Closing the gap between requirement management and system design by requirement tracing

Hayri Verner Hasou, Principal Concept Engineer, Infineon Technologies, Pavia, Italy
(*HayriVerner.Hasou@infineon.com*)

Guillermo Conde, Principal Concept Engineer, Infineon Technologies, Villach, Austria
(*Guillermo.Conde@infineon.com*)

Adrian Rolufs, Director of Solutions, Jama Software, Portland, OR, USA
(*arolufs@jamasoftware.com)*

Dominic Scharfe | Thomas Arndt, COSEDA Technologies GmbH, Dresden, Germany
(*Dominic.Scharfe@coseda-tech.com, Thomas.Arndt@coseda-tech.com*)

***Abstract*—The development of safety critical systems requires traceability of requirements through the product development process. Safety guidelines like DOC178C, ISO26262, IOC61508 or Automotive SPICE/ISO/IEC 15504 require bi-directional requirement traceability. In a system development process, it is required to breakdown high-level requirements like application objectives, goals, aims, aspirations, customer expectations and business needs into development ready, low-level requirements while documenting this process. Requirement engineering is a systematic approach to specify and manage requirements and has the following goals: understanding and documenting the needs of the stakeholders to minimize the project risk, identifying the relevant requirements, documenting the requirements following a certain standard and managing the requirements in a systematic way. Requirements are usually gathered by requirement management tools like Jama Connect. In a state-of-the-art development process, one of the first steps towards implementation is the development of an executable specification which will be refined via the architecture exploration to lower-level models which will be later used as reference for the product implementation [6]. Top-down development implies the partitioning of a complete system into smaller blocks or functions, sometimes mapping to different domains like hardware/software or digital/analog. Modelling languages like SystemC and SystemC AMS can be used for these Model Based System Engineering (MBSE) steps. Tools like COSIDE support the modelling and debug process.**

In this paper we describe how requirements gathered by a requirement management tool are mapped to an executable system level model, how they can be traced, what can be proven using this linkage information and how this traceability makes the design process more structured and reliable.

When tracing beyond requirements to all artifacts located in the tool chain, the usage of traceability can bring several benefits [2], [3]. This is particularly relevant for analog systems where top-down workflows are often slowed down by the limitations of transistor-level simulators [3]. We will show also how the interoperability of tools of different vendors permits the setup of an easy-to-use environment, that minimizes the additional effort and due to the easy access and guaranteed consistency, the overall design time will be reduced.

Requirements; Tracing; System Level Design; Model Based System Engineering, Executable Specification

I. INTRODUCTION

Requirements have a lifecycle: they are linked to stakeholders, then get reviewed and refined to drive the architecture implementation. The right methodology should support requirements development, but also include the possibility to freeze and release a stable configuration and facilitate the further processing in the development cycle. To ensure high quality and meet the customer expectations, requirements has to be traceable during the whole development cycle. The growing complexity and extended configurability driven by multi-market and multi-domain product requirements demands a well-structured handling of those requirement. This becomes only achievable if all stakeholders have clear insights on the project status and can check for feature completeness at any point in time. Furthermore, the technical marketing team, the product definition engineers, as well as the concept engineers have a common database and thus a better common understanding of the product to be implemented.

Requirements are collected by the technical marketing team and reviewed in close cooperation with the customer. Those general requirement items are joined in a requirement management system (see Figure 1). Ideally these requirement items can be re-used, mapped, and annotated to a simulation model to ensure a seamless transition to the concept evaluation phase. Usually, state-of-the-art modelling languages like SystemC [5] and SystemC AMS [4] are used for a model-based design process at a high level of abstraction implementing the features and thus the requirements.

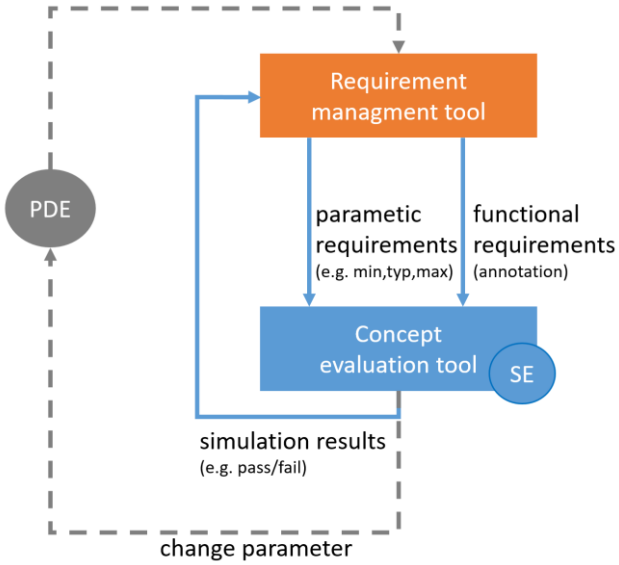

Figure 1: Overview

II. Requirements Overview

In the development of an integrated circuit, there are different types and levels of requirements involved in the product development lifecycle. The levels of requirements are often represented by a system engineering V model, like the one shown in Figure 2.

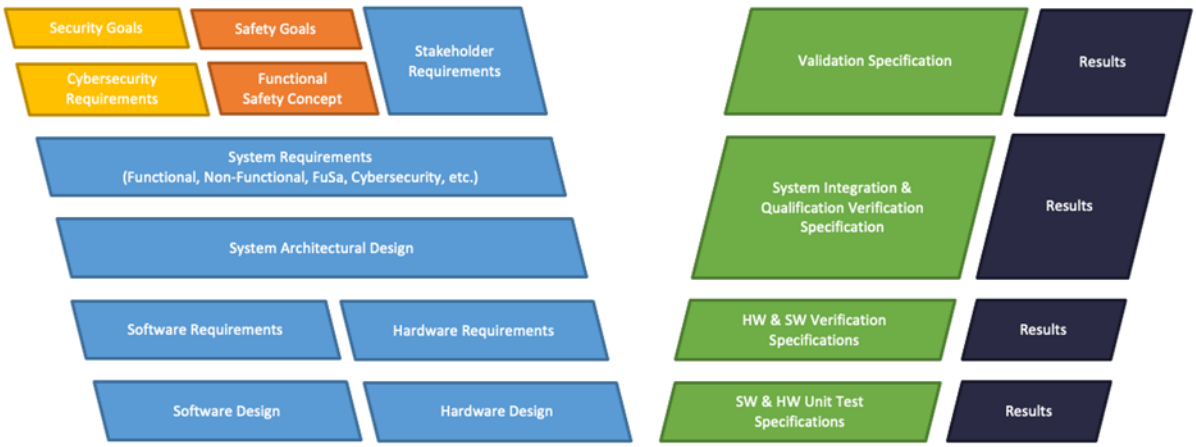

Figure 2: System Engineering V Model

When it comes to hardware design, only a subset of the requirements used in developing a full embedded system are relevant to the hardware-based model. Many of the requirements are at an abstraction level which is above the threshold needed for the hardware developers. Cybersecurity goals, Functional Safety Goals, Stakeholder Requirements, and all system level requirements must be decomposed down to the hardware level before they can be associated with a specific implementation. Many of the system level requirements may also be implemented in software, having no direct impact to circuit design.

Decomposition of requirements is tracked through traceability relationships in the requirements management tool so that each hardware requirement is associated with higher-level requirements. This traceability can then be followed to see how a model satisfies a Stakeholder Requirement, for example, without having to be directly associated with the Stakeholder Requirements.

Within the hardware requirements, there are many different types:

A. Non-functional requirements – Out of Scope

Non-functional requirements specify conditions that are necessary to operate the product under certain conditions. These may include quality-of-service, government regulations, and handling requirements among others. While these requirements are important for developers to consider, they do not directly impact system-level modeling.

B. Architectural requirements – In Scope

Architectural requirements can specify a constraint on how a feature is implemented. For example, configurability achieved by hardware settings (e.g. LASER fuses) or defined by the software. These requirements impact the circuit design concept and therefore should be accounted for in circuit design models but are not the primary focus.

C. Functional requirements – In Scope

Functional requirements describe all the features and functionality that a circuit design must achieve. These requirements are the primary focus of circuit design modeling and there is high value in being able to associate each model or element of a model with the functional requirement(s) that it satisfies.

D. Parametric requirements – In Scope

Parametric requirements describe the electrical characteristics that a circuit design must achieve. This includes power supply voltage ranges, current consumption, and efficiency among other characteristics. Just like the functional requirements, these requirements are a primary consideration for circuit design modeling. Each model will include electrical characteristics and associating those with requirements will ensure that no requirements or requirements changes are missed by the model.

E. Packaing requirements – Partially in Scope

Packaging requirements specify details of the circuit packaging. They are partially relevant for high-level concept evaluation. The performance of a circuit design can be impacted by its packaging and positioning on a PCB. These impacts should be accounted for by the circuit design model.

A requirement management system should provide the means to classify the different requirement types. In this way, the functional and parametric requirements that are the primary focus of circuit design modeling can be easily identified out of all the other requirements. As these requirements are associated with models, automated completeness checks should be performed to make sure that none of the requirements have been missed in the modeling activities. Any missed requirements could lead to developing a product or system that does not satisfy all its requirements, leading to expensive fixes later in the development cycle.

III. USE CASES AND ANALYSIS

Modelling is a way to simplify high-level design and development. It is not intended to replace any existing procedures for chip design. A good approach should seamlessly allow refinement from an abstract level to a detailed level, even allowing to refine the model with information gathered through the detailed design phase.

A. Modelling approaches

There are in principle two general ways to develop and use models:

1) to propose new architectures and circuit concepts, which might be used as reference throughout the project (top-down approach)

2) to generate simplified models of existing blocks – either for re-use purposes or simply to speed up simulation (bottom-up approach)

There are rarely scenarios where system modeling is using only option 1 or option 2. In most cases there is a mix of both approaches. When a system model is based on existing designs, it is common to start with option 2, but often users might add further ideas and develop them using approach 1.

Both of these approaches will also be required when using a model as a vehicle for requirement engineering and application use case development to complement the product description from marketing experts.

To reduce the risk of errors due to missing knowledge, modelling should be as simple as possible and done in a convenient way where the expertise of the individual designer is taken into consideration.

In the case of electronic systems, an analogue designer should be able to develop models based on SPICE-like primitives, while a digital designer may prefer procedural languages.

1) Top down modelling ("agile concept/design engineering")

In a proper top-down approach, the model is developed first, and the product design follows the model behavior and references it through all stages. Relevant implementation findings or feasibility constraints will be directly reflected back to the model.

2) Bottom up modelling ("design-driven concept engineering")

Ideally, after the creation of a SPICE-like representation of the circuitry to model, the process should be already concluded. All further steps require significant additional abstraction. RNM method from analogue towards VHDL, Verilog or SystemVerilog [8], implies many steps and significant risk. A very good knowledge of the overall system is needed to judge if certain abstraction steps are still allowed. Furthermore, a vast knowledge of system theory and linear algebra of underlying solvers is required, and this additional effort may not even bring any speed advantage at the end of the day. SystemC AMS can simplify this flow significantly [7].

Modeling of electronic systems is often based on SystemC AMS extensions because these can be used for a wide variety of use cases such as:

- Executable specification
- Virtual prototyping
- Architecture exploration
- Integration validation

A detailed description of these use cases can be found in the Accellera SystemC AMS users Guide [7]. It is possible to map some use cases to phases of the development and requirements states as in Figure 3.

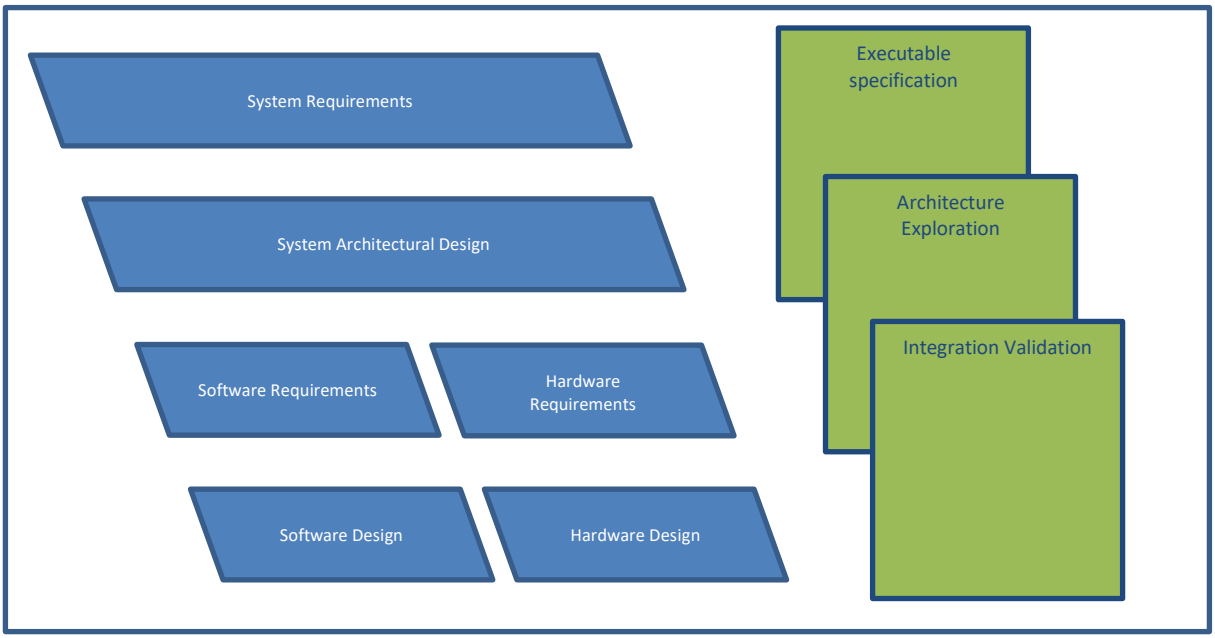

Figure 3: SystemC AMS Use Case Mapped to Development Phases

An *Executable Specification* is a description of the system by using simulation. It complements textual requirements using functional or algorithmic models and is not necessarily related to the implementation of the system.

Architecture Exploration analyzes how the ideal functions defined during the executable specification phase can be mapped into a system architecture. It happens in 2 phases:

1) Executable specification is refined by adding non-ideal properties to evaluate the impact on the system
2) Detailing of the structures and interfaces

Different solutions are compared in terms of factors like cost, design time and associated risks

When all system components are clearly described, the *Integration Validation* setup is used to merge analog and digital HW/SW components to verify the correctness of the overall system. All the interfaces should be identified precisely: analog components shall have robust electrical node description and digital components behavior shall show cycle accurate pin functions. In this setup it should be possible to replace some models of Circuit blocks with actual implementation views like schematic/netlist or RTL code.

IV. Flow Implementation, Tool Support

In this paper the authors show how the product development process can benefit from traceability between requirements and system models. As a case study we outline such a process using Jama Connect as the requirement management tool and COSIDE as the tool for heterogeneous system modelling. Besides those tools being used as reference for this paper, the process can be adapted by other tool vendors as well.

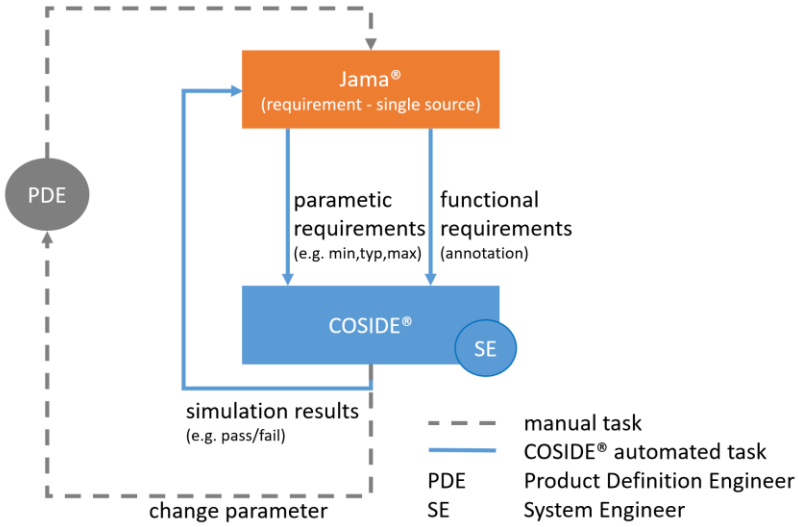

Figure 4: Tool based implementation

Figure 4 refines Figure 1 showing the roles of the case study's tools. The product definition engineer (PDE) populates the requirements collected by the stakeholders (e.g. technical marketing, application engineering) as items into the Jama database. As described in section II, there are two primary requirement categories which are relevant for the system models: parametric and functional requirements. These are accessed by the system engineer (SE), which uses COSIDE to create the system models. This engineer then creates trace entries by connecting requirements to system model elements (e.g. COSIDE schematics, SystemC source code and test benches - see Figure 6). The trace entries are used to visualize the relation between model elements and their functional requirements, to adjust parameters according to parametric requirements or to back propagate simulation results. Insights obtained by the concept studies can then be evaluated by the PDE and SE. The evaluation results can lead to changes of the requirements (e.g. change of parametric requirements).

The requirements management tool Jama is the single source for the requirement data. It provides a database service and an application server. The application server hosts a web application which serves the frontend to the users and provides a machine to machine interface for accessing and changing requirements.

COSIDE is a design environment for heterogeneous systems. It covers the modelling of a holistic system containing the analog and digital hardware as well as the embedded software. The system models are stored in projects, which are usually folders and files on the file system and usually are managed by version control systems (VCS) like svn or git. Projects can include a requirement tracing database which holds requirement and tracing information. By managing this database in the VCS, features like snapshot, baselining and visualizing incremental differences are supported (see Figure 7). Especially when multiple requirement items are changed an overall change summary is crucial to keep track of those changes and their consequences.

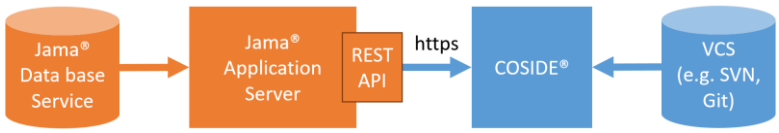

Figure 5: Tool communication, data flow

The communication between the two tools is illustrated in Figure 5. The Jama application server provides a REST API for read and write access. COSIDE projects can be configured to access the Jama application server via its URL and user credentials. A filter mechanism is provided to select only the relevant requirements (e.g. functional

and parametric requirements). The requirements are retrieved and stored inside a tracing database which is contained in the project. Additionally, the database holds all the mapped requirements as trace entries which the user (SE) creates in COSIDE. By providing a synchronization mechanism, external changes of the requirements in the Jama database, as well as in the VCS can be applied at any time.

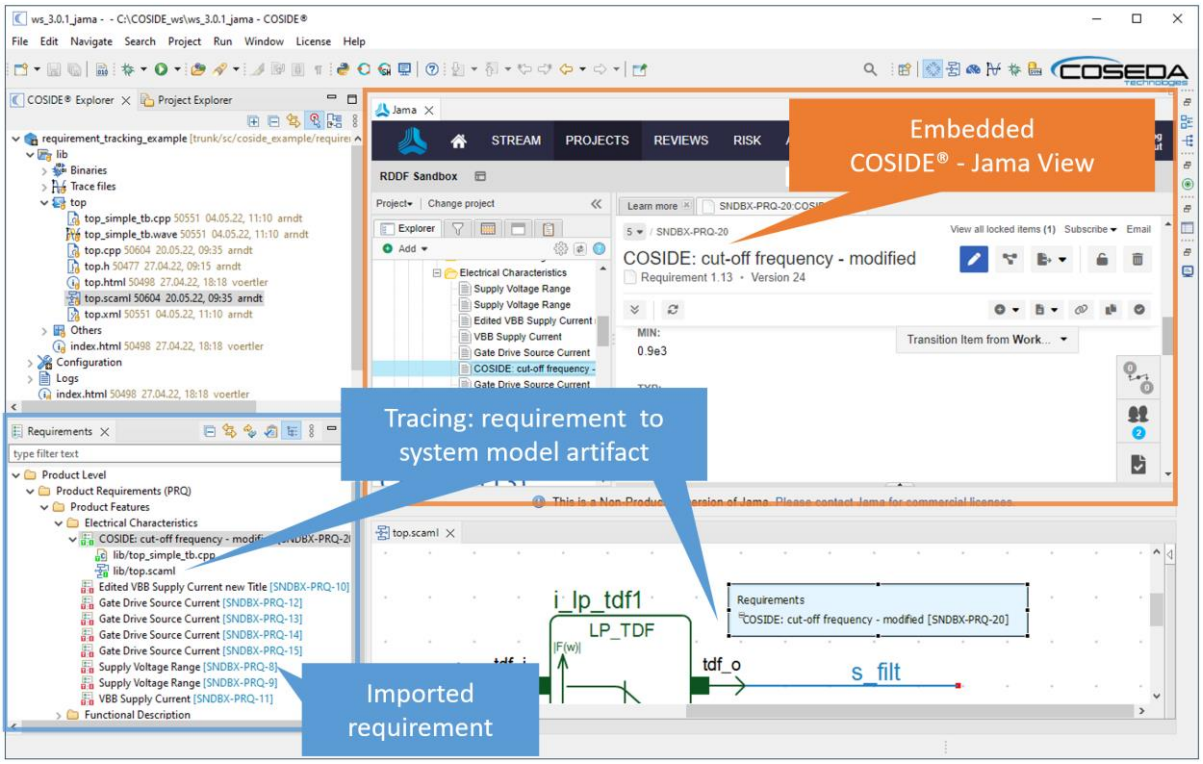

Figure 6: Requirement tracing with COSIDE®

The current status of the successful established relationships between the requirement items and the system model elements gets indicated by a traffic light visualization within the requirement view. Therefore, a good overview is provided if a requirement is kept untouched and still has to be implemented and tested or if e.g. the implementation is already done, but a test is still missing.
As a special kind of requirement, the parametric requirement has a direct impact on the system behavior, since the parameter values are getting extracted from the Jama server and are used directly by the system model.
Once all the requirement items are assigned to the system model elements, which are representing their functional implementation as well as their related tests, a concept validation is done by executing the system simulation.
If this concept validation reveals some feasibility issue, the SE works in close cooperation with the PDE to resolve those issues. Once the PDE decides to change a requirement, a re-synchronization to the Jama server can be performed to get the latest updates. In this case a detailed overview of those changes is displayed while comparing the incoming update from the Jama server with the current requirement tracing database within the project (see Figure 7).
Doing so, the SE can determine precisely if any the system model implementation has to be changed or, in case of a parametric requirement change, only a simulation re-run has to be performed to validate the latest parameter values. To see the status of the mapping between the requirement items and the system model elements, a requirement tracing report is generated at the push of a button (see Figure 8). This report displays a compact overview of the current requirement coverage regarding the implementation and the tests of the requirement. In the traceability matrix the tracing gets reported in a table-based summary.

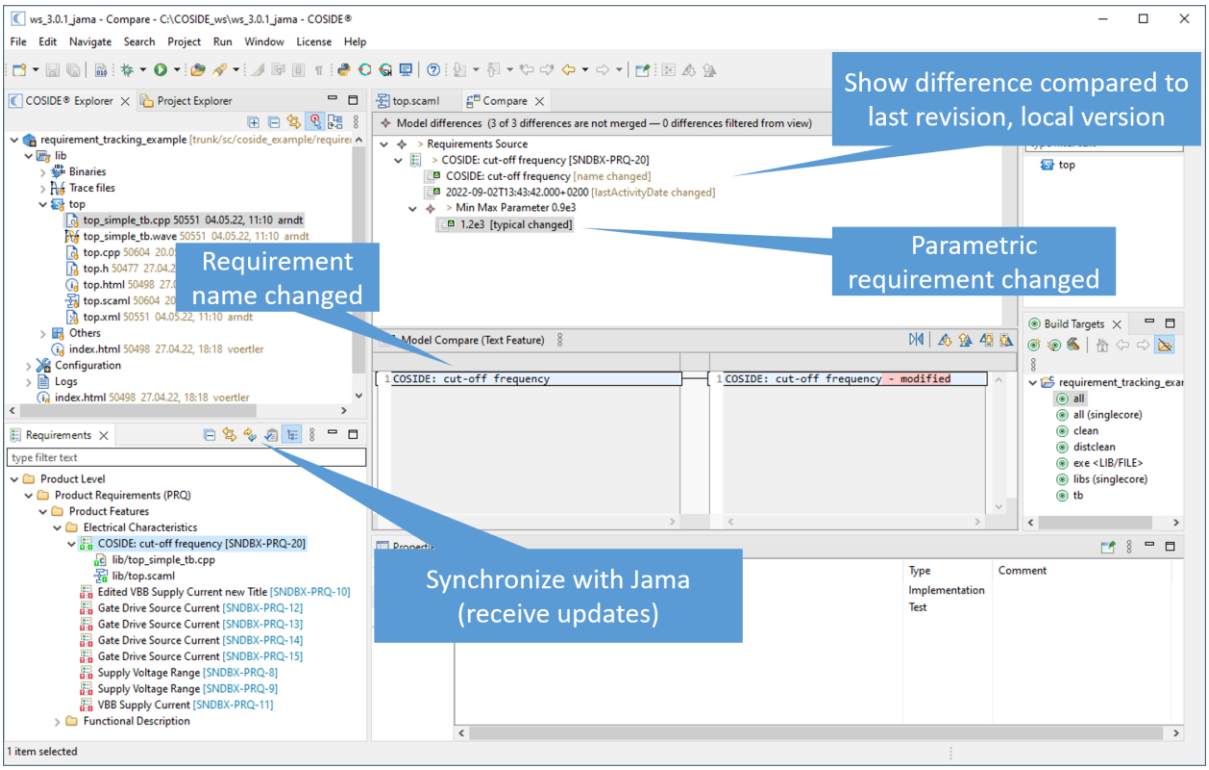

Figure 7: Update scenario – visualizing incremental changes, comparison to reversion management system

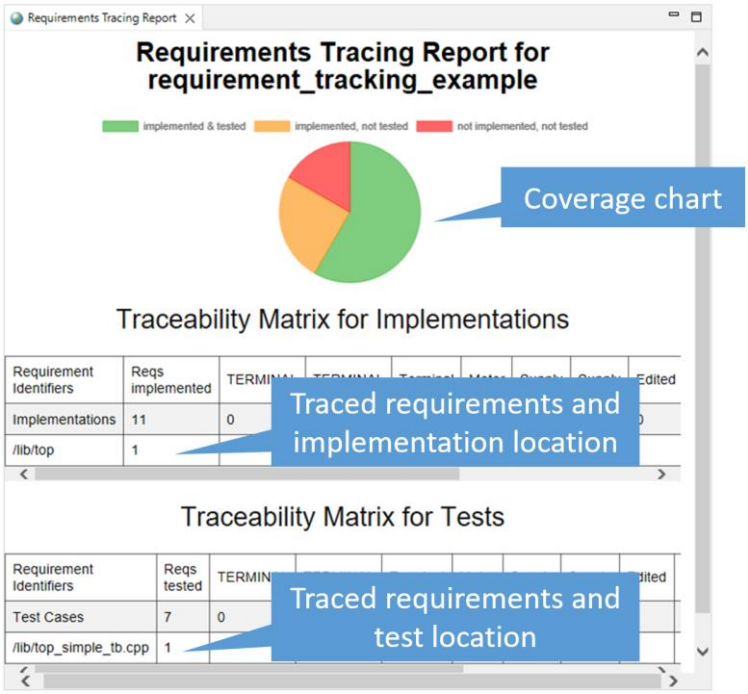

Figure 8: Generated requirements tracing report with coverage chart

V. OUTLOOK

To further improve this flow an automated invocation of statistical analysis dealing with the parametric requirements is planned. In this case, parameter sweep, Monte-Carlo, corner-case analysis, and even constrained random verification with UVM SystemC [9] can be automatically performed by accessing the parametric requirements. This feature will automate the different sensitivity analyses needed to evaluate the feasibility of the product.

The scope of this publication has been limited to the development of hardware-centric products. In the future, our objective is to raise the level of abstraction of this approach by applying the same idea to embedded systems, where a proper HW/SW partition is key for the successful development of the product.

VI. CONCLUSIONS

Bridging the gap between requirements management and circuit design by establishing an automated workflow to provide close cooperation between the product definition engineer (PDE) and the system engineer (SE) is very beneficial. The growing complexity and configurability of today's products and therefore a rapidly growing requirement count makes such an approach crucial.

Furthermore, the promoted mapping of text-based requirements into hardware models improves the communication channels not only between PDE and SE but also with the product developers, by translating the requirements into a language that is more natural for the hardware developer.

By synchronizing our requirement database with our central system-development tool, we can clearly show the dependencies between requirements and hardware blocks. This approach simplifies the digital/analog partition performed by the SE in this kind of heterogeneous hardware-centric systems. This single-source approach reduces the typical overhead caused by the manual updates to the requirement database.

VII. REFERENCES

[1] Wiegers, Karl (2013). "Requirements Traceability: Links in the Requirements Chain, Part 1". jama. Retrieved 2016-12-14.

[2] Lerche, Felix (2019). "5 REASONS WHY A REQUIREMENTS TRACEABILITY MATRIX IS NOT ENOUGH".

[3] Scherr, Wolfgang (2020) "Next Generation System Level Design" (CUAS) ISCD master seminar (see post: guest lecture)

[4] "IEEE Standard for Standard SystemC® Analog/Mixed-Signal Extensions Language Reference Manual," in IEEE Std 1666.1-2016, vol.,no.,pp.1-236, April 6 2016 doi: 10.1109/IEEESTD.2016.7448795

[5] "IEEE Standard for Standard SystemC Language Reference Manual," in IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005) , vol., no., pp.1-638, Jan. 9 2012 doi: 10.1109/IEEESTD.2012.6134619

[6] Accellera SystemC AMS Working Group: "SystemC Analog/Mixed-Signal User's Guide", Jan. 2020

[7] Einwich, K.; Uhle, T. (2010): SystemC AMS - holistic analog, digital, hardware and software system-level modeling. In: *EDA Tech Forum Journal* 7 (1), S. 28–33.

[8] "IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009) , vol., no., pp.1-1315, Feb. 21 2013 doi: 10.1109/IEEESTD.2013.6469140

[9] Thilo Vörtler, Thomas Klotz, Karsten Einwich, Felix Assmann. "Simplifying UVM in SystemC", DVCon Europe 2015.