Software Analytics Tools: An Intentional View

Xavier Franch[1]

[1] *Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain*

Abstract

Software analytic tools consume big amounts of data coming from either (or both) the software development process or the system usage and aggregate them into indicators which are rendered to different types of stakeholders, also offering them a portfolio of techniques and capabilities such as what-if analysis, prediction and alerts. Precisely, the variety of stakeholders and the different goals they pursue justifies the convenience of performing an intentional analysis of the use of software analytics tools. With this aim, we first enumerate the different stakeholders and identify their intentional relationships with software analytics tools in the form of dependencies. Then, we focus on one particular stakeholder, namely the requirements engineer, and identify further intentional elements represented in a strategic rationale model. The resulting model provides an abstract view of the domain which may help stakeholders when deciding on the adoption of software analytic tools in their particular context.

Keywords

Software Analytic Tools, Data-driven Software Engineering, Software Indicators, Software Metrics, iStar

1. Introduction

The increasing availability of big volumes of data produced both during the development of a software product and after its deployment (i.e., at runtime) is paving the way to data-driven software development [15]. Under this paradigm, data gathered from both software repositories and the system while it is in use, is analyzed to get insightful information which guides the evolution of the system.

However, making actionable the collected data is not easy: large volumes need to be collected, cleansed and organized, and multiple data sources need to be reconciled and unified in order to consolidate them as a single (logical) input stream of data [17]. For this reason, it is key the existence of tools that help in analyzing the collected data. These tools are usually known as *software analytic tools* [7][13]. They provide advanced visualization capabilities and analysis techniques such as prediction, simulation, what-if analysis and alerts. Software analytics tools are used by a number of stakeholders such as managers, requirement engineers, project leaders and developers in order to monitor progress, improve practices and decide in future steps in relation to the development of a software product.

Given the variety of such stakeholders, goals and techniques, in this paper we aim at providing an intentional view to software analytic tools using the *i** language [6]. We present an overview of actors, intentional elements and dependencies characterizing these tools and their context of use, with the purpose of facilitating their understanding and fit-for-purpose in the data-driven software development cycle.

iStar'22: The 15th International i* Workshop, October 17th, 2022, Hyderabad, India
EMAIL: xavier.franch@upc.edu
ORCID: 0000-0001-9733-8830

2. Background

According to Gall et al., software analytics tools use data-driven approaches to obtain insightful and actionable information to help software practitioners with their data related tasks [7]. This abstract definition can be accommodated at several levels of detail and with different purposes, e.g. understanding design and code quality with a tool like SonarQube[2] or managing the value stream inside a company using TaskTop[3]. Given their need of great amounts of data, software analytic tools have become also popular in the open source software arena [3] and mobile ecosystems [14].

In particular, and this is the interest of this paper, software analytics has received increased attention in "need for speed" software development methods, as the cases of continuous delivery [9], rapid software development [11] and more generally, continuous software engineering [4]. As shown in **Figure 1**, data mined from software repositories and system usage feed a data ingestion infrastructure, then can be made actionable and enter into the software analytic tool. This tool will provide advanced visualization and analysis techniques to the software development decision-makers who will make and implement the decisions to close the cycle. In what follows, we will make explicit in more details which are the actors, goals and dependencies that altogether provide an intentional view to the use of software analytic tools in the software development process.

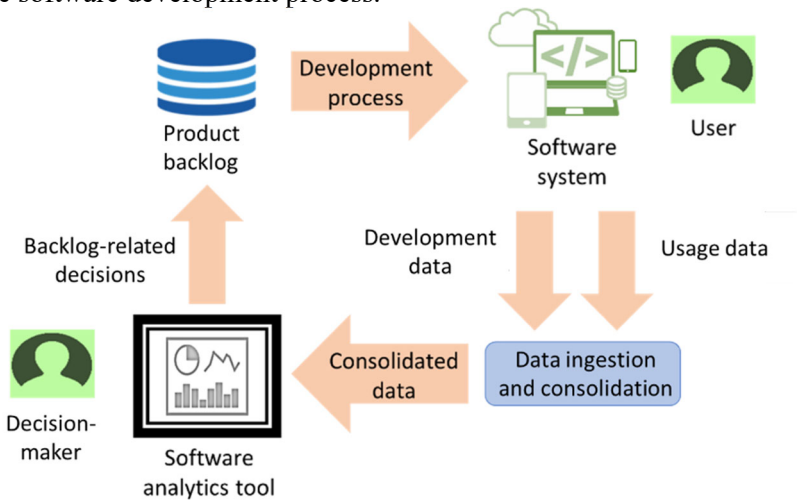

Figure 1: Integrating a software analytics tool into the software development cycle (adapted from [8])

3. Strategic Dependency Diagram for Software Analytics Tools

In this section we identify the different types of actors that are involved in the use of software analytics tools, and what are their dependencies upon such type of tool. Together, they conform a Strategic Dependency (SD) diagram. For compliance with the iStar 2.0 language [2], we will not use positions but only generic actors.

Central in the SD, we include an actor for the Software Analytics Tool itself. It will provide the functionalities required by its stakeholders, roughly grouped into visualization and analysis, with the appropriate qualities, from which we highlight accuracy and understandability, arguably the two most determinant for the tool adoption success. Given that these functionalities and qualities are required by all the stakeholders, we define a Software Analytics Stakeholder abstract actor which acts as depender of these dependencies. The main requirement from the tool to this generic stakeholder is the appropriate configuration parameters (e.g., data sampling intervals, deadlines for releases, etc.).

Table 1 summarizes the stakeholders that are involved in the use of software analytics tools, which are presented below, and defined as subactors of Software Analytics Stakeholder (see **Figure 2** and **Figure 3**):

[2] https://www.sonarqube.org/
[3] https://www.tasktop.com/

- Requirements Engineer. The requirements engineer expects the software analytics tool to detect any system misbehavior (e.g., a feature is missing, a quality requirement is not fulfilled) and eventually generate new requirements in response to them. However, the final decision on whether a candidate requirement needs to be included in the backlog is made by the requirements engineer.
- Developers. They use software analytics tools for improving the quality of their code and their development processes, e.g. to know the test coverage. For this, it is mandatory that they use the development tools that feed data into the software analytics pipeline. For instance, opening issues when they discover new bugs, or changing the task statuses in their project backlog.
- Project Manager. Software analytics tools help project managers in keeping the project in track at different respects, from which we may remark delivery on time and overall quality of the product. Besides, project managers may use these tools to improve the team development process.
- Domain Expert. The domain expert will define the quality model including the different low-level metrics (e.g., number of test cases passed, number of opened issues) and high-level indicators that hierarchically group them (e.g., from number of test cases passed up until product quality indicator). Since a concrete software analytics tool may restrict the kind of model characteristics (e.g., not arbitrary hierarchies), the domain expert needs to be aware of it.
- Data Scientist. The data scientist is responsible for making the software analytics tool properly gather the data. This means connecting the data sources required by the metrics and indicators defined by the Domain Expert, and reconciling their semantics. To do so, the software analytics tool needs to provide appropriate configuration facilities.

Table 1 Actors representing stakeholders for software analytics tools

Actor	Main goal
Software Analytics Tool	Provides advanced software analysis and visualization features
Requirements Engineer	Elicits and prioritizes requirements
Developer	Implement and tests software features
Project Manager	Keeps the project on track
Domain Expert	Defines the concepts that are applicable to a particular project domain
Data Scientist	Takes care of data ingestion

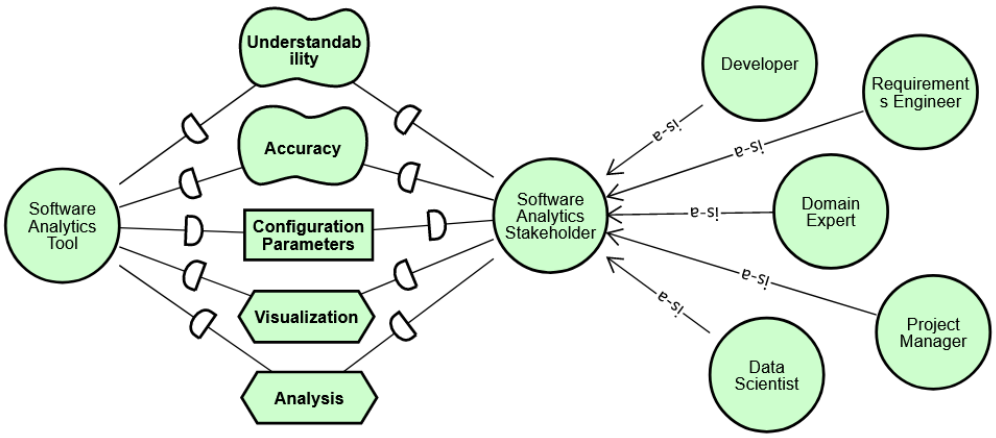

Figure 2: SD diagram for Software Analytics Tools: general view[4]

[4] All diagrams have been drawn using the piStar tool available at https://www.cin.ufpe.br/~jhcp/pistar/tool/# [18]

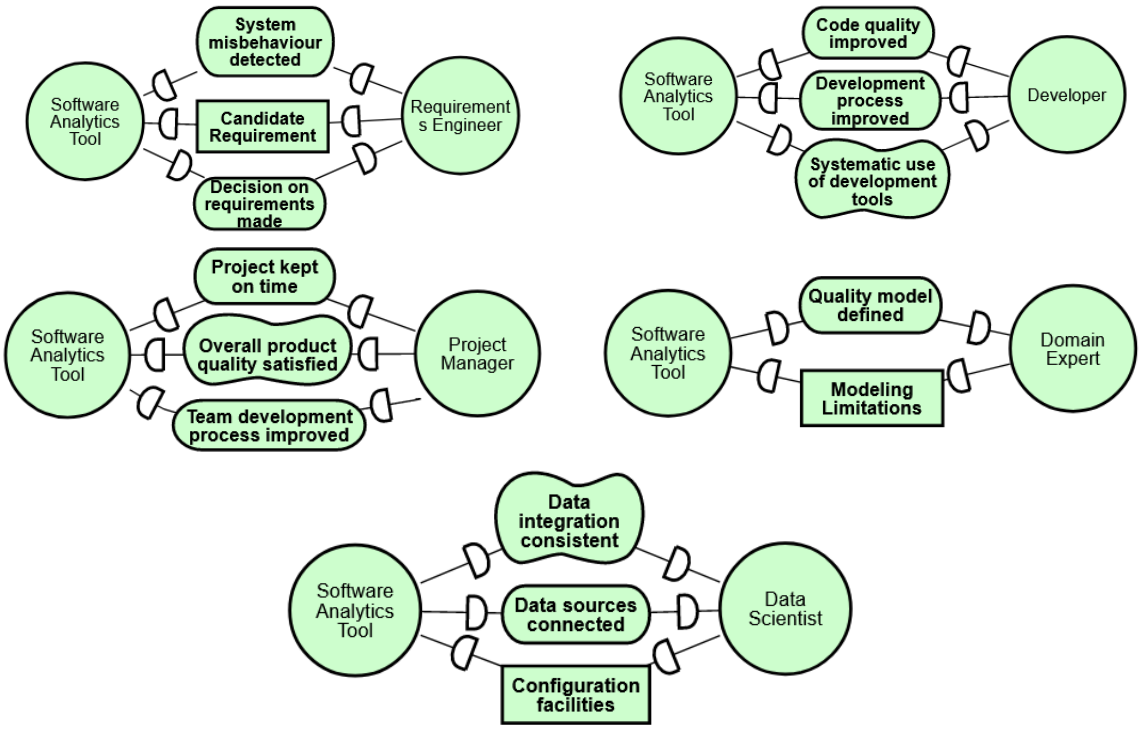

Figure 3: SD diagram for Software Analytics Tools: stakeholders' specificities

4. Strategic Rationale Diagram: The Case for the Requirements Engineer

In this section, we elaborate the details on the intentional view of one of the identified actors, namely the requirements engineer, over software analytics tools. This detail should be elaborated for all other actors in order to get a comprehensive view of the impact of software analytics tools into the development process.

We base our goal in the work by Oriol et al., which describe a data-driven requirements elicitation process guided by a software analytics tool [16]. In summary, this process is composed of four steps: *(i)* detect the violation of a requirement; *(ii)* select candidate requirement to fix the problem; *(iii)* assess these candidate requirements in the context of the system; *(iv)* make an informed decision; *(v)* eventually, update the backlog with the outcome of the decision.

Figure 4 provides the details in terms of an SR diagram including the Requirements Engineer and the required excerpt of the Software Analytics Tools. The general goal of the Requirements Engineer is to made decisions on requirements, which decomposes into several subgoals, roughly corresponding to the steps above:

- Monitoring requirements satisfaction. This subgoal covers the needs of the Requirements Engineer for Step *(i)*. It depends upon the already elicited System Misbehaviour Detected introduced in **Figure 3**. For satisfying this dependency, the Software Analytics Tool offers a task for managing alerts which shall be raised when some monitored value goes beyond a given threshold established in a requirement. For instance, this may be the case when the response time of a given functionality is below the required value. Alert management needs to be Timely, so that the Requirements Engineer is informed of any violation as soon as it occurs.
- Assessing new requirements. The Software Analytics Tool suggests new candidate requirements to be assessed by the Requirements Engineer (Step *(ii)*). These candidate requirements are instantiated from templates defined in a Requirements Pattern Catalogue as defined in [19], whose maintenance relies on the Requirements Engineer. More precisely, the type of requirement which causes the violation is used to browse the catalogue and detect patterns related to this type.
- Providing expert advice. The Requirements Engineer is the ultimate responsible in assessing the adequacy of the candidate requirements. In order to do so, the Software Analytics Tool shall

provide both visualization capabilities and what-if analysis. As dominant qualities, the visualization is expected to be understandable, while what-if analysis is required to be accurate.

- Final decision made. The Requirements Engineer communicates the selected requirements to the Software Analytics Tool. Ideally, the Software Analytics Tool takes care of communicating with a Backlog Management Tool (e.g., Jira, Taiga) in order to store the requirement in the most appropriate form, provided that the Backlog Management Tool offers the appropriate functionalities (usually as an extension). An example appears in [16], which describes the connection of a kind of Software Analytics Tool, the Q-Rapids dashboard, with a Backlog Management Tool, Jira. The requirement is represented as a user story, although in general it could also take the form of an acceptance criterion or even an epic following the advices given in [1].

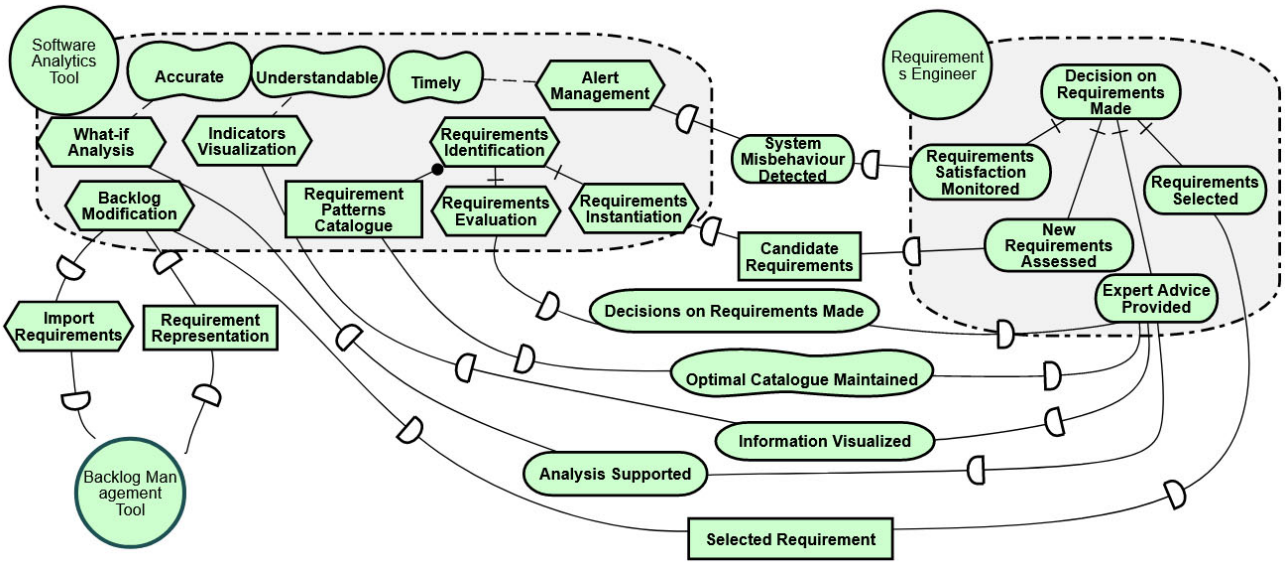

Figure 4: SR diagram for Software Analytics Tools: the case of the Requirements Engineer

5. Conclusions and Future Work

In this paper, we have provided an intentional perspective to the use of software analytics tools in a software development process. The resulting iStar2.0 model (only partial in the case of the SR diagram) shows what are the actors involved, their dependencies upon this kind of tools, and through a more thorough analysis of one particular actor, the requirements engineer, gives light to further capabilities and qualities that are demanded to software analytics tools.

As future work, we plan to use this model as a kind of template from which we can derive more detailed models for particular instances of software analytics tools. To this aim, we plan to use *(i)* modularization [12] to neatly encapsulate the different groups of capabilities and techniques offered by software analytics tools, and *(ii)* specialization [10] to extend and refine capabilities in a gradual manner. In the pursue of scalability, in order to control the complexity of the resulting models, we plan to define some structural complexity metrics [5] and ensure that the evaluation of these metrics in our resulting models does not exceed some (yet to define) pre-established heuristics and thresholds. For instance, as shown in **Figure 4**, we expect that the software analytics tool's SR diagram to be mainly composed of tasks, resources and qualities bound to tasks, while not many goals (if any).

6. Acknowledgements

This paper has been funded by the Spanish Ministerio de Ciencia e Innovación under project / funding scheme PID2020-117191RB-I00 / AEI/10.13039/501100011033.

7. References

[1] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, X. Franch. Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal. In: Proceedings of the 18th International Conference on Product-Focused Software Process Improvement (PROFES), 2017.

[2] F. Dalpiaz, X. Franch, J. Horkoff. iStar 2.0 Language Guide. arXiv:1605.07767 (2016).

[3] S. Dueñas, V. Cosentino, J.M. Gonzalez-Barahona *et al*. GrimoireLab: A Toolset for Software Development Analytics. PeerJ Computer Science 7: e601 (2022).

[4] B. Fitzgerald, K.-J. Stol. Continuous Software Engineering: A Roadmap and Agenda. Journal of Systems and Software, 123 (2017): 176-189.

[5] X. Franch, G. Grau, C. Quer. A Framework for the Definition of Metrics for Actor-Dependency Models. In: Proceedings of the 12th IEEE International Requirements Engineering Conference (RE), 2004.

[6] X. Franch, L. López, C. Cares, D. Colomer. The *i** Framework for Goal-Oriented Modeling. In: Domain-Specific Conceptual Modeling, Springer (2016).

[7] H. Gall, T. Menzies, L. Williams, T. Zimmermann (eds.): Software Development Analytics. Dagstuhl Reports 4,6 (2014): 64–83.

[8] L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, M. Oivo. How can Quality Awareness support Rapid Software Development?–A Research Preview. In: Proceedings of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), 2017.

[9] H. Huijgens, D. Spadini, D. Stevens, N. Visser, A. van Deursen. Software Analytics in Continuous Delivery: A Case Study on Success Factors. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2018.

[10] L. López, X. Franch, J. Marco. Specialization in the iStar2.0 Language. IEEE Access 7, 2019.

[11] S. Martínez-Fernández, A.M. Vollmer, A. Jedlitschka *et al*. Continuously Assessing and Improving Software Quality with Software Analytics Tools: A Case Study. IEEE Access 7 (2019): 68219–68239.

[12] A. Maté, J. Trujillo, X. Franch: Adding Semantic Modules to improve Goal-Oriented Analysis of Data Warehouses using I-star. Journal of Systems and Software 88 (2014): 102-111.

[13] T. Menzies, T. Zimmermann. Software Analytics: So What? IEEE Software 30, 4 (2013): 31-37.

[14] R. Minelli, M. Lanza. Software Analytics for Mobile Applications--Insights & Lessons Learned. In: Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013.

[15] H.H. Olsson, J. Bosch. The HYPEX Model: From Opinions to Data-Driven Software Development. In: Continuous Software Engineering, Springer, 2014.

[16] M. Oriol, S. Martínez-Fernández, W. Behutiye *et al*. Data-driven and Tool-supported Elicitation of Quality Requirements in Agile Companies. Software Quality Journal 28, 3 (2020): 931–963.

[17] M. Oriol, M. Stade, F. Fotrousi *et al*. FAME: Supporting Continuous Requirements Elicitation by Combining User Feedback and Monitoring. In: Proceedings of the 26th IEEE International Requirements Engineering Conference (RE), 2018.

[18] J. Pimentel, J. Castro. piStar Tool – A Pluggable Online Tool for Goal Modeling. In: Proceedings of the 26th IEEE International Requirements Engineering Conference (RE), 2018.

[19] S. Renault, O. Méndez, X. Franch, C. Quer. PABRE: Pattern-based Requirements Elicitation. In: Proceedings of the 3rd International Conference on Research Challenges in Information Science (RCIS), 2009.