Chapter 2
Requirements Engineering: Best Practice

Samuel A. Fricker, Rainer Grau, and Adrian Zwingli

Abstract Many software solutions have failed because they did not meet stakeholder needs. In response to this problem a massive amount of techniques were developed to elicit stakeholder needs, to analyze the implications of these needs on the software, to specify proposed software products, and to check acceptance of these proposals. However, many of these techniques did not become industrial practice because they were not practicable or ineffective when used in real-world projects. To obtain an overview of what common practice is and to understand which techniques reflect best practice because they are particularly effective, we have surveyed a large number of industry projects. Based on 419 valid answers, this chapter gives an overview of commonly used requirements engineering techniques. It also shows which of the techniques, when used in a software project, correlate with requirements engineering success. The chapter concludes with recommendations for software projects and future research to improve requirements engineering practice.

2.1 Introduction

In 1995 the consultancy company Standish Group International published results of an industry survey that showed that only 16 % of the software projects were successful, 53 % were challenged, and 31 % complete failures [1]. Successful projects were those that completed on time and budget and produced a software product with all features and functions as initially specified. The low success rates described in the Standish report generated substantial attention by industry and politics.

S.A. Fricker (✉)
Software Engineering Research Laboratory, Blekinge Institute of Technology, Karlskrona, Sweden
e-mail: samuel.fricker@bth.se

R. Grau
Zühlke Engineering AG, Schlieren, Switzerland
e-mail: rainer.grau@zuehlke.com

A. Zwingli
SwissQ Consulting AG, Zürich, Switzerland
e-mail: adrian.zwingli@swissq.it

S.A. Fricker et al. (eds.), *Requirements Engineering for Digital Health*,
DOI 10.1007/978-3-319-09798-5_2

The Standish survey pointed to software project practices that needed improvement. According to the respondents, the most frequently stated factors that influenced project success were user involvement, executive management support, and a clear statement of requirements. These factors show that requirements engineering is crucial to achieve project success. User involvement is critical for building a software that will be understood by the users, that will be used appropriately, and that creates joy [2]. Management support is critical to align the software with the strategic goals of the organization [3]. Clearly stated requirements contribute to a shared understanding between the project team and the software product's users, management, and other stakeholders. The shared understanding reduces the risk of unsatisfactory outcome and rework of project results [4]. Influenced by these insights, software engineering practice matured over time. Thirty-two percent of the software projects were successful according to the Standish survey published in 2009 [5].

Even though many requirements engineering techniques exist for involving users, for obtaining management support, and for achieving shared understanding, we lack an understanding of whether these techniques make requirements engineering successful. Some researchers believe that no technique would do and claim that good requirements practices are neither sufficient nor necessary [6]. The best we can say today is that the techniques are used inconsistently: some techniques get used by some projects but not by others [7, 8]. Companies that care about requirements engineering seem have a preference for Quality Function Deployment, prototyping, Data Flow Diagrams, role playing, and decision trees [9]. However, we do not know whether any of these techniques correlates with requirements engineering success, thus should be used systematically.

This chapter intends to develop an understanding of what practice makes requirements engineering successful by reporting the results from an own large-scale industry survey. The survey investigated whether the use of requirements engineering techniques differed between projects with successful and unsuccessful requirements engineering. The results show that a few techniques indeed correlated with success. In addition, also the ability to apply a broad variety of requirements engineering techniques is important. These results imply that best practice would be to utilize the few effective techniques and pragmatically select complementing techniques that suit well the type of software being developed and the situation that requirements engineering is confronted with.

The remainder of this chapter is structured as follows. Section 2.2 gives an overview of requirements engineering state of the art that was studied in the industry survey. Section 2.3 describes the survey methodology. Section 2.4 characterizes the projects that have responded to the survey, gives an overview of the requirements engineering practice of these projects, and shows the correlation of requirements engineering practice with success. Section 2.5 discusses the obtained results, gives recommendations for practice, and suggests implications for research. Section 2.6 summarizes and concludes.

2.2 Requirements Engineering State of the Art

2.2.1 Requirements Engineering Techniques

There is a long tradition of research and practice in requirements engineering. One of the early influential works describes requirements engineering as inquiry [10]. During an inquiry the requirements engineer asks questions about a future software product to stakeholders and turns the obtained answers into a specification. While doing so, new questions emerge that are posed again to the stakeholders to initiate the next inquiry.

Since these early days, a large number of techniques have been investigated to advance requirements engineering state of the art [11]. Still, Potts's inquiry remains a good model of how to think of requirements engineering in a software project. Today, a requirements engineer is expected to elicit needs and expectations from stakeholders, to model and analyze the impact of these inputs on the system together with the development team, and to check proposed implementations for acceptance by the stakeholders [12]. Once both the stakeholders and the development team agree, the requirements are used to steer development and, upon release of the solution, check whether the developed product fulfils the agreement. If the inquiry is done well, one can observe that a shared understanding emerges, requirements stabilize, and the stakeholders become satisfied [13, 14].

During elicitation the requirements engineer aims at understanding the project vision and constraints, the context that the product will be deployed into, and the stakeholders that will need to accept the product [15, 16]. Such requirements elicitation results in an overview of users, external systems, and other stakeholder viewpoints and a description of their respective background, interests, and expectations. A large number of techniques are known to elicit such information about the system requirements [17–19]. Table 2.1 gives an overview of selected elicitation techniques.

During analysis the requirements engineer aims at understanding how the requirements will be implemented by the software system [30], how they will be considered in the development plan [31], and how they will be used for the testing of the system [32]. Requirements analysis typically results in one or more prototypes, a definition of project scope or release plan, and a requirements specification for the system. Table 2.2 gives an overview of selected analysis techniques, Table 2.3 of planning techniques, Table 2.4 of relevant requirements types, and Table 2.5 of specification techniques.

During requirements checking, the requirements engineer checks that the right approach has been selected for fulfilling the vision and achieving the system goals and that the system will be accepted by the stakeholders. Requirements checking initiates a new inquiry cycle if the checked requirements turn out to be not good-enough. Requirements checking marks the agreement of the stakeholders on

Table 2.1 Selected requirements elicitation techniques

Technique	Description
Archaeology	Analysis of existing systems to understand their functionality, quality, and usage [20].
Creativity	The generation and selection of ideas to innovate or solve a difficult problem [21, 22].
Data mining	Search and filtering of requirements databases to identify relevant knowledge about stakeholder needs [23].
Interview	Meeting between a requirements engineer and a stakeholder to discuss topics of relevance for the system [24].
Introspection	Use of domain knowledge in combination with reflection and empathy to base requirements on experience [25].
Observation	Study of system use, possibly in the target environment and by real users, to understand usage processes and strengths and weaknesses of a current system [26].
Questionnaire-based survey	Paper or electronic form with questions and space for answers distributed to stakeholders to obtain an overview of stakeholder opinion [27].
Reuse	Use of existing specifications to avoid reinvention of requirements that already are adequate [28].
Workshop	Meeting between a requirements engineer and stakeholders to reach agreement between the workshop participants [29].

Table 2.2 Selected system analysis techniques

Technique	Description
Domain-driven development	Specifying the concepts of relevance in the context the system will be deployed into that are to be implemented or respected by the system [33].
Formal specification	Use of mathematical or formal-logic expressions to enable automated checking of completeness consistency, and correctness [34].
Informal Modeling	Sketching a model of something of relevance to reflect and discuss how the parts of that thing interrelate [35].
OOA	Specifying the structure, functionality, and behavior of the system usually with the object-oriented analysis language UML [36].
Prototyping	Paper- or tool-based approximation of the end-systems to increase the tangibility and authenticity of the planned system [37].
Quality checks	Checking whether the system fulfils its goals and whether functionality and quality are adequate and needed [38].
SA	Specifying the structure, functionality, and behavior of the system with a structured analysis language [39, 40].

contents and scope of the development project if the checking has been successful. Table 2.6 gives an overview of selected checking techniques.

The inquiry cycle leads to a dialogue between stakeholders and development team that can be seen as a negotiation [64]. The negotiation results in an agreement between the stakeholders and the development team about the product to be developed. This agreement, represented by the approved requirements specification, is then

Table 2.3 Selected requirements planning techniques

Technique	Description
Business case	Evaluating whether a set of requirements has to good return-of-investment and should be included into project scope [41].
Prioritizing	Ranking the requirements to obtain an order of how they shall be addressed by the project work [42].
Release planning	Defining the contents of one or more releases to define the scope of the software system [43].
Road mapping	Coarse-grained, long-term planning to agree with stakeholders and suppliers for how the software system shall evolve [44, 45].
Triage	Filtering the requirements to determine what requirements are relevant and what requirements are not [46].
Vision	Defining the problem that is addressed, the key idea of the solution, and how the solution improves state of the art to align the work of developers and stakeholders [47].

Table 2.4 Selected requirement types

Type	Description
Behavior	Behavior is a sequence of states that determine how a system, artifact, or class reacts to events [48].
Formal property	A formal property can be tested for correctness, completeness, and consistency with automated tools [49].
Function	Function is a reaction to inputs or an action of a system [50].
Glossary	A glossary defines terms, abbreviations, acronyms, synonyms, and homonyms [12].
Interface	An interface connects a system with its environment. Typical interfaces are user interfaces [51] and interfaces to other software systems [52].
Process	A process is a series of actions or operations implemented by people, organizations, or software to achieve a goal [53].
Quality	Quality is a characteristic of a software system such as performance, reliability, security, compatibility, portability, usability, and maintainability [54].
Scenario	A scenario is a story of how users and systems interact to achieve a goal [55].
Stakeholder	A person, group, or organization who gains or loses something with the software [56]. May be denoted agent [57] or actor [36].
Structure	Structure refers to entities or systems with their attributes and relationships [36].

Table 2.5 Selected specification techniques

Technique	Description
i* or KAOS	Specifying agents, goals, and formal properties with formal languages to enable reasoning about goals and goal-achievement [57].
Natural language	Specifying requirement with words and sentences to achieve specification flexibility and understandability. Language templates may be used to improve precision [58].
SA diagrams	Specifying functions, processes, structure, and behavior with one of the graphical notations proposed by structured analysis to achieve precision and make structure visible.
Tables	Specifying concepts to achieve an understanding of the terminology [59] and or rules for how conditions affect system behavior [60].
UML diagrams	Specifying functions, scenarios, processes, rules, relations, behavior, and deployment with graphical notations from the Unified Modeling Language to increase precision and show structure.
User screens	Specifying the user interface with paper or tool-based mock-ups to increase the tangibility and authenticity of the planned system.

Table 2.6 Selected checking techniques

Technique	Description
Automated checking	Testing a formal specification of the system to detect conflicting and missing requirements [61].
Inspection	Review of the requirements specification by all relevant stakeholders with a formal process that is effective at discovering problems and leads to in-depth understanding of the specification [62].
Peer review	Feedback by one or more requirements engineers to support and assure the quality of the specification work.
Prototype review	Discussion and use of the prototype, for example in a role-play, to explore uses and check acceptance of the system.
Simulation	Approximation and review of the behavior of the system with an appropriate tool to check correctness of the behavior [63].
Walk-through	Efficient review of the requirements specification by discussing the requirements specification in their sequence with stakeholders.

Table 2.7 Selected requirements negotiation techniques

Technique	Description
Conflict management	Discovering and resolving conflicts among stakeholders and between stakeholders and development team [12].
Handshaking	The review and discussion of implementation proposals to align the planned implementation of the software system with stated and unstated stakeholder needs [65].
Negotiation analysis	Analyzing possible negotiation outcomes and selecting a value-creating, fair agreement [66].
Power Analysis	Analyzing power and influence of stakeholders and planning how to interact with them[67].
Prioritizing	Ranking the requirements to obtain an order of how they shall be addressed by the project work [42].
Strategy alignment	Aligning requirements with company strategy, for example through explicit traceability [3].
Variant analysis	Analyzing and selecting alternative features or ways of solving a problem [68].
Win-win negotiation	Structured, possibly tool-supported approach to identification of options for agreement and selection of the appropriate option [69].

baselined and used to manage the development project and the release of the developed product. Table 2.7 gives an overview of selected requirements negotiation techniques and Table 2.8 of requirements management techniques.

2.2.2 Requirements Engineering Success

For evaluating requirements engineering practices, one needs to understand how to measure requirements engineering success. The most thorough study that answered this question was a survey that tested 32 indicators with 30 requirements

Table 2.8 Selected requirements management techniques

Technique	Description
Baselining	Versioning requirements and specifications and communicating these as a baseline to stakeholders [70].
Change management	Controlled process of collecting change requests, analyzing impact, and deciding about the change [71].
Process measurement	Measuring requirements engineering and implementation efficiency, for example in the form of value stream analysis [72].
Progress tracking	Monitoring the life cycle of requirements from discovery to selection, implementation, and release [73].
Report generation	Generation of reports, such as requirements specifications, from a database of requirements.
Traceability management	Maintaining relationships between requirements and possibly other artifacts to express dependencies, conflicts, and synergies [74].

Table 2.9 Success measurements for requirements engineering [75]

Quality of RE service	Quality of RE products
Business-technical alignment: fit with strategy, ability and willingness to make business changes, and management support.	*Quality of cost–benefits analysis*: completeness and coverage of cost–benefit analysis, new benefits created by the new solution, and sufficient accuracy of cost estimates.
Stakeholder acceptance: awareness of business changes, extent of consensus, willingness to defend solution, and relationship to users.	*Argumentation of impact*: diagnosis of existing solution, traceability of supported processes to problem to be solved and to system goals, and traceability of strengths and weaknesses of new solution to replaced solution.

engineering experts [75]. It showed that requirements engineering success can be measured with *quality of requirement engineering service* and *quality of requirements engineering products*. Table 2.9 gives an overview of the indicators.

The quality of requirements engineering service refers to the effects a requirements engineer wants to achieve. These concern the alignment of the software product with business objectives and the alignment of it with stakeholder needs and expectations. Such alignment can be checked by asking the concerned stakeholders of whether they agree that system will deliver the desired impacts.

The quality of requirements engineering products refers to the work results delivered by the requirements engineer. These should include a comprehensive cost–benefit analysis and a description of impact with detailed traceability to supported processes, system goals, and the replaced solution. Such work results are tangible and can be easily inspected if they are presented in the form of a requirements specification.

In this chapter we use El Emam and Madhavji's success measurements to inquire what requirements engineering goals were important and whether these goals were achieved. This way of assessing the quality of requirements engineering service and products allows taking into consideration the many possible variations of what is important in given projects. It allows the respondents to judge whether requirements engineering was successful according to their own specific contexts.

El Emam and Madhavji's success measurement have the advantage of being measurable immediately when requirements engineering is concluded. However, they fall short in capturing the ultimate objective of requirements engineering. No measurement has been proposed to assess whether the specified system will be successful. For that reason, we extend the measurement framework outlined in Table 2.9 with the additional dimension of requirements engineering outcome. In the survey we thus ask the respondents whether the specified product met the goals the product was conceived for.

2.3 Industry Survey

We investigated the use of requirements engineering techniques and how much they contributed to requirements engineering success with an online survey [76]. We distributed an online questionnaire to people involved in software projects in an attempt to answer the following main research questions.

- RQ1: What requirements engineering techniques are used in software projects?
- RQ2: What are the goals pursued in requirements engineering?
- RQ3: Which requirements engineering techniques correlate with requirements engineering success?

The answers to RQ1 show how frequently each of the requirements engineering techniques is used, thus allows us to say what common practice is. Besides benchmarking practice, these results allow judgment whether techniques that were investigated in research were successfully transferred or not. The answers to RQ2 tell us what requirements engineers try to achieve with their work and with the systems they specify. The results show the priorities that are set for requirements engineering work. The answers to RQ3, finally, tell us what requirements engineering techniques matter most because they are associated with success more than other techniques do.

We built the questionnaire by first basing it on requirements engineering state of the art [11, 12] and then adjusting it based on suggestions from practitioners with broad overview on the software industry. A focus group with experienced practitioners evaluated adequacy, coverage, and understandability of the questionnaire. We then tested and further improved the form by letting respondents fill it in and give feedback in interviews.

To know who was answering, the questionnaire asked respondents to characterize their most recent software project. To answer RQ1, the questionnaire asked multiple-choice questions about requirements-related inquiry, specification, and management techniques in the characterized project. To answer RQ2, it asked multiple-choice questions about the goals of requirements engineering in the project and of the specified product. To answer RQ3, it asked questions about the requirements engineering success. Free-text areas allowed expanding or qualifying the answers.

The theoretical population of the survey was all software projects that were recent when the survey was administered in 2012. The sampling frame was the industrial contacts of our partners in academia and industry. To increase the reach of

the survey we encouraged subjects to recommend the questionnaire to their own contacts.

Six hundred twenty-five respondents, about 10 % of the invited persons, answered the online questionnaire. Filtering for completeness and plausibility reduced the data to 419 valid answers. This number of answers makes this requirements engineering survey by far the largest ever published. The obtained number of samples allowed describing requirements engineering practice with a margin of error smaller than ±5 % for 95 % confidence.

We answered the research questions with statistical analysis. Descriptive statistics of proportions were used for answering RQ1 and RQ2. The difference of proportions test, a variation of the independent samples *t* test, was used for answering RQ3. To control the accumulation of type I error, Holm's step-down method [77] was used to prune the *t* test results for statistical significance. Wilcoxon's rank-sum test, finally, allowed us to explore an additional angle to answer RQ3, whether the number of requirements engineering practices that are used in a software projects would correlate with success.

2.4 Requirements Engineering Practice and Success

2.4.1 Responding Projects

A diverse mix of software projects answered the survey. Figure 2.1 gives an overview of the answering projects, the kinds of software products they developed, and the companies they belonged to.

Many projects were performed at large companies in Switzerland and developed information systems. This distribution is consistent with the Switzerland-oriented contact networks we used for soliciting responses. The key employer in this country is the service sector with IT departments that produce information systems.

A wide spread of industries were addressed with the developed products. Thirty-five responses, 8 % of all responses, were given by projects that developed products for health care. Thus the results reflect practice across industries and are not specific to one of them, for example health care. Product novelty was relatively evenly spread.

A majority of the projects were bespoke and developed tailor-made solutions. The projects used a sequential, incremental, or hybrid development process. Only few did research or used a process like the Spiral model that is designed for experimentation. The long duration of the projects may be explained by the prolonged relationship that IT departments have with the business units they support. The same relationship can also explain the bespoke nature of the projects. Information systems developed for business units are used by a predetermined set of users that can be actively involved into the requirements engineering process.

Project		
Category		
Research	32	8%
Product, Platform	87	21%
Bespoke	237	57%
Tender	42	10%
Other	21	5%
Process (Lifecycle Model)		
Prototyping	23	5%
Evolutionary	39	9%
Incremental	113	27%
Hybrid	104	25%
Sequential	135	32%
Other	5	1%
Size (number of staff)* **		
< 4	54	13%
4-9	107	26%
10-19	89	21%
20-49	84	20%
>= 50	82	20%
n/a	3	1%
Duration (calendar months)* **		
< 4.5	27	6%
4.5-9	97	23%
9-18	119	28%
>= 18	165	39%
n/a	11	3%

Product		
Industry (application domain)		
Banking, Finance	98	23%
Automotive, Transport	54	13%
Software, IT	51	12%
Government, Military	40	10%
Healthcare, Medical	35	8%
Insurance	31	7%
Telecommunications	31	7%
Manufacturing, Supply	22	5%
Other	57	14%
Type		
Information System	260	62%
Software-Intensive	62	15%
Engineering	28	7%
Other	69	16%
Innovation		
Completely New	149	36%
New Features or Use	107	26%
Changed Technology	79	19%
Maintenance	84	20%

Company		
Size (number of employees)* **		
< 10	23	5%
10-49	39	9%
50-249	42	10%
250-4499	144	34%
>= 4500	168	40%
n/a	3	1%
Location		
Europe	368	88%
Switzerland	248	59%
Germany	69	16%
Other Europe	51	12%
Americas	26	6%
Asia-Pacific	25	6%

***: n/a was a possible answer,
light color: sub-categories,
"Other": answers with less than 5% frequency.

Fig. 2.1 Demography of projects that responded to the survey

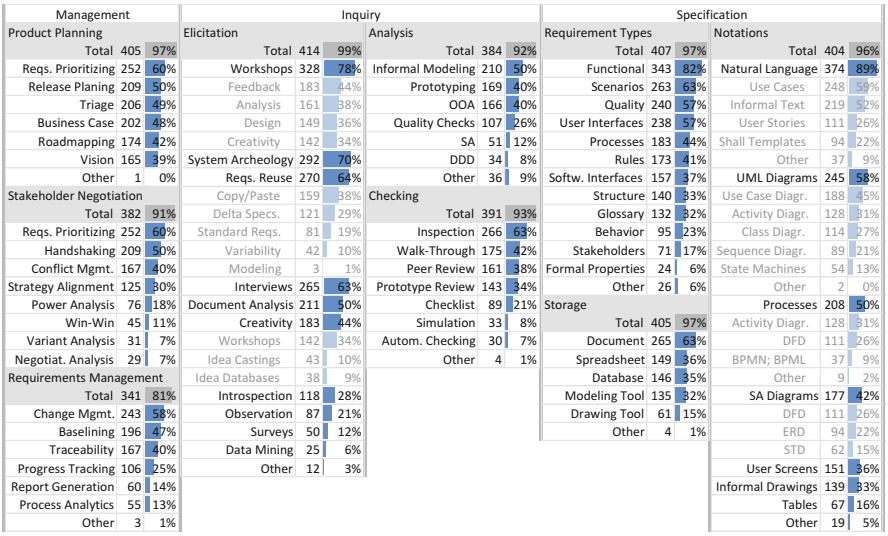
2.4.2 Common Practice

Our data shows that requirements engineering was widely established. However, there was not one way of doing requirements engineering. While only few of the techniques are employed by almost all projects, e.g., workshops, many of the techniques are used by some of the projects only. This result indicates a wide variety of how requirements engineering is done. Figure 2.2 gives an overview of how frequently each requirements engineering practice was used.

Almost every project elicited requirements. The projects tended to do with stakeholder workshops, by studying existing systems, or by reusing specifications. Workshops dominated requirements elicitation practice. Only few projects used techniques like observation, ethnography, surveys, or data mining. These techniques are thus used in special situations only.

Almost every project planned the product to be developed, often by prioritizing requirements. Often a mix of planning techniques was used. No technique was dominant.

Almost every project analyzed requirements. Often a mix of informal modeling, prototyping, and object-oriented analysis was used. No analysis technique was dominant. Historically important techniques like structured analysis, quality function deployment, and decision trees or specialized techniques such as domain-driven development were uncommon.



Management		
Product Planning		
Total	405	97%
Reqs. Prioritizing	252	60%
Release Planing	209	50%
Triage	206	49%
Business Case	202	48%
Roadmapping	174	42%
Vision	165	39%
Other	1	0%
Stakeholder Negotiation		
Total	382	91%
Reqs. Prioritizing	252	60%
Handshaking	209	50%
Conflict Mgmt.	167	40%
Strategy Alignment	125	30%
Power Analysis	76	18%
Win-Win	45	11%
Variant Analysis	31	7%
Negotiat. Analysis	29	7%
Requirements Management		
Total	341	81%
Change Mgmt.	243	58%
Baselining	196	47%
Traceability	167	40%
Progress Tracking	106	25%
Report Generation	60	14%
Process Analytics	55	13%
Other	3	1%

Inquiry		
Elicitation		
Total	414	99%
Workshops	328	78%
Feedback	183	44%
Analysis	161	38%
Design	149	36%
Creativity	142	34%
System Archeology	292	70%
Reqs. Reuse	270	64%
Copy/Paste	159	38%
Delta Specs.	121	29%
Standard Reqs.	81	19%
Variability	42	10%
Modeling	3	1%
Interviews	265	63%
Document Analysis	211	50%
Creativity	183	44%
Workshops	142	34%
Idea Castings	43	10%
Idea Databases	38	9%
Introspection	118	28%
Observation	87	21%
Surveys	50	12%
Data Mining	25	6%
Other	12	3%
Analysis		
Total	384	92%
Informal Modeling	210	50%
Prototyping	169	40%
OOA	166	40%
Quality Checks	107	26%
SA	51	12%
DDD	34	8%
Other	36	9%
Checking		
Total	391	93%
Inspection	266	63%
Walk-Through	175	42%
Peer Review	161	38%
Prototype Review	143	34%
Checklist	89	21%
Simulation	33	8%
Autom. Checking	30	7%
Other	4	1%

Specification		
Requirement Types		
Total	407	97%
Functional	343	82%
Scenarios	263	63%
Quality	240	57%
User Interfaces	238	57%
Processes	183	44%
Rules	173	41%
Softw. Interfaces	157	37%
Structure	140	33%
Glossary	132	32%
Behavior	95	23%
Stakeholders	71	17%
Formal Properties	24	6%
Other	26	6%
Storage		
Total	405	97%
Document	265	63%
Spreadsheet	149	36%
Database	146	35%
Modeling Tool	135	32%
Drawing Tool	61	15%
Other	4	1%
Notations		
Total	404	96%
Natural Language	374	89%
Use Cases	248	59%
Informal Text	219	52%
User Stories	111	26%
Shall Templates	94	22%
Other	37	9%
UML Diagrams	245	58%
Use Case Diagr.	188	45%
Activity Diagr.	128	31%
Class Diagr.	114	27%
Sequence Diagr.	89	21%
State Machines	54	13%
Other	2	0%
Processes	208	50%
Activity Diagr.	128	31%
DFD	111	26%
BPMN; BPML	37	9%
Other	9	2%
SA Diagrams	177	42%
DFD	111	26%
ERD	94	22%
STD	62	15%
User Screens	151	36%
Informal Drawings	139	33%
Tables	67	16%
Other	19	5%

Multiple answers were possible, light color: sub-categories, "Other": answers with less than 5% frequency

Fig. 2.2 Common requirements engineering practice

Almost every project specified requirements. A majority specified functionality, quality, use scenarios, and user interfaces of intended solutions. Functional requirements dominated. Concepts commonly used for formal reasoning, such as agents, goals, and formal properties, were rare. For specifying the requirements, natural language dominated as the notation. Natural language was often complemented with UML diagrams. The use of other diagram types, user screens, and informal drawings varied. Formal-logic and goal-oriented languages like i* or KAOS were almost never used.

The frequency of notations that match requirements analysis techniques was inconsistent with requirements analysis practice. Object-oriented and structured diagrams were much more common than the use of corresponding analysis techniques. User screens were much less frequently documented than prototypes created. Formal specification languages were used as rarely as the corresponding formal methodology.

Almost every project stored requirements. Requirements documents were the most common type of storage. The use of spreadsheets, requirements databases, and modeling tools varied. Drawing tools, wikis, and cards for capturing requirements backlogs were uncommon.

Almost every project checked requirements. Projects tended to prefer manual requirements checking, preferably with rigorous inspections. Simulation and automated formal checking were uncommon.

Almost every project negotiated requirements. To reach an agreement on requirements, most common was requirements prioritization. Uncommon were analytical

techniques such as power, variant, and negotiation analysis, and advanced techniques such as win-win negotiations.

Four out of five projects managed the requirements. This means at the same time that one out of five projects did not use the requirements once they were inquired. Requirements management tended to focus on the handling of requirements. Most common was change management. Requirements were rarely used for analyzing project progress, for reporting, or for measuring the development process.

Overall, the large variety of techniques indicates that there was no one-size-fits-all in requirements engineering practice. Still, there are a few practices that could be seen in a large majority of projects. These include the use workshops to discuss requirements with stakeholders, the specification of functional requirements, and the use of natural language for specification. Among the established methodologies, object-oriented analysis and specification appears to be widely adopted, although not dominating. The older counter-parts, for example structured analysis, were much less important in comparison. Extremely rare were formal techniques. Even-though they are widely researched, they have hardly found their way into current practice.

2.4.3 Requirements Engineering Success

The data from the responding projects showed that there was no dominant way of judging requirements engineering success. Figure 2.3 gives an overview.

The most important requirements engineering goals were shared understanding between the project team and its stakeholders and good quality of the requirements specification. These two objectives of requirements engineering were often complemented with the need for a clear scope for the development, for using little time and

Requirements Engineering Goals		
Total	419	100%
Shared Understanding	214	51%
Specification Quality	197	47%
Clear Scope	160	38%
Efficiency	155	37%
User Satisfaction	145	35%
Timeliness	139	33%
Fit of Solution	94	22%
Estimation Reliability	65	16%
Architecture Quality	58	14%
Cost/Benefit Analysis	26	6%
Other	4	1%

Software Product Goals		
Total	419	100%
Productivity	228	54%
Effectiveness	156	37%
Compliance	143	34%
Satisfaction	137	33%
Flexibility	86	21%
Safety	73	17%
Environment	7	2%
Other	8	2%

Fig. 2.3 Success factors for requirements engineering process and outcome

Achievement of RE Goals		
Total	419	100%
Too little	181	43%
Just enough	229	55%
Too much	9	2%

Achievement of Product Goals		
Total	419	100%
Rather Yes	385	92%
Rather No	34	8%

Fig. 2.4 Requirements engineering success

resources for requirements engineering, for satisfying the users, and for delivering the requirements engineering work results in time.

The most common goal pursued by the software products that were specified with the requirements was productivity improvement. This goal was complemented by a variety of goals that included effectiveness, e.g., to enable users to do things they could not do before, compliance with laws and regulations, and satisfaction of users and stakeholders with the product. Important societal topics, like environmental or societal challenges, were rarely considered to be a goal of the software product.

Figure 2.4 shows how many projects were successful and how many have not been successful when judged according to the criteria summarized in Fig. 2.3. A bit more than half of the projects judged that they fulfilled the requirements engineering goals. A bit less than half judged they did too little. Almost none stated they would have done too much. While positive and negative satisfaction with requirements engineering were rather balanced, product goal achievement was a sharp success. About nine out of ten of the specified products were judged to be a success. Only few were considered failures.

These success rates appear to contradict the success rates identified in other studies. In comparison, Standish presented 32 % project success rate in 2009 by taking into consideration scope, time, and budget adherence [5]. The staggering success rate of 92 % for product goal achievement we observed thus says that while the project may have been problematic, the outcome of the project was not. Also, 55 % satisfaction with the requirements engineering experience is significantly larger than then 32 % project success rate. This may indicate that requirements engineering practice had matured and was less problematic than other disciplines.

2.4.4 Success-Correlating Practice

To identify effective requirements engineering practice we correlated technique use with requirements engineering success. The result of this analysis indicates the techniques that are used in projects with successful requirements engineering significantly more often than in projects that did not meet requirements engineering goals or produced products that did not achieve their goals. Whether practice use leads to success or whether good projects select these practices cannot be concluded from these results and needs to be investigated in future research.

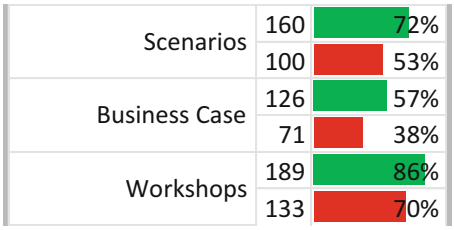


Fig. 2.5 Techniques that correlated with requirements engineering success (*upper rows*: successes, *lower rows*: failures)

Our survey data showed 221 projects with successful requirements engineering and 189 failures according to our success criteria. Only three techniques correlated with requirements engineering success with $p<0.05$ significance after pruning the results with Holm's step-down method to remove false positives. None of the other techniques correlated significantly with success, and no technique correlated negatively. Figure 2.5 gives an overview.

Scenarios are exemplary sequences of system usage [55]. In requirements engineering, they are used to describe concrete stories of how users and external systems interact with the system under consideration to achieve goals that are of value to the user. Scenarios make the functionality of the system concrete and thus enable users to judge whether they feel to be able to use the system meaningfully and whether they like it. Scenarios also allow capturing interaction design knowledge from user experience experts. A common format used to document scenarios is the use case template [78].

Business cases are used to document predicted financial results and other business consequences for one or multiple alternative ways of how a product is built, deployed, and maintained [41]. The business case planning work and results that are obtained with it are determinant for selecting what is in the product scope and what not and for evaluating whether a chosen scope is attractive for the customer of the project. The understanding of a business case allows to stop work on a product that does not make sense or to re-scope the product to make it more attractive.

Workshops create an efficient, controlled, and dynamic setting for quickly eliciting, prioritizing, and agreeing on requirements [29]. The discussion of requirements by the critical stakeholders makes a requirements workshop to be one of the most efficient techniques to perform inquiry and to achieve shared understanding. No other technique allows exposure and resolution of conflicts between stakeholders so efficiently. The same applies for discovery and resolution of misunderstandings.

We also studied whether the number of techniques used and the number of requirement types documented in a project correlates with requirements engineering success. Figure 2.6 shows the distributions with box plots.

According to the Wilcoxon's rank-sum test, successful projects use a significantly larger number of requirements engineering techniques and specify a significantly larger number of requirement types than unsuccessful ones.

Again, the causes for these correlations should be investigated with future research. A hypothesis that should be tested is whether there is a large variety of project context that require more techniques to be used. The observed large variety

http://www.springer.com/978-3-319-09797-8

Requirements Engineering for Digital Health
Fricker, S.A.; Thuemmler, C.; Gavras, A. (Eds.)
2015, VIII, 204 p. 42 illus., 33 illus. in color., Hardcover
ISBN: 978-3-319-09797-8