# The Missing Link - between Requirements and Design

Armin Haße[1], Cees Michielsen[2]

[1]Siemens Industry Software GmbH & Co. KG, Stuttgart, Germany
armin.hasse@siemens.com
[2]R&D Studio b.v.,Maarheeze, The Netherlands
cees@rnd-studio.com

**Abstract** The combination of complex system design and management today requires a profound understanding of the relationships between the requirements and design processes throughout the product life cycle. To practice Systems Engineering merely as a technical matter is a serious misjudgment. Engineering is an interdisciplinary approach for all those involved in the Product Creation Process (PCP). The systematic design and creation of the product, based on intensive interaction between stakeholders, systems and subsystems, is performed on more than only at, what is usually called, the technical level of the PCP. This awareness inspired the authors to describe "Product Abstraction Levels" and align it with for instance the V-Model. It is shown how the information transformation path, starting from the customer, through marketing, engineering to purchasing and back to sales can be paved systematically without losing product-related information. It is shown how a common understanding about the product to be developed can be kept complete and consistent, despite a highly fragmented engineering activities and increasing parallelization of PCP sub-processes. The method how to achieve this, is shown in the way in which requirements are in actually interconnected. At the same time it becomes clear that most of today's existing requirements management tools are not able to support integration with the design process. "The Missing Link" refers to the world behind the various relations between bits of information and also what kind of support we really need for the development of complex systems.

**Keywords** Requirements Management, Product Lifecycle Management, SITIO, Product Abstraction Levels, Design, Design Decisions

## Visualization of relations between requirements

Despite new insights, a persistent image in many books, guidelines, user manuals and tutorials on the subject of requirements management is the *Requirements tree*. This article identifies a number of shortcomings of this type of visualization and suggests improvements. Especially when we look to the overall picture (which we will call the *Product Abstraction Level* landscape later, Fig. 8) it becomes obvious that crucial parts of knowledge are missing in the tree representation. As we go through the product abstraction levels and through the process phases in the Product Creation Process (PCP) we make it clear that the tree representation is not capable of visualizing the actual relationships between the necessary information elements, and in a way forces the users of the tree representation to model their product in an unnatural and often illogical manner.

To clarify the problem, we will use a simple example from the automotive practice: A high-level statement by management is the start of product engineering. As long as just this single statement or requirement is in focus, or even when this would be the only requirement, little seems to go wrong. Only when a second requirement is added and both need to be considered to reflect project priorities, the trouble begins. As we know, even at the highest product abstraction levels (business level, or operational use level with people trying to capture the customer needs) many requirements are identified and specified that need to be considered in conjunction with one another. That's where the problem starts.

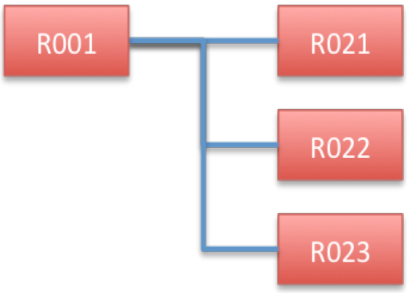


**Fig. 1** A simple requirements tree - How to interpret the relationships between the requirements as depicted in the Figure? Is it a parent-child-like relation or more an Abstract versus Content of an article? Does the picture (syntax) help the viewer to understand what the meaning (semantics) is?

In the following example it becomes clear that understanding the meaning of relations between requirements needs more than what is suggested in Figure 1.
Suppose requirement R001 is a high-level need from top-management of a car manufacturer, saying: "*We need a better car than our previous model X*". This is an unclear, not verifiable statement. There are several ways to interpret the relationship with requirement R001.

**Example A**: "*By 'better' we mean an improved fuel consumption improvement (R021), improved driving performance (R022) and less emission of NOx (R023)*." In this way (a bit) more detail expresses what is meant by R001. One could also say for this example A that *R001 = R021 + R022 + R023*. The requirements are at the same abstraction level, where R001 can be seen as an *umbrella statement*, summarizing the other three.

**Example B**: "*The powertrain shall be optimized for fuel consumption (R021), the stability and brake performance shall be improved (R022), and the exhaust treatment system shall be improved (R023)*". In this case requirements R021, R022 and R023 are derived from a design- and decision-making process and are allocated to systems at a lower abstraction level. Requirement R001 has led to the other three requirements: *R001 → R021, R022, R023*.

In both examples it is not possible to determine if requirements R021, R022 and R023 represent a complete set of requirements to satisfy R001. In example B the design and the following decision-making process is not visible, which in practice will lead to problems when requirements must be met under specific constraints. These constraints are often resource-related: financial, time, personnel, material, capacity et cetera. Typically, these constraints lead to different options during the design process, and therefor also lead to different decisions.
Suppose we add requirement R002 to our structure: "*The sales price of the new car shall not exceed € 30.000 for the European market*."

For example A this requirement could be an addition to R021, R022 and R023 as shown in Figure 2. In that case the set of product requirements is $P_{Reqs} = \{R021, R022, R023, R002\}$. For example B, requirement R002 is at the same abstraction level as R001. Both are requirements that need to be satisfied by the complete product.

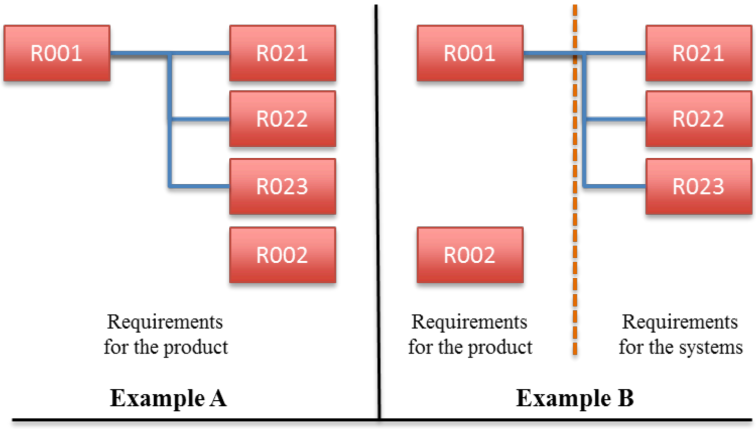

R001
R021
R022
R023
R002
Requirements for the product
Example A
R001
R021
R022
R023
R002
Requirements for the product
Requirements for the systems
Example B

**Fig. 2** a new high-level requirement R002 added**.**

Example B in Figure 2 shows the positioning of R002 at the same level as R001. No relations have been made, i.e. no requirements for the lower levels have been derived. In case R021 (concerning the fuel consumption) would not only have been derived from R001 but also from R002 then this could visually be shown by linking the two requirements. The content of R021 would be changed to reflect this relation

“*The powertrain shall be optimized for fuel consumption, whereby the BOM cost for the powertrain shall not exceed € 2.500*“. And although this relation seen from R021 seems sufficient, it remains unclear how the BOM cost for the powertrain was derived (this information is not stored as part of R002). In addition, how R002 is managed across the systems remains fully unclear.

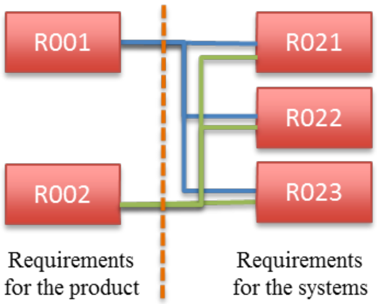


**Fig. 3** integrating requirement R002 - In case the cost for each system has been budgeted in coherence with the other product requirements (R001), this relation can be depicted as in the figure. In this case requirements R021, R022 and R023 are rewritten to reflect the consequences for each of the systems (powertrain, stability & brake and exhaust system).

System requirements R021, R022 and R023 are all linked to both product requirements R001 and R002. And although this would be more in line with what has been decided for both product requirements, the actual trade-off is not visible. Systems requirements cannot be justified purely based on their relation with product requirement without an explicit rationale about the available design options, their pro’s and cons (trade-off) and the final decision on which the derived requirements are based.

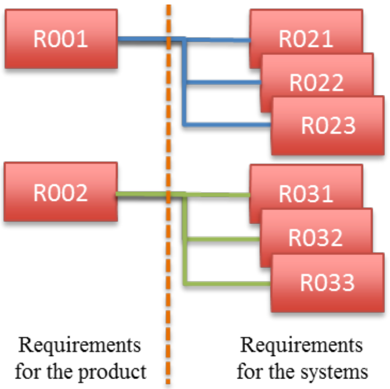


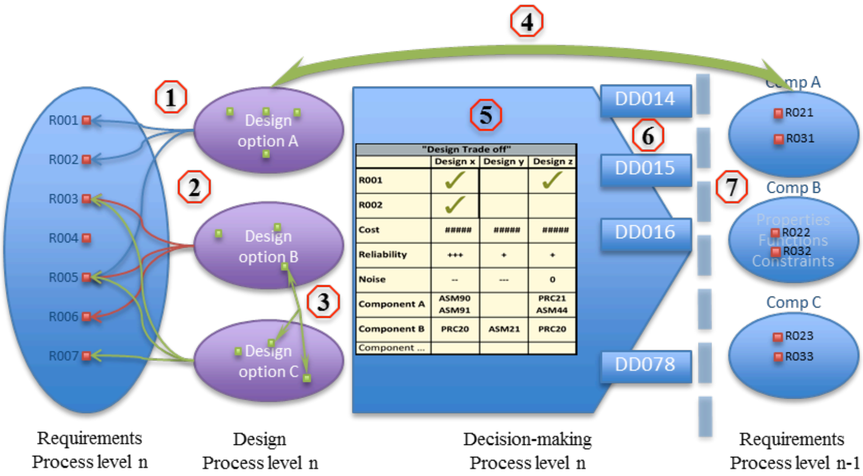
**Fig. 4** two independent requirements trees - In case R002 has not been integrally analyzed and budgeted in coherence with R001, then the cost-related requirements for the systems have no apparent relation with the existing requirements for the “better car”.

The structure in Figure 4 suggests that requirements R001 and R002 are independent. Summarizing we conclude that relationship diagrams or requirements trees that express relations between requirements only,

- Fail to show the difference between requirements that specify a customer need in more detail at the same abstraction level (Figure 1);
- Fail to visualize the multi-objective design and decision-making process results;
- Fail to justify the values in the derived requirements (they are merely capable of showing the result of an untraced decision);
- Fail to help the assessment of completeness for the derivation of requirements (the information of requirements R002, R031, R032, R033 together do not give a decisive answer);
- Fail to support the design and decision-making process at every abstraction level (even when it is made clear that two or more product requirements have been used to build a trade-off and to decide hereupon e.g. in Figure 3, no reference is made to these designs or decisions).

## Visualization of the missing link between requirements

When we look at the development of requirements throughout the lifecycle of a product, we see that requirements are fed to a design process (1). Which requirements are addressed by a design is often implicit, i.e. not or poorly documented. During the design process one or more possible solutions/implementations for one or more requirements are identified (2). For each design option several assumptions or preconditions will be made about the systems/components that are part of the design option (3). During the design process the assumptions and preconditions are verified for feasibility with each system/component responsible (4).



| "Design Trade off" | | | |
|---|---|---|---|
| | Design x | Design y | Design z |
| R001 | ✓ | | ✓ |
| R002 | ✓ | | |
| Cost | ##### | ##### | ##### |
| Reliability | +++ | + | + |
| Noise | -- | --- | 0 |
| Component A | ASM90<br>ASM91 | | PRC21<br>ASM44 |
| Component B | PRC20 | ASM21 | PRC20 |
| Component ... | | | |

**Fig. 5** the requirements derivation steps**.**

A trade-off table is made to enable the decision-making process (5). This table not only shows the relationships of the design options with the affected requirements, but also the required resources (financial, time, personnel, side-effects, …) and the assumptions and preconditions for the systems and/or components that are part of a design option. Once the decision has been made for a particular design option (6), the requirements for the next abstraction level (i.e. systems/components) can be derived from the assumptions and preconditions (7).

These process steps are similar for each product abstraction level (Figure10). It must be said however, that at each level different techniques and tooling can and should be used to accommodate the different types of stakeholders and their communication needs. An important aspect is to identify, specify and maintain the relationships that are made when decisions have been taken. The Design Decision (DD) objects are signposts that point to design options (*references*) and to requirements that are addressed by these design options (*trace links*). In turn, when requirements at lower levels are derived these requirements will point to the Design Decisions.

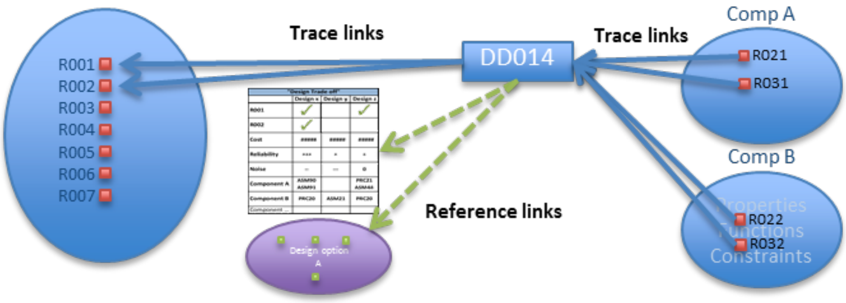


**Fig. 6** the Design Decisions (DD) as signpost

The structure of the DDs themselves is simple: apart from the unique ID and a concise title, the DD points to one or more parent or upstream requirements using *trace links* and also points to zero or more design options, trade-off matrices, reports, and more using *reference links*. When we use the term traceability in the domain of requirements engineering we mean the paths that are created by the trace links. Trace links symbolize the path to *where the requirement originates*, is therefore unidirectional and always point upstream to its parent(s). In our requirements lifecycle model requirements are always linked upstream to one or more design decisions (unless it is the start of a trace also called a *demand*. In figure 6 requirements R001 to R007 are *demands* and have no parents to trace to).

Design Decisions can point to zero or more upstream requirements. There is one important restriction: an upstream requirement can be linked to by no more than **one** design decision. The reason is, that this DD represents the complete fulfillment for the requirement. When a requirement would have more than one DD pointing to it, it would be unclear which one fulfills what part of the requirement. This would bring us back to the problem statement at the beginning of this article.

In case of our example the structure could look like:

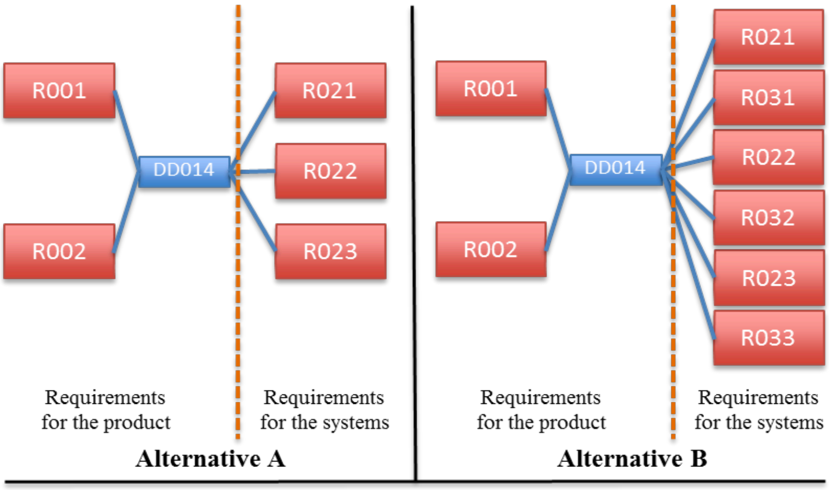

R001
R002
DD014
R021
R022
R023
Requirements for the product
Requirements for the systems
Alternative A
R001
R002
DD014
R021
R031
R022
R032
R023
R033
Requirements for the product
Requirements for the systems
Alternative B

**Fig. 7** Example B with alternative Design Decision

For both alternatives Design Decision DD014 holds the trace links to requirements R001 and R002, and all the reference links (as shown in Figure 6) to design options, trade-off tables, and (if not stated in the descriptive text of the Design Decision itself) to actual decision statements.

The difference between the two alternatives lies at the next abstraction level, where the systems/components either start with a requirement that is derived based on a design decision and that has been rewritten to reflect the priorities in the one requirement (e.g. R021: "*The powertrain shall be optimized for fuel consumption, whereby the BOM cost for the powertrain shall not exceed € 2.500.*").

Or, the systems/components start with two separate requirements (e.g. R021 "*The powertrain shall be optimized for fuel consumption*" and R031 "*the BOM cost for the powertrain shall not exceed € 2.500.*").

Often the allocation of requirements to lower levels is documented as tables. Initially, these tables show the product requirements in the first column, followed by the requirements allocated to the systems and components in the next columns.

| **Incomplete Req table** | *Systems/Components* | | |
|---|---|---|---|
| *Req's (as in Figure 3)* | S1 | S2 | S3 |
| R001 | R021 | R022 | R023 |
| R002 | | | |

| **Incomplete Req table** | *Systems/Components* | | |
|---|---|---|---|
| *Req's (as in Figure 4)* | S1 | S2 | S3 |
| R001 | R021 | R022 | R023 |
| R002 | R031 | R032 | R033 |

In table format the Design Decisions clearly indicate which product requirements are covered, and also which lower level requirements are derived from it. In this way the DDs can be used to measure the requirements coverage: which requirements are covered by a design? The DDs are also used to verify if the set of derived requirements is complete represent all assumptions and preconditions in the design.

| **Alternative A** | | *Systems/Components* | | |
|---|---|---|---|---|
| *Req's* | *Design Decisions* | S1 | S2 | S3 |
| R001 | DD014 | R021 | R022 | R023 |
| R002 | | | | |

| **Alternative B** | | *Systems/Components* | | |
|---|---|---|---|---|
| *Req's* | *Design Decisions* | S1 | S2 | S3 |
| R001 | DD014 | R021, R031 | R022, R032 | R023, R033 |
| R002 | | | | |

The tables above represent the same information as in Figure 7. In addition, the last three columns also show the Systems/Components to which the requirements are allocated.

## Product Abstraction Levels

Examples from industry show combinations of a V-model or pyramids with abstraction levels in it. The basis for creating levels often is a common goal to manage and control information that has a logical coherence. Examples of these goals are: to manage risks [2] or to manage the product requirements transition [3]. In practice not only the information is structured at specific levels, also the roles, responsibilities and processes that are relevant to manage the goals are part of the level definition. In this section the product requirements transition is the central theme for defining product abstraction levels (PALs).

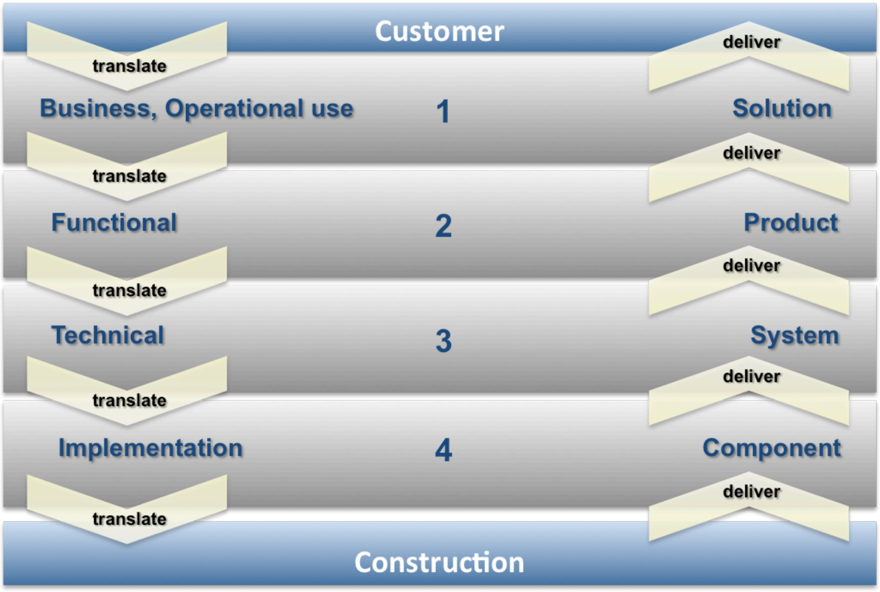


**Fig. 8** Product Abstraction Levels: Operational use level (1), Functional level (2), Technical level (3), Implementation level (4)

A typical enterprise objective is to provide products (goods or services) to sell to customers. In turn, customers often need bespoke solutions to meet their needs. At the *operational use level* (business or solution level) the business analysts capture these customer needs as the basis for finding a solution for them.

The first translation is made at the business level where the customer needs are placed in context with the operational use of the product, including the legal, cultural, environmental and political aspects. The results of the analyses are the *Stakeholder Requirements*.

The solution (i.e. the fulfillment of the Stakeholder Requirements) can be made either from available products (i.e. the cars in the showroom) of from products to be developed. The search for a solution at the business level can also be seen as a design process: in what way can we meet the stakeholder requirements in the most effective and efficient manner?

When the solution is not available (yet) at the business level, the requirements must be specified for the product to be developed. At the business level the *product* is seen as a black box defined by its functions and properties and limited by the product's constraints. The translation at the business level can be described as a process where the stakeholder requirements are translated into functional and quality requirements for the product to be developed:

- What are the required properties of the product?

  - *Drivability, comfort, reliability, ergonomics, styling, safety, security, …*
- What are the required functions of the product, and how well should these functions perform under which conditions?
  - *Central door locking, park assistance, cruise control, …*
- What are the limitations or constraints for the product design?
  - *Total Bill of Material cost, external interfaces, design envelope, …*

When a complete translation is made for all Stakeholder requirements into Product requirements, and the (relevant) stakeholders agree to this translation, the product development can go to the next phase and can start designing the product. This is one of the crucial decisions made within product development and is symbolic for the responsibility at the business level. It means that the people responsible at business level for ensuring that the stakeholder requirements are understood and met as well as the people responsible for *running the business*, and the people responsible for developing the product, have a common understanding (represented by the functional product requirements) about the outcome of the development process.

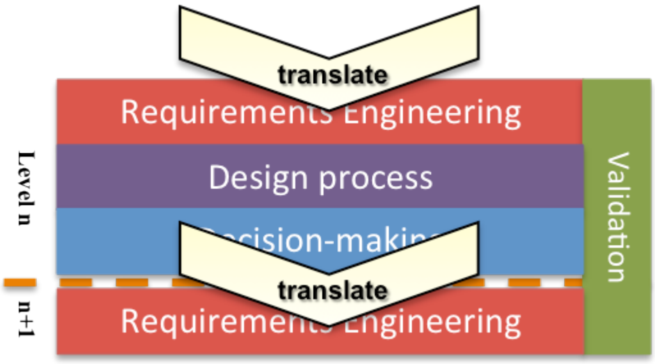


**Fig. 9** Information translations in between the product abstraction levels. The Requirements Engineering process is at the receiving end of the translation. In each level the same three processes are involved in the translation process.

The idea behind the PALs model is to acknowledge the need on each level to express the requirements and design process in such a way that the goals of the level are achieved. The number of levels is determined by the necessity of each level to find a solution outside its own scope. In case we were to determine the number of different levels for a car shop that buys and sells cars, one level would suffice. In the model we use in this article our reference is a car manufacturer; In this case the number of levels is 4. The nature and content of each level differs. You will find a lot of decompositions at the Technical level, right up to the disciplines (mechanics, electronics, software, …) that generate the requirements specifications for the Implementation level.

From a Model-Based Systems Engineering perspective often only the technical (3) and implementation levels (4) are documented and supported by methods and tools. We have experienced that it is worth the trouble to extend the model with the business (1) where the customer solution is delivered and with the functional level (2) as an intermediate between the business and technical level to describe the required product properties and functions.

The *Operational use* level is focused on

- Capturing the customer needs;
- Analyzing the operational use terms and conditions;
- Determining the product roadmaps;
- Determining the product's scope;
- Determining the product development process conditions;
- Finding solutions;
- Determining the Functional architecture, including the interaction and dependencies between the properties and functions;
- Transforming customer demands into a deliverable solution.
- Deliver the solution to the customer;

Normally, the creation of solutions follows a business interest to sell available products that fulfill the solution requirements and constraints (sufficiently for the customer). Sometimes the constraints are more decisive for customer satisfaction than the product requirements ("*I need a new car this week!*"), in which case the customer buys a car from the show room that comes closest to his needs, rather than waiting for the product to be developed.

The *Functional* level functional level focuses on

- Capturing the product's Properties, Functions and Constraints;
- Simulating and validating the overall behavior and feasibility;
- Finding integral solutions (design options);
- Determining the Systems architecture, including the interfaces between the systems;
- Develop trade-offs in which the design options are compared to each other;
- Transforming functional, quality and limiting product requirements into requirements for the specific systems that play a role in the solution, where assumptions made in the product's design are allocated to these systems and requirements are derived from it (see Fig. 5);
- Agree on acceptance criteria and process with people responsible for the systems or subsystems at the Technical level;
- Integrate the systems into functionally complete product;
- Test and release the product, including the product specification (documented results of the tests and measurements to describe the precise behavior and properties of the product);

Just as all the other levels, the functional level defines the **complete** product in its own way. All functional, quality and limiting requirements must be allocated to specific systems at the Technical level.

The *Technical* level focuses on:

- Capturing the Property, Function and Limiting requirements for each system (each system must be fully specified in relation to the overall product;
- Decomposition of each system, including the intra-system interfaces;
- Design for each system;
- Deriving requirements a result of the systems and subsystems design process to the various disciplines that play a role in the development of the product;
- Developing discipline-based architectures and designs;
- Specifying the requirements for the components or elements at the Implementation level;
- Agree on acceptance criteria and process with people responsible for the components/elements at the Implementation level;
- Integrate the components into functionally complete systems;
- Test and release each system, including the system specification (documented results of the tests and measurements of each system to describe the precise behavior and properties of each system);

The *Implementation* level focuses on

- Using the requirements and detailed design to construct, make, manufacture, or program the components/elements.
- Perform unit tests;
- Release the components (including a component specification describing the precise behavior and properties of the component as found during test and measurement);

It is important to understand that the levels described above represent the product dimension by means of a specification structure in which the dependencies and decompositions of all the product elements are shown in relation to each other.

When we look at the process dimension, we see all the processes that are requires for the transformation in the product dimension, are performed in parallel to each other. Typical milestones and toll-gates, phase descriptions are part of the process dimension and are not discussed in this article. Besides the process dimension we also see the need for the people dimension to complete the whole picture: *who* does *what*, *why*, *when*, and for *how much*. The people dimension is crucial to have good quality transformation of information between the levels. Using the SITIO [1] method, a high level of confidence can be achieved about the conformance to expectation, the fitness-for-purpose and the intrinsic quality of requirements.

## A Product Abstraction Level in a Nutshell

For each level two major *inputs* are known: (**$I_1$**) needs/decisions/trends of the level above and (**$I_2$**) known properties & functions/deliveries/outputs of the level below. For each level two major *outputs* are known: (**$O_1$**) requests & assumptions/decisions and (**$O_2$**) deliveries. This means that within the scope of each level, processes are used to produce the outputs, based on both inputs for that level. It also means that the responsibilities can be directly coupled to the roles within these processes. As an example, the figure below shows 7 major processes that either support the information transition downstream or support the solution delivery upstream.

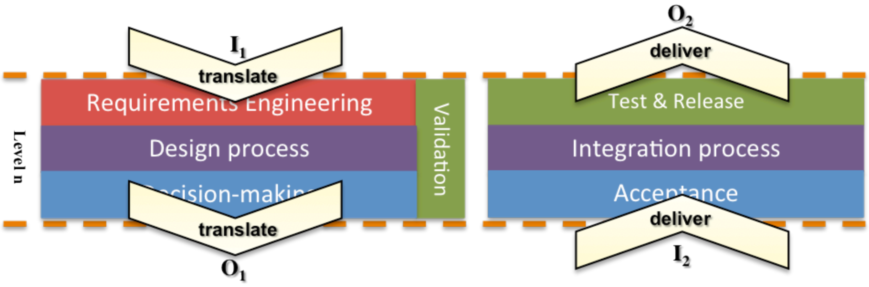


**Fig. 9** the seven main processes and inputs and outputs of each level

The four levels describe four basic views of the product. These views are in essentially determined by the active roles at the levels (people) and their objectives. Each level in turn can operate independently, providing results (outputs, $O_x$) and processing requirements (inputs, $I_x$). The arrows *translate* and *deliver* symbolize the direction of the information/delivery and also the source of this information/delivery.

In practice these two movements are a common process of continuously interaction of roles at these levels. It also indicates that the quality of the translation results depends on the persons that have specific responsibilities at these two levels. E.g. the customer can raise the quality of the requirements by being more specific about his/her needs, and at the same time the marketing representative who is responsible for gathering the product requirements can raise the quality by applying the skills to gather, analyze, specify and validate the requirements. For more see SITIO (Secure Information Transformation from Input to Output) [1] methodology.

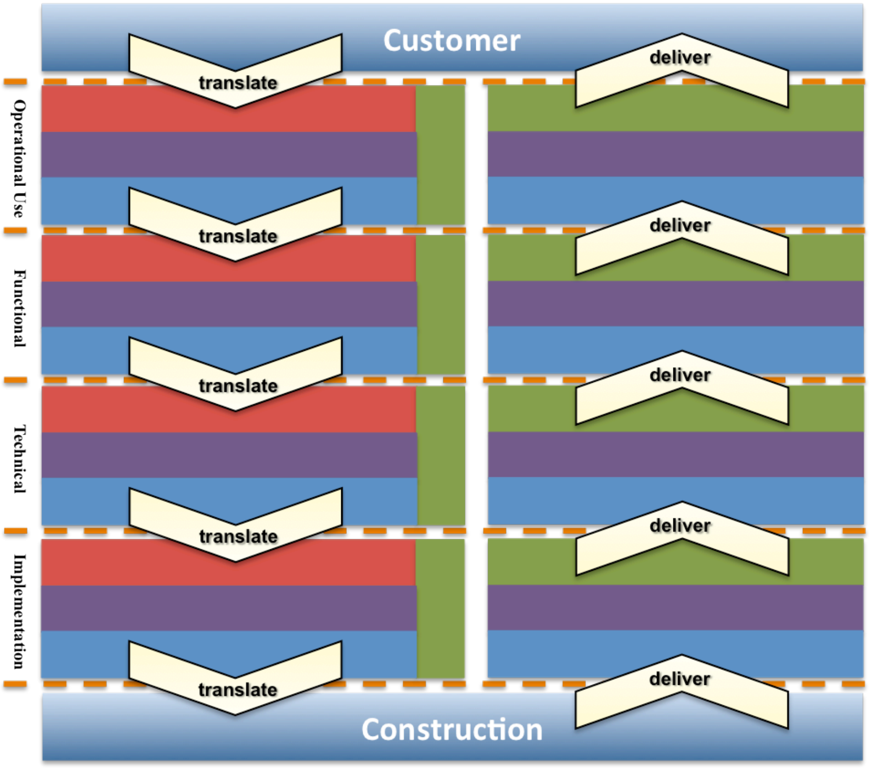


**Fig. 10** The seven major processes are found at each level.

Before going into more detail about the processes in the abstraction levels it is important to understand the function of translate and deliver.

**Translate** – at the *Business level* the incoming *Translate process* ($I_1$) is often triggered by customer needs or demands. Characterized by pushing (customer) and pulling (marketing) activities. The translate process itself encompasses all the steps and means to communicate these demands to the requirements responsible (Business Analyst or Requirements Manager roles typically in marketing departments). The communication means differ at every level. Whilst e-mail or meeting minutes can be used to transmit the demands between customers and the business, other means (e.g. user stories or use cases) are more appropriate at lower abstraction levels, e.g. at the Business level for the outgoing translate process ($O_1$).

**Deliver** – at the Solution level the incoming *Deliver process* ($I_2$) delivers the already produced/available/released products. These products will then combined and configured (assembled) to the desired individual costumer solution. This is often done by agents (e.g. shop assistant or sales persons). Then the individual solution will delivered to the customer via the outgoing delivery process ($O_2$). A typical example could be a car dealer. He has usually not only the product “Car” available, but he also has banking products (like lease or loan agreements) as well as various service options/products. The combination of these products allows him to respond to individual customer needs and assemble a customized solution.

**About the processes** – at each level the design process is fed with the decisions and the derived requirements from the level above. In addition, the design also has knowledge of the deliveries done in the past by the level below. Design options are created that fulfill the requirements. Design options are validated, weighed and decided upon. Requirements for the next level are derived from these decisions. This sequence is repeated until the implementation is completed. Then the integration and test processes start to go upwards in the product dimension.

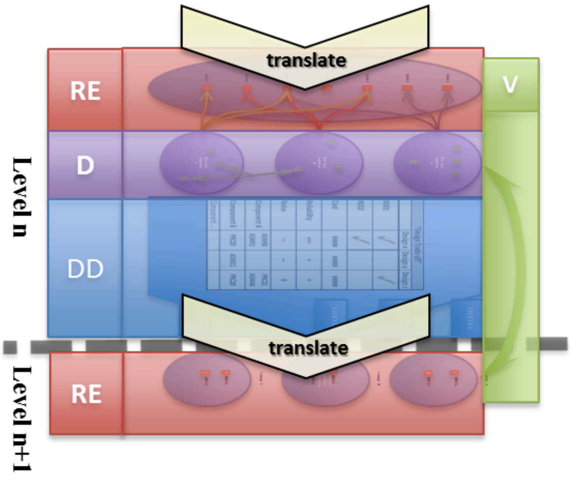


**Fig. 11** Figure 5 turned 90° labeled with the requirements & design processes. Requirements Engineering (RE), Design (D), Decision Making (DD), Validation (V)

By themselves these processes are not new, the model presented in this article merely emphasizes the application of these processes in the transformation of information in the product dimension. The model also makes clear that the seven main processes are used at every abstraction level throughout the entire life cycle of the product. When one would describe each process individually, one must take care that the nature of the deliverables differs at each abstraction level. I.e. requirements do not have the same format at each abstraction level. Designs use different communication means and formats to ensure correct interpretation of the requirements. Where SysML can work very well on the Technical level to communicate between the systems, it may be fully inadequate to communicate at business level. The authors which to stress that this awareness is a huge advantage when complex products are developed in large organizations, where the decision-making process from a political or social viewpoint are troubling the view on the product’s structure and dependencies. In other words, understanding the product’s structure helps to get a grip on the other two dimensions: process and people.

## Conclusion

In summary, we conclude that the traditional (*tree structured*) way of visualizing requirements does not do justice to the inherent complex nature of current products and product development processes. The structured and systematic method described in this article makes the information transformation process transparent and explicit throughout the life cycle of the product.

By adding the *missing links* (Design Decisions) to the traces, a complete, traceable, verifiable and justifiable structure is made throughout all product abstraction layers. Information is structured and described by means that are appropriate to the direct stakeholders for each level. No information gets lost or gets degraded by deploying the product abstraction levels, the design decisions and SITIO.

The awareness of professionals to know where the parts they work on fit in the overall structure; the constant awareness of what is *up* and what is *down* in the product structure helps to determine how complete and how good the specifications are and what the dependencies are. By using the product abstraction levels as a map, everyone involved in the product development process is better equipped to help others and at the same time can be helped by others more effectively.

In this way complex systems can be developed and managed in a disciplined and structured manner.

## References

1. Rebel, Martin; Fischer, JörgW; Haße, Armin; Michielsen, Cees (2013): Enhancing Interpretation-Quality of Requirements Using PLM Integrated Requirements-Communication in Cross Company Development Processes. In: Michael Abramovici und Rainer Stark (Hg.): Smart Product Engineering: Springer Berlin Heidelberg (Lecture Notes in Production Engineering), S. 43–50.
2. H. Negele, S. Wenzel, T. Pfletschinger and G. Getto: Successful Implementation and Application of Continuous Risk Management to Complex Systems Development in the Automotive Industry. In: Published and used by INCOSE 2005, S.4
3. Prof. A. Katzenbach: Informationstechnik und Wissensverarbeitung in der Produktentwicklung. Vorlesung (Handout) 2009/2010 Universität Stuttgart 2009, S.28(14)