Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 16 (2013) 796 – 805

Procedia Computer Science

Conference on Systems Engineering Research (CSER'13)
Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.

# A Literature Survey on International Standards for Systems Requirements Engineering

Florian Schneider[a]*, Brian Berenbach[b]

[a]*Chair for Applied Software Engineering, Technische Universität München, Boltzmannstr. 3, Garching, 85748, Germany*
*Siemens Corporation, Corporate Technology, 755 College Road East, Princeton 08540, USA*

**Abstract**

Since 2005, the international organization for standardization (ISO) has published a wealth of standards and reports that deal with requirements engineering for systems. ISO/IEC/IEEE 24765 defines a standard vocabulary for systems and software engineering. ISO/IEC 24766 defines requirements for requirements engineering tools. ISO/IEC/IEEE 29148 describes processes for requirements engineering. The ISO 25000 series targets product quality metrics. This review paper shall provide a high-level description of each of these standards and highlights their interconnection. It thus provides to the systems engineer some guidance as to the relevance of those standards to his or her work.

*Keywords:* systems engineering; requirements engineering; international standards

## 1. Introduction

There are a plethora of standards available for systems engineering, published by the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC), and the Institute of Electrical and Electronics Engineers (IEEE), Many of the standards cross-reference each other, and some are specific to a domain, e.g. software vs. systems. Furthermore, as technology changes, some of the standards become obsolete or outdated. For example, ISO/IEC Technical Report 24766 is a nice guideline for evaluating requirements engineering tools, yet it does not consider support for product lines (e.g. capture of variation points). This paper provides an overview of some of the most common standards that systems engineers have to work with today, including areas where the authors identified potential weaknesses.
The standards described here often contain project execution requirements (i.e. requirements governing how a product is built) in projects with legal authorities, showing up in public requests for proposals. These often end up in contracts, and the systems engineer then must be able to show that the development process and the specifications

* Corresponding author. Tel.: +49 89 289 18233; fax: +49 89 289 18207.
*E-mail address*: florian.schneider@in.tum.de

produced are compliant with the requested standards. So a certain familiarity with these standards is of benefit to the engineer, and more so if the engineer has plans to become a certified requirements engineer (e.g. through the International Council on Systems Engineering (INCOSE) or the International Requirements Engineering Board (IREB)). With ISO, IEC, and IEEE harmonizing their standards and due to the complexity of the domain, it is hard not to get confused, with the various relationships between standards.
We chose standards that define the most important terms of requirements engineering, impose requirements on the engineering process, the tools used therein and the artifacts created with the tools in the process. As they are relatively new and thus might not be known by many, we also added two standards regarding quality requirements. The relationships between the chosen standards are shown in Figure 1.

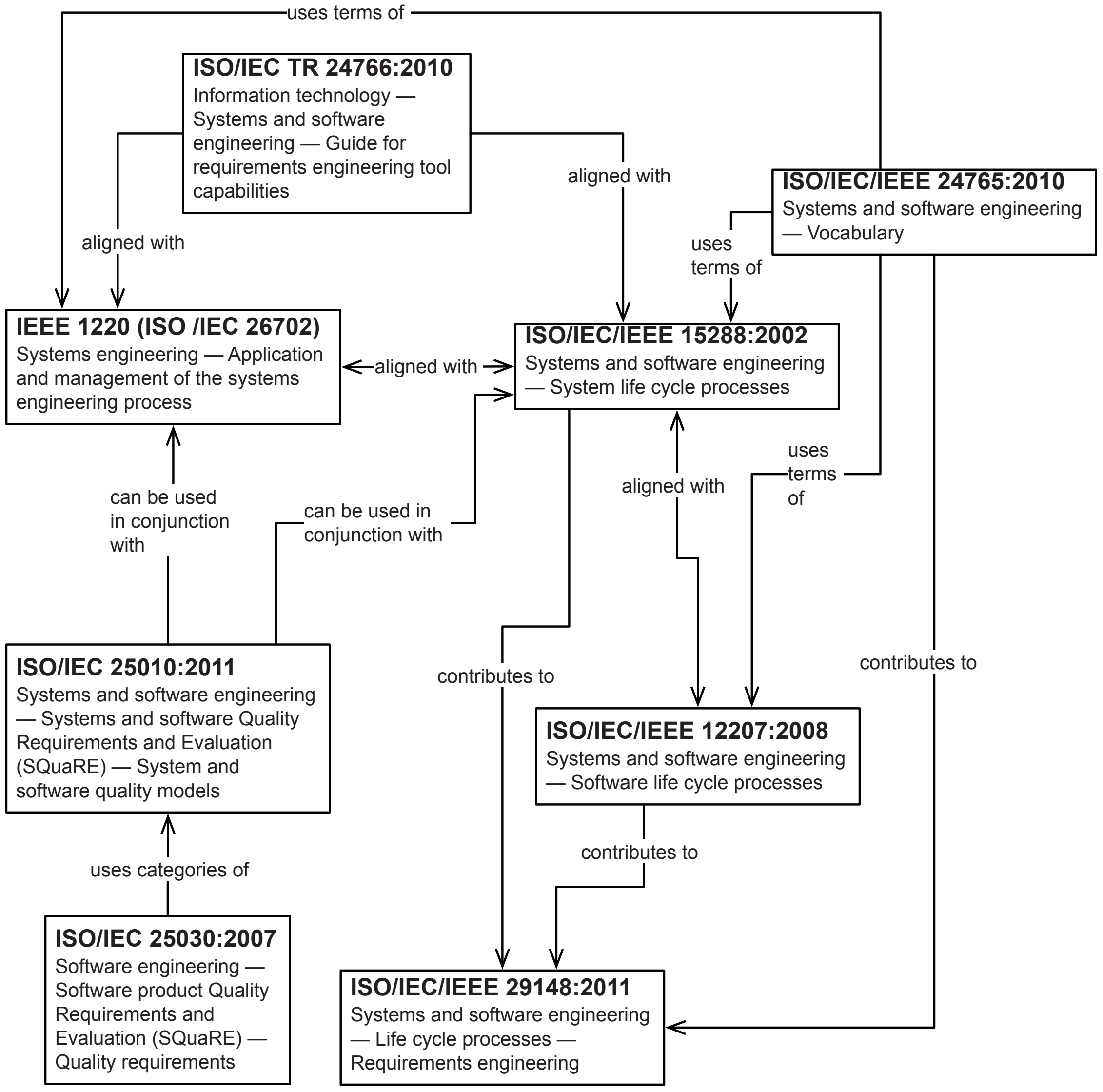
uses terms of
ISO/IEC TR 24766:2010
Information technology — Systems and software engineering — Guide for requirements engineering tool capabilities
aligned with
aligned with
ISO/IEC/IEEE 24765:2010
Systems and software engineering — Vocabulary
uses terms of
IEEE 1220 (ISO /IEC 26702)
Systems engineering — Application and management of the systems engineering process
aligned with
ISO/IEC/IEEE 15288:2002
Systems and software engineering — System life cycle processes
uses terms of
aligned with
can be used in conjunction with
can be used in conjunction with
ISO/IEC 25010:2011
Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models
contributes to
contributes to
ISO/IEC/IEEE 12207:2008
Systems and software engineering — Software life cycle processes
uses categories of
contributes to
ISO/IEC 25030:2007
Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements
ISO/IEC/IEEE 29148:2011
Systems and software engineering — Life cycle processes — Requirements engineering

Figure 1: Relationships of the standards described in this paper

Section 2 provides a description of each of the standards we chose and a critical discussion. Section 3 provides some general remarks regarding the relationships between standards. A critique that is applicable to the selected standards is given in section 4.

## 2. Description of Standards

For each standard we provide a short summary of its content, where possible mirroring the introductory sections of the standard itself. Subjective commentary suggests other points that should be of interest to the systems requirements engineering community. A short section that outlines related standards follows the summary. A critical discussion of the standard from the author's point of view is then given, sometimes as a discussion between the two authors.

### *2.1. ISO/IEC/IEEE 24765:2010 [1]*

This standard provides a *standard vocabulary for systems and software engineering*. Included are definitions of a number of terms that are important to requirements engineering. A restatement and discussion of all such terms would go beyond the scope of the paper[1]. The authors would like however to share their own interpretation of the requirements-related parts. We extracted all the terms with the suffix "requirement" and mapped them to classes or properties of classes. A Unified Modeling Language (UML) class diagram notation was chosen to visualize this. As the standard does not make any taxonomical statements, the figure is merely our interpretation of the standard. The standard does only state which terms are related to each other (e.g. "customer requirement is in the cf. section of "contractual requirement"), but not exactly how. According to Figure 2, requirements are either product requirements or non-technical requirements. Product requirements either are physical, functional, or non-functional. Other requirements terms were mapped to Boolean attributes, where a Boolean attribute can have only a true or false value. For example, the attribute "user" will be either true or false and can be used to search the requirements in a database to find any requirements that were expressed by presumable users of the system. With these attributes, any requirement could be marked whether it was allocated, stated by a customer, stated by a user, part of a contract, or optional. Product requirements can be marked whether they are related to design or implementation, or affect the interface of the system. One term ("derived requirement") was mapped to a relationship: Any requirement can be derived from another.

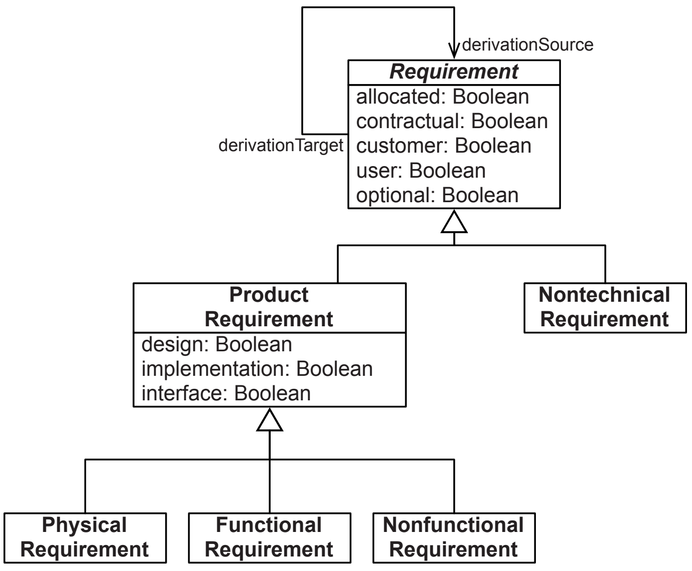

Figure 2: Requirements taxonomy that we derived from ISO/IEC 24765:2010

[1] The vocabulary itself is maintained in a database accessible through www.computer.org/sevocab

#### *2.1.1. Relationships to other standards*

ISO/IEC/IEEE 24765:2010 obsoletes IEEE's 630.12 of 1990, the IEEE Standard Glossary of Software Engineering Terminology. Apart from that standard, several other ISO and IEEE standards contribute to the vocabulary, among them the standards 12207, 15288, and 26702. With the existence of the new vocabulary standard, other standards as the Guide to the Systems Engineering Body of Knowledge (SEBoK) [2] can refer to it.

#### *2.1.2. Critical discussion*

A point of dispute among the authors was whether a physical requirement should be interpreted as its own class of requirements, or whether it should rather be mapped to a Boolean attribute of ProductRequirement. The way it is presented in Figure 2, it only makes sense if physical requirements describe, for example, desired material or size of a product or product part. This way, physical requirements would be clearly separated from functional requirements (that describe desired functions) and non-functional requirements (that describe desired quality of the functions).

### *2.2. ISO/IEC TR 24766:2009 [3]*

ISO/IEC TR 24766 is a technical report that provides a guide for the evaluation of requirements engineering tools. It has a short definition of terms, a brief mention of the processes needed to perform requirements engineering, and a reasonably comprehensive discussion of needed tool capabilities for each process. The report does a nice job of describing the capabilities that the tool must have in order to effectively support each process (e.g. risk analysis, elicitation, verifying contract compliance, etc.) without falling into the trap of describing implementation. Finally, there is a nice summary of requirements quality characteristics in annex A.

#### *2.2.1. Relationships to other standards*

As requirements engineering crosscuts so many areas, there is a bibliography listing related ISO/IEC standards, with an emphasis on SquaRE (Software product Quality Requirements and Evaluation).

#### *2.2.2. Critical discussion*

The technical report is well written and clear. However, there are two omissions, which may or may not impact an organization using the report as a checklist for tool selection. First, there is no discussion of **extensibility.** Most complex engineering tools today do not provide all needed functionality "out of the box" and require custom scripting to meet project needs. Moreover, for tool chain integration, it is usually necessary to "plug" a requirements tool into a tool chain using software code or scripts as the glue between tools to automate the transfer of information, e.g. tracing from requirements tool to design tool to testing tool. Most of the commercial RE tools on the market do provide custom programming or scripting facilities, however some checklist for what might be needed would be helpful. The second omission is that of **product line** support. For those potential users of an RE tool who do not have product lines, this may not be an issue. Many systems engineers, however, need to document variation points in their products in order to support the testing of different configurations that may be deployed in a single product or as part of a product line. For example, a model of a car may be ordered with a manual or automatic transmission. Depending on which transmission a customer orders, certain other components need to be specified. To ensure the proper documentation of variations for manufacturing and testing, a requirements database may need to support product lines or at the least product variations. Unfortunately, 24766 makes no mention of such capabilities.

### *2.3. ISO/IEC 25010:2011 [4]*

ISO 25010 establishes a *quality model for software products and software-intensive systems* that guides the formulation of quality requirements and metrics to measure their satisfaction[4]. This standard is a revision of ISO/IEC 9126-1:2001[5]. The standard defines two quality models, which describe desired quality characteristics of a system. The two models described in this standard are the "quality in use" model and the "product quality" model. The quality in use model consists of five characteristics (effectiveness, efficiency, satisfaction, freedom from risk, and context coverage). It relates to the aspects of the usage of a system by a user, in a particular context. The product quality model consists of eight characteristics (functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability). It relates to static properties of the software product and dynamic properties of the computer system. The quality characteristics may be seen as categories that can be used to characterize quality requirements. Quality requirements describe the desired quality of a system. The effective quality of a system can be measured as follows. First, for every quality characteristic (for which a quality requirement was formulated) a set of properties of the system, that together cover the characteristic in question, has to be identified. For each of the properties a measure has to be computed and a derived quality measure for the combination of these values.

### *2.3.1. Relationships to other standards*

The characteristics defined by this standard form the basis for the formulation of quality requirements (see ISO 25030[6]) and the measurement of these requirements (see ISO 25040[7]).

### *2.3.2. Critical discussion*

As the software product quality might also be affected by the way the project creating the product is organized, one might argue that the standard does not take non-technical requirements as formulated by ISO 24765 into account.

## *2.4. ISO/IEC 25030:2007 [6]*

This standard in effect states *requirements for quality requirements models or specifications*. It does so by formulating requirements for the expression of requirements, requirements regarding the contents of a requirements document, and requirements regarding activities to be performed when creating such a document. The intention as mentioned in the introduction is to "to improve the quality of software quality requirements". While officially targeting only quality requirements, the standard requires a quality requirement specification to include other parts as well: a list of stakeholders, the system boundaries, and the software boundaries. It so adapts the general model of the other ISO 250xx standards: Stakeholders may be interested in complex systems, which may not consist of software alone. So it is important to decide when analysing stakeholder needs and mapping them to quality requirements which requirements affect which part of the system. Implicit to this model is the process of mapping the stakeholder needs to quality requirements. Therefore the standard requires quality requirement specifications to describe all stakeholders of a product with the roles they can play and with the needs they have. Furthermore, the boundaries of the system and the boundaries of the software being part of a system shall be documented. The relationship of stakeholder needs to quality requirements needs to unambiguous, so that it is clear which requirement was derived from which need. It should also be clear which function is constrained by the quality requirement, so there needs to be an unambiguous relationship between a functional requirement and its corresponding quality requirements. As stakeholders often can't tell the requirements engineer about all the needs they have, the standard makes clear that quality requirements may exist that were found without a preceding formulation of a stakeholder need – but should nevertheless be documented. Stakeholders shall approve the quality requirements then, and approval shall be documented. Very important is the definition of the term quality requirement: A requirement either is typed a "quality in use requirement" or a "software product quality requirement". Then it becomes a specification of a quality measure (measuring quality of one of the characteristics of the given category) together with the specification of a target value. It is very important that software quality requirements shall be identifiable (important for traceability) and measurable (important for verification).

### *2.4.1. Relationships to other standards*

The ISO 25030 was among the first of the ISO-9126 follow-ups to be published. Because it was finished earlier than the ISO 25010, it has relics referring to the ISO 9126-1 quality model. It seems like the ISO itself is considering redrawing or revising that standard[1]. So the next version of the ISO 25030 will be more aligned with

[1] See http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=35755 (Last access: October 1st, 2012). The state of the standard upon last access of the webpage was 90.60, which means that the ISO has currently had a vote to decide whether the standard needs revision.

ISO 25010. As an important part of a quality requirement is the quality measure, the standard is tightly coupled to the ISO 2502x standards that give guidance on how to construct and document quality measures.

### *2.4.2. Critical discussion*

The requirements regarding the formulation of quality requirements aren't always specific to quality requirements. Those that are not specific could as well be part of ISO 29148, which also imposes requirements for "good requirements specifications". Second, the standard is very much focused on software product requirements. Though it has a model of systems and software, it does not say anything specific about system requirements, which may cause problems when using it as the basis of a requirements process for system construction.

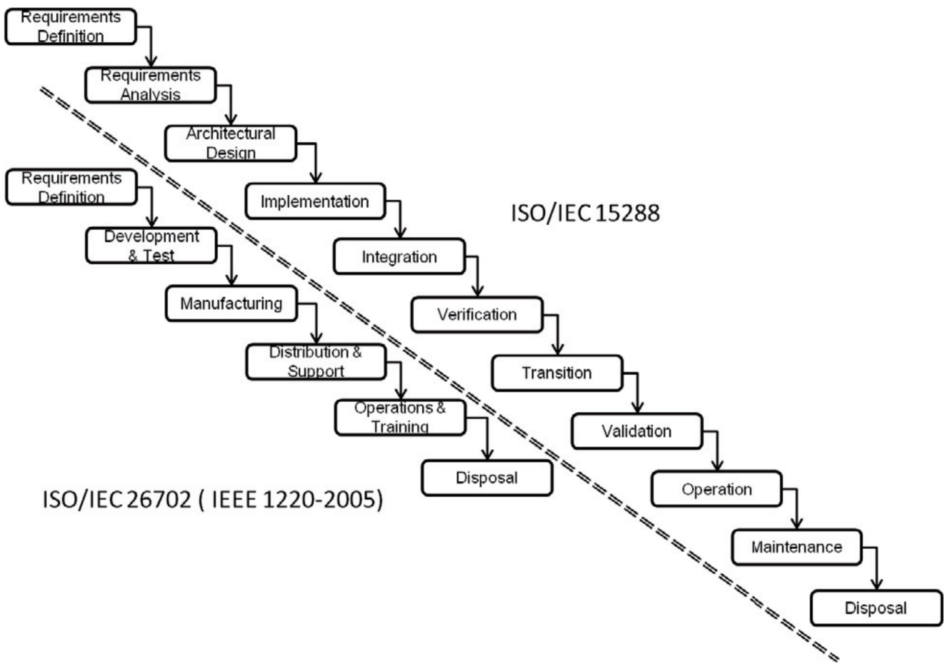

Figure 3: Comparison of 15288 and 26702 Process View

### *ISO/IEC/IEEE 15288 [8] (and IEEE 1220 (ISO/IEC 26702) [9])*

15288 is a foundation standard that focuses on system life cycle processes, as opposed to its sister standard 12207 which focuses on software standards. Both take the approach of defining process requirements, e.g. "The process shall…" It takes a high level view of processes, providing, for the most part, a comprehensive yet terse description of the processes involved in systems engineering. Note that the systems engineering body of knowledge (SEBOK) [2] and the INCOSE Systems Engineering Handbook [10] are closely aligned with 15288. Systems engineers studying for INCOSE certification are encouraged to become familiar with 15288. 15288 describes the system lifecycle in detail, with each necessary or optional step of a process listed, describing what needs to be done but not how it is to be done. While the system breakdown process is iterative, the only types of elements are "system" and "system element", combined recursively. The lack of further decomposition may be somewhat frustrating to an organization looking for more layers of definition.

### *2.5.1. Relationships to other standards*

15288 is closely related to IEEE 1220 (ISO/IEC 26702) as they both cover elements of the same material with a different focus. Annex F lists the relationship of 15288 to other standards in some detail, including IEEE standards 1220, 1228, 1233, 1362 and 1471, and ISO/IEC/IEEE 12207.

### *2.5.2. Critical discussion*

While 15288 holistically covers the entire lifecycle (including acquisition), for those engineers building a product IEEE 1220 may be more informative. 15288, as mentioned above, only has "system" and "system element", hierarchically decomposed with the part under a "system element" being another system. IEEE 1220 provides a finer grain of decomposition, defining System->Product->Subsystems->Assemblies->Components. From that point, there are two alternate decomposition strategies, Components->Subcomponents->Parts, or alternatively, Subassemblies->Subcomponents->Parts. At this point recursive decomposition is possible. Interestingly, while 1220 defers to other standards for information about requirements processes, 15288 devotes several pages to defining requirements processes. So 15288 and 1220 complement each other. The company defining an organizational systems engineering process will need to be clear about which standards the approach (or a hybrid) was derived from. However, as mentioned elsewhere, standards do not provide the "how to", but merely a checklist for those engineers wanting to ensure a comprehensive set of processes.

## *2.6. ISO/IEC 12207 [11]*

12207 is a foundation standard that focuses on software life cycle processes, as opposed to its sister standard 15288 which focuses on system standards. As with 15288, processes are defined that are applicable to the entire project/product (e.g. project metrics, requirements, etc.) and software specific processes (e.g. software requirements analysis). The process sets are defined as tailorable models, with Annex A providing tailoring guidelines. Annex B contains a reference model that can be used to perform process assessments of project or organization software processes.

### *2.6.1. Relationships to other standards*

Since a software product or component may be part of a larger system, Annex D provides some guidance for process alignment, i.e. the alignment of the overall systems processes with the subset used for software development. Annex G provides a comprehensive description of the relationship of 12207 to other related IEEE processes, e.g. 1362 (describes concept of operations).

### *2.6.2. Critical discussion*

12207 is quite comprehensive, and, unlike some earlier standards, provides guidance for the acquisition of software as well as the construction process. Processes are defined using sets of one-line statements for each of the atomic processes. So while the engineer will see a line item "Risk management policies describing the guidelines under which risk management is to be performed shall be defined", there will be no clue how to create the definitions. Furthermore, there are no inline references or footnotes to other documents, standards or procedures that might help in understanding how to define the risk management policies. So 12207 makes for a nice checklist to ensure that required processes are in place, but provides little or no guidance as to "how to" create and execute processes. 12207 is for use by experienced staff setting up or assessing process, but typically would not be recommended as a starting point for the novice.

## *2.7. ISO/IEC/IEEE 29148:2011 [12]*

In short, the standard does not only define the processes of the requirements engineering activity, it also formulates requirements for requirements documentation. For the purpose of this paper, it is especially worth noting that the standard provides guidelines for applying the requirements-related processes of the 12207 and 15288 standards. This standard might be considered to be the mother of all "requirements standards" as it gives a rather extensive description of the domain of requirements engineering. All the other standards can be used in conjunction with this one. The standard starts with defining the important concepts of requirements engineering. If first defines the term requirements engineering itself. Then it talks about stakeholders and that the stakeholders' needs should be transformed into requirements. It tells us what a "well-formed requirement" is, proposes three templates that can be used to formulate textual requirements, and provides characteristics of "good requirements" and "good requirements specifications". Apart from the "dos", it also provides some "don'ts" regarding requirements language (e.g. avoid subjective language). Subsequently, requirements types (including the quality requirement type of the ISO 250xx standards) are proposed. To conclude the concept sections, the relationship between requirements processes and the resulting "requirements information items", especially their scope, is highlighted. The standard then points our attention to the processes of requirements engineering. Here we can see that the two referred to standards 15288 and 12207 are not obsoleted and reformulated, but annotated and detailed. So the 29148 can be seen as an elaboration of the two standards.

### *2.7.1. Relationships to other standards*

The 29148 standard is a "harmonization standard" that results from the harmonization of standards ISO/IEC/IEEE 12207:2008, ISO/IEC/IEEE 15288:2008, ISO/IEC/IEEE 15289:2011, ISO/IEC TR 19759, IEEE Std 830, IEEE Std 1233, IEEE Std 1362, ISO/IEC TR 24748-1, and ISO/IEC/IEEE 24765.

### *2.7.2. Critical discussion*

It seems that the ISO/IEC/IEEE 29148:2011 is actually the standard that every requirements engineer should be familiar with. It is not only a collection of the most basic principles of requirements engineering, it also hints at how high-quality requirements can be achieved. So it is a good starting point for knowledge acquisition as it relates to other standards that detail certain aspects of requirements engineering (e.g. it points to the 250xx series for quality requirements). Finally, requirements engineers do not need to read the standards 15288 and 12207, as all the tasks relevant to requirements engineering are cited in original form and then elaborated. The standard however is a little bit short with the reader regarding categories of requirements. As this still seems to be a point of discussion in the research community, the standard would have benefited by providing stronger statements regarding a basic useful set of requirements categories (e.g. would three categories, namely functional, non-functional, and other, suffice?). The standard weakens this part by only saying what examples of requirements types are ("important examples" nonetheless, but it would be a stronger statement to say "these are the types of requirements that are useful). Last, we are a bit sceptical whether every comment on the requirements of 15288 and 12207 actually helps in addressing these requirements. This might be something only experience can show. E.g. the verification of requirements can be accomplished by traceability, but research shows that traceability still is a tedious thing to achieve. So the 29148 does not fully address the criticism we discussed regarding the 15288 and 12207 standards.

## 3. Relationships between standards

Some relationships are historical, stating which standard was revised by which standard. Other relationships express a shared vocabulary. Then adherence to one standard can be required in a process described by another. As all standards we found heavily rely on text, we would like to encourage the use of graphics more heavily in standards. Especially the visualization of the shared vocabulary (e.g. as in Figure 4) or historical aspects (e.g. as in Figure 5) could be helpful. Figure 4 shows, for the 29148 standard, which other standards contribute terms (solid line arrow) and from which standards terms were adapted (dotted line arrow). A book icon [13] was used to clarify

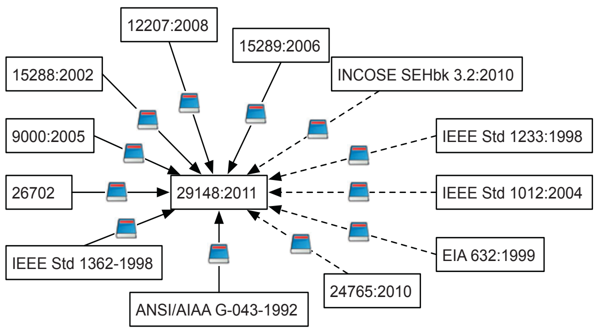

Figure 4: How other standards contribute to the vocabulary of the 29148 standard

the meaning of the arrows. Figure 5 visualizes the history of the ISO quality requirements standards. Here, a clock icon [14] was used to clarify the meaning of the arrows. With both icons, combined diagrams showing history and contribution would be possible.

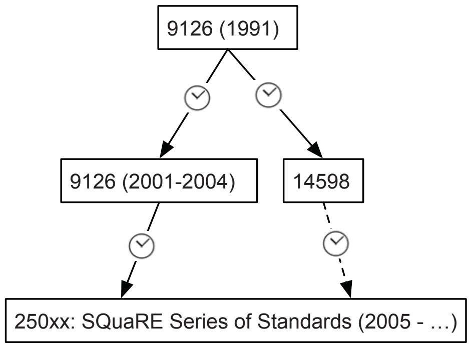

Figure 5: Historical evolution of ISO's quality requirements statement

## 4. Challenges

Some standards have a long history and have been revised under the same standard number. Confusion might arise when speaking about ISO 9126 one time and ISO 9126:2001 or ISO 9126:1991 the other time. Are they both ISO 9126? It is always safer referring to a standard by including the date of its publication. The second thing we observed is that it is hard to talk about multipart standards. ISO 9126 per se does not exist in reality. It is rather shorthand for referring to four documents (ISO 9126-1:2001, ISO TR 9126-2:2003, ISO TR 9126-3:2003, and ISO TR 9126-4:2004). Not all parts of a standards family are actually standards. Some of them are technical reports. We do not know whether all of them have a "TR" in their name. Many standards are joint standards. So for a correct name, all standard bodies should be acknowledged when talking about a standard. It is difficult to speak about ISO/IEC/IEEE 24765 instead of ISO 24765.

## 5. General Discussion

While we discussed each of the standards standalone, we want to make some general remarks that cover multiple standards. The ISO 25000 series of standards in general can easily be criticized for not fully addressing the systems engineering domain. We would suggest in any event that any taxonomy of quality requirements for systems engineering should take the categories of ISO 25010 into account. Furthermore, many of these standards should be easily transferable to the systems engineering domain. It would be interesting to investigate whether the ISO is actually taking steps toward such an effort. A general critique of most of the standards presented here involves the proper use of citations. While some standards do a good job at least regarding referring to related standards, some others don't. What all have in common that rarely works external to the "standards domain" are cited. No journal or conference paper would get by with this. For standards, it seems to be acceptable. It is not always clear whether a standard supersedes another. While the standards organizations certainly have a well-defined process and vocabulary, the formulations in the introductory standards paragraphs are sometimes quite irritating to the novice reader. When standard A revises standard B, is B then obsolete? When standard A is a result of harmonization of standards B and C, are B and C then obsolete?

## 6. Using the standards

The standards have been used at Siemens as guides on internal product development projects, and, in some cases, as regulatory requirements on contract-based projects. It was found that in every case where they were used, application of the standards had to be done carefully. For example, IEC 26702 was used on rail and medical projects. The work product decomposition, e.g. system->product->subsystem needed to be tailored (e.g. eliminate assembly) and the defined processes were not an exact fit. Moreover, on contract based work, the processes needed were markedly different than those in the standard, as the standards had not taken client-supplier interactions into consideration. We recommend that where possible, the standards be used as starting points, and tailored based on need and project type and size.

## 7. Conclusion

In this review paper, we have described seven international standards and one tech report (sometimes jointly) published by ISO, IEC, and IEEE. Apart from the summary, we shared our personal views regarding these standards. Our goal is to initiate a discussion of the standards with the systems engineering community, with the objective of seeing them improved. Our review of the standards discussed in this paper is by no means complete, and might be outdated soon after publication. Perhaps in the future a follow on to this paper will include all the parts of the ISO 25000 ("SQuaRE") Series. Last but not least, we have shown diagrams that would enhance readability of the standards. A relationship of one standard to the other could easily be found in a diagram. Scanning the whole vocabulary chapter takes much more time. We also hope to get feedback from the standards community. Finally, we apologize in advance to the authors of standards who work very hard to create them for any misperceptions or misrepresentations. Our objective is to initiate a dialog that might eventually lead to a coherent, more easily comprehended set of interlocking standards.

## References

[1] International Organization for Standardization, "ISO/IEC/IEEE 24765:2010 - Systems and software engineering -- Vocabulary," ISO/IEC/IEEE, 24765, Dec. 2010.

[2] A. Pyster, D. H. Olwell, N. Hutchison, S. Enck, J. F. Anthony Jr, D. Henry, and A. Squires, Eds., *Guide to the Systems Engineering Body of Knowledge (SEBoK)*. [Online]. Available: http://www.sebokwiki.org/1.0/index.php/Main_Page. [Accessed: Oct.-2012].

[3] International Organization for Standardization, "ISO/IEC TR 24766 - Information technology — Systems and software engineering — Guide for requirements engineering tool capabilities," ISO/IEC, Dec. 2009.

[4] International Organization for Standardization, "ISO/IEC 25010 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models," ISO/IEC, Mar. 2011.

[5] International Organization for Standardization, "ISO/IEC 9126-1 - Software engineering -- Product quality -- Part 1: Quality model," International Organization for Standardization, 9126, 2001.

[6] International Organization for Standardization, "ISO/IEC 25030 - Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality requirements," ISO/IEC, Jun. 2007.

[7] International Organization for Standardization, "ISO/IEC 25040 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process," ISO/IEC, Feb. 2011.

[8] International Organization for Standardization, "ISO/IEC 15288:2008 - Systems and software engineering -- System life cycle processes," ISO/IEC, Mar. 2008.

[9] International Organization for Standardization, "ISO/IEC 26702 IEEE Std 1220-2005 - Systems engineering — Application and management of the systems engineering process," ISO/IEC/IEEE, Jul. 2007.

[10] SE Handbook Working Group, *Systems Engineering Handbook*. International Council on Systems Engineering (INCOSE), 2011, pp. 1–384.

[11] International Organization for Standardization, "ISO/IEC/IEEE 12207:2008 - Systems and software engineering -- Software life cycle processes," ISO/IEC, Mar. 2008.

[12] International Organization for Standardization, "ISO/IEC/IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering," ISO/IEC/IEEE, Nov. 2011.

[13] "GNOME Dictionary Icon." [Online]. Available: http://commons.wikimedia.org/wiki/File:Gnome-dictionary.svg. [Accessed: 04-Oct.-2012].

[14] "Simple Clock Icon." [Online]. Available: http://commons.wikimedia.org/wiki/File:Clock_simple.svg. [Accessed: 04-Oct.-2012].