Workflow for the Assessment of ITER Plasma Control System Design

L. Pangione, T. Ravensbergen, L. Zabeo, P. De Vries, G. De Tommasi, *Senior Member, IEEE*, M. Cinque, and S. Rosiello

***Abstract*— The plasma control system (PCS) for the first nonactive ITER operation phase will require simultaneous active monitoring and control of many continuous and discrete quantities. Considering the unique challenges ITER will face, all the controllers will be integrated and deployed with very little experimental time dedicated to PCS tuning and development. In order to maximize the efficiency of the ITER PCS design, a formal system engineering approach has been adopted. In a simplified way, the design process starts with the definition of the requirements. Functionalities are then designed and developed in order to meet these requirements. As a last step in the design process, it is important to assess that all the designed functionalities meet the associated requirements and that all the requirements are covered. The many different control functions will be designed and implemented in ITER PCS simulation platform (PCSSP) by different designing teams, both internal and external to ITER Organization. Although each team will be responsible for the independent assessment of the modules they deliver, an extra step is, nevertheless, necessary to guarantee that all the modules still continue to work when connected together. Therefore, integrated assessments will be built from independent assessments and will prove the controllers continue to meet the requirements. For this reason, it is necessary to have a unified workflow for the assessments performed by all the different designing teams. In fact, in order to guarantee a smooth integration assessment, it is important that all the assessments follow the same rules, use the same tools, are provided with the correct information, and are performed on the same platform. In this article, we present the proposed assessment workflow for ITER PCS components and some early impressions gathered from assessments of first delivered modules.**

***Index Terms*— Integrated assessments, ITER, plasma control system (PCS).**

I. Introduction

THE ITER plasma control system (PCS) for the first prefusion power operation phase will monitor and control many continuous and discrete plasma and device quantities simultaneously. Considering the uniqueness of the ITER project, where small margin is left for mistakes and whose design challenges span decades, the PCS is designed using a formal and systematic approach.

In this article, we focus on the final part of the design process in which the functionalities are assessed, that is, they are proven to satisfy the corresponding requirements. We present a framework, consisting of a set of rules and a collection of software, to prepare and execute the assessments. This article is organized as follows. Section II introduces the system engineering approach, which characterizes the design of the ITER PCS. In Section III, the focus is on the definition of the assessment phase. In Section IV, the assessment framework is presented and explained in detail with Section V providing some final remarks.

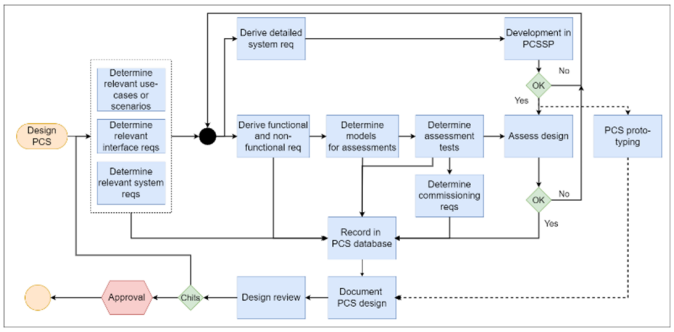

Fig. 1. Flow diagram of the ITER PCS design process.

II. System Engineering Approach

The design process of the ITER PCS has been organized as a relatively linear chain of steps, as shown in Fig. 1. The design begins with the system requirements stating the high-level functionalities the PCS has to accomplish [1], [2]. It then continues with the refinement of these requirements into more accurate functional and nonfunctional requirements, respectively, system requirements (SYRs) and performance requirements (PERFs), so that a functional breakdown is possible.

In our approach, we consider functional requirements, the requirements that define *what* the systems must be able to do, in contrast to nonfunctional requirements, which describe *how* the systems must perform. According to this definition, nonfunctional requirements are measurable [1], [3].

Manuscript received 3 October 2023; revised 11 January 2024; accepted 5 March 2024. This work was supported by ITER Organization. The review of this article was arranged by Senior Editor R. Chapman. *(Corresponding author: L. Pangione.)*

L. Pangione, T. Ravensbergen, L. Zabeo, and P. De Vries are with ITER Organization, 13067 St. Paul Lez Durance Cedex, France (e-mail: Luigi.Pangione@iter.org).

G. De Tommasi, M. Cinque, and S. Rosiello are with the Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università degli Studi di Naples Federico II, 80138 Naples, Italy, and also with Consorzio CREATE, 80125 Naples, Italy.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPS.2024.3376955.

Digital Object Identifier 10.1109/TPS.2024.3376955

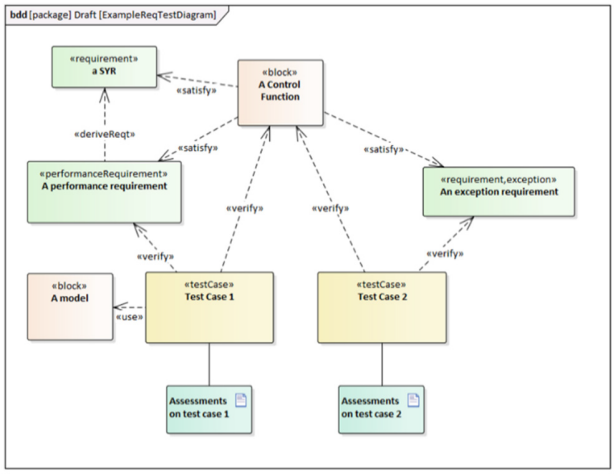

Fig. 2. Example of relationships between different artifacts as produced by the PCSSDB.

Once the functionalities are designed and implemented in the PCS simulation platform (PCSSP) [4], it is necessary to prove that they meet the associated requirements. The assessment process, namely to guarantee through simulations that the associated SYRs and PERFs are satisfied, is, together with commissioning, part of our verification and validation (V&V) process.

Ultimately, the design is documented, defended in a formal review process, and finally approved. The whole process is described in detail in [1].

The designed functionalities are assessed by simulating them in representative challenging situations, called test cases, to ultimately prove that the measurable performance requirements (PERFs) are satisfied.

Clearly, feedback simulations require a model that mimics the behavior of the quantities considered. The accuracy of the model varies as a function of the aspects of the assessed functionality, and therefore, multiple versions of the same model can exist with different performances and final scopes. Nevertheless, the careful selection of the model is an important part of the assessment and is, therefore, a crucial element of the PCS design itself.

All the artifacts of the process described above, i.e., requirements, functional blocks, models, test cases, and their results, are put into the PCS design database (PCSDB) [1], which is a system engineering repository to store design artifacts in the form of SysML [5] elements and diagrams. The PCSDB is implemented using Enterprise Architect [6], a commercial software that fully supports the system engineering approach and includes SysML among the available modeling languages. By linking properly the artifacts, it is possible to have a complete traceability of each of the designed functionalities. Fig. 2 shows a very simplified example of the graphical output extracted from the PCSDB.

In particular, it is possible to see how the functional block "a control function" (pale orange) satisfies a (functional) requirement "an SYR," a (nonfunctional) performance requirement "a performance requirement," and a (nonfunctional) requirement "an exception requirement" necessary to properly handle an exception. The requirements blocks are shown in pale green rectangles. Nonfunctional requirements are measurable, and it is, therefore, possible to verify them in test cases, represented by pale yellow rectangles.

It is interesting to note that each test case is associated with a one-to-one relationship to its assessment. This represents the valid assessment performed to be included in the final design. Moreover, the example in Fig. 2 highlights that the test case "Test Case 1" makes use of a specific functional block representing the model "a model" (pale orange). This is extremely important to create a logical link between the assessment process and the commissioning process, as will be better explained later in Section V.

It is also worth noticing that we decided not to include elementary unit tests concerning the implementation of the designed functionalities in our design process. This is because our SYRs and PERFs do not necessary reach the level of details required for their code implementation. We still perform unit tests on the designed functionalities, but we simply do not trace them back into the PCSDB.

To better manage the design process, all the PCS functionalities are organized into independent tasks, following a reasonable level of decoupling of the physical behaviors. For example, magnetic control, fueling control, error field control, and ELM control belong to different tasks, and, based on the current knowledge, they are known to be decoupled in a first iteration of the design process. Each task is assigned to a specific PCS design team either internal or external to ITER Organization, and each team is fully responsible for the design of the envisaged functionalities. As explained earlier, the design process involves the definition of the requirements, both SYRs and PERFs, the definition of the test cases, and, ultimately, also the execution of the corresponding assessments. These assessments can involve a number of designed functionalities belonging to the same task.

Since all the functional blocks are assessed independently within the scope of their tasks, this is not yet sufficient to guarantee that they will meet the requirements when they will act simultaneously. In fact, on the real machine, most of the PCS functionalities will not be perfectly decoupled, since they are acting on the same physics phenomena by using the same actuators. For this reason, a separate task, responsible for the integration of the functionalities and their assessment, has been established.

III. Design Assessment

It is evident that the integrated assessments need to rely on the quality of the assessments produced by the single tasks.

It is possible to imagine the assessments as a pyramid, shown in Fig. 3, where the width represents the number of tests, while the height represents the level of integration. At a low level, where a single functionality is assessed, it is possible to perform a large number of simulations, but as the integration increases, and the simulations become more complex and computationally more expensive, the number of assessments is reduced. In this picture, the integrated assessments remain of a manageable number, while still being exhaustive.

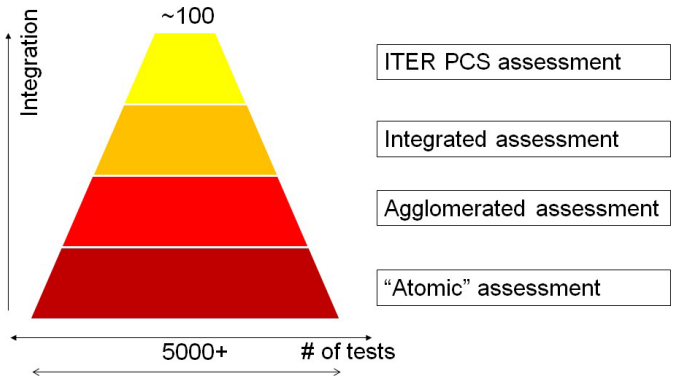

Fig. 3. Representation of the different levels of assessments as defined for the ITER PCS design.

For the design of the ITER PCS, we decided to classify the assessments into four categories.

1) *Atomic Assessments:* Here, only a single functionality is assessed.
2) *Agglomerated Assessments:* Here, multiple functionalities belonging to the same task are assessed.
3) *Integrated Assessments:* Here, multiple functionalities belonging to different tasks are assessed.
4) *ITER PCS Assessments:* Here, we aim to assess the whole PCS, or at least most of its functionalities.

As is clear from the picture, each level of assessment must rely on the quality of the level below. This is particularly important when the responsibility is handed over from the design team to those responsible for the integration (namely, from “agglomerated assessment” to “integrated assessment”).

Obviously, the pyramid in Fig. 3 represents a simplification of reality. In fact, due to the physical coupling described above, it is not always possible to properly assess a single functionality in isolation. It is, therefore, not always possible to have an “atomic assessment” below an “agglomerated assessment.” In addition, the boundaries between these layers are not always well delimited and can become blurry. In fact, it is possible that an “integrated assessment” fails even if all the associated “agglomerated assessments” and “atomic assessments” have passed. This can happen for various reasons, for example, because different models have been used during the simulations, or because the “integrated assessment” challenges the functionalities in a new way not tested before, or even, in the worst case, that the functionalities present a problem not highlighted before. These example situations require a certain level of cooperation between teams, so that the level of assessment becomes blurrier. A further important consideration is that models used for integrated assessments can be derived in different ways; for example, in some cases, it will be possible to have a single model that covers multiple behaviors, while in other cases, it will be necessary to connect together multiple models each covering a single aspect.

Ultimately, the pyramid will present in reality multiple peaks, because the entire PCS cannot necessarily be tested simultaneously. Instead, the whole functionalities are assessed in a few simulations covering most of them.

IV. Assessment Framework

Considering the heterogeneity of the PCS design teams and the high number of assessments involved, it is clear that a formal framework to perform the assessments is needed.

This framework must provide a unique way to perform all the assessments independently of their level in the pyramid and from the team that executes them.

In particular, it is extremely important that all the assessments are written using the same tool in the same way. This reduces ambiguity and increases reproducibility and readability. In fact, even if assessments are not meant to be executed constantly and frequently, it should be always possible to re-execute them while retaining a historical trace. This can be useful, for example, if an improvement of the design is made or if a further investigation is needed at a second stage. Moreover, the assessments must be executed using a single command invoked only when needed and in batch mode, without human intervention during their execution.

Finally, the framework must be able to store and retrieve the data produced by an assessment if required. Even if the ultimate result of an assessment can be either pass or fail, there is extra information, such as internal state values, waveforms, and commands produced during the simulation, which are important to store. These values are not part of the design assessment and are, therefore, not supposed to be stored in the PCSDB, but they should be accessible for later analysis.

As explained earlier, the proposed and implemented solution to assess all the PCS functionalities is the ITER PCS assessment framework. This consists of a set of rules and a collection of software packages, which allow all the developers involved in the design of the PCS to assess the designed functionalities in a simple and uniform manner, while keeping the level of traceability required by the system engineering approach.

At the base of the PCS, assessment framework is an in-house MATLAB© custom plugin called the `PCSDB_assessment_plugin`, which extends the functionalities of the standard MATLAB Test-Runner by adding specific features to the runner itself. The MATLAB Unit Test framework defines the TestRunner class as the class needed to run the tests which the user can define as specialization of the superclass TestCase. Comparison of values is provided by standardized verification functions implemented in the MATLAB verification package.

In particular, when invoked, the plugin is responsible for the following.

1) checking that the assessments are correctly prepared;
2) building the output structure of the filesystem to store generated results and data;
3) executing the appropriate assessments;
4) providing a light form of sandboxing to isolate the execution of each assessment and of the data generated;
5) capturing the results generated using the MATLAB qualifications package;
6) generating the reports.

The process adopted to execute the assessment is summarized in Fig. 4. The process starts from the left-hand side of the figure with the designer working on specific functionalities. As MATLAB and PCSSP support both Windows and Linux

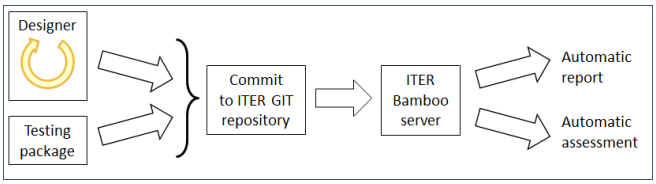

Fig. 4. Flow diagram representing the adopted assessment process.

platforms, this can be done at the designer teams' convenience. Nevertheless, ITER provides on request access to its Linux cluster where all the necessary software is already available. Once the design of some functionalities is completed, the design team adds to its GIT [7] commit a testing package containing all the instructions necessary to execute the assessments.

The testing package consists of two parts. The first is the PCSSP code implementing each assessment, for which the main file is a MATLAB class, which extends the default MATLAB `TestCase` class, and which contains a method for executing each assessment. Naturally, the code will also make use of additional functionalities implemented in separate files, such as MATLAB scripts/functions and Simulink© files. The assessment's methods must follow a set of rules and must be tagged as "assessment." The methods can also be tagged with an additional task-specific string value to allow the framework to restrict the execution of the assessments.

The second part of the testing package is a cover sheet supporting each assessment. This cover sheet contains extra information to facilitate the readability of the simulation code. In particular, the cover sheet contains the following:

1) inputs and outputs of the assessment;
2) the assessment's prerequisites, i.e., what has to be true for the assessment to be valid;
3) the names of the variables being assessed;
4) a list of files used during the simulation;
5) the filename and the data saved at the end of the assessment.

This information is required to facilitate the review of the assessment by an independent audit. In order to keep the framework light, only simple checks are performed on the content of the test cover sheets. Whenever these checks fail, for example, the test cover sheet is missing or incomplete, the assessment is performed anyway, but the result, either passed of failed, is marked as "tainted." The "tainted" label does not invalidate the assessment itself, but shows that some information is missing.

Once the code for the assessment is stored in the ITER PCS GIT repository, it is not automatically executed together with the rest of the unit tests. In fact, the assessment process is only needed to prove that the modules provided by the design teams meet the requirements at the time of the deliverable. The responsibility of these modules is then handed over to ITER, which is then responsible for their implementation and maintenance.

The execution of the assessments is triggered by tagging the GIT commit with a specific and unique tag, set by the designer when the code is completed. To reduce the possibility of mistakes, the tag is automatically generated using a custom GIT command. The assessment tag allows the ITER continuous integration server to invoke the assessment pipeline in which the `PCSDB_assessment_plugin` is executed. The first report generated by the plugin is a global assessment output file, which contains all the assessment results, the GIT commit hash to identify the assessed code inside the GIT repository, the date of the assessment, and the user name of the user who launched the assessment. All this information is then manually imported into the PCSDB, by means of a dedicated Enterprise Architect plugin, to "officially" include the assessment results into the design. It is important to observe that this output file is automatically generated by the framework without any intervention of the design team. This guarantees homogeneity in its structure and content.

A further report, containing statistics on the execution environment, the figures generated and the output sent to the MABLAB console, is also automatically generated at the end of each execution. This information, together with the data saved, can be used later for further analysis.

V. Conclusion

The assessment framework being implemented, and described here, has many advantages. It guarantees the level of traceability required by the adopted system engineering approach. In fact, once the results are imported in the PCSDB, during a design review, it is possible to uniquely link the assessment results (passed or failed) with the execution of the assessments (job number), with the actual code used during the assessment (GIT hash), and with the additional information generated (reports and saved data). There is, moreover, a clear distinction among the design environment, which can be any machine, the environment, which stores the designed code, the GIT repository, and the environment, which executes the assessments and stores its results (the continuous integration server). Note that this advantage does not limit the freedom of the designer to being constantly connected with ITER network. In fact, since PCSSP is available on different platforms, the whole process can be performed on many other machines. Nevertheless, only results obtained by running on the continuous integration server are imported into the PCSDB. As such, the assessments are always executed on the same platform, starting from scratch and following the same steps every time. This is beneficial to guarantee the correctness and the reproducibility of the results.

The assessments are executed only when needed, avoiding unnecessary waste of resources and, more importantly, avoiding confusion on the "final design version" of the code, which shall be considered for the design of ITER's PCS. An example of the benefits, which the traceability provided by the combined use of the assessment framework and the PCSDB brings, is reported in Fig. 5.

Fig. 5 shows how the commissioning process can benefit from the assessment process, as already anticipated early in this article. In fact, some commissioning procedures will be the repetition of some assessments, for example, in the case we need to validate a modeled assumption. Here, commissioning procedure number 101 is directly connected with the test case

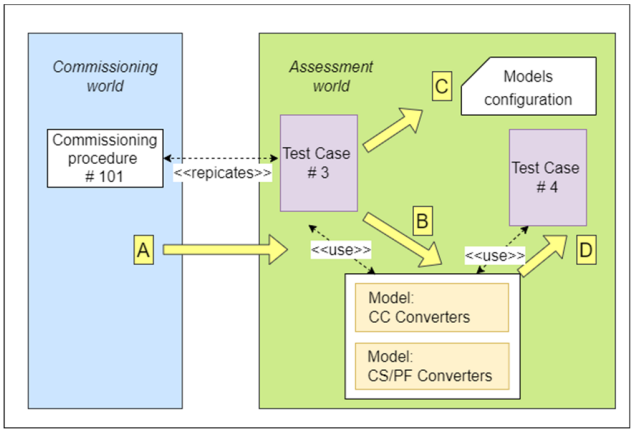

Fig. 5. Example of benefits for the ITER PCS commissioning generated by combined use of the assessment framework and the PCSDB.

"Test Case 3." This makes use of two models, named "correction coil (CC) converters" and "central solenoid and poloidal field (CS/PF) converters," which represent, respectively, the magnetic behavior of the CC converters and of the CS/PF coils converter. These models are also used in the test case "Test Case 4." If the commissioning procedure invalidates one model, we can easily trace back to all the simulations (in this case only "Test Case 4"), depending on the same model. Since the test case will have an associated assessment, with the whole chain of information, it will be easy to pin down all the affected functionalities and assess the impact of the model mismatch.

At the time of writing, the first set of "atomic" and "agglomerated" assessments have been delivered and assessed using the proposed framework. First statistics show "atomic" assessments execution times that span from the few minutes to several hours for the most detailed cases. Because of the variety of the designing teams involved and to the number of tests, the framework has already evolved and reached the maturity level required for the "integrated assessment" phase.

ACKNOWLEDGMENT

The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

REFERENCES

[1] M. Cinque et al., "Management of the ITER PCS design using a system-engineering approach," *IEEE Trans. Plasma Sci.*, vol. 48, no. 6, pp. 1768–1778, Jun. 2020, doi: 10.1109/TPS.2019.2945715.

[2] G. De Tommasi et al., "System-engineering approach for the ITER PCS design: The correction coils current controller case study," *Fusion Eng. Design*, vol. 185, Dec. 2022, Art. no. 113317.

[3] N. Hutchison, *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, document Version 2.9, Stevens Inst. Technol. Syst. Eng. Res. Center, Int. Council Syst. Eng., Inst. Elect. Electron. Engineers Syst. Council, Trustees Stevens Inst. Technol., Hoboken, NJ, USA, 2023. [Online]. Available: https://www.sebokwiki.org

[4] T. Ravensbergen et al., "Strategy towards model-based design and testing of the ITER plasma control system," *Fusion Eng. Design*, vol. 188, Mar. 2023, Art. no. 113440. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920379623000248

[5] *SysML Open Source Project-What is SysML? Who Created SysML?* Accessed: Sep. 7, 2023. [Online]. Available: https://sysml.org/

[6] *Enterprise Architect*. Accessed: Sep. 7, 2023. [Online]. Available: https://sparxsystems.com/products/ea/

[7] *GIT Reference Manual*. Accessed: Mar. 6, 2024. [Online]. Available: https://git-scm.com/docs