini et al. [12]	2008	Fenton's categorization [22]	Measurement theory; Alternative methods to collect and analyze core measure; Metrics concepts, and how to identify metrics
Kitchenham et al. [13]	2010	OO metrics, web-metrics, and other code metrics	Identify trend in influential software metrics studies
Abilio et al. [14]	2012	Software attribute they measure in paper	Software maintainability metrics
Kupiainen et al. [15]	2015	Fenton's categorization [22]	Using metrics in Agile software
Nuñez-Varela et al. [16]	2017	Object oriented programming, aspect oriented programming, feature oriented programming, and procedural programming	The trend of source code metrics
Meidan et al. [17]	2018	Process, developer, project, and organization	Understand the measurement of the software development process
This work	2022	A detailed framework based on Fenton et al.'s categorization [22]	Specific to metrics in software process modeling, including classification, causal relationships between metrics, selection of metrics in modeling, and data issues for measurement.

Fig. 1: Meta-model of the ontology for software metrics in software process simulation modeling

is to specify a model that represents those parts of the problem domain that are included in the simulation model. According to the model development process suggested by Kitchenham et al. [19], domain analysis and the definition of model requirements are needed before specifying a model. Domain analysis is to confirm the information needs in the real world. Not all information needs should be included in the model requirements since they should be valid, credible, feasible, and useful [36], in other words, they are calculable concepts. A simulation model represents the conceptual model in a specific computer code, which means that simulation modeling is to quantify the input and output of the conceptual model. From the perspective of software metrics, it is to measure attributes of entities and quantify the causal relationships between attributes.

4 RESEARCH METHOD

This section describes the SLR process on metrics in SPS modeling that followed the SLR guidelines [38]. Four researchers and their supervisor were involved in this study.

Strictly speaking, this study is not an SLR research. We use the SLR research method to study metrics, which are usually not the research focus of the primary studies we included.

4.1 Research Questions

This study focuses on metrics and elements shown in the meta-model (Fig. 1) that are related to metrics. RQ1 studies metrics and their corresponding attributes from different perspectives. RQ2 studies the causal relationships between metrics presented in SPS models. RQ3 studies the purposes and paradigms of SPS models, which affects the selection of metrics. RQ4 studies issues and solution strategies of data for measurement. As an instantiated executable model of the conceptual model, the simulation model can cover the relevant information of the conceptual model. Hence, RQs are not specifically addressed to conceptual model, information need, and calculable concept that are involved in the meta-model. To achieve the research objective, four research questions are defined to drive this study following the meta-model (as follows).

RQ1: *What metrics have been used and studied in the SPS studies?* RQ1 aims to discuss what to measure in depth as well as build a classification framework of software metrics used and studied in the SPS models. As shown in the meta-model, metrics quantify attribute that is associated with entity. It will also be a multi-level classification of metrics that group them based on the attribute they measured and the entity corresponding to the attribute. In addition, the unit and scale type of each metric would be presented to show how to measure more specifically.

RQ2: *What are the causal relationships between metrics used in different SPS studies?* RQ2 helps to identify the common causal relationships among software metrics used in existing SPS models. The simulation model consists of causal relationships among metrics. The causal relationships between the metrics are the basis for understanding the structure of SPS models.

RQ3: *What are the considerations for selecting metrics in terms of different modeling purposes?*

It is almost impossible to simulate all the factors and details in one model. Modelers select the most relevant metrics and ignore others as they concentrate on different purposes. Besides, modelers would adopt different modeling paradigms for different modeling granularities, which would affect the measurement of metrics. The goal of RQ3

TABLE 2: Software metrics ontology: glossary of concepts

Concept	Definition	Attribute: *Description*
Entity	Object that is to be characterised by measuring its attributes [37].	Name: *Name of an entity.* Description: *An unambiguous description of the entity meaning.*
Attribute	A measurable physical or abstract property of an entity.	Name: *Name of an attribute.* Definition: *An unambiguous description of the attribute meaning [37].* Type: *Attributes can be internal or external.*
Metric	A metric is the number or symbol assigned to an entity by this mapping in order to characterize an attribute. [22] Variable and parameter are synonyms of metric in an SPS model.	Name: *Name of a metric.* Unit: *Particular quantity defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity.* Scale type: *The type of scales depends on the nature of the relationship between values of the scale. These types of scales are commonly defined: nominal, ordinal (restricted or unrestricted), interval, ratio, and absolute.*
Data	The data that need to be collected in the real world for measurement.	Source: *The source where the data can be obtained.*
Conceptual Model	A non-software specific description of the computer simulation model (that will be, is or has been developed), describing the objectives, inputs, outputs, content, assumptions and simplifications of the model [36].	Name: *Name of a conceptual model.* Modeling Purpose: *A specific purpose corresponding to the real world that why we develop a model.*
Information Need	Insight necessary for the specific modeling purpose [37]	Description: *An unambiguous textual statement describing the information needs.*
Calculable Concept	Abstract relationship between attributes of entities and information needs [37].	Name: *Name of a calculable concept.* Description: An unambiguous description of the calculable concept meaning.
Simulation Model	A simulation model is a computerized model that represents some dynamic system or phenomenon [1].	Name: *Name of a simulation model.* Modeling Purpose: *A specific purpose corresponding to the real world that why we develop a model.* Modeling Paradigm: *A distinct set of concepts or thought patterns for simulation modeling, e.g., System Dynamics.*

TABLE 3: Software metrics ontology: relationship description

Relationship	Description
Sub-Attribute	An attribute may be composed of none or several sub-attributes, which are in turn attributes.
Associated with	One or more measurable attributes are associated with one or more entities.
Quantifies	One or more metrics can quantify an attribute.
Requires	It requires real-world data for measuring a metric.
Combines	A calculable concept combines (associates) one or more measurable attributes.
Consists of	A simulation model consists of a number of metrics.
Instances of	A simulation model is an instance of a conceptual model.
Relates to	One metric relates to another another metric.

is to investigate the considerations of metrics in terms of modeling purposes and paradigms, which are the two main properties of different simulation models as indicated in the meta-model.

RQ4: *What are the issues and strategies for obtaining data for metrics.* The lack of data remains a problem in SPS modeling, since Raffo and Kellner [39] have analyzed different situations and solutions. RQ4 aims to revisit the status quo of data issues based on evidence, as well as investigate existing solution strategies and specific attributes they can measure.

4.2 Selection Criteria

The inclusion and exclusion criteria for the relevant studies are shown in Table 4. The included paper should be published in English and we retrieved the time span up to 2021. Our research included the studies that applied simulation modeling paradigms for software process research, software education, and software practice. Since the metric is our review focus, the selected studies should clearly claim the metrics used in their model. Regular papers can give us

more comprehensive information to help us collect significant evidence, so we would not select the studies which have no more than 5 pages. Studies should be published as journal articles or conference papers rather than other forms that presented in C5 and C6. We extracted data from primary studies that could provide the first-hand metrics used in SPS research rather than from relevant secondary studies. Furthermore, evidence from primary studies could help us to summarize the relationships among different metrics in different SPS research. In addition, no secondary study was found that investigated the metrics used in SPS. To be specific, although Pfahl [40] presents the example set of metrics used in the SD model which aims to support the analysis of the effectiveness of key SPI in the automotive industry (satisfy C1, C2, C3), the paper is excluded because it is a short paper (meet C4). In another example, Zhang et al. [41] mapped four typical simulation paradigms to the appropriate maturity levels of CMMI to adopt them; however, no specific model is presented in the paper (does not satisfy C2, C3).

TABLE 4: Selection criteria

Inclusion criteria
C1. Published before 2021 and written in English.
C2. Primary studies on employing simulation modeling paradigms for software process research, education and practice.
C3. Primary studies that claimed the metrics used in the simulation model.

Exclusion criteria
C4. Short papers (no more than 5 pages).
C5. In the forms of editorial, abstract, keynote, poster, and book.
C6. Opinion pieces, comments, corrections, notes, slides alone or position papers.
C7. Secondary studies summarizing the outcomes of the existing research work, e.g. road-map, review, survey, etc.

4.3 Search & Selection Process

Fig. 2 shows the search process of this study that consists of five stages. Stages I and II were completed in our initial review [28]. In stage I, the manual search and the automated search were performed by two research students. The venues for manual search, which include five conferences and six journals, are listed in Gao et al.'s work [28]. The search string is shown in Fig. 2. It was further coded into the equivalent forms to match the search syntax of different digital libraries. Four digital libraries (IEEE Xplore, ACM digital library, ScienceDirect, SpringerLink) were searched to retrieve as many SPS studies as possible. In stage II, the forward snowballing was applied as a supplementary [28]. As a result, a total of 331 candidate studies were identified by scanning the title, keywords, and abstract.

The only difference in the selection criteria between this study and the study by Gao et al. [28] is C3, which identifies the studies reporting modeling metrics (instead of the hybrid simulation modeling in the previous study [28]) from all relevant SPS studies. Consequently, in stage III, the 331 candidate studies were checked against the selection criteria by further reading the introduction, conclusion, and even full text iteratively until the final consensus was reached. Candidate papers were assigned to four research students and each paper was assigned to at least two students to enable parallel review. All inconsistencies and disagreements were thoroughly discussed and resolved during weekly meetings with supervisors. In this study, only the latest versions of primary studies were selected for review if different versions were published based on the same model.

In stage IV, we replicated the automated search using the same search string and extended the search scope to 2021. We identified 19 papers that met the selection criteria published between 2016 and 2021.

In stage V, we conducted forward snowballing using Google Scholar. The seminal set for the forward snowballing include "Abdel91" [2], "Kellner99" [1], and "Zhang10" [25], which were cited by most relevant studies from the early stages of the review. We identified 5 studies after deduplication.

4.4 Data Extraction

To answer the research questions, we defined a data extraction scheme to collect important information from the reviewed studies. Each research question is answered by at least one extraction item. As shown in Table 5, the data extraction scheme includes the citation information (e.g., title, year and modeling paradigms) and the information specifics to the research questions. Software metrics (names, units and original descriptions) are extracted for answering RQ1,2,3,4. Causal relationships between (two) metrics and modeling purposes are identified for RQ2 and RQ3, respectively. Data sources and issues and solutions are identified for RQ4.

We started with a pilot extraction, in which 35 randomly picked papers was allocated to all researchers. We noticed that some of the papers clearly introduced metrics used for their SPS modeling or presented the SPS model. For these cases, we can easily identify the software metrics and their causal relationships. Some papers did not present software
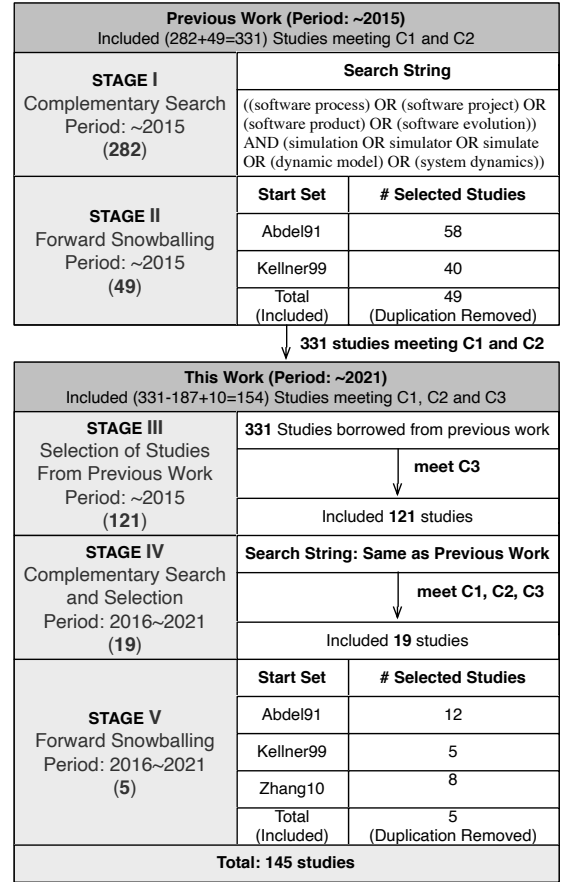
Previous Work (Period: ~2015)		
Included (282+49=331) Studies meeting C1 and C2		
STAGE I Complementary Search Period: ~2015 **(282)**	**Search String**	
	((software process) OR (software project) OR (software product) OR (software evolution)) AND (simulation OR simulator OR simulate OR (dynamic model) OR (system dynamics))	
STAGE II Forward Snowballing Period: ~2015 **(49)**	**Start Set**	**# Selected Studies**
	Abdel91	58
	Kellner99	40
	Total (Included)	49 (Duplication Removed)

↓ **331 studies meeting C1 and C2**

This Work (Period: ~2021)		
Included (331-187+10=154) Studies meeting C1, C2 and C3		
STAGE III Selection of Studies From Previous Work Period: ~2015 **(121)**	**331** Studies borrowed from previous work	
	↓ **meet C3**	
	Included **121** studies	
STAGE IV Complementary Search and Selection Period: 2016~2021 **(19)**	**Search String: Same as Previous Work**	
	↓ **meet C1, C2, C3**	
	Included **19** studies	
STAGE V Forward Snowballing Period: 2016~2021 **(5)**	**Start Set**	**# Selected Studies**
	Abdel91	12
	Kellner99	5
	Zhang10	8
	Total (Included)	5 (Duplication Removed)
Total: 145 studies		

Fig. 2: Thorough literature search process

TABLE 5: Data extraction scheme

Item	Description	RQ(s)
Title	The title of the study.	Info
Year	The published year of the study.	Info
Paradigms	System Dynamics; Discrete-Event Simulation; Agent-Based Simulation; Hybrid simulation, etc.	Info
Software metrics	The metrics involved in the SPS studies, including their names, units, and original descriptions.	RQ1-4
Causal relationships between metrics	the predecessor metric that output to or affects the current metric, and the successor metric that is inputted from or affected by the current metric.	RQ2
Modeling purposes	The reasons form SPS models.	RQ3
Data issues	The issues of selecting metrics due to data availability.	RQ4
Solutions of data issues	The strategies that solve the data issues.	RQ4
Data source	The source of data related to solutions for measuring attributes, one study may use multiple data sources.	RQ4

metrics explicitly; the metrics and causal relationships might be identified from its context by iteratively reading the full text. As a result, we extracted 2130 metrics and identified 183 types of causal relationships from the 145 identified papers. These metrics, which are also called variables, comprise the SPS models.

4.5 Data Synthesis & Classification

We applied the thematic synthesis method [42] to construct our findings in a systematic manner. As the metric

TABLE 6: Example of coding

Metric	Code-I1	Code-I2	Code-I3	Code-I4
E1: Residual defect density (i.e. actual reported defects that were not corrected after 1094 days)	Defect, Density, Residual	Defect, Density, Remaining	Defect, Density	Defect
E2: Number of tasks completed	Task, Number, Completed	Task, Number, Completed	Task, Size	Task
E3: Delay from the completion of this until next release is delivered to users	Delay, Release	Delay, Release	Delay, Release	Time

descriptions vary significantly between process modelers, to answer RQ1, we synthesized the list of the extracted software metrics using the coding technique [43]. It is an iterative process to develop a consistent set of codes from the diverse descriptions in the reviewed studies. Table 6 shows three examples of the evolution of codes. In the first iteration, two researchers coded all the metrics at a detailed level based on their descriptions independently, then we discussed the differences between the two sets of codes, and finally reached agreement. In the second iteration, we identified similar codes in the Code-I1 which developed in the first iteration and replaced them into the same code, for example, we coded 'issue', 'error', 'fault', 'flaw', 'bug' as 'defect'; coded 'fix', 'fixed', 'correct', 'correction' as 'fixing'. For the Example 1 (E1), the 'residual' is replaced by 'remaining' from Code-I1 to Code-I2. Some codes have no synonyms that could be replaced; then Code-I2 would be the same as Code-I1, e.g., E2. We gradually made the code more abstract in the third and fourth iterations through discussion. Some codes did not change from I2 to I3 to replace few similar metrics, e.g., E3. We developed Code-I4 based on Code-I3 and partially referred to the Fenton et al's software metric taxonomy [22] which has been widely accepted in the community. As a result, all metrics can be classified into 29 different categories according to Code-I4, which can be grouped by product internal/external, process internal/external and resource internal/external as suggested by the study [22]. The categories were classified into sub-categories according to Code-I3.

In the process of coding-based classification of metrics, metrics that measure time, effort, and cost, etc. can be easily identified and classified. But we have also encountered some thorny problems, which are not covered in Fenton et al's work [22]. There are a variety of factors and multipliers that affect other diverse variables in the model. We grouped the factors into two major categories, i.e. Process factor and Manpower factor, based on the variables that are affected by them. For example, various policies were modeled in existing SPS models; we classified *pair programming policy* which is a boolean variable that determines whether to apply pair programming in the Process factor; and classified *staffing policy* which determines the number of person months assigned to the Manpower factor. In Fenton et al.'s framework [22], *defect* is the only category related to defects, and they suggested that *defect* should be the Process attribute. In our opinion, although defects are generated and fixed in the process, defects that exist at a certain point in time exist objectively as part of the product. Therefore, we distinguish *defect activity* from *defect* and classify *defect* into Product category as it is more reasonable. We classified metrics such as *defect fixing rate* into *defect activity* which belongs to Process and classified metrics such as # *fixed*

defects into *defect*. Similarly, we perform the classification based on the results of coding (i.e. the abstract meaning of metrics) on the one hand and on the other hand, based on the context of the metrics. We refer to existing classifications, but we are not bound by them.

The content (frequency) analysis was performed throughout the study to analyze the nominal data across groups of variables. To identify the most common causal relationships between metrics, the percentage of the occurrence of causal relationships was counted when answering RQ2.

For RQ3, we counted the frequencies of different categories of metrics used for two hierarchical levels of modeling purposes as well as for different paradigms to locate the evidence that leads to the difference so that we could conduct an in-depth analysis.

5 RESULTS

This section presents the distribution of included studies from three aspects, i.e. publication years, paradigms used in studies and modeling purposes.

5.1 Years

The review identifies 145 papers after the five stages (as presented in Fig. 2). The selected papers for review were published from 1997 to 2021. The complete list of references of the included studies is shown in the APPENDIX.
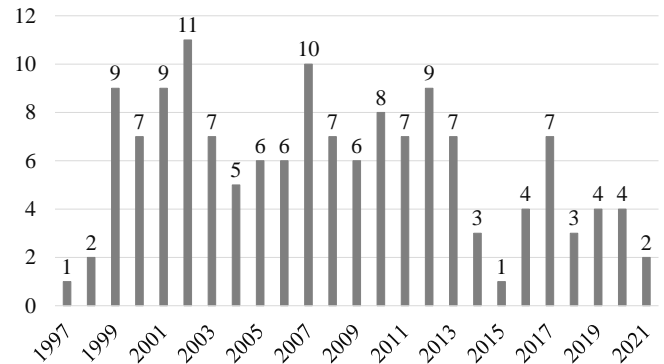

Fig. 3: Study distribution per year

5.2 Paradigms

We classified the modeling paradigms applied in the SPS studies as shown in Fig. 4. System Dynamics (SD, 43%) is the most popular modeling paradigm in SPS. The rest includes Discrete-Event Simulation (DES, 19%), Hybrid Simulation (Hybrid, 12%) and Agent-Based Simulation (ABS, 8%). A hybrid model may adopt two or more modeling paradigms

together. Other studies use a variety of modeling paradigms that simulate software processes at different abstraction levels distinct from the above, e.g., Qualitative Simulation (QSIM), Parametric Estimating (PE), etc.
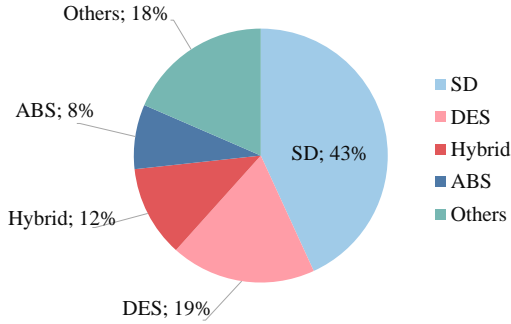

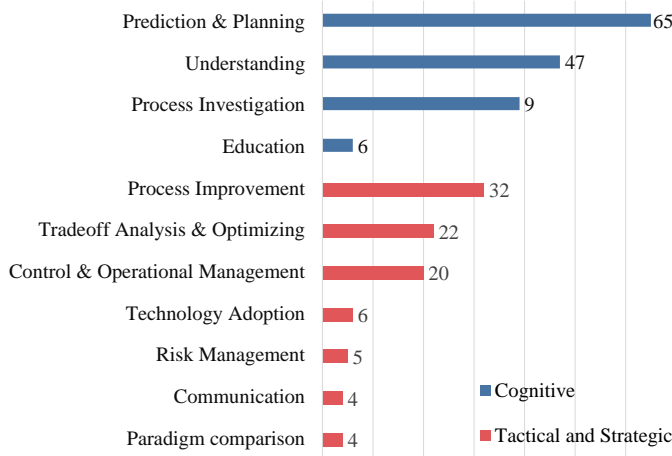Fig. 4: Study distribution per modeling paradigm

5.3 Modeling Purposes

Fig. 5: Study distribution per modeling purpose (One study may have multiple modeling purposes)

Kellner et al. [1] identified the reasons for SPS models and clustered them into six specific modeling purposes to perform the simulation of software processes. Zhang et al. [25] extended it to ten purposes based on the systematic review of published SPS studies. In their study, these purposes were grouped into three levels, i.e. cognitive, tactical & strategic. The cognitive level includes understanding, communication, process investigation, and education. The rest of the purposes are at both tactical & strategic levels.

We grouped the modeling purposes of the studies into eleven purposes (we identified a new modeling purpose). Different from the other ten purposes, the new purpose, paradigm comparison, is related to the modeling paradigm, but not for studying the software process. The purpose is to compare the strengths and weaknesses of different paradigms. In the study whose goal is to compare paradigms, the paradigm comparison is the sole purpose of building the models. For example, a qualitative and a quantitative model of the typical software evolution process

were built for comparing SD with qualitative modeling diagram and there is no redundant discussion about the value of the models themselves [Zhan 09]. The models built for paradigm comparison are simple but complete enough which are based on the degree to which the simulated behaviors interpret the process. From this point of view, these models are at the cognitive level.

As shown in Fig. 5, we grouped 42 studies into the cognitive level and the remaining 103 into the tactical & strategic level. One study may have multiple modeling purposes, therefore, one study may have both cognitive and tactical & strategic modeling purposes. For such research, we consider its modeling granularity to be at the tactical & strategic level. Understanding and process investigation are the most common modeling purposes at the cognitive level. Prediction & planning and process improvement are the most common modeling purposes at tactical & strategic level.

6 FINDINGS

Kaner et al. [44] collected the definitions of measurement, a concise definition they recommend is provided by Fenton and Pfleeger as below [22].

Formally, we define measurement as a mapping from the empirical world to the formal, relational world. Consequently, a metric is the number or symbol assigned to an entity by this mapping to characterize an attribute.

In SPS modeling, a metric is also called variable sometimes, in our opinion, metric is a more appropriate name since it implies the mapping from real software process to the SPS model. However, the boundary between attribute and metric is not strict in SPS studies, e.g., *size of code* and *number of defects* are the common names of two variables, however, the former is an attribute and the latter is a metric according to the definitions in the software measurement area. The metric of *size of code* should be *lines of code*. This kind of ambiguity will cause trouble when we classify them. Hence, we treat all variables as metric and keep their original representation as much as possible.

Although there exists a body of knowledge on software metrics [22], no systematic and comprehensive research on metrics for process modeling has been reported in the SE community. The selection of metrics in SPS models turns out to be more challenging than static models because of the dynamic nature and the extra requirements for executability. Therefore, this study concentrates and reports on the metrics used and studied in SPS modeling only.

6.1 Metrics and Classifications (RQ1)

We extracted a total of 2130 metrics used (or studied) in SPS models from the 145 reviewed studies. There are many identical or similar metrics in the data set; hence, we classified them for analysis. Based on the meta-model, our classification framework would contain four levels of categories, from abstract to concrete, these are entities, attributes, subattributes, and metrics.

6.1.1 Categories of Entities

We refer to categories and definitions in Fenton et al.'s classification [22]. We classify software metrics into three categories as *products*, *processes*, and *resources* (aligned with a Goal-Based framework for software measurement), where *processes* are activities that evolve over time and have to be completed in sequence during development, *products* are generated from process activities including artifacts, deliverables, and documents, and *resources* are the entities needed in performing activities.

In each category, we further distinguished two types of metrics, i.e. *internal* and *external* metrics. An *internal* metric can be measured only by the intrinsic properties of *product*, *process*, or *resource* on its own without considering their observable behaviors. On the contrary, an *external* metric is measured purely by taking impacts that it may make on the *product*, *process*, or *resource* into account.

6.1.2 Categories of Attributes

There is a large amount of studies in software metrics/measurement, however, the roles of metrics in SPS studies are different from the roles in them, which would easily make modelers get confused. For example, the software measurement related studies are full of research on coupling and cohesion whilst these are rarely used in existing SPS models. It implicates that existing classification frameworks, which were built based on software measurement research and knowledge, are not suitable for SPS modeling. Furthermore, one of the most recognized frameworks, Fenton et al.'s classification framework, does not fully cover the metrics used in modeling. In the study [22], only examples of attributes are presented for each entity, which is far from enough to guide modeling.

In this study, we built a new classification framework of metrics used in SPS models following the thematic synthesis method described in Section 4.5. The framework is divided into four figures, i.e. Fig. 6-9, due to the space constraints of one page. In these figures, the first column shows whether the measured attributes belong to an internal or external entity. The numbers in brackets indicate the number of metrics that measure these attributes. The second and third columns show the attributes and their sub-categories. The last column lists typical metrics with their units. Each metric consists of two parts; the upper rectangle shows the name of the metric and the lower rectangle shows its unit. For example, the first metric in Fig. 6 is *size of code* and its unit can be LOC, DSI or Function Points (FPs). The height of the rectangle of a metric is within the scope of the sub-category it belongs to, e.g, *size of code* measures *artefact size*, specifically, code size. It should be noted that *size of code* should be a name of attribute from the semantic point of view, however, these are the minimum units in SPS models. We name these metrics following the coding technique and comply with the original expression of them in reviewed studies. To achieve the goal of modeling, various means (indicate by units) of measuring a specific metric were used in different studies. It is largely determined by the modeling object and the method of measurement that may lead to different units of the metric. Separating all similar metrics (e.g., both LOC and DSI are the metrics of code size) will make the results and discussion too trivial. Hence, we leave out some of the non-essential results.

The unit also implies the scale type of a metric, especially for numerical data. As shown in Table 2. Scale types are nominal, ordinal (restricted or unrestricted), interval, ratio, and absolute according to ISO/IEC 15939. The absolute and ratio are numerical data, which can be indicated by units. For instance, # *defects* is absolute and *defects %* is ratio. Besides, the type name of boolean, interval, and ordinal types are presented instead of presenting the unit since these metrics are dimensionless (i.e. there is no specific unit). For those metrics with a clear ordinal set or range, the ordinal set or range is also presented, e.g., *expertise of developer* commonly have three levels 0,1,2 and *knowledge level* have a finer level of granularity ranged from 0 to 100.

The classification framework presented in the figures is elaborated from *product*, *process*, and *resource* categories as follows.

6.1.3 Product Metrics

As shown in Fig. 6, the internal metrics are further divided into three sub-categories, i.e. *artefact size*, *artefact property* and *defect*.

Artefact Size measures the amount of work product to be produced from the process. Generally, size is used to measure the scale of a project and further compute indirect attributes. Metrics for various types of artefacts were used in different studies, the collected metrics can be classified into five sub-categories as shown in Fig. 6. *Size of code* that measures program size is most used metric in all the size metrics and is associated with the effort of development and maintenance and the faults generated [22]. Meanwhile, *size of requirement document*, *size of design document*, and *number of (#) test cases* are widely used in SPS models. In detail, test cases are generated by either manual or automated approaches, while the latter is in consideration of the tools supporting model-based testing and made by software specifications written in more formal notation or structure [Aran 08]. In some studies such as [Wern 02, Zhan 12a], the artefact size is abstracted into a number of arbitrary-sized "units" in order to represent some suitable measure to the size of the system in reality. The # *user stories* and # *features* are used in different project contexts.

Artefact Property is used to measure products at a more detailed level; e.g., code can be further measured by # *decision statements in the code*, # *local and global variables*, document can be measured by *words per page*, etc. To be specific, # *control flows identified in the skeleton* is used to estimate the effort required to perform the requirements analysis activity in a DES model [Aran 08]. *Coupling degree* and *code complexity* also belong to this category. The former is used only in one study and is measured by the probability that one component will be affected by another [Padb 11]. The latter can be measured in different ways. Smith et al. [Smit 06] used McCabe complexity to measure code complexity. Raffo et al.s [Raff 00] suggest that *number of decision statements* in the code can be the metric of code complexity, *number of local and global variables* and *level of control-flow nesting* are the alternative metric. *Flesch-Kincaid Grade Level Score* and *Gunning Fog Index* are suggested as metrics of document complexity.

ENTITY	ATTRIBUTE	SUB	METRIC		
INTERNAL	Artefact Size (208)	Code	**Size of Code**	**Size of Component**	
			LOC, DSI, FPs	Files, Units	
		Requirement Document	**Size of Requirement Document**		
			Pages, FPs, User Stories, Units		
		Design Document	**Size of Design Document**		
			Pages, FPs, Tasks, Units		
		Test Case	**Size of Test Code**	**# Test Cases**	
			LOC	Test Cases	
		User Story	**# User Stories**	**# User Stories Completed**	
			User Stories	User Stories	
	Artefact Property (84)	Feature	**# Features**		
			Features		
		Code	**# Decisions**	**# Classes**	**# Functions per Class**
			Decisions	Classes	Functions/Class
			# Local and Global Variable	**# Keywords**	
			Variables	Keywords	
			# Control Flows Written in the Use Case		
			Control Flows		
			Code Complexity	**Coupling Degree**	
			McCabe Complexity	Probability	
		Document	**# Statements**	**# Paragraphes**	
			Statements	Paragraphes	
			Document Complexity		
			McCabe Complexity		
	Defect Size (158)	Size	**# Injected Code Defects**	**# Injected Requirement Defects**	
			Defects	Defects	
			Detected Defects in Test	**# Detected Defects after Test**	
			Defects/LOC; Defects/Test Case	Defects	
			# Fixed Defects	**# Remaining Defects**	
			Defects	Defects	
		Density	**Defect Density**		
			Defects/LOC; Defects/Unit		
			Design Defect Density	**Released Defect Density**	
			NA	Defects/FP	
		Percentage	**Defects Detected in Reviews %**		
			%		
			Overlap of Detection of Reviewers		
			%		
			Percentage Fixed Code Defects		
			%		
			Percentage of Fixed Requirement Defects		
			%		
	Defect Property (8)	Type	**Defect Type**		
			Ordinal {Phases}		
		Severity	**Defect Severity**		
			Ordinal {1,2,3,4}		
EXTERNAL	Quality Attribute (32)	Quality	**Code Quality**	**Document Quality**	
			# Defect/LOC	# Defects	
			System Quality		
			# Replacing Requirements		
		Maintainability	**Maintainability**		
			Numeric Data		
		Reusability	**Reusability**		
			Numeric Data		
		Reliability	**Reliability**		
			Numeric Data		
		Usability	**Usability**		
			# Usability Problems		

Legend:
Metric Name
Unit1;
Unit2
…

Fig. 6: Classification of Product Metrics

Defect Size measures the amount of defects that are injected, detected, and may remain throughout the development process. According to the life cycle of defects, they roughly include injected, detected, fixed, and remaining defects. From the perspective of measurement, the metrics are classified into size, density, percentage, and property subcategories. The # *defects* is the most frequently used type. The *defect density* and *percentage of fixed defects* evaluate the quality of the product from two different angles, the former evaluates from the product itself and the latter evaluates from the progress of fixing defects.

Defect Property is used to measure defects at a more detailed level. Properties such as *type* and *severity* are introduced in several studies [Rus 03, Zhan 08a, Cang 08, Lune 21] to make the models behave closer to reality. The *type* can be the ordinal data that indicates the phase of the

defect injected [Rus 02]. The *severity* can be ordinal data with three to four levels, e.g., four levels of defect severity are modeled (i.e. "easy", "medium", "hard", and "very hard") by Zhang et al. [Zhan 08a].

The **external metrics** of the product were not commonly used in SPS. They measure the **quality attributes** of product such as *reusability*, *maintainability*, *quality*, etc. While *quality* is the commonly used metrics in this sector, the means to measure *quality* of code and document vary from model to model. Pfahl and Lebsanft adopted an SD model to analyze the impact of software requirement volatility, the *quality of the system* is measured by the number of replacing requirements [Pfah 00b]. From business value concerns, Madachy [Mada 05] measured the *quality* by the number of defects. Only one study used the *maintainability* metric and measured it using the equation $Maintainability = 13.12 * Complexity + 0.17 * Effort + 3.87 * Size$ [Arau 12]. The *reusability* and *reliability* are commonly based on CO-COMO II, they are used as cost drivers with a nominal value of 1 [Uzza 13b, Silv 13].

6.1.4 Process Metrics

With the focus on process modeling, Fig. 7, 8 indicate that it is a large set of process-related metrics that can be further classified into a number of categories such as *time, task, effort*, etc.

As a distinction between dynamic process models and static process models, there are four main types of **time metrics**. The first is *duration* which denotes the time spent on a single phase or the entire process. The model simulates a real process over a period of time, no matter whether *duration* is an explicit metric indicated in an SPS model. *Calendar date* could be an alternative metric to *duration* when the model has many overlapped duration metrics. In a stochastic simulation model for risk management, *the start dates and end dates of every risk* is measured [Zhou 12]. The third type is *work time* which is often used as a multiplier in the model to simulate # *hours work per day* [Wake 05]. *Delay* is the last type of time metrics such as *hiring delay, delay from completion to release, new requirements feedback delay*, etc.

As an alternative metric of artefact size in SPS, **task size** runs through phases of the development process as an indicator of the job size of the corresponding process (phase).

Change occurs during requirement, design, implementation, or throughout the entire process. It is one of the major reasons for *rework*. From the reviewed studies, this kind of metric is not widely used in SPS models.

Increment represents the process that the new requirements or increments are adding into the development process. Both # *new requirements* added and the *new requirement generation rate* are metrics that measure this process.

Rework is the feedback of *change* and *defect activity*. The *amount and percentage of task that required rework or were reworked* are *rework* metrics. We classified the *rate of rework* into the *work rate* category to emphasize the activity and the 'speed' of the process.

We introduced the **defect** in product entity. There are also some defect related metrics should be process metrics, we classified them into *defect activity*. Metrics in *defect* measures the static attributes of product defect, such as defect density.

ENTITY	ATTRIBUTE	SUB	METRIC						
INTERNAL	Time (300)	Duration	**Interval between Requirements** Days	**Project Duration** Hours, Days, Weeks, Months	**Task Duration** Days	**Fixing Duration** Hours, Days	**Answer time** Days	**Lead time** Days	**Delivery time** Days
			Requirement Selection Time Days/Requirement	**Coding Duration** Days	**Review Duration** Hours, Days	**Test Duration** Hours, Days, Weeks, Months	**Rework Duration** Hours, Days, Weeks	**Issue Assessment Duration** Hours	
		Delay	**New Requirements Feedback Delay** Months	**Domain Discovery Delay** Months	**Analysis Discover Delay** Months	**Design Filling Delay** Months			
		Date	**Project Completion Date** Date	**Project Start Date** Date	**Start Time of Inspection** Date Time	**Time of Occurrence of Risk** Date Time			
		Work Time	**Work Time Multiplier** Hours/Day, Hours/Week						
	Task (114)	Size	**Task Size per Inspection** Pages, LOC	**Test Task Size** # Requirements	**# Maintenance Requests** # Requests	**Amount of Implementation Work** # Tasks, User Stories	**Total Tasks** # Tasks	**Required Tasks** # Tasks/User # Stories	
		Progress	**The Percentage of Task Complete** Numeric Data [0,1]	**The Percentage of Task Increased Due to Requirements Volatility** Numeric Data [0,1]		**The Percentage of Task Reworked Due to Requirements Volatility** Numeric Data [0,1]			
		Rate	**Task Rate** Person Weeks/Task, Hours/Task						
		Dependency	**Task Dependency** Percentage of task-related effort that has to be consumed before a subsequent task can begin						
		Level	**Level of Task Underestimation within Project** Task Underestimation%		**Degree of Tasks to be Rebudgeted** Numeric Data [0,1]				
	Change (54)	Size	**# Requirement Changes** # Changes	**# Design Changes** # Changes	**# Code Changes** # Changes	**Cumulative Changes Released** # Changes			
		Rate	**Rate of Requirement Changes Input** # Changes/Time	**Change Request Acceptance Rate** %	**Rate of Change Completion** # Changes/Day	**Requirements Volatility Rate** %			
		Distribution	**The Percentage of Change in Product** %	**The Percentage of Change in Product Due to Requirements Volatility** %					
	Increment (22)	Size	**Specification Increments** # Specification Docs	**High Level Design Increments** # Increments	**Low Level Design Increments** # Increments				
			Code Increments LOC	**Demand of New Features** # Features					
		Rate	**New Specification Generation Rate** # Specifications/Day	**New Requirement Generation Rate** # Requirements/Day	**New Requirement Feedback Rate** # Requirements/Day				
	Rework (6)	Size	**Amount of Rework** # Reworks						
		Percent	**Percentage of Rework** Reworks%						
	Defect Activity (78)	Injection Rate	**Defect Injection Rate during Coding** Defects/LOC, Defects/Hour Defects/Month, Defects/Task, Defects/Unit		**Defect Injection Rate by Each Developer** Defects/LOC	**Defect Regeneration Rate** Defects%			
		Detection Rate	**Defect Detection Rate** Defects/Hour	**Individual Defect Detection Rate** Defects/Hour					
		Fixing Rate	**Defect Fixing Rate** Defects/Hour						
	Work Rate (117)	Development Rate	**Development Rate** DSI/Day, LOC/Day, FPs/Day, Tasks/Iteration						
		Design Rate	**Design Rate** # Documents, # Documents/Month						
		Implementation Rate	**Implementation Rate** LOC/Day						
		V&V Rate	**Test Rate** LOC/Day	**Component Validation Rate** Components/Time	**System Validation Rate** Elements/Time				
		Inspection Rate	**Inspection Rate** NA						
		Rework Rate	**Rework Rate** Hours/Defect, Tasks/Day	**Rework Discover Rate** Tasks/Hour	**Revision Rate** Tasks/Day				
		Release Rate	**Delivery Rate** Units/Time	**Release Rate** Modules/Month					
	Effort (149)	Total Effort	**Total Effort** Person Hours, Person Days, Person Weeks, Person Months			**Effective Effort** Person Days	**Planned Effort** Person Days		
		Effort per Unit	**Effort per Test Case** Person Weeks	**Test Effort Rate** Hours/Unit	**Test Effort** Person Days	**Meeting Effort** Person Hours			
		Effort per Activity	**Rework Effort** Person Days	**Fixing Effort** Person Weeks, Person Days/Defect		**Preparation Effort** Person Days			
		Effort per Phase	**Design Effort** Hours	**Coding Effort** Person Hours	**Review Effort** Person Days	**Testing Effort** Person Days			
	Overhead (17)	Communication Overhead	**Communication Overhead** Numeric Data						
		Training Overhead	**Training Overhead** Numeric Data						
		Other	**Task Switch Overhead** Minutes; Hours						

Fig. 7: Classification of Process Metrics (Part1)

ENTITY	ATTRIBUTE	SUB	METRIC				
INTERNAL	Risk (13)	Size	**# Occurred Risks** Risks	**# Identified Risks** Risks			
		Impact	**Impact of Risk occurs** Numeric Data [0,1]	**Overall Impact of All Occurred Risks** Numeric Data [0,1]			
	Process Factor (110)	Policy	**Pair Programming Policy** Boolean Switch	**Review Adjustment Policy** Boolean Switch			
		Method	**Coding Method** KLOC/Day due to the method used; # Defects/KLOC due to the method used			**Reading Technique** Ordinal (Different Techniques)	
		Schedule Pressure	**Schedule Pressure** Normal Effort %				
		Discovery Factor	**Domain Discovery Factor** Passed change requests %	**Analysis Discovery Factor** Passed change requests %	**Design Discovery Acceptance Factor** Passed change requests %		
		Other	**Requirements Volatility Multiplier** Numeric Data	**New Requirements Feedback Factor** Numeric Data	**Maintenance Adjustment Factor** Numeric Data	**Effort Multiplier** Numeric Data	
	Personnel Continuity (33)	Rate	**Turnover Rate** Persons/Day	**Hiring Rate** Persons/Day	**Assimilation Rate** Persons/Day	**Learning Rate** Experience/Day	**Loss Rate** Persons/Day
	Process Static Status (20)	Phase of Development	**Software Development Phase** Ordinal (Phases)	**No. Sprint** Ordinal (No.)			
		Other	**# Inspections** Inspections	**# Process Increments** Process Increments			
EXTERNAL	Cost (52)	Single Cost	**Single Cost** $	**Tester Cost** $/Hour			
		Total Cost	**Project Total Cost** $	**Approved Budge** $			
		Activity Cost	**Defect Cost** Hours/Defect	**Cost per Effort** $/Person Days	**Implementation Cost** $	**Deadline Delay Penalty** $/Day	
	Effectiveness (35)	Effectiveness	**Test Effectiveness** Defects Detected %	**Review Effectiveness** Defects Detected %	**Effectiveness of the V&V Process** Defects Detected %		
	Process Performance(6)	Performance	**Cost Performance Index** Numeric Data	**Schedule Performance Index** Numeric Data	**Development Performance** Effort/Hour		

Fig. 8: Classification of Process Metrics (Part2)

Metrics in *defect activity* measures the attributes of dynamic activities, e.g., *defect injection rate during coding*. The rate can be measured by the number of defects injected per time unit, also can be measured by the number of defects per artefact size (LOC).

Work rate represents the 'speed' of activities are executed, which covers the entire life-cycle, including the processes of design, implementation, review, testing, releases, etc. *Development rate* is the most common one, which measures the primary activity simulated in a model. Only using the *development rate* would regard the process as a whole and will not distinguish the speed of different activities in the process. On the contrary, the speed of different activities can be measured respectively by *design rate*, *implementation rate*, etc.

Effort is used (or studied) in numerous models as it is a special interest of process modeling. Its metrics appear as for example *design effort*, *rework effort*, or even *total effort*, in corresponding phases except requirements. The effort can be typically measured with *person-hours*, *person-days* and *person-months* which depends on the resolution of the model. In software process research, *Effort* is regarded as the major contributor to *cost* or its alternative [Rus 14].

Risk rarely appears in SPS models. In the model that aims to investigate the impact of risk, the number, impact and occurrence time of risks are used in the model [Zhou 12].

Overhead results from the situation where extra work beyond the tasks cannot be avoided. *Communication overhead* is the most common type of overhead. We classify them into process category because it is generated from the communication activity (or other activities) in teamwork. Other kinds of overhead such as *task switch overhead* and *training overhead*

are also considered in several studies [Baum 17, Garo 16].

Process factors that generally denote the managerial and organizational factors make impacts through the whole process. In SPS models, *policies* are the most metrics in this subcategory but vary significantly among different models. *Schedule pressure* is a common factor of *productivity* or *work rate*. Various factors were used in studies, which are difficult to exhaust here. Various of *discovery factors* which determines the proportion of elements which might possibly progress to the next stage can be calibrated using historical data [Wern 02]. The determination of what factors should be considered depends on specific project characteristics and organizational characteristics.

Personnel continuity relates to the training and turnover process and metrics measure the 'speed' of these processes. For example, to model the human resource evolution process, the *hiring rate*, *dismissal rate*, *turnover rate* can be considered [Ruiz 01].

Process static status includes the metrics that are predefined and do not change during the modeled process, such as *stage of the development* life-cycle, # *process increments*, etc. The *stage of the development* can be combined with product features such as *system type* to measure the *capacity* [Hurt 15].

Process external metrics are occasionally observed in the reviewed models. They include *effectiveness*, *cost*, and *process performance*, in which *effectiveness* and *cost* were used a lot. For software process, there are many concerns that can be defined as the indicators of *effectiveness*, such as *test requirement defect detection effectiveness* [Cang 08]. Likewise, *cost* is commonly measured by money value as well as effort in SPS studies for various purposes, such as *coders cost*, *estimated budget*, *cost to repair defects* and so forth.

6.1.5 Resource metrics

ENTITY	ATTRIBUTE	SUB	METRIC			
INTERNAL	Manpower (154)	Size	**# Staffs** Staffs	**# Testers** Staffs	**# Experienced Staffs** Staffs	**# Teams** Teams
		Role	**Role in Development** Developer, Tester		**Role of Experience** Experienced, New	
	Environment (65)	Facility	**# Available Inspection Rooms** Rooms		**# Servers** # Servers	
			Test Facility Availability Facilities * Hours/Week		**# Test Machines** # Machines	
			Database Size Numeric Data		**Main Storage Constraint** Numeric Data	
		Other	**Multisite Development** Numeric Data	**# Zooms** Zooms	**Customer Trust** %	
			Time Zone Boolean Switch	**Native Language** Boolean Switch	**Culture** Boolean Switch	
EXTERNAL	Manpower Property (236)	Productivity	**Coding Productivity** LOC/Day		**Testing Productivity** LOC/Day	
			Inspection Productivity LOC/Day		**Productivity of New Staff** LOC/Day	
			Productivity of Experienced Staff LOC/Day			
			Individual Productivity LOC/Day		**Nominal Productivity** LOC/Day	
			Average Productivity LOC/Day			
		Capability	**Required Capability** LOC/Day		**Developer Capability** LOC/Day	
			Capacity for Training Numeric Data [0,1]		**Capacity for Testing** Person Days	
			Capability for Detecting Defects Defects Detected %		**Staffing Level** Numeric Data [0,1]	
		Expertise	**Required Level of Knowledge** Numeric Data [0,100]		**Domain Expertise** Ordinal {0,1,2}	
			Expertise of Developer Ordinal {0,1,2}		**Expertise of Evaluator** Ordinal {0,1,2}	
			Developer's Knowledge Level Numeric Data [0,100]			
		Experience	**Language Experience** Numeric Data		**Tool Experience** Numeric Data	
			Individual Experience Ordinal {0,1,2}		**Average Experience** Ordinal {0,1,2}	
			Inspection Experience Ordinal {0,1,2}		**Analyst Experience** Numeric Data	
		Skill	**Required Skill** Ordinal {1,2,3}		**Skill Level** Ordinal {1,2,3}	
			Coding Skill Ordinal {1,2,3}		**Testing Skill** Ordinal {1,2,3}	
	Manpower Factor (52)	Policy	**Staffing Policy** # Person Months Assigned		**Scaling Policy** Resource Assigned %	
			Anti-Regressive Work Policy Resource Assigned to Anti-Regressive Work %			
		Personal Need	**Self Esteem** Numeric Data	**Achievement Need** Numeric Data	**Staff Dedication** Numeric Data	
		Other	**Training Factor** Numeric Data	**Team Cohesion** Ordinal {0...5}	**Member familiarity** Ordinal {1,2}	
	Utilization (7)	Utilization	**Team Utilization** Not Idle %	**Staff Utilization** Not Idle %		

Fig. 9: Classification of Resource Metrics

Different from product and process, Fig. 9 shows that modelers pay more attention to external entity for resource attributes since only the size or role of the resource can be measured by itself. The resource properties such as manpower skill cannot be measured only directly by itself without any other external reference.

Manpower measures the number of human resources for development. According to the modeling granularity and the scale of the simulated project, studies may regard all the resource as a team or allocate individuals to different phases.

Environment Few studies considered *environment* in SPS models. As an example, *test facility availability* is introduced to model the constraint of resource on test phase [Hous 10]. whether the project is *multi-site development*, *Multi-site development* is a factor suggested by COCOMO II.

To quantify the effect or the value of the manpower in a process, five basic **manpower property**, i.e., *productivity,*

capability, *expertise*, *experience*, and *skill* can be measured. *Productivity* forms the basis to contribute to the *development rate*. The rest four metrics are the factors that determine *productivity*. The rest four metrics are the factors that determine *productivity*. The *expertise* and *knowledge* are the same metric. Different from *expertise*, *experience* already takes the domain knowledge into account, and were often simply measured by years without historical data.

Manpower factor consists of all the influencing factors as metrics related to resource with a great diversity, ranging from *self esteem*, *team cohesion*, to *native language*.

Only the human resource **utilization** is considered in existing studies, although both individual and team *utilization*s were used, they were only occurred in a few studies.

> **Findings:** 1) For most reviewed studies only the metrics that are deemed to be significant are described in detail. 2) Product and process external metrics are not used frequently in process simulation modeling whilst resource external metrics are widely used.

6.2 Causal Relationships between Metrics (RQ2)

The SPS models are built to simulate the process of the development team producing software products by carrying out a series of development-related activities under the constraints and support of the environment. In the process, activities will be organized together in a certain workflow and affect each other, and people are the specific actors of activities. People will be influenced by the environment and various other factors, and may also react to them. SPS models use different blocks (defined by paradigm) to depict different elements in the process, including people, activities, environments, etc. These blocks are instantiated implementations of the metrics discussed in this study in a specific simulation paradigm. At the implementation level, a SPS model is made up of blocks and their relationships to each other. The implementation of the model depicts different metrics and their interrelationships in reality. We discussed the metrics used in existing research in RQ1, and RQ2 will discuss causal relationships between metrics.

We collected all the causal relationships (from metric A to metric B) can be identified in included papers. As a result, we identified 183 types of causal relationships. Table 7 presents high-frequency (with more than 10 occurrences in all SPS models and a relationship may appear multiple times in a model.) casual relationships and we discuss them in more detail below.

From **manpower property** to **manpower property**. It is the most used causal relationship. The most direct relationship between software developers and software development activities is the metric *productivity*, which describes a person's ability to participate in activities. Productivity is the most common one among the metrics of manpower property. *Productivity* can be affected by factors such as *experience* and *knowledge*, which also belong to manpower property [Hana 98, Klun 18, Oors 18]. In addition, *productivity* can be more refined. For example, *real-time productivity* can be composed of *growth* and *baseline/average productivity* [Lehm 10, Zawe 13]. Metrics such as *experience* can also be more refined. For example, the *experience of inspectors* can

be affected by *development experience*, *inspection experience*, and *domain experience* [Neu 02b].

From *defect size* to *defect size*. Defects can be in different states in the software life cycle. The change of state will be reflected in the change of quantity. The *total of defects* can be the sum of *open defects*, *in progress defects*, *waiting to test defects*, *reopen defects*, and *resolved and closed defects* [Zhan 18]. Besides, The *number of detected defects* can be affected by the *number of generated defects* [Zhan 08a]. The *number of escaped defects* can be the difference between the *number of detected defects* and the *number of fixed defects* [Lake 03]. Furthermore, defects can also be described as different types. The *total of detected defects* can be the sum of *detected passive defects* and *detected active defects* [Zhan 08b].

From *change* to *change*. This type of causal relationships was mainly found in study [Zhan 18] and study [Klun 18]. In study [Klun 18], the *sprint change capacity* is based on the *daily change capacity*. The rest of relationships were found in study [Zhan 18], and the *change* is also known as issue request in this work. The *change* is modeled in detail based on the life-cycle of issues in the Issue Tracking System. For example, the *number of issues waiting for review* is affected by the *number of sprint issues*. The *total number of issues* is the sum of *duplicated and invalid issues*, *open issues*, *n progress issues*, *sprint issues*, *issues waiting for review*, *resolved and closed issues*, and *reopen issues*.

From *work rate* to *work rate*. One type of *work rate* may be a composite of many other rates. For example, the *nominal development rate* can be the sum of *experienced employee development rate* and *new employee development rate* [Zhan 06b]. Besides, any type of *actual work rate* can be based on a type of *baseline work rate* [Zhan 18].

From *artefact size* to *artefact size*. Artefact can have different origins or be in different stages of development. The *project size* can be the sum of *remaining size* and *completed size* [Zhan 06b]. The *rate of generating requirements* can be affected by feedback of the *number of generated new requirements* [Zhan 09]. The *number of generated new requirements* can be based on the *number of exogenous requirements* [Hall 05].

From *artefact property* to *artefact property*. The refinement of *artefact property* may not be so common, but it may be very fine-grained. Study [Al E 08] used 15 kinds of different *complexity* metrics to quantify the *comprehensive complexity* of software systems. Study [Aran 08] modeled the relationships among *number of control flows identified in the skeleton*, *number of control flows written in the use cases that required rework after inspection*, and *number of control flows written in the use cases*. Study [Aker 17] used *ambiguity and brittleness of test code* to quantify the *smell of test code*.

From *process factor* to *process factor*. One *process factor* can be broken down into a number of different factors. For example, the *relevance of factoring* is based on a combination of multiple factors including *relevance produktivity degree*, *relevance of sprint course*, *revlevance of reviews*, *relevance of commitment-loyalty*, and *relevance of customer satisfaction*.

From *time* to *time*. *Time* usually describes the duration or delay of an activity. One activity contain a start time and an end time, and the difference between the two is the duration [Zhou 12]. The *duration or delay* of an activity can be based on a *mean or baseline value* [Ruiz 02b]. The *average*

productive time may be affected by the *time loss due to work partitioning* [Hsia 99].

From *manpower* to *manpower*. Manpower can be differentiated based on experience or other factors. The *total number of employees* can be the sum of *number of trained employees* and *number of experienced employees* [Zawe 13]. The *experienced employees* can be the sum of *experienced employees for development* and *experienced employees for training Zhang2006Semi*. The *daily available workforce* can be based on the *total workforce* [Ambr 11].

From *work rate* to *artefact size*. This type of relationship describes a common situation where the size of an aretefact accumulates based on the rate of work. For example, the *specification units to be processed* is based on the *specification unit completion rate* [Wern 99]. In addition, combining the *software development rate* with other factors can also decide that *requirements not met correctly*. Under this type of relationships, the latter is not a simple accumulation of the former [Hall 05].

From *effort* to *effort*. Effort can be differentiated based on different activities. For example, the integration effort can be the sum of *project assessment effort*, *project tailoring effort*, and *glue code effort* [Naun 07]. Furthermore, the *development effort* can be calculated by the *expected development effort* and *growth effort* [Choi 06].

From *environment* to *manpower property*. The environment is the main factor that affects manpower. For example, the environmental factors such as *market salary*, *working environment*, *team management* and *reward* may affect the *motivation* of developers [Fate 18].

From *manpower property* to *work rate*. The rate of development (*work rate*) is based on human productivity (*manpower property*). For example, the *new employee development rate* is based on the *new hired workforce productivity* [Zhan 06b].
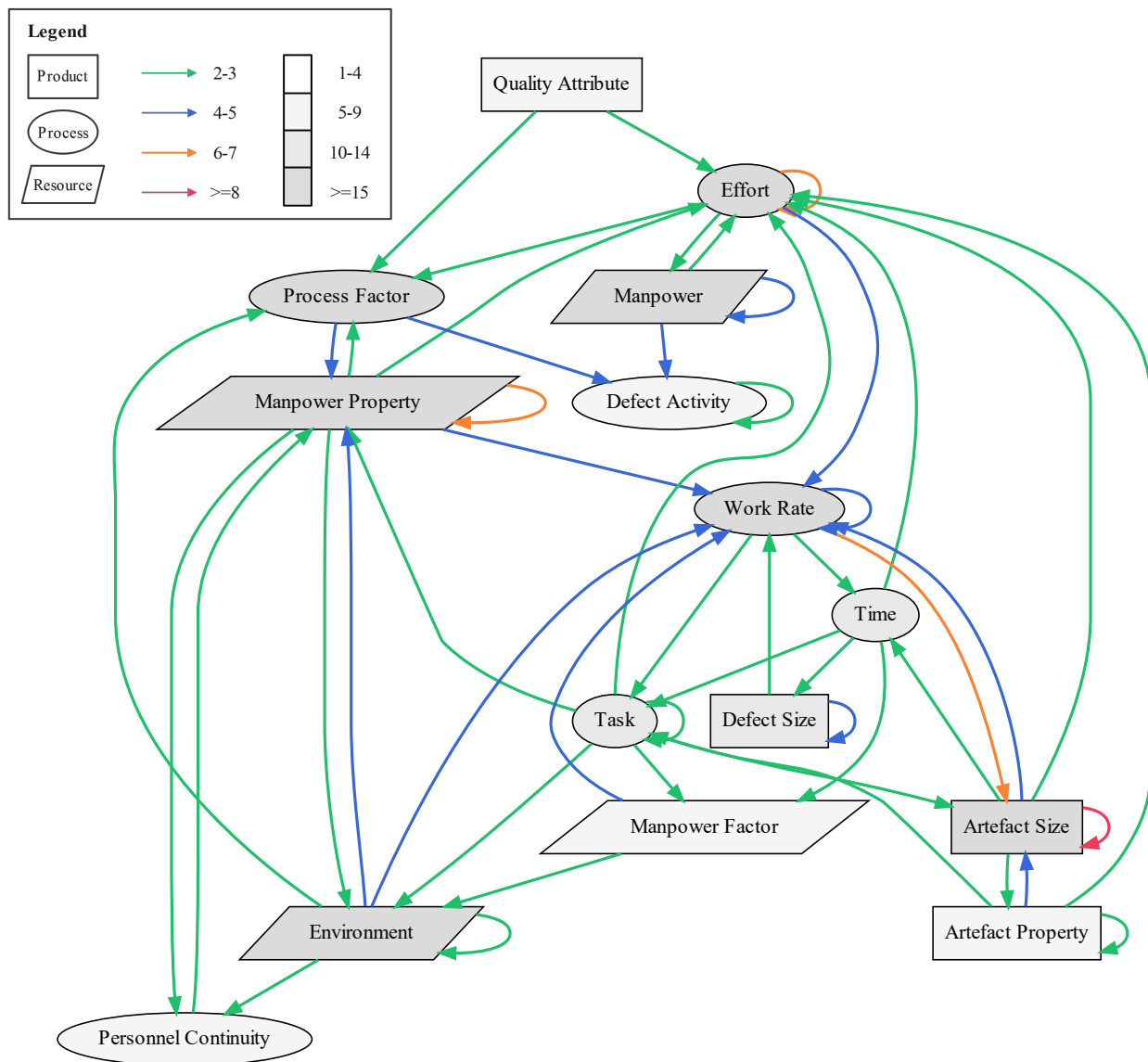

Findings: 1) High-frequency causal relationships are mainly relationships between metrics of the same type. It indicates that SPS models typically refine the main types of metrics such as *manpower property*, *defect size*, *work rate*, *change*, *artefact size*, *artefact property*, *process factor*, *time*, *manpower*, etc. 2) Furthermore, even though the main line of the software life cycle is so clear, we do not see a relationship appearing in more than 10% of the models. This fully demonstrates the diversity of SPS models, which requires specific analysis of specific problems as well as even the same problem may be implemented differently.

6.3 Metrics Selection Based on Purposes and Paradigms (RQ3)

As shown in the meta-model, the metrics selected for building the simulation model is determined by the information needs for building the corresponding conceptual model which depends on the modeling purposes. Kellner et al. [1] also indicated that many aspects of what to simulate are inter-related and driven based on the purpose. Modelers simulate process at different granularity levels since it is nearly impossible as well as not necessary in some cases for a modeler to simulate every detail of a process. Hence, the

TABLE 7: High-frequency casual relationships between metrics

Casual Relationships		Frequency	Citations
From	To		
Manpower Property	Manpower Property	44	[Mada 07, Lehm 10, Zawe 13, Neu 02b, Neu 02a, Zhan 08a, Cher 10, Spas 12, Oors 18, Hana 98, Fate 18, Klun 18]
Defect Size	Defect Size	22	[Lake 03, Mart 00, Zhan 08a, Zhan 08b, Zhan 18]
Change	Change	22	[Zhan 18, Klun 18]
Work Rate	Work Rate	20	[Zhan 06b, Wern 02, Oors 18, Wern 99, Zhan 18]
Artefact Size	Artefact Size	19	[Zhan 06b, Hall 05, Aran 08, Zhan 09, Wern 99, Zhan 18, Klun 18, Duga 21]
Artefact Property	Artefact Property	18	[Aran 08, Shuk 12, Aker 17]
Process Factor	Process Factor	15	[Oors 18, Zhan 18, Klun 18]
Time	Time	15	[Hsia 99, Zhou 12, Mish 16, Wake 04, Ruiz 02b, Zhan 18, Klun 18]
Manpower	Manpower	12	[Zawe 13, Mada 05, Zhan 06b, Ambr 11, Tram 16, Kahe 01, Zhan 18]
Work Rate	Artefact Size	12	[Lehm 10, Wern 02, Hall 05, Zhan 09, Wern 99, Zhan 18, Duga 21]
Effort	Effort	10	[Choi 06, Naun 07, Aran 08, Noll 16]
Environment	Manpower Property	10	[Hurt 15, Fate 18, Klun 18]
Manpower Property	Work Rate	10	[Lehm 10, Zhan 06b, Mada 00, Wern 02, Zhan 08a, Cocc 11, Mish 16, Oors 18]

Fig. 10: Diagram of causal relationships between metrics in cognitive models (frequency ≥ 2).

Fig. 11: Diagram of causal relationships between metrics in tactical & strategic models (frequency ≥ 4).

selection of the appropriate metrics is critical to meet the needs of different modeling granularity. To a great extent, it depends on the specific purpose of a study. Existing SPS models were classified into Cognitive Models (CMs) and Tactical & Strategic Models (TSMs) according to their modeling purposes as shown in Fig. 5. RQ3 discussed the commonalities and differences between CMs and TSMs from the perspective of causal relationships used.

In addition, the diversity and complexity of software processes, which can be reflected in modeling purposes, determine the different capabilities of simulation paradigms needed [25]. RQ3 will also discuss the differences in modeling granularity between different paradigms.

6.3.1   Comparison between two levels of purposes in modeling

We identified 217 causal relationships (the same type can be counted multiple times according to the frequency) in CMs and 462 causal relationships in TSMs. Fig. 10 and Fig. 11

present diagrams of causal relationships between metrics in CMs and TSMs respectively. If we included all the identified relationships, the diagram would be too complex to be understood. By compromise, we end up with a frequency greater than or equal to 2 times (about 1% of the total) as a threshold for CMs. Considering that the number of relationships identified in TSMs is approximately twice as many as those identified in CMs. We set the threshold as 4 times for TSMs. As a result, the Fig. 10 contains a total of 51 types of causal relationships (the same type is only counted 1 time) consisting of 15 types of metrics, and the Fig. 11 contains a total of 39 types of causal relationships (the same type is only counted 1 time) consisting of 17 types of metrics.

In diagrams, we use box, oval, and parallelogram to denote product, process, and resource metrics respectively. The darker the fill indicates the higher the frequency of the metric used. We used different colored arrow curves to indicate the frequency of the relationship. The green, blue,

orange, and red colors indicate frequencies greater than or equal to 2 (4), 4 (8), and 8 (16) for CMs (TSMs) respectively.

From the perspective of metrics: There is a clear overlap in the metrics of high frequencies in CMs and TMs, including *artefact size*, *defect size*, *work rate*, *task*, *time*, *process factor*, *manpower*, and *manpower property*. These all represent the basic elements of a software process, namely actors, artefacts, and activities. With these metrics, it is possible to describe a software development process at a macro level. This is probably the reason for their high frequency. The frequency of *effort* is not high in TSMs but high in CMs. The frequencies of *change* and *defect activity* are high in TSMs but not high in CMs. Furthermore, there is *effectiveness* in TSMs but not in CMs.

From the perspective of causal relationships: There are four types of relationships that are high-frequency in both CMs and TMs, including *from artefact size to artefact size*, *from defect size to defect size*, and *from manpower to manpower*, *from manpower property to manpower property*. It shows that CMs and TMs have commonalities in the fineness of modeling artefact, manpower, and defect, that is, to distinguish different artefact, manpower and defect, as well as the properties of manpower would be further refined. It should be noted that although *from manpower property to manpower property* is a high-frequency relationship in both CMs and TSMs, its frequency is as high as 38 (8.2%) in TSMs, but the frequency in CMs is only 6 (2.8%). Besides, the frequencies of *from defect size to defect size* in TSMs and CMs are 18 (3.9%) and 4 (1.8%) respectively. Most of the high-frequency relationships in CMs are in the form of a relationship *from one metric to work rate*, whilst these are not high-frequency relationships in TSMs (except *from work rate to work rate*). Even so, the types of relationships around *work rate* are the most diverse in both CMs and TSMs. *From time to process factor* and *from manpower to time* are high-frequency relationships in TSMs, whilst there is no such relationships in CMs. *From effort to effort* is high-frequency relationship in CMs, but not in TSMs. *From artefact property to artefact property*, *from time to time*, *from process factor to process factor* are high-frequency relationships in TSMs, but not in CMs.

> **Findings:** There are significantly more causal relations between metrics of the same type in TSMs, whilst relationships around *work rate*, *artefact size*, and *effort* are more frequent in CMs. It indicates that CMs tend to directly model around the final result (e.g., effort, output artefact) at a macro-level granularity, whilst TSMs are more concerned with a micro-level granularity. TSMs tend to use multiple metrics within the same type to model an aspect (such as *process factor*, *manpower property*, *defect size*, etc.) in detail.

6.3.2 Comparison of different modeling paradigms in modeling

In order to simulate processes with different granularity, it is need to adopt appropriate modeling paradigms. For example, ABS is able to simulate each developer as an agent to study the impact of *module complexity* on *individual productivity* and *motivation*. At this fine-grained level, where different developers need to be simulated separately, the model would be very complex using the SD paradigm. In the existing research, the DES and ABS paradigms have never been used in models that only stay at the cognitive level. Using different simulation paradigms mean that some of the metrics used will also change. Metrics that measure the same attribute need to change for different simulation paradigms, which is indirectly determined by the modeling granularity. Below are some examples to illustrate the main differences.

Probability metrics which indicates the probability that an event or state may change only used in DES and statistical models. For example, a DES model used *the base probability* $P(D_k^i(t))$ that specifies how likely it is that team i will finish component k after having worked on it for t time units [45]. If it is in an SD model, *# components per day* may be used to indicate the work rate. For paradigms that are good at macro-processes, like SD, it would be complicated to simulate activities that different teams implement different components into separate events.

For similar reasons, ABS models are better at modeling different developers in details. For example, Cherif and Davidsson [46] built an ABS model that simulates the difficulty of task j as *the difference between level of knowledge* b_{ij} *and the required level of knowledge* θ_j, where b_{ij} denotes *developer* i *'s knowledge about activity* j. Cherif and Davidsson compared the ABS model with SD model and indicated that the main difference is that an SD model inputs average value of individual characteristics (*manpower property*) as an alternative.

> **Findings:** Compared with the SD model, DES and ABS can provide more detailed simulations from the perspective of development activities and individual developers, respectively, which is reflected in the more detailed metrics they use. In contrast, SD tends to use the mean as an alternative. This reduces the difficulty of obtaining metrics under the premise of less impact on the macro-process research.

6.4 RQ4: Revisit Data Issues and Coping Strategies

As early as two decades ago, Raffo and Kellner et al. [1, 39] discussed several situations that might arise when measuring process metrics in simulations and the possible coping strategies, which laid the foundation for subsequent research on SPS models. SPS models have encountered many new challenges over the years. Data issues were undoubtedly one of the major challenges of SPS.

As a result of reviewing 145 included papers, despite the fact that not every paper fully discussed their own data problems through modeling, we still summarized most of the data problems encountered during the data preparation stage in simulation modeling through retrieval. Furthermore, we included two related studies [39, 47] as a supplementary data source for RQ4. We identified these two studies in the literature selection stage. Although these two studies did not meet our selection criteria C1 and C2, they reported experiences related to data issues. The inclusion of these two studies can help us to discuss data issues more comprehensively.

TABLE 8: Data issues encountered in SPS data preparing process (evidence from our included studies and related work).

Data Preparing Steps	Encountered Data Issues	Specific Issues	Metrics(E.g.,)	Ref.
Data Provenance	**Availability**	No readily available meta data in real-world projects Variables in the model required data not available in the literature Records missing No quantitative information available	*Rework Effort* *Requirements Volatility; Affected Job Size(%)* *Not Specified* *How TDD decrease Defects Injected Numbers*	[Ruiz 01, Ruiz 02a, Neu 02b, Berl 03, Ruiz 04, Choi 06, Chen 06, Turn 06, Park 07, Topi 08, Ferr 09, Dan 13, Baum 17], [39]
	Scarcity	Sample used were limited to a few projects Scarcity of available information on samples	*Not Specified* *#Maintenance Requests*	[Turn 06, Conc 13, Hurt 15], [39]
	Reliability of metadata	Systematic biases are likely to exist in the repository data Expert estimates are impeded by a lack of intuition for some parameters	*Task's State* *Not Specified*	[Kous 07, Baum 17],[47]
	Diversity	Data diversity due to different organizational structures and project environments	*Not Specified*	[47]
	Traceability	Data "silos" between different artifacts and sources	*Requirements; Defects*	[47]
Data Collection &Processing	**Understanding**	Misunderstanding of the real meaning of the data Difficult to find accurate descriptions for data fields because of distributed in various data sources	*Not Specified* *Not Specified*	[47]
	Accessibility	Inability of various interfaces	*Not Specified*	[47]
	Time-span	Difficulty in choice of that time-span of historical data	*Not Specified*	[47],[Conc 13]
	Processing	Time and effort consuming	*Software Developer Time & Project Costs and Effort*	[Pfah 99, Raff 00, Ferr 09]
		Considerable variability, outliers and noise	*Not Specified*	[Conc 13, Lune 17], [39]
Data Measurement	**Definition**	Loosely defined	*Not Specified*	[39]
	Granularity	Simulation model is an abstraction of real-world process, the metrics are coarse measures than actual data records.	*Not Specified*	[Raff 03, Al E 08, Zhan 12b]
	Accuracy	Many of model's parameters could be estimated only,the accuracy of the measurements is questionable	*Not Specified*	[Baum 17], [39]
	Completeness	Not possible to measure all variables and their relevance accurately	*Not Specified*	[Choi 06],[47]
	Measurement Method	Existing studies rely on the use of industrial data averages, expert estimates, or values acquired from analytical models	*Not Specified*	[39]
	Quantifiability	Unquantifiable variables	*Skill Levels, Manpower, Communication Overhead, Review Support, Process Maturity, and Tool Support values*	[Choi 06, Armb 03, Conc 13, Lune 17]
		Unquantifiable variables' relationships	*Interactions among developers->Maintenance efforts*	[Choi 06, Xiao 10, Conc 13]
	Dynamic	Do not know the empirical distribution of variables variations	*Growth and changes of developers' behaviors; effort variations; developers' allocation*	[Baum 17, Lune 21]
	Reliability of outcome	Does the above measure provide reliable	*Not Specified*	[Choi 06, Klun 18, Lune 21]

6.4.1 Data issues encountered in SPS data preparing

Specifically, we focused on the data issues encountered in the construction of SPS models. To facilitate understanding, we divided the data issues into three steps that align with the data preparation process of SPS, namely data provenance issues, data collection and processing issues, and data measurement issues.

Data Provenance Issues.

1) Availability. Software processes generate all sorts of complex data, and the availability of data has always been the greatest challenge in simulation modeling research. It was evident from all 17 studies that data availability was a problem. Not all activities are documented, and most of the data are not available through relevant literature. There are missing data records, for example, there are no records for *rework efforts* on any activities [Choi 06]. In particular, quantitative data sources are unavailable (e.g., *rate of job size added by requirements volatility* [Ferr 09]).

2) Scarcity. Even if data are obtained, the data scarcity problem is a common issue for modelers. In most simulation studies, only a few data were available for simulation since the histories of selected projects are too short [Conc 13]. A general problem with the record data in projects is that it *does not provide adequate information* to simulate the real variable change and interrelationship [Turn 06].

3) Reliability of metadata. Modelers would be concerned about the authenticity and reliability of the data when they come into contact with the real data source. Both quantitative and qualitative data is obtained from software repositories, manual record documents and expert estimation whilst these data sources are more or less error-prone. For example, software developers may incorrectly choose defect types, forget the defect location time, delay in updating the requirements and planning schedule documents [Baum 17],[47].

4) Diversity. As a result of the differences in data types, record forms, project processes, and organizational

structures, there will naturally be a variety of process data [Hurt 15]. Because of different organizational and project development environments, data sources can be distributed in a variety of online and offline sources without direct correlation. If data from multiple stages and activities of the software process are involved in the simulation model, it is challenging to ensure the consistency of the data.

5) Traceability. It is of interest to restore the true evolution of process data by building traceability between them. There is a lack of discussion of this aspect of the problem in the previous studies and how they deal with it. Ali et al. [47] noted that real-world projects have "silos" between process data information. The requirements repository stores process information about requirements, while defect reports are stored in another database, i.e. issue tracking system. As a result, despite the existence of both data points, we are unable to use them since their connections have been missed.

Data Collection & Processing Issues.

Even the data sources for modeling are available, the modeler still struggles with how to collect the data and how to process it to meet subsequent simulation modeling demands.

1) Understandability. The process data may distributed in various data sources across various departments, therefore it might be difficult to understand accurate meanings of descriptions for data-fields in documentation [47]. Modelers must consult domain experts frequently about data fields, even if the name seems straightforward. The field names differ between organizations, teams, and databases. For instance, in the issue tracking system, the release version number may be defined as "baseline", whilst it might be defined as "version" in the requirements management repository.

2) Accessibility. In order to export data from multiple data sources, diverse interfaces must be accessible. Accessing the interface still requires permissions and authorizations from various business departments, making the data collection process extremely complicated and time-consuming. A risk of data collection failure exists due to data sensitivity and permission concerns.

3) Time-span. It has been recommended by Ali et al. [47] to discuss the selection of historical data time-span for the construction and calibration of SPS models with domain experts. In order to simulate the change distribution of variables and the interaction between them, a long enough time period is required [Conc 13]. When simulating the current reality with historical data, sufficient historical data is required so that we can understand potential changes adequately. For example, if the development process, programming language, or development platform changes, modelers need to be aware of these changes.

4) Processing. Obtaining, collecting, processing and analyzing data is undoubtedly a complex process that requires considerable time and effort [Pfah 99, Ferr 09, Raff 00] due to issues related to data diversity, reliability, understandability, etc. The current data collection and processing activities in SPS research still relies on manual methods, e.g., data format conversion, outlier and noisy data processing.

Data Measurement Issues.

The primary purpose of data preparation should be to provide the data necessary for the simulation model to measure model metrics, which requires measuring the real values of model variables and their relationships. Due to this, SPS places a great deal of importance on the measurement of metrics. Based on a summary of measurement issues reported in previous works, we classified them into eight categories, which are definitions, granularity, accuracy, completeness, measurement method, quantiability, and dynamic of metrics, as well as the reliability of outcome of SPS models.

1) Definition. The meta-model illustrates that the metrics used in SPS models are based on the information needs used in the construction of the conceptual model, which depends on the purpose of modeling. Due to cognition differences, modelers may not be able to comprehend the details of the development process. They might ignore the difficulty of measurement and define inappropriate metrics as a result.

2) Granularity. Often a modeler cannot simulate every detail of a process because it is either impossible or not necessary to do so. Metrics should therefore be selected and measured based on the modeling granularity. However, in many cases, real-world data is too fine-grained to match the assumptions of the model [Zhan 12b]. As an example, SD model assumes no difference between the input variables (e.g., new requirements). However, there is a huge difference between different requirements in actual circumstances, including the difficulty, priority, dependency, and the amount of developers should be invested in them [Al E 08]. Both of these factors may affect the size of the job.

3) Accuracy. The accuracy of measurement is also a common problem in SPS modeling. The accuracy is often questioned due to the quality of the data and the choice of measurement methods. When available data sources are limited, many variables in the model have to be estimated by experts or referred to other literature. Expert estimates, however, are derived from subjective perceptions and always hampered by a lack of intuition [Baum 17, Klun 18].

4) Completeness. Moreover, not all variables can be accurately defined and measured. Due to issues relating to data availability, quality, and metric measurement, modelers have to adjust the structure and scope of the model, which may also affect the completeness of SPS models.

5) Measurement methods. Measurement methods typically include mean value estimation, probability estimation, distribution fitting, and stochastic simulation using Monte Carlo. Moreover, existing SPS studies use industry averages, expert empirical estimates, or values derived from analytical models [48]. These methods are far too simplistic and wild to meet the modeling demands of capturing real changes of variables and their reciprocal effects.

6) Quantifiability. Many process metrics, such as human aspect factors [Choi 06] (e.g., skill levels, interactions among developers), are difficult to quantify, but are extremely critical for the simulation model because they influence productivity greatly [Lune 17]. Quantifying the causal-relationships between variables accurately is a classic challenging task[47].

7) Dynamic. Despite improvements in data sources and quality, modelers still are not able to fit the true dynamic distribution of variables from historical data. For example,

TABLE 9: Coping strategies

ID	Coping Strategies	Ref.
CS1	Directly go to the source documents and repositories.	[Baum 17, Raff 00, Menz 02]
CS2	Look for data in other parts of the organization.	[Menz 02],[47]
CS3	Develop a survey to collect the needed data.	[Menz 02, Ferr 09]
CS4	Work with experienced process participants or using experts' opinion, e.g., developer, project manager.	[Pfah 99, Menz 02, Chen 06, Topi 08, Rus 14], [39]
CS5	Get data from the literature.	[Berl 03, Ruiz 04, Turn 06, Rus 14]
CS6	Use approximate data or easily configurable data.	[Kous 07]
CS7	Adjust model variables: another calculated method, using replacement metrics, adjust scope of variables.	[Baum 17], [39]
CS8	Adjust model scope.	[39]
CS9	Adjust the scale of historical data.	[Conc 13]
CS10	Drop the variable from the model.	[39]

it is difficult to measure the growth of developers as well as the impact of the growth on their behavior [Lune 21].

8) Reliability of outcome. All of the data preparation work ultimately serves to construct and calibrate the simulation model. This in turn would affect the reliability of the simulation results directly. In other words, it is challenging to validate and ensure that the artefacts, activities, and actors modeled in an SPS model are reliable enough to be able to generate plausible results [Choi 06, Lune 21]. This is something we ultimately need to address to meet the modeling purpose.

Findings: 1) Throughout the evolution of modeling work, the availability of data has been a major concern. 2) Additionally, modeling requirements for the model metrics (i.e. modelling changes of real process variables and the mutual influence relationships) present a big challenge to modelers. 3) There would be specific problems encountered at various stages of the preparation process, including problems with reliability, diversity, traceability, historical data time span, the data interface, the format of the data, and the processing time, etc.

6.4.2 The relationship between data issues and coping strategies

We identified ten coping strategies (CS1-CS10) as shown in Table 9, of which most of them have already been discussed by Raffo and Kellner [39], CS3 and CS9 are newly discovered strategies. Figure 12 shows a visualization map that allows to assess the relationship between data issues, the strategies used to cope with them, and the number of evidence in SPS studies provided to support those conclusions.

Since data availability has been a major challenge for modelers over the years, researchers have proposed a variety of solutions. When it is possible for a modeler to access multiple data sources, they could simultaneously use the source documents, repositories, and working with experienced practitioners, getting estimate values from experts, getting data from the literature, etc., to reduce bias. There three types of common data sources can be used when the available data for modeling is not sufficient as shown in Table 10, which are experienced experts, literature, and organization's documents.

Experienced experts (CS4) is one of the main data sources for collecting and verifying metrics. For example, a manager in the distributed support department was conducting a process improvement program and provided us with an estimate of the range of potential productivity improvements [Pfah 04]. Furthermore, advice from experts is especially needed for qualitative data.

Literature (CS5). In some cases, If project data are not available statistical, metrics and typical functions from literature can be used initially. A significant part of the cause and effect relationships covered in the Quality Assurance model is quantitatively supported by the data provided in the study [51], and they provides data concerning errors, costs and duration. In the study [49], the effect of different integration starting points on delivery time, productivity and cost were analyzed. To calibrate GENSIM 2.0, Frost et al. [52] provided an example of a defect containment matrix from which values for the calibration fault injection rates and verification effectiveness can be derived. Wagner [53] provided much data on typical (average) verification and validation rates, and rework efforts for defects of various document types.

Organization's documents (CS1, CS2). Organizations maintain a wide variety of documents which contains a number of metrics. Different organizations adopt different processes and use different tools, hence, the available evidence could not cover all types of documents.

If measurement data sources are not available, it is suggested to use an approximation value (CS6) or adjust variables' measurement methods (CS7). In case of all the above measures fail, modelers may need to simplify the scope and structure of the model again, or maybe just drop the variables.

Studies[Pfah 99, Menz 02, Chen 06, Topi 08, Rus 14], [39] indicated that modelers work closely with experienced practitioners to fully understand software processes, so as to build models based on an in-depth understanding of the data. It is necessary to consult internal experts to determine whether the project has historical data of a certain scale within it and whether the necessary information has been recorded so as to avoid the issue of data scarcity in modeling.

As a means of avoiding repetitive data collection and processing steps due to information asymmetry and cognitive differences, researchers recommend integrating software repositories, data documents, and internal practitioners' opinions together. Without those strategies, manpower and time would be wasted. There is little discussion of misunderstandings, data accessibility risks, or time spans for history data, even though these issues were raised by Ali et al. [47] and other modelers may face similar challenges.

There are not many novel strategies provided by researchers for data measurement. Aside from that, we observed that the measurement problem still has a number of difficulties, such as matching modelling granularity, quantifying variable models, fitting dynamic distributions to variables, etc., for which there were few papers that gave appropriate countermeasures. Undoubtedly, this will result in modeling bottlenecks. As a result of this situation, the subsequent simulation work might be held to an even higher standard. A subsequent study is expected to examine
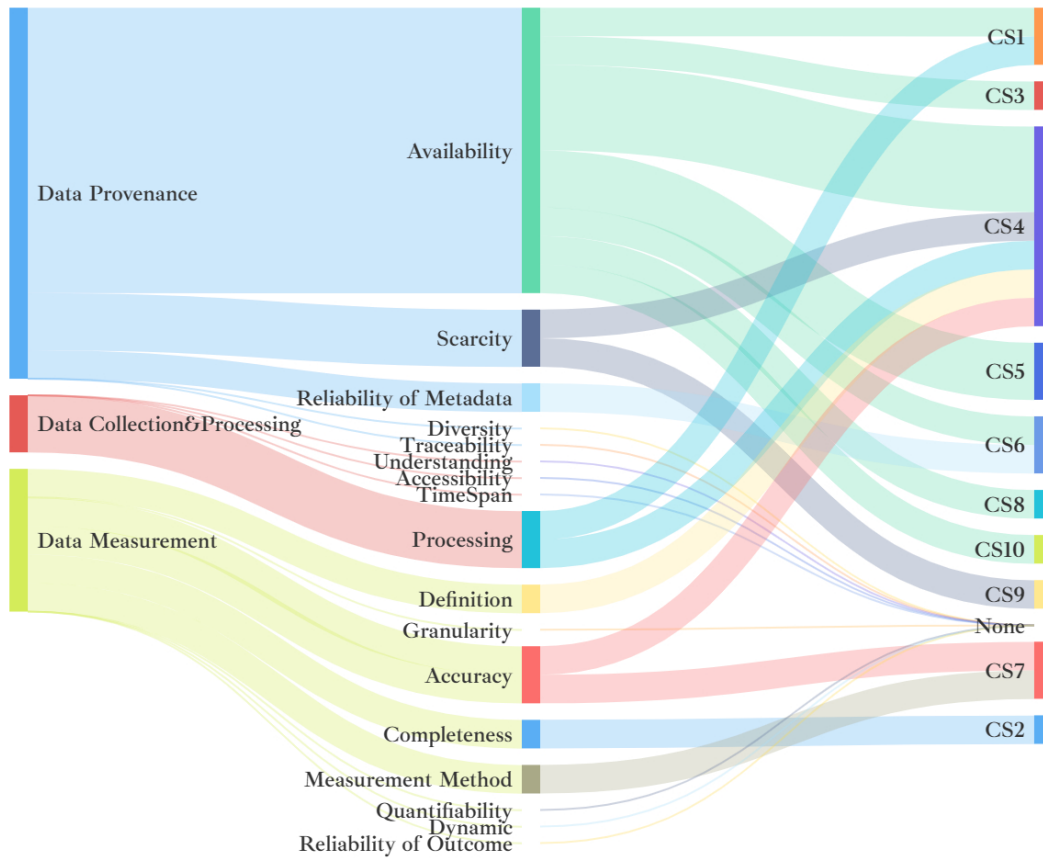
Fig. 12: The relationship between data issues and coping strategies

these problems in more detail and provide comprehensive and detailed solutions, thereby serving as a useful guide and practice for modelers.

> **Findings:** 1) We found that relatively few SPS studies discussed appropriate strategies for coping with specific data quality problems from the data provenance aspect, such as reliability of metadata, diversity of process data, and data traceability, etc. 2) There are also a lack of details regarding effective communication with practitioners, minimizing data accessibility risks, and selecting the appropriate time-span for project history information. 3) In terms of data measurement, researchers have not provided many novel strategies. However, it is really critical to solve these issues in the construction and application of SPS models in real-life projects. Further studies are expected to provide comprehensive and detailed solutions to these issues.

7 THREATS TO VALIDITY

We check the validity from four aspects based on the mapping provided by Zhou et al. [54].

Construct Validity: Finding all relevant papers was a common threat for all SLR studies. We may have missed a few related studies for SPS in paper selection process To overcome this, we have done extensive research on SPS

and using the search string has been validated in previous studies. To search the papers published after 2015, we also used Wohlin's forward snowballing strategy [55].

Internal Validity: Selection bias is also a standard threat to all SLRs. Our selection method was based on the QGS method [56] and snowballing in order to reduce bias. A forward snowballing strategy was employed using three seminal papers which were cited by most relevant studies with higher citations in SPS research in order to extend the pool of relevant studies until 2021. Another threat was how to minimize inaccuracies when extracting data. Study selection and data extraction are mainly carried out independently by two students, and they conduct a cross check before synthesizing the data. The supervisor and students met weekly to discuss all the inconsistencies and disagreements. One threat is that most studies did not provide a complete list of the metrics they used, much less present the details of whether a metric is calibrated or not. We only extracted the metrics and their relationships reported in the study that were considered as the important and general metrics by the authors.

External Validity: We ensure that the primary study of the selection is high generalizability which may lead to the high generalizability of the study conclusion.

Conclusion Validity: During the data synthesis phase of the SLR, grouping extracted metrics and data issues was another challenge. Based on Fenton et al.'s classification [22]

TABLE 10: Available data sources

Data sources	Metric	Ref.
Experienced experts		
Experienced programmers and testers	*Amount of incoming work; # programmers; # injected defects per day per programmer,* etc.	[Berl 03]
Industrial project managers' intuitions and experience	*workload for the activity; developers' abilities; development duration*	[Hana 02]
A manager in the distributed support department	*range of potential productivity improvements*	[Pfah 04]
Survey with professionals of different areas	*effort*	[Shuk 12]
Past organizational data and expert judgment	*estimated requirements volatility*	[Mada 00]
Expert judgment	*review skill; global blocker issue risk; global blocker issue suspend time; review remark fix duration; issue assessment duration; planning duration; task switch overhead; review fix to task factor,* etc.	[Baum 17]
Literature		
Abts et al. [49]	*delivery time; productivity; cost*	[Ruiz 04]
Lum et al. [50]	*cost*	[Uzza 13a]
Jones et al. [51]	*# defects; costs; duration covered in the QA model*	[Drap 99]
Frost et al. [52]	*defect injection rate; verification effectiveness*	[Garo 09]
Wagner et al. [53]	*typical (average) verification and validation rate; rework effort for defects of various document types*	[Garo 09]
Organization's documents		
Project management documents	*schedule; effort; code size*	[Pfah 00a, Menz 02, Raff 02]
Individual inspection reports	*defect detection rate; inspection effectiveness*	[Menz 02, Raff 02]
Holon history records	source code;change history data; estimate of WIP	[Chat 00]
Company's ticket system	*defect injection rate; task duration; review duration; issue fix overhead; # developers; dependency*	[Baum 17]
Time reporting system	*ratio of new development to rework*	[Pfah 04]
Configuration management system	move-to-production statistics	[Pfah 04]
IT recruiting and training department	*attrition rate;* training statistics	[Pfah 04]

on metrics into three categories, we strictly followed the definitions of the different categories to ensure the correctness of metric classification. To construct a systematic set of findings, we applied the thematic synthesis method [42] and coding technique [43] in an iterative process. This allowed us to develop a consistent set of codes based on the diverse descriptions of metrics. By applying thematic synthesis [42], we summarized the data issues encountered in the construction of SPS models by aligning them with the data preparation process of SPS.

There is also a common threat for us, which is the lack of details from primary studies. Most of our primary studies did not provide such details because they generally focused on building models and analyzing their results. Therefore, we were unable to extract and synthesize them comprehensively and completely. It is therefore expected that the subsequent study on SPS will examine more problems in more detail and provide comprehensive and detailed information regarding SPS metrics. This includes measurement methods, values, categories, data issues, coping strategies, and data sources, etc. This serves as a useful guide and practice for modelers.

8 CONCLUSION

SPS modelers bear the burden of the high cost of selecting appropriate metrics, which is regarded by the community as one of the crucial barriers in adopting SPS in practice. This paper reports an SLR on the metrics and the associated attributes used in SPS models. The implication of this work can be extended to software process modeling in general considering the continuity and similarity between the static models and dynamic models and the higher standard required for dynamic models. The study is driven by a meta-model of the ontology of metrics so that we are able to comprehensively study the metrics in SPS models from the perspective of modeling.

As the result of an exhaustive search, we identified a total of 145 papers report SPS models until 2021. We identified 2130 metrics from these models and classified metrics and their corresponding attributes into six entity categories from the measurement perspective. Coding technique is applied to build the classification framework. The framework is more suitable as a reference for modeling than existing software measurement frameworks since it includes all metrics limited to SPS and has multiple levels of categories to show the similarities and differences of metrics. Different from Fenton et al.'s framework [22], we distinguish *defect activity* from *defect size* and *defect property* as well as classify *defect size* and *defect property* into product category since it is more reasonable. The choice of paradigm is constrained by both modeling granularity and available data. Therefore we further discussed data issues of measurement and coping strategies.

This study helps to address one of the two major challenging tasks of modeling, i.e. identifying the key metrics (elements of the model) and their relationships (structure of the model) from real processes. Although we discussed data issues and coping strategies reported in existing studies, it is still a challenging task for modelers to collect the data from industrial settings. SPS studies usually assume that organizations can collect required information but it is recognized that this is not feasible in some cases. In recent years process mining turns to be an effective tool to distill process information from various software repositories. Many mining algorithms and tools aiming at different purposes and data formats are mapped in our previous study [57]. Investigating how to apply an appropriate mining technique to obtain the data on the required metrics can be an opportunity for future work to improve the practical adoption of SPS.

Furthermore, the metrics and knowledge synthesized from SPS studies are not limited to process simulation but can be extended to software process modeling in general. Taking simulation metrics as standards and references can

further drive software developers to improve the collection, governance and application of process data in practice.

REFERENCES

[1] M. I. Kellner, R. J. Madachy, and D. M. Raffo, "Software process simulation modeling: Why? what? how?" *Journal of Systems and Software*, vol. 46, no. 2, pp. 91 – 105, 1999.

[2] T. Abdel-Hamid and S. E. Madnick, *Software Project Dynamics: An Integrated Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[3] R. Ahmed, T. Hall, P. Wernick, S. Robinson, and M. Shah, "Software process simulation modelling: A survey of practice," *Journal of Simulation*, vol. 2, no. 2, pp. 91–102, 2008.

[4] T. Birkhölzer, C. Dickmann, J. Vaupel, and L. Dantas, "An interactive software management simulator based on the cmmi framework," *Software Process Improvement & Practice*, vol. 10, no. 3, pp. 327–340, 2010.

[5] D. Crespo and M. Ruiz, "Decision making support in cmmi process areas using multiparadigm simulation modeling," in *Simulation Conference*, 2012, pp. 1–12.

[6] J. Li, M. Li, D. Wu, and H. Song, "An integrated risk measurement and optimization model for trustworthy software process management," *Information Sciences*, vol. 191, no. 9, pp. 47–60, 2012.

[7] D. M. Raffo, J. V. Vandeville, and R. H. Martin, "Software process simulation to achieve higher cmm levels," *Journal of Systems & Software*, vol. 46, no. 2–3, pp. 163–172, 1999.

[8] D. Mishra and B. Mahanty, "A study of software development project cost, schedule and quality by outsourcing to low cost destination," *J. Enterprise Inf. Management*, vol. 29, no. 3, pp. 454–478, 2016.

[9] H. Zhang, R. Jeffery, D. Houston, L. Huang, and L. Zhu, "Impact of process simulation on software practice: an initial report," in *2011 33rd International Conference on Software Engineering (ICSE)*, 2011.

[10] H. Zhang, D. Raffo, T. Birkhölzer, D. Houston, R. J. Madachy, J. Münch, and S. M. S. Jr., "Software process simulation - at a crossroads?" *Journal of Software: Evolution and Process*, vol. 26, no. 10, pp. 923–928, 2014.

[11] O. Gómez, H. Oktaba, M. Piattini, and F. García, "A systematic review measurement in software engineering: State-of-the-art in measures," in *ICSOFT 2006, First International Conference on Software and Data Technologies,*

Setúbal, Portugal, September 11-14, 2006, 2006, pp. 224–231.

[12] C. G. Portobellini, R. D. C. D. Fariapereira, and J. Luizbecker, "Measurement in software engineering: From the roadmap to the crossroads," *International Journal of Software Engineering & Knowledge Engineering*, vol. 18, no. 01, pp. 37–64, 2008.

[13] B. Kitchenham, "What's up with software metrics? - a preliminary mapping study," *Journal of Systems & Software*, vol. 83, no. 1, pp. 37–51, 2010.

[14] R. Abilio, P. Teles, H. Costa, and E. Figueiredo, "A systematic review of contemporary metrics for software maintainability," in *Sixth Brazilian Symposium on Software Components*, 2012.

[15] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development - a systematic literature review of industrial studies," *Inf. Softw. Technol.*, vol. 62, no. C, pp. 143–163, Jun. 2015.

[16] A. S. Nuñez-Varela, H. G. Pérez-Gonzalez, F. E. Martínez-Perez, and C. Soubervielle-Montalvo, "Source code metrics: A systematic mapping study," *Journal of Systems & Software*, vol. 128, pp. 164 – 197, 2017.

[17] A. Meidan, J. A. García-García, I. M. Ramos, and M. J. Escalona, "Measuring software process: A systematic mapping study," *ACM Computing Surveys*, vol. 51, no. 3, pp. 58:1–58:32, 2018.

[18] O. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, 1994.

[19] B. A. Kitchenham, L. Pickard, S. G. Linkman, and P. Jones, "A framework for evaluating a software bidding model," *Information & Software Technology*, vol. 47, no. 11, pp. 747–760, 2005.

[20] B. Kuipers, "Qualitative reasoning: Modeling and simulation with incomplete knowledge," *Automatica*, vol. 25, no. 4, pp. 571–585, 1989.

[21] K. Akingbehin and B. Maxim, "A three-layer model for software engineering metrics," in *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, ser. SNPD-SAWN '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 17–20.

[22] N. E. Fenton, *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, 2014.

[23] H. Zhang, B. A. Kitchenham, and D. Pfahl, "Reflections on 10 years of software process simulation modeling: A systematic review," in *Making Globally Distributed Software Development a Success Story, International Conference on Software Process, ICSP 2008, Leipzig, Germany, May 10-11, 2008, Proceedings*, 2008, pp. 345–356.

[24] H. Zhang, B. Kitchenham, and D. Pfahl, "Software process simulation modeling: Facts, trends and directions," in *15th Asia-Pacific Software Engineering Conference APSEC 2008, 3-5 December 2008, Beijing, China*, 2008, pp. 59–66.

[25] H. Zhang, B. A. Kitchenham, and D. Pfahl, "Software process simulation modeling: An extended systematic review," in *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, Paderborn, Germany, July 8-9, 2010. Proceed-

ings, 2010, pp. 309–320.

[26] H. Zhang, D. R. Jeffery, D. Houston, L. Huang, and L. Zhu, "Impact of process simulation on software practice: an initial report," in *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May 21-28, 2011*, 2011, pp. 1046–1056.

[27] S. Jiang, H. Zhang, C. Gao, D. Shao, and G. Rong, "Process simulation for software engineering education," in *Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24 - 26, 2015*, 2015, pp. 147–156.

[28] C. Gao, H. Zhang, and S. Jiang, "Constructing hybrid software process simulation models," in *Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24 - 26, 2015*, 2015, pp. 157–166.

[29] H. Gong, H. Zhang, D. Yu, and B. Liu, "A systematic map on verifying and validating software process simulation models," in *Proceedings of the 2017 International Conference on Software and System Process, Paris, France, ICSSP 2017, July 5-7, 2017*, 2017, pp. 50–59.

[30] B. B. França and G. Travassos, "Are we prepared for simulation based studies in software engineering yet?" vol. 16, p. 8, 2013.

[31] N. B. Ali, K. Petersen, and C. Wohlin, "A systematic literature review on the industrial use of software process simulation," *Journal of Systems and Software*, vol. 97, pp. 65 – 85, 2014.

[32] D. Pfahl, *Process Simulation: A Tool for Software Project Managers?* Springer Berlin Heidelberg, 2014.

[33] F. García, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, and M. Genero, "Towards a consistent terminology for software measurement," *Information & Software Technology*, vol. 48, no. 8, pp. 631–644, 2006.

[34] L. Olsina and M. de los Angeles Martín, "Ontology for software metrics and indicators," *Journal of Web Engineering*, vol. 2, no. 4, pp. 262–281, 2003.

[35] S. Robinson, "Conceptual modeling for simulation," in *Proceedings of the 2013 Winter Simulation Conference on Simulation: Making Decisions in a Complex World*, 2013, pp. 377–388.

[36] ——, "Conceptual modelling for simulation part i: definition and requirements," *Journal of the Operational Research Society*, vol. 59, no. 3, pp. 278–290, 2008.

[37] I. Standard, "Adoption of iso/iec 15939:2007— systems and software engineering— measurement process," 2009.

[38] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering version 2.3," Software Engineering Group, School of Computer Science and Mathematics, Keele University and Department of Computer Science University of Durham, Tech. Rep., 2007.

[39] D. Raffo and M. I. Kellner, "Empirical analysis in software process simulation modeling," *Journal of Systems and Software*, vol. 53, no. 1, pp. 31–41, 2000.

[40] D. Pfahl, "Pl-sim: a generic simulation model for studying strategic spi in the automotive industry," in *W11L Workshop - 26th International Conference on Software Engineering*, vol. 2004, 2004, pp. 149–158.

[41] H. Zhang, B. Kitchenham, and R. Jeffery, "A framework for adopting software process simulation in cmmi organizations," in *International Conference on Software Process*, 2007, pp. 320–331.

[42] D. S. Cruzes and T. Dybå, "Recommended steps for thematic synthesis in software engineering," in *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*, 2011, pp. 275–284.

[43] J. M. Corbin and A. L. Strauss, "Basics of qualitative research: techniques and procedures for developing grounded theory," *Thousand Oaks Ca Sage Tashakkori A & Teddlie C*, vol. 36, no. 100, p. 129, 1998.

[44] C. Kaner and W. P. Bond, "Software engineering metrics: What do they measure and how do we know?" *Metrics IEEE Cs*, 2004.

[45] F. Padberg, "A discrete simulation model for assessing software project scheduling policies," *Software Process: Improvement and Practice*, vol. 7, pp. 127–139, 2002.

[46] R. Cherif and P. Davidsson, "Software development process simulation: Multi agent-based simulation versus system dynamics," in *Multi-Agent-Based Simulation X, International Workshop, MABS 2009, Budapest, Hungary, May 11-12, 2009 Revised Selected Papers*, 2009, pp. 73–85.

[47] N. Ali and K. Petersen, "A consolidated process for software process simulation: State of the art and industry experience," 09 2012, pp. 327–336.

[48] B. B. França and N. Ali, *The Role of Simulation-Based Studies in Software Engineering Research*, 08 2020, pp. 263–287.

[49] C. M. Abts, "Cots software integration cost modeling study," June 1997.

[50] K. Lum, J. Powell, and J. Hihn, "Validation of spacecraft software cost estimation models for flight and ground systems," 2002.

[51] C. Jones, "Applied software measurement: assuring productivity and quality," *McGraw-Hill software engineering series*, 1996.

[52] A. A. Frost and M. J. Campo, "Advancing defect containment to quantitative defect management," 2007.

[53] S. Wagner, "A literature survey of the quality economics of defect-detection techniques," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE '06, 2006.

[54] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, "A map of threats to validity of systematic literature reviews in software engineering," in *23rd Asia-Pacific Software Engineering Conference, APSEC 2016, Hamilton, New Zealand, December 6-9, 2016*, 2016, pp. 153–160.

[55] C. Wohlin, "Second-generation systematic literature studies using snowballing," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE 2016, Limerick, Ireland, June 01 - 03, 2016*, 2016, pp. 15:1–15:6.

[56] H. Zhang, M. A. Babar, and P. Tell, "Identifying relevant studies in software engineering," *Information & Software Technology*, vol. 53, no. 6, pp. 625–637, 2011.

[57] L. Dong, B. Liu, Z. Li, O. Wu, M. A. Babar, and B. Xue, "A mapping study on mining software process," in *24th Asia-Pacific Software Engineering Conference, APSEC 2017*, 2017.

APPENDIX

REFERENCE

[Aker 17]   O. Akerele. "Using System Dynamics for Agile Cloud Systems Simulation Modelling". *Strategic Engineering for Cloud Computing and Big Data Analytics*, pp. 101–117, 2017.

[Al E 08]   A. Al-Emran, P. Kapur, D. Pfahl, and G. Ruhe. "Simulating worst case scenarios and analyzing their combined effect in operational release planning". pp. 269–281, Springer, 2008.

[Ambr 11]   B. G. Ambrósio, J. L. Braga, and M. A. Resende-Filho. "Modeling and scenario simulation for decision support in management of requirements activities in software projects". *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 23, No. 1, p. 35–50, 2011.

[Aran 08]   E. Aranha and P. Borba. "Using process simulation to assess the test design effort reduction of a model-based testing approach". In: *Software Process, 2008 International Conference on Making Globally Distributed Software Development A Success Story*, pp. 282–293, 2008.

[Arau 12]   M. A. P. Araújo. "Towards a model to support in silico studies of software evolution". In: *Acm-Ieee International Symposium on Empirical Software Engineering and Measurement*, pp. 281–290, 2012.

[Armb 03]   O. Armbrust. "Using Empirical Knowledge from Replicated Experiments for Software Process Simulation: A Practical Example". In: *International Symposium on Empirical Software Engineering, 2003. Isese 2003. Proceedings*, pp. 18–27, 2003.

[Baum 17]   T. Baum, F. Kortum, K. Schneider, A. Brack, and J. Schauder. "Comparing Pre Commit Reviews and Post Commit Reviews Using Process Simulation". In: *IEEE/ACM International Conference on Software and System Processes*, pp. 26–35, 2017.

[Berl 03]   T. Berling, C. Andersson, M. Höst, and C. Nyberg. "Adaptation of a simulation model template for testing to an industrial project". pp. 3–4, 2003.

[Cang 08]   J. W. Cangussu and R. M. Karcich. "Using Dynamic Models for the Evaluation of Integration and System Testing". In: *International Conference on Software Engineering and Knowledge Engineering*, pp. 436–441, 2008.

[Chat 00]   B. W. Chatters, M. M. Lehman, J. F. Ramil, and P. Wernick. "Modelling a software evolution process: a long-term case study". *Software Process Improvement & Practice*, Vol. 5, No. 2-3, pp. 91–102, 2000.

[Chen 06]   Y. Chen, G. C. Gannod, and J. S. Collofello. "A software product line process simulator". *Software Process Improvement & Practice*, Vol. 11, No. 4, pp. 385–409, 2006.

[Cher 10]   R. Cherif and P. Davidsson. "Software Development Process Simulation: Multi Agent-Based Simulation versus System Dynamics". *Lecture Notes in Computer Science*, Vol. 5683, No. 2010, pp. 73–85, 2010.

[Choi 06]   K. Choi, D. Bae, and T. Kim. "An approach to a hybrid software process simulation using the DEVS formalism". *Software Process: Improvement and Practice*, Vol. 11, No. 4, pp. 373–383, 2006.

[Cocc 11]   L. Cocco, K. Mannaro, G. Concas, and M. Marchesi. "Simulating Kanban and Scrum vs. Waterfall with System Dynamics". In: *International Conference on Agile Software Development*, pp. 117–131, 2011.

[Conc 13]   G. Concas, M. I. Lunesu, M. Marchesi, and H. Zhang. "Simulation of software maintenance process, with and without a work-in-process limit". *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 25, No. 12, pp. 1225–1248, 2013.

[Dan 13]   X. H. Dan and D. J. Buettner. "Modeling user story completion of an agile software process". In: *International Conference on Software and System Process*, pp. 88–97, 2013.

[Drap 99]   A. Drappa and J. Ludewig. "Quantitative modeling for the interactive simulation of software projects". *Journal of Systems and Software*, Vol. 46, No. 2-3, pp. 113–122, 1999.

[Duga 21]   G.-L. Dugarte-Peña, M.-I. Sánchez-Segura, A. de Amescua, F. Medina-Domínguez, and S. Armenia. "Using system dynamics to teach about dependencies, correlation and systemic thinking on the software process workflows". *IET Software*, Vol. 15, No. 6, pp. 351–364, 2021.

[Fate 18]   I. Fatema and K. Sakib. "Using Qualitative System Dynamics in the Development of an Agile Teamwork Productivity Model". *International Journal on Advances in Software*, Vol. 11, No. 12, pp. 170–185, 2018.

[Ferr 09]   S. Ferreira, J. Collofello, S. Dan, and G. Mackulak. "Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation". *Journal of Systems & Software*, Vol. 82, No. 10, pp. 1568–1577, 2009.

[Garo 09]   V. Garousi, K. Khosrovian, and D. Pfahl. "A customizable pattern-based software process simulation model: design, calibration and application". *Software Process Improvement & Practice*, Vol. 14, No. 3, pp. 165–180, 2009.

[Garo 16]   V. Garousi and D. Pfahl. "When to automate software testing? A decision-support approach based on process simulation". *Journal of Software: Evolution and Process*, Vol. 28, No. 4, pp. 272–285, 2016.

[Hall 05]   T. Hall and P. Wernick. "Program slicing metrics and evolvability: an initial study". In: *IEEE International Workshop on Software Evolvability*, pp. 35–40, 2005.

[Hana 02]   Hanakawa, Noriko, Matsumoto, Kenichi, Torii, and Koji. "A Knowledge-Based Software Process Simulation Model". *Annals of Software Engineering*, Vol. 14, No. 1-4, pp. 383–406, 2002.

[Hana 98]   N. Hanakawa, S. Morisaki, and K. Matsumoto. "A Learning Curve Based Simulation Model for Software Development". In: *Forging New Links,*

Proceedings of the 1998 International Conference on Software Engineering, ICSE 98, Kyoto, Japan, April 19-25, 1998., pp. 350–359, 1998.

[Hous 10] D. Houston and M. Lieu. "Modeling a Resource-Constrained Test-and-Fix Cycle and Test Phase Duration". In: *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, Paderborn, Germany, July 8-9, 2010. Proceedings*, pp. 211–221, 2010.

[Hsia 99] P. Hsia, C. Hsu, and D. C. Kung. "Brooks' Law Revisited: A System Dynamics Approach". In: *23rd International Computer Software and Applications Conference COMPSAC '99, 27-19 October 1999, Phoenix, AZ, USA*, pp. 370–375, 1999.

[Hurt 15] N. Hurtado, M. Ruiz, E. Orta, and J. Torres. "Using simulation to aid decision making in managing the usability evaluation process". *Information & Software Technology*, Vol. 57, pp. 509–526, 2015.

[Kahe 01] G. Kahen, M. M. Lehman, J. F. Ramil, and P. Wernick. "System dynamics modelling of software evolution processes for policy investigation: Approach and example". *Journal of Systems & Software*, Vol. 59, No. 3, pp. 271–281, 2001.

[Klun 18] J. Klünder, F. Kortum, T. Ziehm, and K. Schneider. "Helping teams to help themselves: An industrial case study on interdependencies during sprints". In: *International Conference on Human-Centred Software Engineering*, pp. 31–50, Springer, 2018.

[Kous 07] K. G. Kouskouras and A. C. Georgiou. "A discrete event simulation model in the case of managing a software project". *European Journal of Operational Research*, Vol. 181, No. 1, pp. 374–389, 2007.

[Lake 03] P. B. Lakey. "A hybrid software process simulation model for project management". In: *Proceedings of the 2003 International Workshop on Software Process Simulation and Modeling (ProSim'03)*, 2003.

[Lehm 10] M. M. Lehman, G. Kahen, and J. F. Ramil. "Behavioural modelling of long-lived evolution processes—some issues and an example". *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 14, No. 5, pp. 335–351, 2010.

[Lune 17] I. Lunesu, M. Marchesi, J. Münch, and M. Kuhrmann. "Using measurement and simulation for understanding distributed development processes in the cloud". In: *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, pp. 1–11, 2017.

[Lune 21] M. I. Lunesu, R. Tonelli, L. Marchesi, and M. Marchesi. "Assessing the Risk of Software Development in Agile Methodologies Using Simulation". *IEEE Access*, Vol. 9, pp. 134240–134258, 2021.

[Mada 00] R. J. Madachy and D. Tarbet. "Case studies in software process modeling with system dynamics". *Software Process: Improvement and Practice*, Vol. 5, No. 2-3, pp. 133–146, 2000.

[Mada 05] R. Madachy. "Integrated Modeling of Business Value and Software Processes". Vol. 3840, pp. 389–402, 2005.

[Mada 07] R. Madachy, B. Boehm, and J. A. Lane. *Assessing hybrid incremental processes for SISOS development: Research Sections*. John Wiley & Sons, Inc., 2007.

[Mart 00] R. H. Martin and D. Raffo. "A model of the software development process using both continuous and discrete models". *Software Process: Improvement and Practice*, Vol. 5, No. 2-3, pp. 147–157, 2000.

[Menz 02] T. Menzies, D. Raffo, S. O. Setamanit, Y. Hu, and S. Tootoonian. "Model-Based Tests of Truisms". In: *IEEE International Conference on Automated Software Engineering, 2002. Proceedings. ASE*, pp. 183–191, 2002.

[Mish 16] D. Mishra and B. Mahanty. "A study of software development project cost, schedule and quality by outsourcing to low cost destination". *Journal of Enterprise Information Management*, Vol. 29, No. 3, pp. 454–478, 2016.

[Naun 07] P. Naunchan and D. Sutivong. "Adjustable Cost Estimation Model for COTS-Based Development". In: *Software Engineering Conference, 2007. Aswec 2007. Australian*, pp. 341–348, 2007.

[Neu 02a] H. Neu and U. Beckerkornstaedt. "Learning and Understanding a Software Process through Simulation of Its Underlying Model". *Kornstaedt*, 2002.

[Neu 02b] H. Neu, T. Hanne, J. Münch, S. Nickel, and A. Wirsen. "Simulation-Based Risk Reduction for Planning Inspections". pp. 78–93, 2002.

[Neu 03] H. Neu, T. Hanne, J. Münch, S. Nickel, and A. Wirsen. "Creating a Code Inspection Model for Simulation-Based Decision Support". In: *Intl. Workshop on Software Process Simulation and Modeling*, 2003.

[Noll 16] J. Noll and A. Butterfield. "Teaching global software development through game design". In: *2016 IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW)*, pp. 55–60, IEEE, 2016.

[Oors 18] K. E. van Oorschot, K. Sengupta, and L. N. Van Wassenhove. "Under pressure: The effects of iteration lengths on agile software development performance". *Project Management Journal*, Vol. 49, No. 6, pp. 78–102, 2018.

[Padb 11] F. Padberg and D. Weiss. "Model-Based Scheduling Analysis for Software Projects". In: *Computer Software and Applications Conference Workshops*, pp. 304–309, 2011.

[Park 07] S. H. Park, K. S. Choi, K. Yoon, and D. H. Bae. "Deriving Software Process Simulation Model from SPEM-based Software Process Model". In: *Software Engineering Conference, 2007. APSEC 2007. Asia-Pacific*, pp. 382–389, 2007.

[Pfah 00a] D. Pfahl and K. Lebsanft. "Knowledge Acquisition and Process Guidance for Building System Dynamics Simulation Models: an Experience Report from Software Industry". *International Journal of Software Engineering and Knowledge Engineering*, Vol. 10, No. 4, pp. 487–510, 2000.

[Pfah 00b] D. Pfahl and K. Lebsanft. "Using simulation to analyse the impact of software requirement volatility on project performance". *Information & Software Technology*, Vol. 42, No. 14, pp. 1001–1008, 2000.

[Pfah 01] D. Pfahl, M. Klemm, and G. Ruhe. "A CBT module with integrated simulation component for software project management education and training". *Journal of Systems and Software*, pp. 283–298, 2001.

[Pfah 04] D. Pfahl, M. Stupperich, A. G. Daimlerchrysler, and T. Krivobokova. "PL-SIM: A Generic Simulation Model for Studying Strategic SPI in the Automotive Industry". *Journal of Modern African Studies*, Vol. 20, No. 1, pp. 87–105, 2004.

[Pfah 99] D. Pfahl and K. Lebsanft. "Integration of system dynamics modelling with descriptive process modelling and goal-oriented measurement". *Journal of Systems and Software*, Vol. 46, No. 2, 1999.

[Raff 00] D. Raffo, W. Harrison, and J. Vandeville. "Coordinating models and metrics to manage software projects". *Software Process: Improvement and Practice*, Vol. 5, No. 2-3, pp. 159–168, 2000.

[Raff 02] D. Raffo, W. Harrison, and J. Vandeville. "Software process decision support: making process tradeoffs using a hybrid metrics, modeling and utility framework". In: *Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002*, pp. 803–809, 2002.

[Raff 03] D. Raffo and S. Setamanit. "Supporting Software Process Decisions Using Bi-Directional Simulation". *International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, No. 5, pp. 513–530, 2003.

[Ruiz 01] M. Ruiz, I. Ramos, and M. Toro. "A simplified model of software project dynamics". *Journal of Systems and Software*, Vol. 59, No. 3, pp. 299–309, 2001.

[Ruiz 02a] M. Ruiz, I. Ramos, and M. Toro. "A Dynamic Integrated Framework for Software Process Improvement". *Software Quality Journal*, Vol. 10, No. 2, pp. 181–194, 2002.

[Ruiz 02b] M. Ruiz, I. Ramos, and M. Toro. "Integrating Dynamic Models for CMM-Based Software Process Improvement". In: *International Conference on Product Focused Software Process Improvement*, pp. 63–77, 2002.

[Ruiz 04] M. Ruiz, I. Ramos, and M. Toro. "Using Dynamic Modeling and Simulation to Improve the COTS Software Process". In: *Product Focused Software Process Improvement, International Conference, Profes 2004, Kausai Science City, Japan, April 5-8, 2004, Proceedings*, pp. 568–581, 2004.

[Rus 02] I. Rus, S. Biffl, and M. Halling. "Systematically combining process simulation and empirical data in support of decision analysis in software development". In: *Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002*, pp. 827–833, 2002.

[Rus 03] I. Rus, M. Halling, and S. Biffl. "Supporting Decision-Making in Software Engineering with Process Simulation and Empirical Studies". *International Journal of Software Engineering and Knowledge Engineering*, Vol. 13, No. 5, pp. 531–545, 2003.

[Rus 14] I. Rus, H. Neu, and J. Münch. "A Systematic Methodology for Developing Discrete Event Simulation Models of Software Development Processes". *CoRR*, Vol. abs/1403.3559, 2014.

[Shuk 12] R. Shukla, M. Shukla, A. K. Misra, T. Marwala, and W. A. Clarke. "Dynamic software maintenance effort estimation modeling using neural network, rule engine and multi-regression approach". In: *International Conference on Computational Science and ITS Applications*, pp. 157–169, 2012.

[Silv 13] L. C. e Silva and A. P. C. S. Costa. "Decision model for allocating human resources in information system projects". *International Journal of Project Management*, Vol. 31, No. 1, pp. 100–108, 2013.

[Smit 06] N. Smith, A. Capiluppi, and J. Fernández-Ramil. "Agent-based simulation of open source evolution". *Software Process Improvement & Practice*, Vol. 11, No. 4, p. 423–434, 2006.

[Spas 12] B. Spasic and B. S. S. Onggo. "Agent-based simulation of the software development process: A case study at AVL". In: *Simulation Conference*, pp. 1–11, 2012.

[Topi 08] G. Topic, D. Jevtic, and M. Kunstic. "Petri Net-Based Simulation and Analysis of the Software Development Process". In: *Knowledge-Based Intelligent Information and Engineering Systems, 12th International Conference, KES 2008, Zagreb, Croatia, September 3-5, 2008, Proceedings, Part II*, pp. 418–425, 2008.

[Tram 16] M. T. I. Trammell, A. Moulton, and S. E. Madnick. "Effects of Funding Fluctuations on Software Development: A System Dynamics Analysis". *Engineering Management Journal*, Vol. 28, No. 2, pp. 71–85, 2016.

[Turn 06] I. Turnu, M. Melis, A. Cau, A. Setzu, G. Concas, and K. Mannaro. "Modeling and simulation of open source development using an agile practice". *Journal of Systems Architecture the Euromicro Journal*, Vol. 52, No. 11, pp. 610–618, 2006.

[Uzza 13a] M. Uzzafer. "A contingency estimation model for software projects". *International Journal of Project Management*, Vol. 31, No. 7, pp. 981–993, 2013.

[Uzza 13b] M. Uzzafer. "A simulation model for strategic management process of software projects". *Journal of Systems & Software*, Vol. 86, No. 1, pp. 21–

37, 2013.

[Van 17]   K. Van Oorschot, K. Eling, and F. Langerak. "Measuring the Knowns to Manage the Unknown: How to Choose the Gate Timing Strategy in NPD Projects". *Journal of Product Innovation Management*, 2017.

[Wake 04]   W. W. Wakeland, R. H. Martin, and D. Raffo. "Using design of experiments, sensitivity analysis, and hybrid simulation to evaluate changes to a software development process: a case study". *Software Process: Improvement and Practice*, Vol. 9, No. 2, pp. 107–119, 2004.

[Wake 05]   W. W. Wakeland, S. Shervais, and D. Raffo. "Heuristic optimization as a V&V tool for software process simulation models". *Software Process: Improvement and Practice*, Vol. 10, No. 3, pp. 301–309, 2005.

[Wern 02]   P. Wernick and T. Hall. "Simulating global software evolution processes by combining simple models: an initial study". *Software Process: Improvement and Practice*, Vol. 7, No. 3-4, pp. 113–126, 2002.

[Wern 99]   P. Wernick and M. M. Lehman. "Software process white box modelling for FEAST/1". *Journal of Systems & Software*, Vol. 46, No. 2–3, pp. 193–201, 1999.

[Xiao 10]   J. Xiao, L. J. Osterweil, Q. Wang, and M. Li. "Disruption-Driven Resource Rescheduling in Software Development Processes". In: *New Modeling Concepts for Today's Software Processes, International Conference on Software Process, ICSP 2010, Paderborn, Germany, July 8-9, 2010. Proceedings*, pp. 234–247, 2010.

[Zawe 13]   A. Zawedde and D. Williams. "Determinants of Requirements Process Improvement Success". In: *International System Dynamics Conference*, 2013.

[Zhan 06a]   H. Zhang, M. Huo, B. Kitchenham, and D. R. Jeffery. "Qualitative Simulation Model for Software Engineering Process". In: *17th Australian Software Engineering Conference (ASWEC 2006), 18-21 April 2006, Sydney, Australia*, pp. 391–400, 2006.

[Zhan 06b]   H. Zhang and B. Kitchenham. "Semi-quantitative Simulation Modeling of Software Engineering Process". In: *Software Process Workshop*, pp. 242–253, 2006.

[Zhan 08a]   H. Zhang, D. R. Jeffery, and L. Zhu. "Hybrid Modeling of Test-and-Fix Processes in Incremental Development". In: *Making Globally Distributed Software Development a Success Story, International Conference on Software Process, ICSP 2008, Leipzig, Germany, May 10-11, 2008, Proceedings*, pp. 333–344, 2008.

[Zhan 08b]   H. Zhang, J. Keung, B. A. Kitchenham, and D. R. Jeffery. "Semi-quantitative Modeling for Managing Software Development Processes". In: *19th Australian Software Engineering Conference (ASWEC 2008), March 25-28, 2008, Perth, Australia*, pp. 66–75, 2008.

[Zhan 09]   H. Zhang, B. Kitchenham, and D. R. Jeffery. "Qualitative vs. Quantitative Software Process Simulation Modeling: Conversion and Comparison". pp. 345–354, 2009.

[Zhan 12a]   H. Zhang, B. Kitchenham, and R. Jeffery. "Toward trustworthy software process models: an exploratory study on transformable process modeling". *Journal of Software Maintenance and Evolution: Research and Practice*, Vol. 24, No. 7, pp. 741–763, 2012.

[Zhan 12b]   H. Zhang, G. Klein, M. Staples, J. Andronick, L. Zhu, and R. Kolanski. "Simulation modeling of a large-scale formal verification process". In: *International Conference on Software and System Process*, pp. 3–12, 2012.

[Zhan 18]   X. Zhang, X. Wang, and Y. Kang. "Change-oriented open source software process simulation". *IEEE Access*, Vol. 6, pp. 70145–70163, 2018.

[Zhou 12]   P. Zhou and H. K. N. Leung. "A Stochastic Simulation Model for Risk Management Process". Vol. 1, pp. 737–742, 2012.