

Best practices for software maturity improvement: a GÉANT case study ^{*}

Bartosz Walter¹, Branko Marovi², Ivan Garnizov³, Marcin Wolski¹, and
Andrijana Todosijevic⁴

¹ PSNC, Pozna, Poland {bartek.walter,marcin.wolski}@man.poznan.pl

² University of Belgrade, Belgrade, Serbia, branko.marovic@rcub.bg.ac.rs

³ Friedrich-Alexander-University of Erlangen-Nrnberg, Erlangen, Germany,
ivan.garnizov@fau.de

⁴ AMRES, Belgrade, Serbia, andrijana.todosijevic@amres.ac.rs

Abstract. Maturity models for software indicate the key areas that contribute to quality improvements. They usually combine technical, organisational and human aspects relevant for effective software development, to focus the efforts and draw the direction for optimisations. In this paper, we present the process of defining best practices that support the GÉANT Software Maturity Model (GSMM), aligned to the needs of a distributed, innovation-driven, pan-European organisation. Based on the identification of specific goals relevant for GÉANT and a preliminary maturity assessment, we created a catalogue of best practices that help the software teams to attain the goals defined in the GSMM.

Keywords: maturity evaluation · best practices · software process improvement · SPI

1 Introduction

Managing software process improvement endeavours is a complex challenge that usually involves the effort of several teams and individuals. In particular, it is the case for large organisations focused on innovation, with an established culture of diversity and openness [14]. The process improvement usually requires identification of factors relevant in a given context, setting attainable objectives, defining metrics for tracking the progress, but also coordinating the efforts in various areas: technical, human and organisational.

The concept of maturity, which captures the capability of an organisation to deliver high-quality products, is widely accepted as an effective method of

^{*} This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

The scientific/academic work is financed from financial resources for science in the years 2019-2022 granted for the realisation of the international project co-financed by the Polish Ministry of Science and Higher Education.

improving the processes. Maturity models provide frameworks capturing the essential dimensions of quality that are relevant in a given context, to set objectives and propose methods of addressing them. Several models have been proposed for software development, both generic [12,6] and tailored [9].

However, defining the model is only one side of the coin. Apart from identifying the goals and defining the metrics, the subject teams and individuals also need guidance, support and actionable recommendations on how to work to attain the objectives. Although it is quite feasible to directly implement the model in uniform and hierarchical organisations, for internally diversified and independent structures it could be a difficult task.

This also is the case of GÉANT, a pan-European organisation, established and funded under several EU programmes for the development and operation of a fast, reliable networking environment for research and education, which offers software-based services to various end-users, including students and researchers. It involves many independent software teams that are free to define their processes and obliged only to adhere to the common organisation-wide recommendations.

In this paper, we report on how a custom maturity model for GÉANT could be supported by a catalogue of best practices. The catalogue guides on how the specific objectives of the model could be addressed and implemented by the GÉANT software teams.

The paper consists of seven sections. In Sec. 2 we report a literature overview; in 3 we shortly introduce the GÉANT organisation and its specifics concerning software development. In Sec. 4 we present the entire process of maturity improvement, from defining the model to constructing the catalogue of best practices. Next, in Sec. 5 we present how the best practices are described, formatted and presented in a catalogue. Sec. 6 reports the early results of the evaluation, and the Sec. 7 provides concluding remarks and the summary.

2 Related work

Maturity of software organisations is a topic widely explored in literature. Several software-related maturity models have been developed that refer to specific areas and scopes, e.g. software process capability models (CMMI) [12], software analysis [8], operational management [13] or business process management [5]. Although maturity is usually related to traditional methods of software development, the concept of maturity has been also tailored for agile approaches: Agile Maturity Model (AMM) [7] defines levels of agility, which address the common agile practices and values starting from basic ones, e.g., planning and requirements management, up to managing uncertainty and defect prevention.

A number of maturity models have been also proposed in EU-funded projects. They address various areas that could be partially relevant in a software-related context, e.g., communication [11] or selected education techniques⁵. However, they usually focus on a single, selected dimension of the project.

⁵ <https://embed.eadtu.eu/>

As a consequence, although numerous models exist, they still need to be merged, customized or redefined to reflect the specific requirements and settings and to embrace all areas relevant for software development. To respond to this, in previous papers [14] we presented a preliminary version of the GSMM, a maturity model dedicated for the GÉANT organisation, along with recommendations on how to define models and implement them [15].

Effective implementation of the objectives and goals defined in maturity models requires also adequate guidance and recommendations. They could take the form of best practices that are well-founded on both the experience and the existing knowledge, and are applicable in the relevant context. This approach is widely adopted in software engineering, e.g., in SWEBOK [3]. Catalogues of such practices dedicated to specific areas have been proposed by various authors. Gamma et al. [10] expressed the collected experience in designing object-oriented software as a set of design patterns. They documented key design templates and presented them in the form of a catalogue of abstract structured recommendations. Also, anti-patterns have been defined, capturing practices that should be avoided [4]. Similar efforts have been undertaken also in several other software-related areas, e.g., testing, documentation etc.

Ambler [1], based on his observations, emphasized the importance of the context in the analysis of best practices. He argued that most practices are not applicable in all cases, and they needed to be either adapted before being applied or to be implemented only in specific environments.

We believe there is still a need for presenting custom, organisation-specific models for improving software maturity, supported by structured, practice-originated experience.

3 Background

GÉANT is a pan-European project focused on the development and maintenance of e-infrastructure and services for the research and education community. It operates the backbone network and associated services interconnecting national research and education networks (NRENs) across Europe and enables their collaboration. Also, it is a distributed, innovation-oriented organisation involving participants from many countries and organisations that develop and maintain network-based products and services, frequently based on dedicated or customized software. GÉANT portfolio comprises currently 30 software projects: some are used directly by GÉANT; some more are shared or used by NRENs; yet, others contribute to wider open-source communities.

Members of the software teams have specific working arrangements: they simultaneously work for GÉANT and their native organisations, can be simultaneously involved in several projects, are geographically distributed, and are placed in different cultural and professional backgrounds. The teams share the common software development framework provided by GÉANT, but are allowed to choose and customize their processes, methodologies and approaches. Their

developments are focused on innovative and often prototypical applications for the high-performance network, based on the novel and often federated services.

A previous analysis of the GÉANT software development practices [16] showed that there is a need within the GÉANT software development community for optimisation of the software development processes. This need could be addressed by providing the software teams with guidance on adopting and using software development methodologies and practices effectively and efficiently.

The motivation for the establishment of a software maturity model for GÉANT was to determine properly the improvements schema for software teams. Moreover, the practical appliance of maturity model would align the improvement effort with governance frameworks, commonly approved models, and with industry practices. The apparent attractiveness and popularity of the maturity model in the management of the software process and, more broadly, IT management, has contributed to our effort to develop a maturity model specifically for software process improvement (SPI) within GÉANT, while respecting the seven suggested requirements for the development of maturity models [2].

The GÉANT Software Maturity Model (GSMM) has been designed to achieve two primary goals: (1) to capture key practices that already help the teams to successfully deliver software, and (2) to identify areas for further improvements that could be applied by the teams [15,14].

The extracted practices can then be shared, adapted and applied by the teams to streamline and align software development processes and governance. The leading factors considered include the lasting nature of GÉANT, its products and services, distributed nature of conducted collaborations and the existence of many practices that have been established for some time.

The resulting model consists of categorised software engineering topics and processes into five key thematic areas, referred to as target areas (TA), namely: requirements engineering; design and implementation; software maintenance; quality assurance; and team organisation. These target areas were elaborated by providing specific content to the maturity model, with each one consisting of several specific goals (SGs) that capture sub-objectives and related activities.

4 Process of defining the best practices

Best practices are commonly accepted procedures that aim at accomplishing certain objectives. They are applicable in a given context. They are considered as the gold standard for attaining specific objectives.

The process of defining the best practices comprised the four main steps (see Fig. 1):

1. Defining the software maturity model which would determine our improvements schema;
2. Preparing the questionnaire which would be consistent with the structure of the software maturity model;

3. Interviewing selected GÉANT software teams to collect both qualitative and quantitative information and opinions on how the specific goals are addressed;
4. Determining the set of Common Best Practices (CBPs), based on the results of the survey, literature review and own observations made by the software management team.

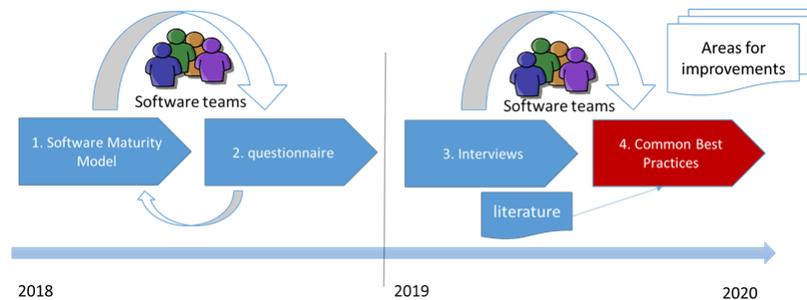


Fig. 1. The process of defining best practices adopted for GSMM

Additionally, the interviews highlighted areas for potential improvements in teams.

The first two steps (defining the software maturity model and preparing the questionnaire) have been accomplished within a few iterations: the model, its underlying conceptual framework, as well as the questionnaire, were iteratively developed in cooperation with several software teams. The other two steps (3-4) were accomplished sequentially.

4.1 Defining the maturity model and questionnaire

Maturity models are widely applied managerial instruments used for the evaluation and improvement of organisational practices and processes. The GSMM focuses on software development processes within GÉANT, considering the particular constraints in which the software teams operate. As a result, the GSMM identifies 29 Specific Goals (SGs) grouped into five Target Areas (TAs), which are essential for effective software development in GÉANT. The goals indicate objectives that need to be addressed by the software teams in the technical, organisational and human domain.

The process of defining and implementing the GSMM, its elements, produced outputs and related actions and enhancements are presented in Fig. 2. The process of identification of specific elements of the GSMM and their further refinements is conducted iteratively, based on external and internal sources of the domain knowledge. In particular, a number of pilot interviews with the teams helped to identify key areas that are relevant in GÉANT, and further to

decompose them into individual objectives. What is important, the identification process was not limited to the activities directly related to software development. We were also interested in capturing organisational, communication and human perspectives that are relevant for software teams.

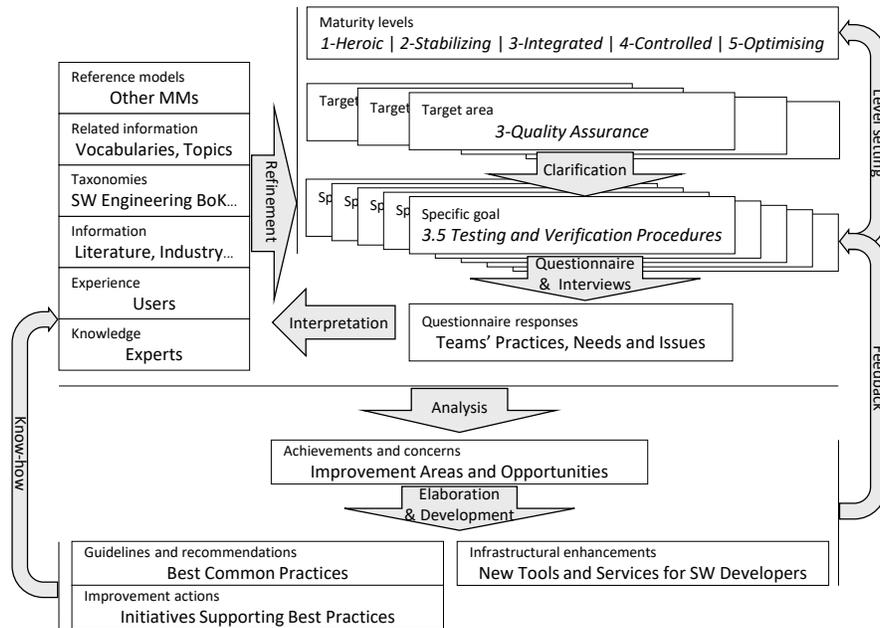


Fig. 2. The overview of GÉANT Software Maturity Model development

This model has been elaborated in close cooperation with selected software teams, by applying iterative refinements, improvements and collecting feedback. Specifically, the process included the following phases:

1. **Defining a preliminary version of GSMM.** Based on the observation of software teams, pilot interviews with the teams and a literature review, an initial version of GSMM was drafted.
2. **Validating the preliminary version.** The preliminary version of GSMM was presented to representatives of selected teams and the executive managers responsible for supporting software development in GÉANT, to collect early feedback that could be immediately addressed.
3. **Revising the GSMM.** Based on the collected feedback, the core elements of GSMM have been revised to better align it with the needs and practice of the teams.
4. **Formalising the GSMM.** In parallel to that, the GSMM has been formalised into a framework by defining its elements. As a result, it can also be adapted to other application domains, beyond software development.

4.2 Data collection for the Common Best Practices

To identify and capture the practices applied by the SWD teams, as well as the teams expectations concerning maturity, we created a questionnaire for GSMM and used as a blueprint. The questionnaire covers us meant to extract comments from the teams on each GSMM-specific goal by asking a number of closed, open and rating questions. The questions were formulated to elicit useful explanations on whether some aspects are covered or significant, how the evaluated team addresses each specific goal, what resources and approaches are used, what outputs are produced, and in which areas support may be needed. In total, we interviewed 12 teams, a representative subset of the active software teams with varied sizes.

This resulted in an extensive but targeted and streamlined process of data collection, in which the participants were able to effectively and purposefully capture their teams software development and management practices, but during which it was also possible to capture the teams' attitude and perception of the specific topics.

5 Common Best Practices

The GSMM, presented in Sec. 4, identifies the core areas and goals that contribute to the effective software development in GÉANT. However, software teams need not only the targets but also the guidance on how to organise the efforts towards meeting these them, considering the specific context of the organisation. To address this, we created a catalogue of Common Best Practices (CBPs). They have been identified based on three sources of information:

- practices currently applied by the software development teams, extracted during the questionnaire interviews (see Sec. 4),
- recommendations provided in the literature and case studies concerning similar process optimisation efforts,
- direct or indirect observations made by the software management team.

The catalogue includes recommendations showing how the particular GSMM specific goals could be addressed, considering the constraints and opportunities present in GÉANT. Therefore, the best practices balance between the need for providing detailed operational guidance and giving a general direction. Both of these are extremes that would result in missing the objective of providing the teams with effective guidance that they could adapt and implement in their own practice.

5.1 Template of a Best Practice

Description of a best practice includes a large volume of diversified information of various nature and format, which can hinder its readability. To make the practices more accessible for the team members and, consequently, facilitate the

adoption of practices, we decided to define a template which presents the data in a structured way. Each practice is presented as a set of attributes that describe key elements of the practice. The template comprises the following attributes:

- **Objective** that the practice is expected to address;
- **Applicability**, describing types of projects or their phases, in which the practice could be effectively applied;
- **Context** that captures a specific setting (a set of technical, managerial or organisational constraints), for which the practice was identified and for which it is recommended; The practice can be used in other settings, but it may require additional validation;
- **Addressed elements in GSMM**, linking the practice with the SGs in the GSMM;
- **Prerequisites**, listing the condition necessary to apply the specific practice;
- **Recommendation**, outlining actions that should be undertaken to meet the SG; It includes high-level directional advice on *what* to do, with lower-level details on *how* to reach the goal.
- **Risks**, describing possible risk factors and their consequences in case of misusing the practice;
- **Related practices**, indicating other practices that could be applied in a similar context;
- **Origin**, providing details on how the practice originated.

5.2 Example

As an example, we present a practice related to managing stakeholders that belongs to the Requirements Engineering target area. It deals with the problem of identification of relevant organisations, teams and individuals, who would affect the project or are interested in its outcomes, and properly addressing their needs and expectations.

- **Objective:** Identify relevant stakeholders that can contribute to the project or have an impact on it
- **Context:** The practice applies to all projects.
- **Addressed elements in GSMM:** RE-1. Identification and overall management of stakeholders
- **Prerequisites:** none
- **Recommendation(s)**
 1. Identify an initial group of stakeholders
 - (a) Consider teams, NRENs or individuals that could be affected or could impact the project.
 - (b) Look for similarities to other projects, either previous or current.
 - (c) Look for a dominant stakeholder, who is mostly interested in the outcome of the project
 2. Maintain (update) the group of stakeholders
 - (a) Publish the list of stakeholders and their representatives

- (b) Periodically update (involve and retire) the group of stakeholders
 - (c) Apply snowballing to identify new stakeholders
 - (d) Categorise the stakeholders according to their relevance for the project
- **Risks:**
1. The identified group does not include all relevant stakeholders
 - (a) The project may be subject to tensions, sudden changes or drifting.
 - (b) The decisions could be made/affected by people not officially involved in the project.
 - (c) The project would be not driven by stakeholders, but rather by the project team.
 2. Group of stakeholders is not properly updated
 - (a) The group may not reflect the actual balance of interests.
- **Related practices:** BP-A-2: Create a strategy to communicate with stakeholders.
- **Origin:** This practice has been defined based on the survey, supported by the observation by the software management team.

The recommendations do not provide direct instructions for the teams on how to proceed in a step-by-step manner, but rather give directional guidance. It includes advice concerning the factors and issues that facilitate addressing the respective goal in the GSMM and that should be considered by the team. As a result, the recommendations presented in a practice are a trade-off between the desire to deliver actionable procedures on one hand and the necessary abstractness on the other. The software teams are expected to analyse and adapt the recommendations to their local context.

The risk factors capture possible consequences of applying the practice improperly. In this case, they are mostly related to issues of stakeholders identification and management, including prioritisation and identification of relationships among them.

This specific practice is closely related to the process of defining the communication policy and maintaining contact with stakeholders. These two practices should be used together to ensure the maximum benefit from their implementation.

The entire catalogue is available for the GÉANT staff and currently includes 24 practices divided into five target areas that directly correspond to areas in the GSMM. Each practice addresses one or more SGs defined in the GSMM and, currently, all the goals are covered.

6 How to identify key areas for improvements?

One of the goals for analysis was to identify the areas that deserve most attention and effort. The data collected so helped in the selection of topics that should be promoted, worked on and supported through the improvement incentives. Below, we provide comments on them:

- First, we focused on the areas in which the declared knowledge and satisfaction of the software teams was the most diversified. For those items, we can rely on the immediately available internal expertise of some teams, which reduces the effort needed for process improvement. Such harmonisation of processes among teams could additionally strengthen their collaboration.
- Another issue refers to areas for which the survey scores were generally low or mediocre. These can be interpreted two-fold: either (1) the survey has identified a relevant topic that has been underestimated and has not been covered adequately, or (2) the topic is considered irrelevant by developers. To determine the actual status of these topics, teams and their leaders should be consulted. For that reason, it is useful at this point to establish a regular mechanism for collecting feedback from software teams. The apparent difficulty associated with these topics is the insufficient internal experience, so it may be necessary to look for external expertise and support to achieve the expected improvements.
- The third group of interest includes the areas with uniformly high marks. Here, we can expect relatively small improvements, even if the collected data is not completely accurate. High grades given by all teams may indicate that the reached level cannot or should not be further improved.

The instruments established to collect feedback from software teams should not be used just to address the dilemmas related to the selection of improvement areas. They could be also applied to get a response about the ongoing improvement initiatives and track the metrics that are relevant for the implementation of the GSMM. However, conducting extensive interview-based surveys, like the one described in Section 4, requires significant effort; at next stages, they could be supported with simpler and frequently run online questionnaires that focus on the key elements.

Optionally, this approach could be taken even further by associating the indicators with maturity levels. Maturity is assessed against key capability and practice characteristics linked with the selected target areas at each level of the model. Maturity levels are typically established using a five-point Likert scale where the higher the level, the higher is the level of maturity. These progressive levels guide the planning and development of roadmaps. Currently, there is no need for GÉANT to develop such a far-reaching maturity model. A future move in this direction would require reaching beyond GÉANT into an even larger base of software teams to capture the indicators and link their values to maturity levels. Given the number of potential indicators and certain subjectivity in associating them with maturity levels, it is necessary to draw from the opinions and attitudes of a larger community to get non-disputable criteria. Although other related standardisation processes and schemes and maturity models could additionally support this development, it is crucial to establish and employ a simple, inclusive and quality process in which the indicators and levels are produced and challenged. Again, a series of simple and low-effort online surveys could be used to iteratively establish and maintain the set of used indicators and maturity criteria.

7 Summary and conclusions

The presented approach aims to establish an adaptive and practical framework for improving the quality and maturity of practices within GÉANT. It is primarily aimed at improving coordination of software process improvement efforts, fostering the collaboration of software teams, and supporting the entire software development life cycle with best practices.

In particular, the catalogue of best practices does not only help the teams to address and implement the objectives set in GSMM but also allows them to adapt the recommended activities to their contextual constraints. As such, it fits well into the SPI Manifesto⁶ that emphasizes the need for embedding the maturity improvement efforts in practice and the actual needs of software teams. We believe that the catalogue of best practices will become a live, ever-growing toolbox of recommendations that would provide substantial guidance for the teams, but also receive updates from them.

References

1. Ambler, S.: Questioning Best Practices for Software Development: Practices are Contextual, Never Best (2011 (accessed April, 2020)), <http://www.ambyssoft.com/essays/bestPractices.html>
2. Becker, J., Knackstedt, R., Pppelbu, J.: Developing maturity models for IT management. *Business & Information Systems Engineering* **1**(3), 213–222 (2009). <https://doi.org/10.1007/s12599-009-0044-5>
3. Bourque, P., Fairley, R.E., Society, I.C.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0. IEEE Computer Society Press, Washington, DC, USA, 3rd edn. (2014)
4. Brown, W.J., Malveau, R.C., McCormick, H.W.S., Mowbray, T.J.: *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis: Refactoring Software, Architecture and Projects in Crisis*. John Wiley & Sons, 1. edn. (1998)
5. de Bruin, T., Rosemann, M.: Using the Delphi technique to identify BPM capability areas. *ACIS 2007 Proceedings - 18th Australasian Conference on Information Systems* (01 2007)
6. Burnstein, I., Suwanassart, T., Carlson, R.: Developing a testing maturity model for software test process evaluation and improvement. In: *Proceedings International Test Conference 1996. Test and Design Validity*. pp. 581–589 (Oct 1996). <https://doi.org/10.1109/TEST.1996.557106>
7. Chetankumar, P., Ramachandran, M.: Agile maturity model (AMM): A software process improvement framework for agile software development practices. *International Journal of Software Engineering* **2** (01 2009)
8. Covey, R., Hixon, D.: The creation and use of an analysis capability maturity model (ACMM). Tech. rep., AEROSPACE CORP EL SEGUNDO CA LAB OPERATIONS (2005)
9. Fontana, R., Jr, V., Reinehr, S., Malucelli, A.: Progressive outcomes: A framework for maturing in agile software development. *Journal of Systems and Software* **102**, 88–108 (04 2015). <https://doi.org/10.1016/j.jss.2014.12.032>

⁶ <https://2019.eurospi.net/index.php/manifesto>

10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1 edn. (1994)
11. Muszynska, K.: Communication maturity model for organizations realizing eu projects. *Procedia Computer Science* **126**, 2184–2193 (01 2018). <https://doi.org/10.1016/j.procs.2018.07.230>
12. Paulk, M., Curtis, B., Chrissis, M., V. Weber, C.: Capability Maturity Model for Software, Version 1.1 (01 1993)
13. Renken, J.: Developing an IS/ICT management capability maturity framework. In: Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries. pp. 53–62. South African Institute for Computer Scientists and Information Technologists (2004)
14. Stanisavljevic, Z., Walter, B., Vukasovic, M., Todosijevic, A., Labedzki, M., Wolski, M.: GÉANT software maturity model. In: 2018 26th Telecommunications Forum (TELFOR). pp. 420–425 (Nov 2018)
15. Walter, B., Wolski, M., Stanisavljevic, Z., Todosijevi, A.: Designing a Maturity Model for a Distributed Software Organization. An Experience Report, pp. 123–135 (08 2019)
16. Wolski, M., Adomeit, M., Golub, I., Martinovic, R., Kupański, S., Labedzki, M., Javaid, S., Todosijevic, A., Radulovic, A., Visconti, S., Mazurek, C.: Deliverable D5.3 - Analysis of Requirements for Software Management. Tech. rep. (2017), https://www.geant.org/Projects/GEANT{}_Project{}_GN4/deliverables/D5-3{}_Analysis-of-Requirements-for-Software-Management.pdf