Management of the ITER PCS design using a system-engineering approach

Marcello Cinque, Gianmaria De Tommasi *Senior Member, IEEE,* Peter C. De Vries, Francesco Fucci, Luca Zabeo, Giuseppe Ambrosino, Sylvain Bremond, Igor Gomez, Damien Karkinsky, Massimiliano Mattei, Remy Nouailletas, Alfredo Pironti, Fernanda G. Rimini, Joseph A. Snipes, Wolfgang Treutterer, Micheal L. Walker

***Abstract*—The Plasma Control System (PCS) in a tokamak device is in charge of controlling the evolution of the plasma parameters according to prescribed models against uncertainties and disturbances. The PCS accomplishes its mission by receiving data from the diagnostics, and by computing in real-time the commands to the actuators that affect the plasma behaviour. The design of the ITER PCS includes many different aspects, which are not limited to the design of the control algorithms, including the definition of the verification and validation tests for the various components, and commissioning procedures. Moreover, contributions come from different parties that adopt heterogeneous sources. To homogenize these contributions and to keep track of the PCS life-cycle throughout the various design stages, a specific system-engineering approach has been adopted on top of the *standard* ITER life-cycle and requirement management process. Such an approach relies on a database implemented using Enterprise Architect® that allows modelling the various aspects of the PCS design using SysML. This paper gives an overview of the adopted approach, and describes the structure and the current content of the PCS database.**

***Index Terms*—System-engineering, SysML, ITER, Plasma Control System.**

I. Introduction

THE ITER tokamak is currently under construction in France, and it will be the world's largest fusion device, designed as the next step between existing research machines and future power plants. The Plasma Control System (PCS, [1], [2]) is an essential ITER component that plays a major role towards the achievement of the final goal, which is to reach a fusion gain factor $Q = 10$.

M. Cinque, G. De Tommasi and A. Pironti are with Dipartimento di Ingegneria Elettrica e delle Tecnoologie dell'Informazione, Università degi Studi di Napoli Federico II and with Consorzio CREATE, via Claudio 21, 80125 Napoli, Italy. email: detommas@unina.it.

P. C. De Vries, J. Snipes and L Zabeo are with ITER Organization, Route de Vinon sur Verdon, CS 90 046, 13067 St Paul Lez Durance Cedex, France.

F. Fucci is with Critiware, c/o Centro Meridiana, Torre Antares, 80013, Casalnuovo di Napoli, Italy.

G. Ambrosino is with Consorzio CREATE, via Claudio 21, 80125 Napoli, Italy.

S. Bremond and R. Nouailletas are with CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France.

I. Gomez and W. Treutterer are with Max Planck Institute for Plasma Physics Boltzmannstr 2, 85748 Garching, Germany.

D. Karkinsky and F. G. Rimini are with CCFE, Culham Science Centre, Abingdon OX14 3EA, United Kingdom.

M. Mattei is with Dipartimento di Ingegneria Industriale, Università degli Studi della Campania "Luigi Vanvitelli", 80131 Aversa, Italy and with Consorzio CREATE, via Claudio 21, 80125 Napoli, Italy.

M. L. Walker is with General Atomics, PO Box 85608, San Diego, California 92186-5608, USA.

The PCS is the real-time system that robustly control the evolution of the plasma parameters such as plasma current and density, against uncertainties and disturbances. PCS accomplishes its mission by receiving data from the diagnostics, and by computing the commands to the actuators that affect the plasma behaviour (e.g., the gas injection valves, and the additional heating systems).

An international team is carrying out the PCS design, mainly focusing on first plasma operation ([2], [3]), which is the first phase of the so called *staged approach* that includes the following:

- First Plasma Operation (FP);
- Pre-Fusion Plasma Operation 1 (PFPO1);
- Pre-Fusion Plasma Operation 2 (PFPO2);
- Fusion Plasma Operation (FPO).

For the first stage, i.e. for First Plasma Operation, the design activities have been further breakdown in tasks as follows:

i) design of the overall PCS functional architecture;
ii) design of the control algorithms and support functions (including the definition of the interfaces with other ITER systems, such as diagnostics and actuators);
iii) definition and execution of the procedures for the performance assessment of the PCS components;
iv) definition of the pulse schedule;
v) definition of the commissioning procedures.

It should be noticed that task **i)** will provide an architecture which is not strictly limited to FP, but it will include all the features envisaged during the ITER life-time [4], such as the handling of different exception classes [5]. Similarly, also task **ii)** will go beyond what is strictly related to FP, that includes the plasma breakdown and the plasma current ramp up to about 1 MA. Indeed, task **ii)** will look also for some *advanced* functions, like the control of elongated plasmas ([6], [7]), that are needed during early operation soon after the FP phase.

To manage all the different aspects of such a complex project, where contributions are given by different participants, it is essential to rely on a system-engineering approach. Such an approach, not only enables the assessment of the compliance of the design with the requirements, but permits also to have a centralized and coherent view of all the different aspects of the design itself. System-engineering has been successfully adopted for large and complex projects in avionics [8], aerospace [9], [10] and big physics [11], and has

been recently applied also to manufacturing in the Industry 4.0 context [12]. System-engineering has proved to be beneficial also for the management of civil nuclear fusion projects [13], and for this reason has been envisaged since the very beginning of the ITER project [14].

To adopt a system-engineering approach specific to the PCS, a database, called the PCS Database (PCS DB), has been deployed at ITER and used as the central collaboration tool. The PCS DB is implemented using the Enterprise Architect® [15], a commercial software that fully supports the system-engineering approach. While during the preliminary design phase, the use of the PCS DB was limited to the management of the PCS requirements [16], in the current design phase its use has been extended to support and document the tasks **i)-v)**.

This paper introduces the overall structure of the PCS DB, focusing on how the PCS design has been modelled using SysML [17], as well as on the relationships (*links*) between the different design *objects* (called *artifacts* in the system-engineering jargon) represented in the PCS DB. Indeed, thanks to these relationships the model SysML can be exploited to check the completeness of the design in view of the PCS implementation [18]. Moreover, part of the information stored in the PCS DB will be used to setup the PCS components implemented within the ITER Real-Time Framework (RTF, [19], [20]).

The remainder of the paper is organised as follows. The overall structure of the PCS database and the adopted modelling approach is described in Section II. How the different types of requirements are represented in the PCS DB is briefly mentioned in Section III. The requirements and constraints related to the interaction of the PCS with the other plant systems is traced in a specific components of the PCS DB described in Section IV. Starting from all the captured the requirements that are relevant for FP, the functional architecture described in Section V has been derived. Following the work-flow described in [20], each PCS function is first prototyped in the PCS Simulation Platform (PCSSP, [21]) to assess its performance [22]; Section VI describes how the relevant information from the simulation environment is captured in the PCS DB. Moreover, Section VII describes how the various PCS functions are mapped into the actual PCS architectural components, while Section VIII shows how the PCS commissioning procedures are modelled. Eventually some conclusive remarks are given.

II. The PCS database

The PCS DB is a centralized repository supporting the design team on the collaborative definition of PCS design artifacts (i.e., elements used to model the design) in SysML, a widely used modelling language to specify systems designs. Basically, the PCS DB captures all design information, from requirements to verification and validation of the various PCS functions, and it is used to *trace* requirements to system functions and architectural elements, to both (i) formally demonstrate the coverage of requirements imposed on the PCS, and (ii) justify design choices.

SysML provides a semi-formal language to structure the PCS model in hierarchical entities, named package diagrams. A package is a model artifact that acts as a container for other artifacts stored in the PCS DB.

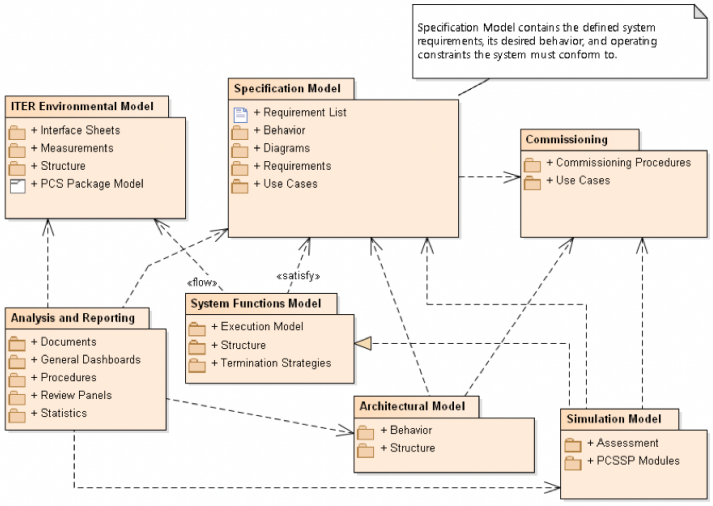

Fig. 1. Main packages of the PCS model within the PCS DB. According to the SysML formalism, the arrows show the dependencies between the packages. In some cases SysML is extended by means of stereotypes, such as $<< satisfy >>$ and $<< flow >>$, to are used to extended SysML, and to make explicit the type of dependecy.

Fig. 1 depicts the current structure of the PCS DB in terms of packages and of the dependency links between them, which aim at capturing the logical connections among all the components involved in the design. The rationale behind the subdivision of packages relates to the classical *V-model* [23], so that root-level packages are containers of artifacts produced at the various stages of the design flow, from specification, to design, development (of simulation models), and validation.

The package structure of the PCS model includes six main packages:

1) **Specification Model**; this package contains all the artifacts concerning the description of the requirements and the use-cases of the Plasma Control System. The PCS DB will be used to assess the compliance of the design and the architecture to high-level requirements, by linking them to design artifacts contained in the other packages.
2) **ITER Environmental Model** that includes all the elements concerning the interfacing of the PCS with other plant systems, such as, actuators and diagnostics. It contains the interface requirements extracted from the official ITER interface documents, and structured information, in the form of measurements signals, including network messages formats, where available. This information is useful to know in detail what information the PCS can expect to send/receive to/from other plant systems.
3) **System Functions Model**; this package includes the functional decomposition of the PCS that includes not

only control functions (i.e., control algorithm), but also data processing functions (called *support functions* in the ITER PCS jargon), and exception handling functions. This package contains also the related execution model and the signals internal to the PCS. The PCS functions modelled in this package are designed to satisfy the requirements in the *Specification Model*, and to be compliant with the interface requirements contained in the *ITER Environmental Model*.

4) **Architectural Model** that contains the artifacts concerning the design of the PCS architecture, including SysML behavioural and structural diagrams; for example, this package contains the artifacts related to the design of the so called *compact controller* [4], as one of the PCS architectural solutions to implement the continuous control functions specified in the *System Functions Model* package.
5) **Simulation Model**, which includes all the artifacts concerning the simulation of PCS control algorithms. It contains references to the models of the PCS functions in the PCSSP simulation environment [21]. It contains also references to the assessments conducted on such models [22], to demonstrate the performance of control functions in given experimental scenarios, modeled as test-cases. Assessment models and results are linked with the relevant PCS control functions in the *System Functions Model* package. In this way, it is possible to understand what specific function a simulation model implements and, vice versa, what are all the simulation models (including variants, if relevant) that have been specified for a given control function.
6) **Commissioning**; this package contains all the artifacts about the PCS commissioning; it contains the list of PCS use-cases for system commissioning and integrated commissioning activities, and the related commissioning procedures. Use-cases are linked with functional or architectural elements involved, and to the relevant test-cases.

Other than the above six packages, the PCS DB also includes an *Analysis and Reporting* package, that provides summarized views and scripts to generate documentation from the design model. Moreover, modeled artifacts can link, as sources, to external documents stored in the ITER document management system.

III. The Specification Model

In the PCS DB requirements and use-cases for the PCS are modelled within a *Specification Model* package (see also [16]). Requirements are further decomposed into *stakeholder requirements* (SHR), used to describe high-level needs for the PCS, and *system requirements* (SYR), detailing how the high-level needs reflect into desired system-level characteristics, such as, control goals, exception handling strategies, or architectural constraints.

The relationships between requirements are modeled using SysML requirement diagrams, that allow capturing the decomposition of requirements into sub-requirements, and to trace what are the originating SHRs of given SYRs (using the *derivation* relationship). Then, SYRs need to be further traced down to system functions and architectural elements (using the *satisfy* relationship) to keep track of how they are then reflected in the design. Some SYRs imply (but often don't state explicitly) some assumed level of control performance. In this case, *performance requirements* are specified too, and linked with originating SYRs, to clearly state what is the expected performance level. This is particularly useful to guide performance assessment activities (e.g., to define simulation scenarios and test-cases) and to demonstrate, at least via simulation, if the designed control functions are in line with the expected performance.

As an example, from the following SYR:

"The PCS shall be able to simultaneously control currents in CS and PF coils, using the power supplies in voltage control mode",

two separate performance requirements are derived, corresponding to the application of the controller for first plasma operation and for commissioning:

"Performance Requirement 1: For First plasma operations, the PCS shall be able to simultaneously control currents in CS and PF coils, using the power supplies in voltage control mode, with step response overshoot $< 5\%$ *and settling time* $< 0.5\ s$*, and after the user specified expected breakdown time achieve a tracking error* $< 30\ A$*".*

"Performance Requirement 2: For coil commissioning, the PCS shall be able to simultaneously control currents in CS and PF coils, using the power supplies in voltage control mode, with step response overshoot $< 5\ \%$ *and settling time* $< 0.5\ s$*, and guaranteeing a zero steady-state error to a ramp reference".*

IV. The ITER Environmental Model

The PCS works closely collaborating with other plant systems, which means that the PCS should not be treated alone during its design. In particular, PCS functions need plasma parameters as well as information from other plant systems to perform their actions. A plasma parameter could not be linked to just a diagnostic or a single operation instrumentation. A direct link – i.e., one diagnostic to one plasma parameter – is generally limited to a restricted number of cases. In principle, plasma parameters are computed differently according to the machine operation stage. For example, during FP the line-averaged electron density will come only from the Density Interferometer Polarimeter (DIP), while at PFPO1 it will come from the DIP, the TIP and the Core Thomson Scattering diagnostic.

To allow a clear traceability of the diagnostic requirements in the above-mentioned cases, the methodology should take into account the following two cases: (i) a single diagnostic generates one or more parameters; and (ii) multiple diagnostics generate one or more parameters.

The information related to the interaction of the PCS with the other plant systems is contained in the *ITER Environmental Model* package, which consists of three sub-packages, whose content is detailed in the following sub-sections.

A. Measurements

The objective of the *Measurements* package is to contain the measurement requirements, which specify the accuracy, the time resolution, and the constraints about the real-time measurements provided from diagnostic systems to the PCS. The measurement requirements collect the specifications of the plasma measurements (i.e. plasma current, neutron fluence, etc.) that constrain the PCS functionality.

B. Interface Sheets

The objective of the *Interface Sheets* package is to be the container of the requirements, the use-cases and the constraints that the PCS imposes on other plant systems and vice versa. The internal structure of the package is designed to convey such information. The content of the *Interface Sheets* package is structured into three packages, which are the *Actuators*, the *Diagnostics*, and the *Passive Systems*, to group the interface information by plant system type.

C. Structure

The *Structure* package contains the SysML blocks used to model, in detail, what are the provided data and, if any, the received commands for each of the ITER plant systems that is relevant for the PCS design.

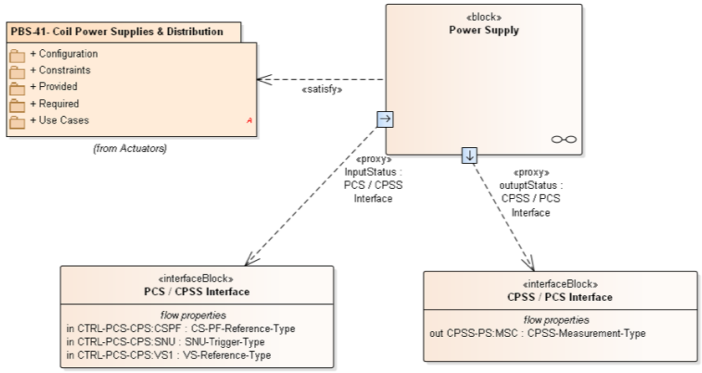

Fig. 2. Model of the power supply for the superconductive CS and PF coils within the *ITER Environmental Model* package of the PCS DB.

As an example, Fig. 2 specifies the high-level design of the Power Supply for the superconductive coils as a block (a black box, from the PCS design point of view), that accepts a set of commands (specified in the PCS/CPSS interface), and that provides the related measurements (specified in the CPSS/PCS interface), through its input/output ports. The interfaces contains the details in terms of **i)** name of the message (the *topic*) to be transferred on the Synchronous Data Network (SDN, [24]), that is, the networking bus used at ITER to interconnect the plant systems, and **ii)** for each message, the corresponding data type, that contains the detailed specification of the data structure that will be sent/received to/from the SDN.

V. PCS Functional Architecture

The PCS can be considered both as a *container* and as an *orchestrator* of control algorithms, which are configurable according to the operational purposes. Although the structure of the system does not change during different operational phases (i.e., FP, PFPO1), the control algorithm details can change. For example, the magnetic controller does not require the plasma shape control for FP, but this becomes necessary to operate in PFPO1.

The purpose of the *PCS Functional Model* is to act as an abstraction layer which separates the responsibility of a controller from the control algorithm implementation. The primary element of the PCS Functional Architecture is the PCS *Function Block*, which is represented as a SysML block containing the following information: **a)** the interface of the block, that is the input/output ports of the system function; **b)** the description of the function; **c)** its dependency with the system requirements.

Fig. 3 shows a *Block Definition Diagram* (BDD) of the PCS control model, where the *whole/part composition* link is used to model the composition/aggregation relationship between PCS function blocks.

The control functions have been divided in sub-packages. The "Basic Functions" sub-package, which contains the following blocks for FP operation:

- **CS/PF Control**, which contains the abstractions needed for the control of the currents/voltages in the CS/PF coils;
- **ECRH Control**, embodies the ports for the control of the ECRH;
- **GIS Control**, which contains the artifacts needed for the control of the gas valves;
- **Prefill Control**, which contains the ports for the control of the prefill in the vacuum vessel;
- **Density Control**, which contains the port for the initial density control for First Plasma operation;
- **SNU Control**, which contains the ports for the control of the *Switching Network Units*;
- **Magnetic Field Control**, which contains the ports tracking the information for magnetic field control in the breakdown region. It is a control function that provides the references to the CS/PF control function.

Moreover, the "Support Functions sub-package contains additional PCS functions which are not strictly control functions, such as the **Magnetic Map** one, which aims at estimating in real-time the magnetic field components in the control points inside the vacuum chamber, needed as inputs by the **Magnetic Field Control**.

Fig. 4 depicts the internal structure of the PCS in an *Internal Block Diagram* (IBD). The IBD shows the internal connection between the system function blocks, it models the data-flow processing of the system. The PCS takes signals coming from the supervisor and dispatches them to each single controller.

In SysML, functional blocks have a clear and intuitive relationship with requirements, which are modeled with the $<< satisfy >>$ dependency link. A satisfy link conveys that a given functional block is responsible to fulfill a set of functional requirements, that is, the function block will cover

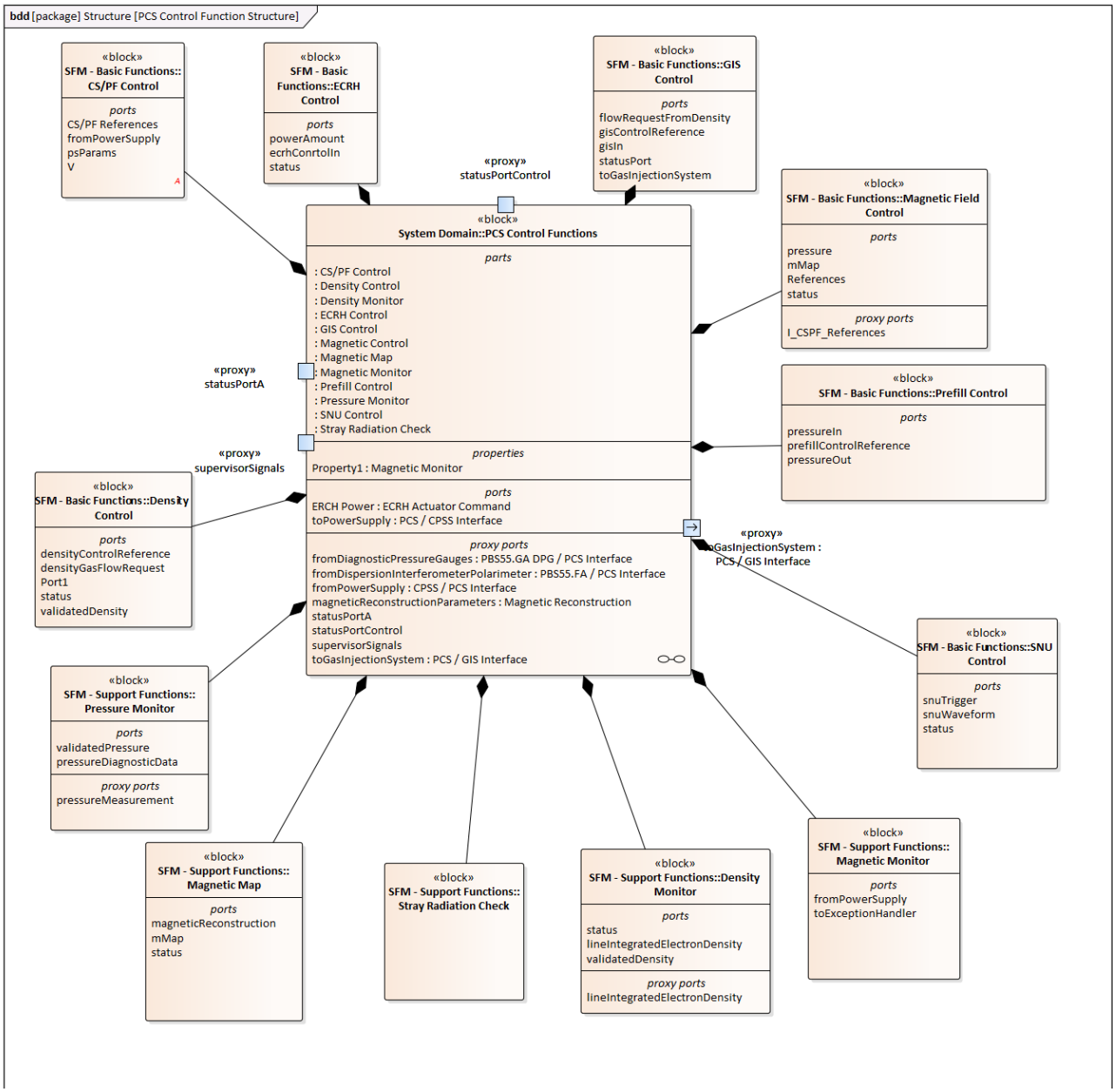

Fig. 3. Block definition diagram of the PCS Control Model for First Plasma within the System Functions Model (SFM) package.

in its implementation the feature described in the requirement. In the PCS design, we use this relationship to bond the PCS functional element to continuous control requirements. The set of connections of the PCS functional elements with the requirements establishes the responsibility of the block. The PCS functional elements are a central element of the functional decomposition, since they relate the functions to the architectural model and the simulation models. Moreover, the satisfy relationship is crucial since it allows both to check the coverage of functional requirements and to analyze **i)** if the system design covers all the needs, which are stated in the requirements, and **ii)** if there exist functional components that do not have any relationship with requirements.

In the latter case, the information can be used to check the state of the design, and to justify at what level of details the functional decomposition can be stopped, according to what is expressed in the requirements.

The mapping of functional blocks to requirements is modeled in the PCS DB using $<< satisfy >>$ relationships that can be practically defined in EA both with relationship matrices or SysML requirements diagrams.

To compute if a block covers at least one requirement, it is just needed to check, for each column, if there exists at least one arrow. To check that a system requirement is covered by a block, the following cases have to be considered: **i)** the requirement has only dependency links; **ii)** the requirement is directly connected to a block with a satisfy dependency link.

The evaluation of the first is less direct and it requires

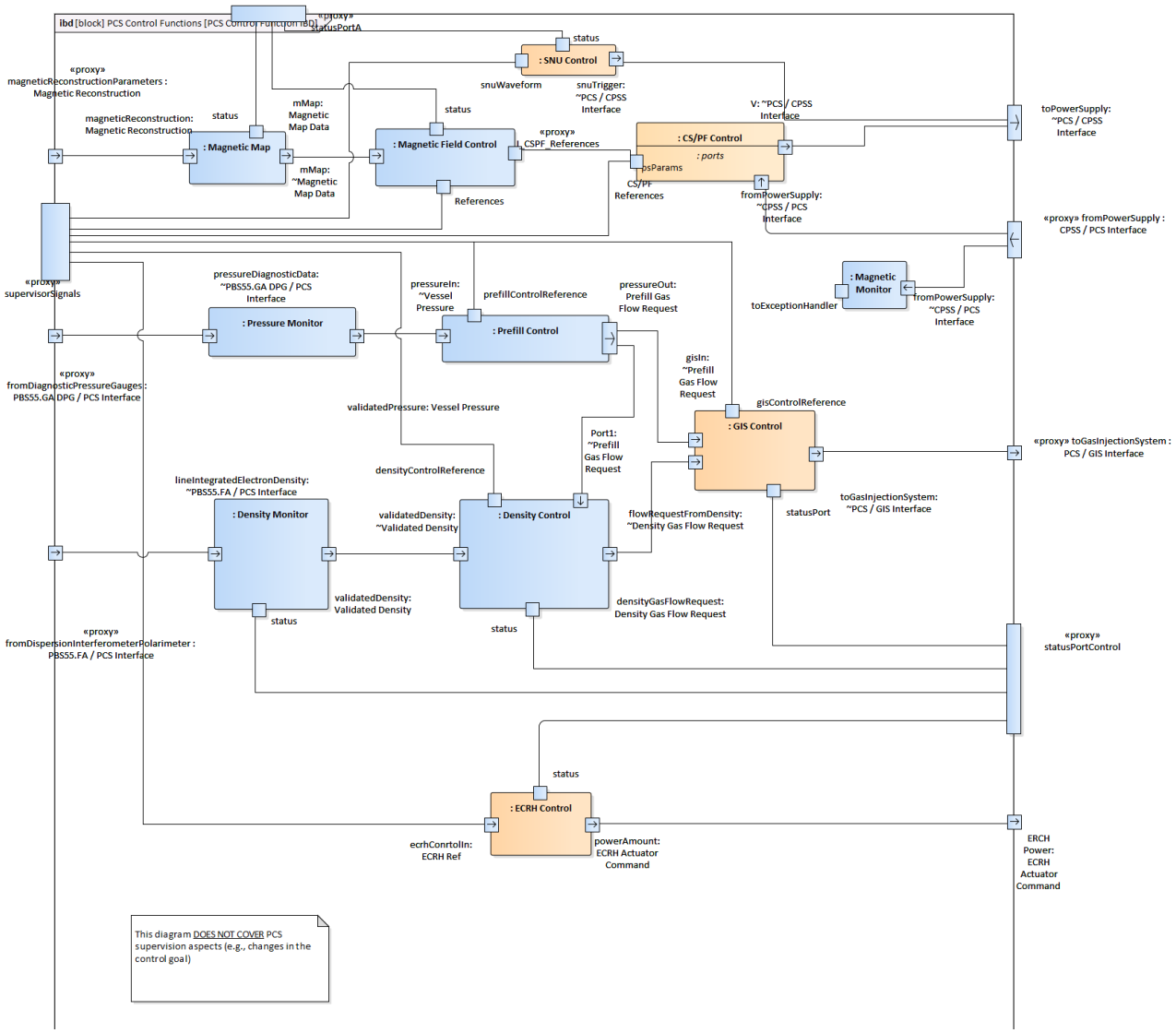

Fig. 4. Simplified version of the *Internal Block Diagram* for the PCS function model.

constructing a graph of dependent requirements, namely the *dependency graph*. The graph can be trivially constructed from the dependency relationship matrix (*DEP*-matrix), since the dependency matrix is the relationship matrix between system requirements that are connected by a dependency link.

To check if the system requirement is satisfied by a block, both the information of the *SAT*-matrix and of the dependency graph are used. The evaluation algorithm performs a depth-first visit of the dependency graph and for each node encountered during the visit we evaluate if for the visited node, there is or is not a $<< satisfy >>$ link in the *SAT*-matrix.

It is worth to notice that all these consistency checks have been automated by using the external scripts that make use of the EA API.

Blocks may have ports that can be used as interfaces to connect them with other components or with other plant systems that are modeled in the *ITER Environmental Model* package. For instance, as depicted in Fig. 5, considering the CS/PF Control function block, the "V" port is connected to the PCS/CPSS interface of the Power Supply, to send commands to coils, while the "fromPowerSupply" port is connected to the CPSS/PCS interface, to receive measurements from the power supplies. The details on available measurements and commands are then available in the power supply interface modeled in the *ITER Environmental Model* package.

The diagram to be used to formalize the relationship is the IBD, where congruent ports, i.e. ports of the same type, can be connected using *Connectors* links between them. Clearly, the connection is logical and it is meant to verify that the implementation of the block is compliant with the interface specification of the corresponding plant system. In practice, systems will communicate by exchanging messages (compliant with the same interface specification) over the SDN.

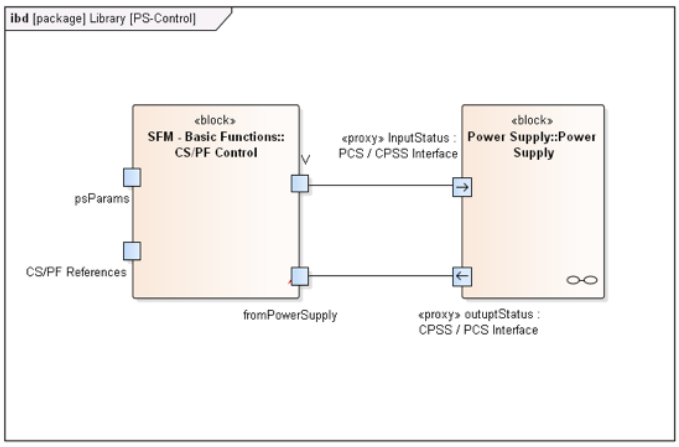

Fig. 5. CS/PF control function to Power Supply plant system connection example.

VI. Simulation Prototypes and Performance Assessment

Since the PCS preliminary design [2], its functions (either for control, for diagnostic data processing or for exception handling) and its architecture have been prototyped in the PCSSP, which is a simulation environment [21] developed in Simulink®. On the long time scale, the use of PCSSP is not limited to the prototyping, and its use is envisaged also to automatically generate code and/or configuration files for the RTF, similarly to what is provided by the MARTe framework [25], [26], or is done for TCV [27].

Referring to the main packages introduced in Section II, all of the relevant information about the Simulink models developed in PCSSP and the corresponding simulation results aimed at assessing the performance requirements, is captured in the *Simulation Model*. More in detail, the *PCSSP Modules* subpackage contains the simulation prototypes of the PCS functions, while the performance assessments are modeled in the *Assessment* subpackage (see Fig. 1).

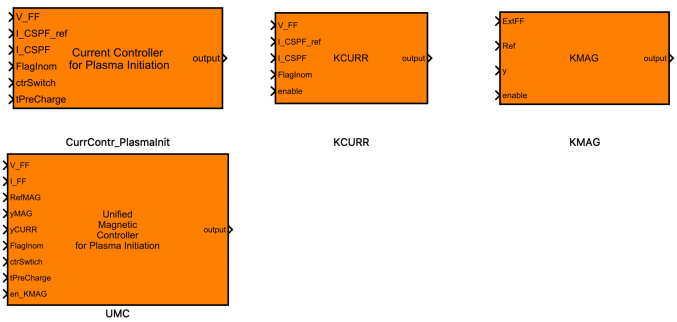

Fig. 6. PCSSP Modules of the magnetic control functions envisaged for FP.

As an example, Fig. 6 shows the PCSSP modules that have been developed as prototypes of the magnetic control functions for the FP phase. In particular, *KCURR* and *CurrContr_PlamsaInit* are two different solutions for the CS/PF Control function shown in Fig. 5, while *KMAG* is the prototype of the controller of the magnetic field in the breakdown region and the *Unified Magnetic Controller* (UMC) is the combination of both *KMAG* and *CurrContr_PlamsaInit*. These blocks are used to assess performance running closed loop simulations, which include models for the actuators, the diagnostic systems and the plasma.

The *PCSSP Module* package contains a representation of each of the PCSSP modules shown in Fig. 6, which can then be mapped to the corresponding PCS functions contained in the *System Function Model* (see Fig. 7, where the artifacts that represents *KCURR* are mapped to the CS/PF Control block), and interface requirements modeled in the *ITER Environmental Model* (see Fig. 8, where *KCURR* I/O is mapped to the PCS/CPSS and CPSS/PCS interfaces). Hence, through the corresponding PCS function blocks, the PCSSP modules are linked to the relevant SYRs. These links allows to assess the fulfillment of each functional requirement relevant for FP by at least one PCSSP prototype.

As introduced in Section III, SYRs are refined into non functional requirements that specify performance goals [22]. Such performance requirements are assessed running PCSSP simulations, whose results are traced in the PCS DB by means of the *testCase* objects. The relationship between the assessment results and the corresponding performance requirements is modeled in the PCS DB using the $<< verify >>$ relationship that links testCase objects to requirements, as shown in the example in Fig. 9. Moreover, the *testCase* object contain links to the simulation data and results stored in the ITER archive (see Fig. 10).

VII. PCS Architectural Components

During the PCS preliminary design, the main architectural components for the PCS architecture have been identified by abstracting the specific PCS functions required for commissioning and FP operation [4]. The simplified block diagram in Fig. 11 models the main PCS architectural components and shows how they can be specialized to implement the required PCS functions.

The architectural components are modeled in the *Architectural Model* which will also trace the configuration parameters that are needed to specialize a general purpose component to implement the needed PCS function. For example, the *KCURR* controller introduced in Section VI can be also realized by configuring the compact controller component. Such a possibility will be modeled in the PCS DB by tracing the CS/PF Control block also with the compact controller artifact, and by attaching to this relationship the corresponding configuration.

VIII. PCS Commissioning Procedures

PCS commissioning procedures will eventually be captured in PCS DB. However, at present, the design team is focused on identifying commissioning requirements through the use of commissioning use-cases. Commissioning use-cases are different from system use-cases because they have different goals. In particular, these are accumulated assurances about correct implementation, while minimising risks of failures and

identifying activities that can be carried out in parallel, thus streamlining the commissioning process.

Commissioning use-cases are also useful for identifying entities (stakeholders or management systems) with which the PCS will need to integrate in some way during the commissioning phase. The full extent of the entities and their roles in PCS commissioning remains to be worked out at ITER. However, they will eventually be actors in commissioning use-cases and interfaces in commissioning procedures.

Currently, commissioning use-cases are captured in PCS DB using the SysML *Use-Case* object and contained in the *Commissioning* package shown in Fig. 1.

There are two stages in the PCS commissioning; system commissioning, where components of the PCS are validated, and integrated commissioning where PCS is linked with other plant systems. Therefore, commissioning use-cases may involve one or more PCS functions and it is useful to keep track of this relationship for two reasons; demonstrating completeness that all PCS functions have a commissioning use-case, and managing engineering changes by identifying those commissioning use-cases that need to be reviewed following a change in a given PCS function. Eventually these verification requirements will be translated to the commissioning procedures.

An example is shown in Fig. 12 where the *CS/PF Control* function is associated to a commissioning use-case on PCS feedback control. This link is indirectly confirmed also through the relationship between requirements, test-cases, and use-cases. Specifically, for the considered example, CS/PF Control satisfies a performance requirement about the magnetic field at breakdown, which is verified by a test-case that, in turn, is used by the same commissioning use-case on which the block was allocated. Checking these types of relationships is useful to understand the design, and to build the test-suites that are necessary to verify PCS functions, according to requirements and commissioning procedures.

Finally, it is planned that PCS DB will eventually provide the ability to perform full impact analyses, by following all relationships mentioned so far.

Conclusions

The design of the ITER PCS involves multiple aspects, from the definition and refinement of the system requirements starting from the very high-level ones, to the assessment of the performance and the definition of the commissioning procedures, including the prototyping of the control functions in the PCSSP simulation environment. The contributions to the design come from many parties and, hence, need to be kept under control by using a system engineering approach. To this aim, the PCS DB has been deployed at ITER to manage and document the overall PCS design process. This paper describes how the PCS DB has been structured and implemented using Enterprise Architect®. Some examples of the contents for first plasma operation have been also presented. Relying on a general structure and on a documented design process, the use of the PCS DB is not limited to the FP phase, but is envisaged also to support the next phases of the staged approach. Moreover, once the PCS will start to operate, the PCS DB could also be used to trace the performance of the real-time system; such information can be exploited to justify the further design choices during the next phases of the staged approach.

Acknowledgement

This work has been partially supported by the RT-CASE project, funded by the Dept. of Electrical Engineering and Information Technology of the University of Naples Federico II, Italy. The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

References

[1] J. Snipes, Y. Gribov, and A. Winter, "Physics requirements for the ITER plasma control system," *Fusion Engineering and Design*, vol. 85, no. 3-4, pp. 461–465, 2010.

[2] J. Snipes *et al.*, "Overview of the preliminary design of the ITER plasma control system," *Nuclear Fusion*, vol. 57, no. 12, p. 125001, 2017.

[3] P. de Vries *et al.*, "Preparing the Plasma Control System final design for ITER first plasma operations," *Fusion Engineering and Design*, vol. 129, pp. 334–340, 2018.

[4] W. Treutterer *et al.*, "Towards a preliminary design of the ITER plasma control system architecture," *Fusion Engineering and Design*, vol. 115, pp. 33–38, 2017.

[5] G. Raupp *et al.*, "Preliminary exception handling analysis for the ITER plasma control system," *Fusion Engineering and Design*, vol. 123, pp. 541–545, 2017.

[6] G. Ambrosino, M. Ariola, G. De Tommasi, A. Pironti, and A. Portone, "Design of the plasma position and shape control in the ITER tokamak using in-vessel coils," *IEEE Transactions on Plasma Science*, vol. 37, no. 7, pp. 1324–1331, 2009.

[7] R. Ambrosino *et al.*, "Design and nonlinear validation of the ITER magnetic control system," in *Proc. of the 2015 IEEE Multi-Conf. Syst. Contr.*, Sydney, Australia, September 2015, pp. 1290–1295.

[8] T. Le Sergent, F. Dormoy, and A. Le Guennec, "Benefits of model based system engineering for avionics systems," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, 2016.

[9] D. Kaslow, B. Ayres, P. Cahill, L. Hart, and R. Yntema, "A Model-Based Systems Engineering (MBSE) approach for defining the behaviors of CubeSats," in *2017 IEEE Aerospace Conference*, 2017, pp. 1–14.

[10] M. Waseem and M. Sadiq, "Application of model-based systems engineering in small satellite conceptual design-A SysML approach," *IEEE Aerospace and Electronic Systems Magazine*, vol. 33, no. 4, pp. 24–34, 2018.

[11] S. Herzig, R. Karban, G. Brack, S. Michaels, F. Dekens, and M. Troy, "Verifying Interfaces and generating interface control documents for the alignment and phasing subsystem of the Thirty Meter Telescope from a system model in SysML," in *Modeling, Systems Engineering, and Project Management for Astronomy VIII*, vol. 10705, 2018, p. 107050V.

[12] M. Arantes, R. Bonnard, A. Mattei, and P. de Saqui-Sannes, "General architecture for data analysis in industry 4.0 using SysML and model based system engineering," in *2018 Annual IEEE International Systems Conference (SysCon)*, 2018, pp. 1–6.

[13] D. Wolff, R. Brown, P. Curson, R. Ellis, T. Galliara, and M. Harris, "Early Lessons From the Application of Systems Engineering at UKAEA (May 2017)," *IEEE Transactions on Plasma Science*, vol. 46, no. 5, pp. 1725–1734, 2018.

[14] S. Chiocchio, E. Martin, P. Barabaschi, H. Bartels, J. How, and W. Spears, "System engineering and configuration management in ITER," *Fusion Engineering and Design*, vol. 82, no. 5-14, pp. 548–554, 2007.

[15] "Enterprise architect," 2019, accessed 07.06.2019. [Online]. Available: https://sparxsystems.com/products/ea/

[16] M. Cinque *et al.*, "Requirements management support for the ITER PCS in view of first plasma operations," *Fusion Engineering and Design*, vol. 146, pp. 447–449, 2019.

[17] "SysML Open Source Project - What is SysML? Who created SysML?" 2019, accessed 07.06.2019. [Online]. Available: https://sysml.org/

[18] A. Winter *et al.*, "Implementation strategy for the ITER plasma control system," *Fusion Engineering and Design*, vol. 96, pp. 720–723, 2015.

[19] B. Bauvir, R. Lange, and A. Winter, "Software Architecture and Design Document for the ITER real-time framework," ITER Organization, Tech. Rep., 2017. [Online]. Available: https://user.iter.org/?uid=PKT5S7
[20] L. Zabeo *et al.*, "Work-flow process from simulation to operation for the Plasma Control System for the ITER first plasma," *Fusion Engineering and Design*, vol. 146, pp. 1446–1449, 2019.
[21] M. Walker *et al.*, "The ITER plasma control system simulation platform," *Fusion Engineering and Design*, vol. 96, pp. 716–719, 2015.
[22] ——, "Assessment of controllers and scenario control performance for ITER first plasma," *Fusion Engineering and Design*, vol. 146, pp. 1853–1857, 2019.
[23] G. Ambrosino *et al.*, "Rapid prototyping of safety system for nuclear risks of the ITER tokamak," *IEEE Transactions on Plasma Science*, vol. 38, no. 7, pp. 1662–1669, 2010.
[24] A. Neto *et al.*, "Conceptual architecture of the plant system controller for the magnetics diagnostic of the ITER tokamak," *Fus. Eng. Des.*, vol. 96–97, pp. 887–890, 2015.
[25] G. De Tommasi, R. Vitelli, L. Boncagni, and A. Neto, "Modeling of MARTe-based real-time applications with SysML," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2407–2415, 2012.
[26] G. Manduchi *et al.*, "Integration of Simulink, MARTe and MDSplus for rapid development of real-time applications," *Fusion engineering and design*, vol. 96, pp. 645–648, 2015.
[27] F. Felici *et al.*, "Development of real-time plasma analysis and control algorithms for the TCV tokamak using Simulink," *Fusion Engineering and Design*, vol. 89, pp. 165–176, 2014.

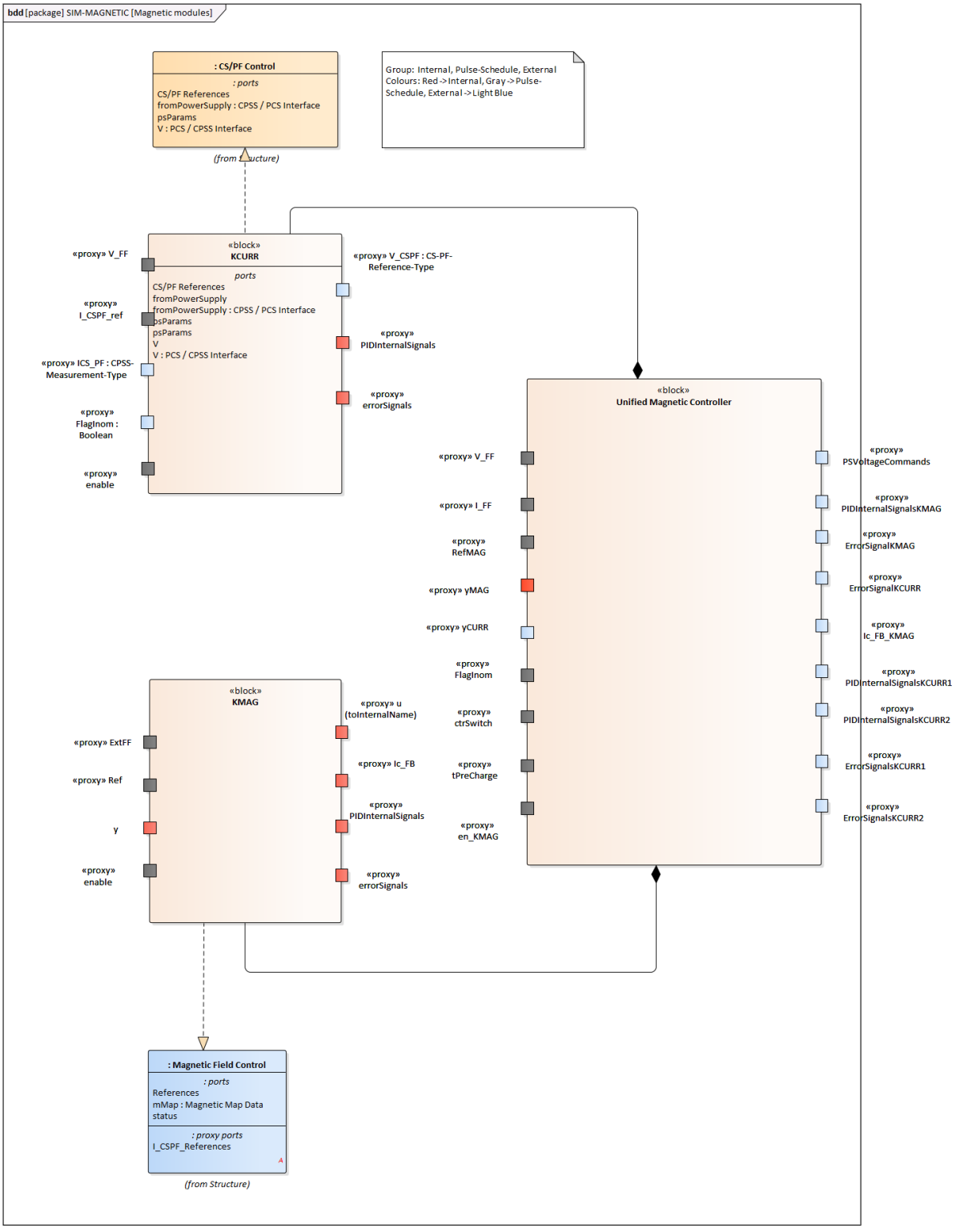

Fig. 7. Example of mapping of the PCSSP module artifacts to the PCS function blocks contained in the *System Function Model* package.

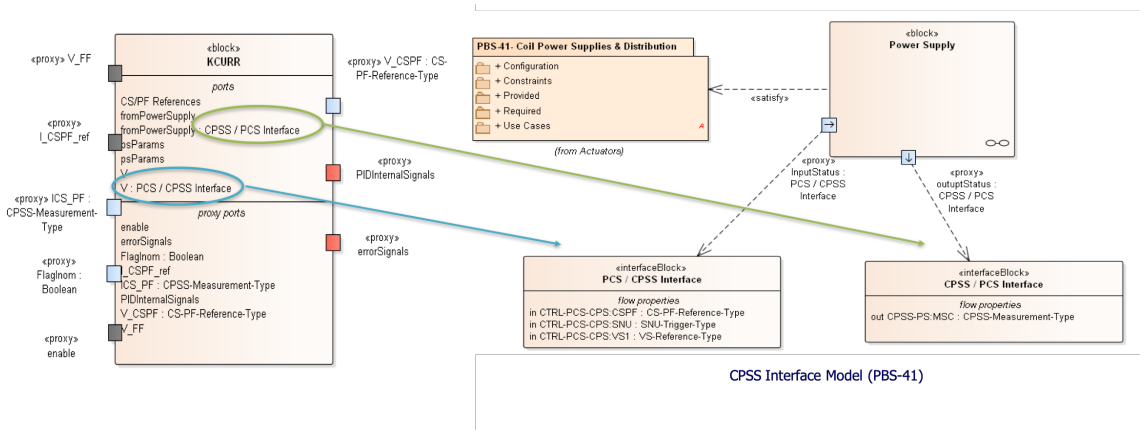

Fig. 8. Example of mapping of the PCSSP module artifacts to the interface model contained in the *ITER Environmental Model* package.

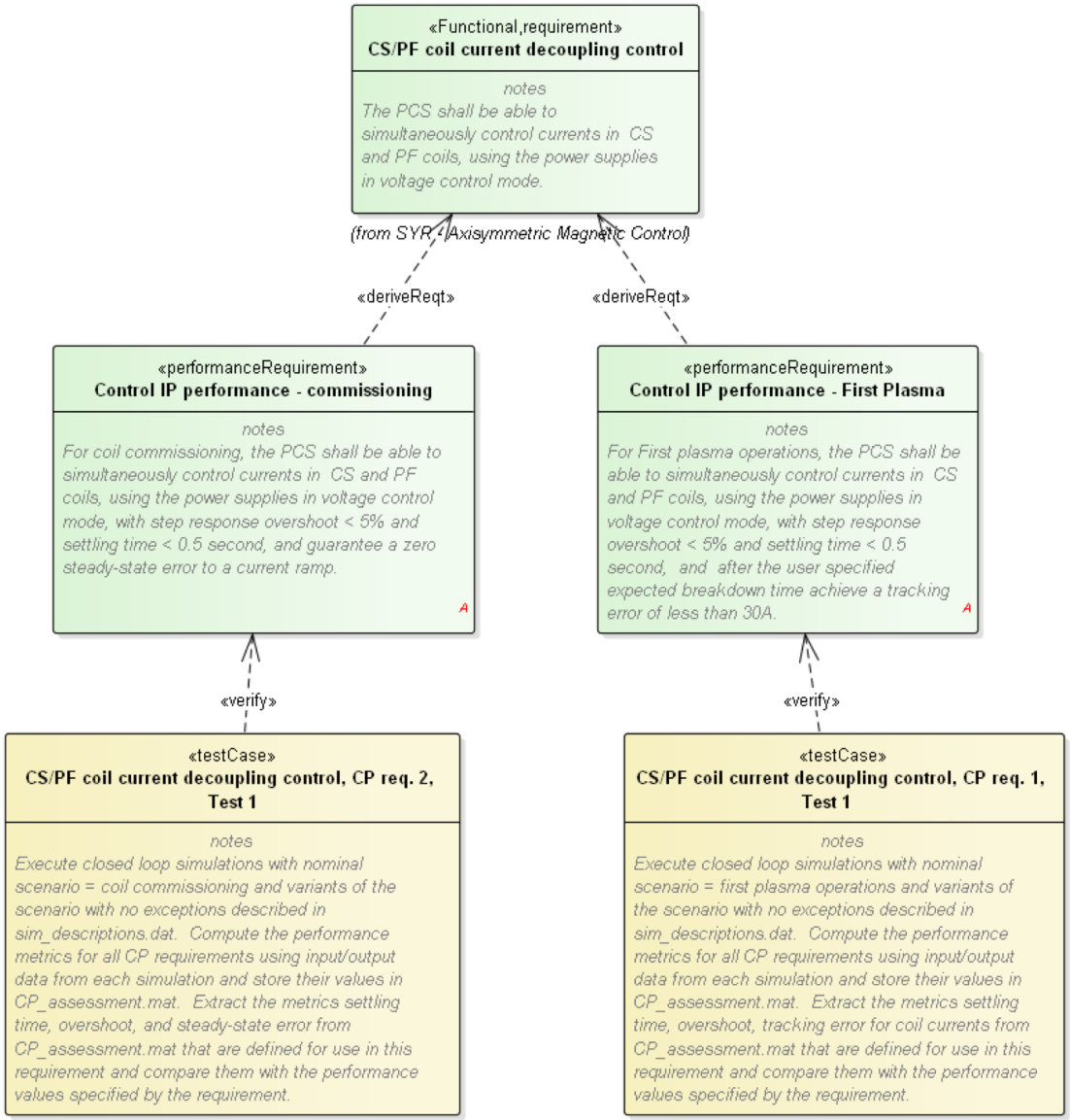

Fig. 9. Relationship between performance requirements and *testCase* objects used to trace the corresponding assessment results.

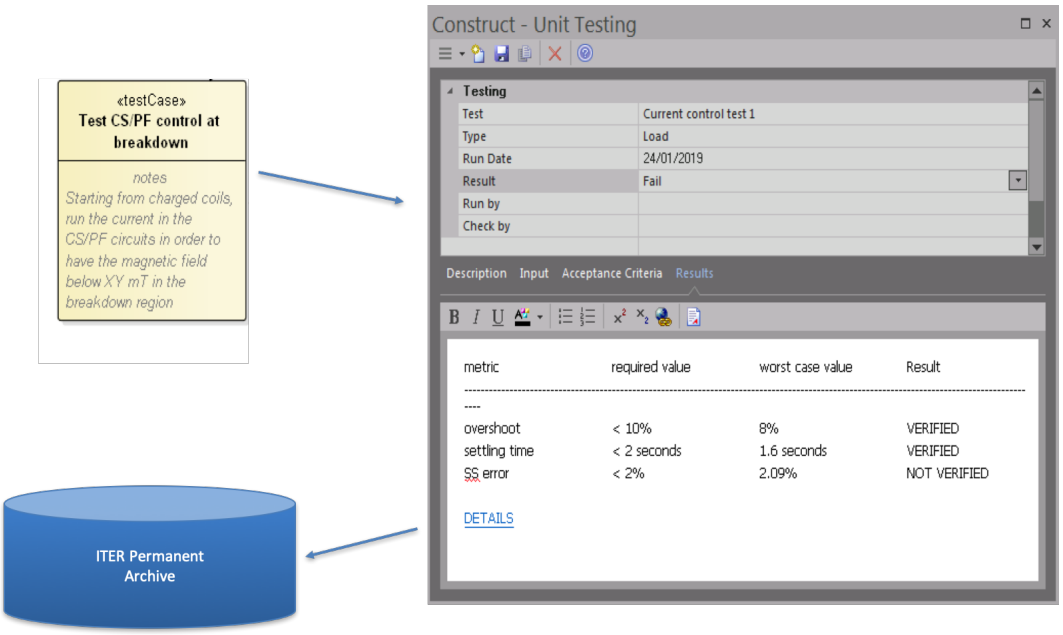

Fig. 10. The simulation results of each assessment can be retrieved using the link stored in the corresponding *testObjects*.

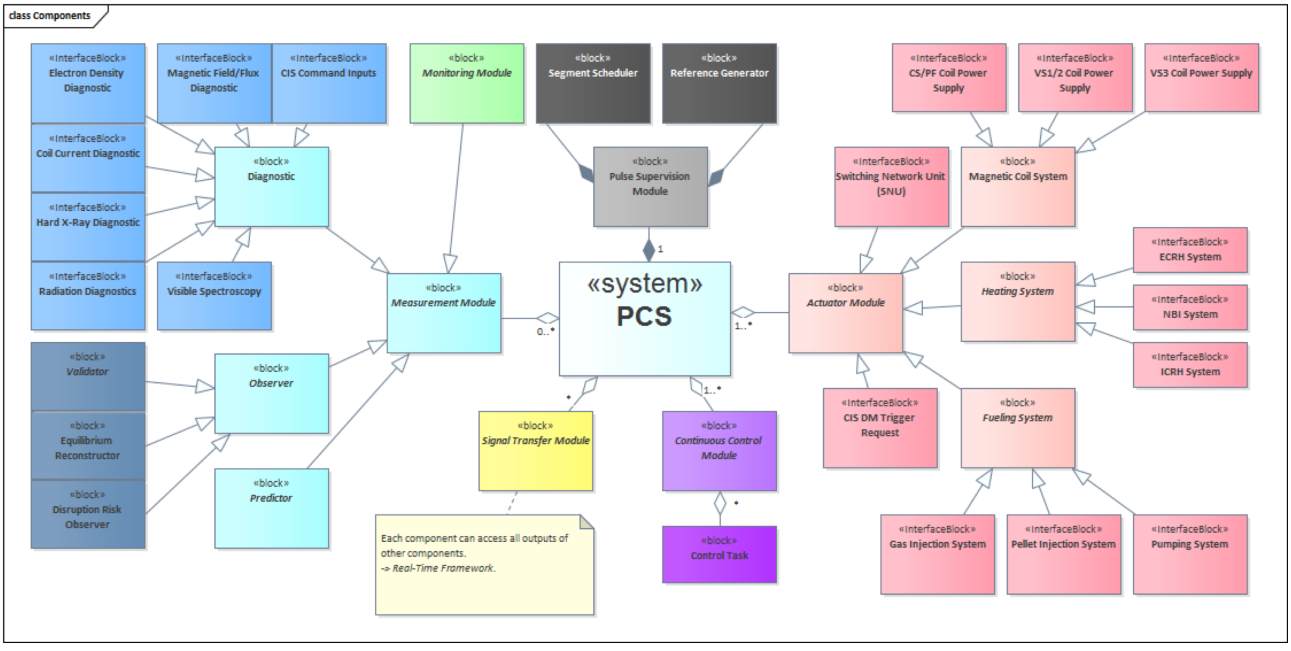

Fig. 11. Simplified block diagram of the PCS architectural components.

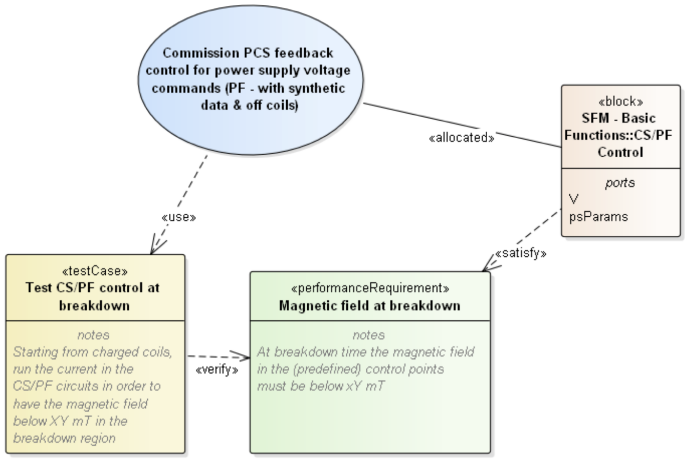

Fig. 12. Example of relationships of a functional block with requirements, use-cases, and test-cases