SDD TOOLKIT: ITER CODAC PLATFORM FOR CONFIGURATION AND DEVELOPMENT

L.Abadie, F.Di.Maio, D.Stepanov, A.Wallander,
ITER Organization, Route de Vinon sur Verdon, 13115 St. Paul lez Durance, France,
K.Bandaru, H.Deshmukh, P.Nanware, R.Patel, TCS, Mumbai, India
G. Darcourt, A.Mariage, Sopra Group, Aix-en-Provence, France,
A.Zagar, Cosylab, Slovenia

Abstract

ITER will consist of roughly 200 plant systems I&C (in total millions of variables) delivered in kind which need to be integrated into the ITER control infrastructure. To integrate them in a smooth way, CODAC team releases every year the Core Software environment which consists of many applications. This paper focuses on the self description data toolkit implementation, a fully home-made ITER product. The SDD model has been designed with Hibernate/Spring to provide required information to generate configuration files for CODAC services such as archiving, EPICS, alarm, SDN, basic HMIs, etc. Users enter their configuration data via GUIs based on web application and Eclipse. Snapshots of I&C projects can be dumped to XML. Different levels of validation corresponding to various stages of development have been implemented: it enables during integration, verification that I&C projects are compliant with our standards. The development of I&C projects continues with Maven utilities. In 2012, a new Eclipse perspective has been developed to allow user to develop codes, to start their projects, to develop new HMIs, to retrofit their data in SDD database and to checkout/commit from/to SVN.

INTRODUCTION

SDD toolkit has been developed to promote PCDH standards [1] as CODAC will have to integrate around 200 systems produced by different institutes and companies around the world. Standardization being a key criterion for success, it is important to clearly set a development framework for I&C designers to minimize heterogeneous code and implementation. We also try to automate as much as possible code generation to avoid human errors. The toolkit is a home-made product and is part of Core System infrastructure [2]. It is based on a relational database and uses modern technology such as Eclipse [3], Hibernate [4] and Spring [5]. The toolkit consists of many tools which will be described in the next sections.

CHALLENGES

In this section, we describe the main challenges that SDD is trying to address.

EPICS and CSS

CODAC uses EPICS [6] as a conventional control system framework. EPICS is based on records (or variables) to describe the system. A record has a type (e.g. ai – analog input) and a set of static fields. We expect to have millions of EPICS variables. Each variable requires a unique name. Moreover EPICS variables can be viewed as a graph where variables can be linked with each other (aka as record plumbing). EPICS community has developed CSS [7], which consists of a set of applications such as HMIs framework development, alarm handling and archiving system.

I/O Modules

PCDH recommends a set of I/O modules to be used. Each I/O module requires an EPICS device support which allows controlling the board via EPICS.

SDN

Fusion devices require a fast feedback control loop with plasma control. The data is getting transported on another network called SDN (Synchronous Data Network). It is based on UDP multicast and uses the concept of SDN topics for data exchange. A SDN topic is a C-structure which has two parts (metadata and data). Then programs can subscribed and/or publish SDN topics.

Validation

As we will receive systems developed by different people, it is important that we validate the project. Checking that the deliverable is compatible with CODAC standards is essential to a smooth integration, e.g. the variable is unique, links between EPICS variables are valid, SDN topics are published once.

SDD MODEL

The main purpose of the model is to describe the required information so that we can generate configuration files for the different services and make necessary validations.
Figure 1 shows the architecture layout of the SDD toolkit. Each component will be described in the next subsections.

ISBN 978-3-95450-139-7

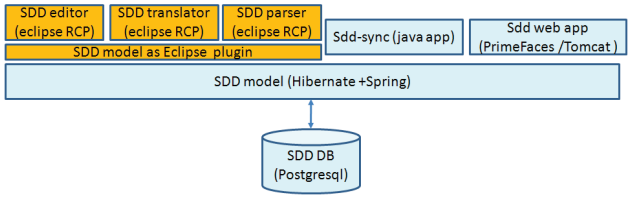

Figure 1: SDD toolkit architecture.

Database Centric

The core component of the toolkit is the database and its library access. The schema uses a relationship model and describes necessary information to generate configuration files.

The main key points of the database schema are the following two aspects:

- Static data which consists of metadata used for validation: e.g. declaration of EPICS record type and set of fields, metadata required to describe alarm, list of I/O modules and its description of EPICS interface.
- Concept of variables: a variable has a type which can be EPICS, SDN, etc. It has a unique name. Depending on the type, we can impose or not a structure.

We use PostregSQL [8] to implement the database level.

Three Different Views

The SDD model uses three views to describe the system as shown in Figure 2.

The global container is called the **I&C project**. It has a name and a version. An I&C project has three views:

- **Physical** which describes the list of components, signals, controllers and I/O modules and location inside cubicles/chassis
- **Functional** which describes the list of variables and associated I&C functions
- **Control** which describes the relationship between variables and controllers.

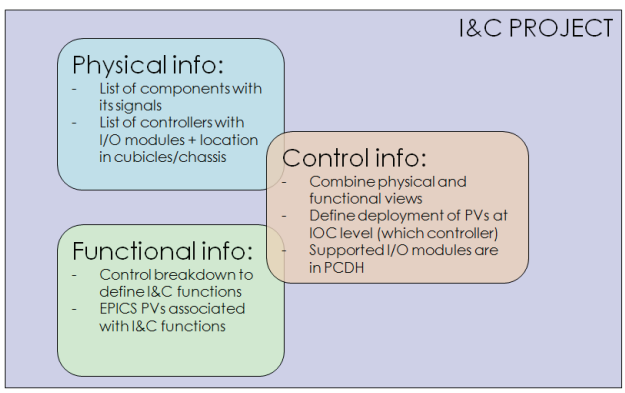

Figure 2: The different views of the SDD model.

Data Access

We have implemented a data access library to interact with SDD database, using Hibernate as ORM (Object relational model) framework, Spring to manage transactions.

We also developed mass import/export mechanism based on Excel.

The data access offers the necessary methods to save and query data. Due to the amount of data to be saved and loaded, we developed the toolkit using lazy saving and loading.

We implemented three levels of validations:

- Minimum, to warrantee that the data is consistent (e.g. unique name of variables)
- Medium, all information to generate configuration files have been filled
- Full validation, the deliverable is complete and compliant with CODAC standards.

SDD migration

As we have more and more users, we need to provide a smooth upgrade path. Whenever, there is a SDD DB schema change, we provide a migration script to allow migrating data from one version of SDD to another one.

SDD Liefecycle

Another key point of the SDD toolkit is the duality local versus central SDD DB. It is important to highlight that all deliverables need to be submitted to the central DB as shown in Figure 3. Next section explains which tool is responsible for communication between central and local databases. By submitting the deliverables to the SDD central DB, it allows monitoring of the progress of the deliverables.

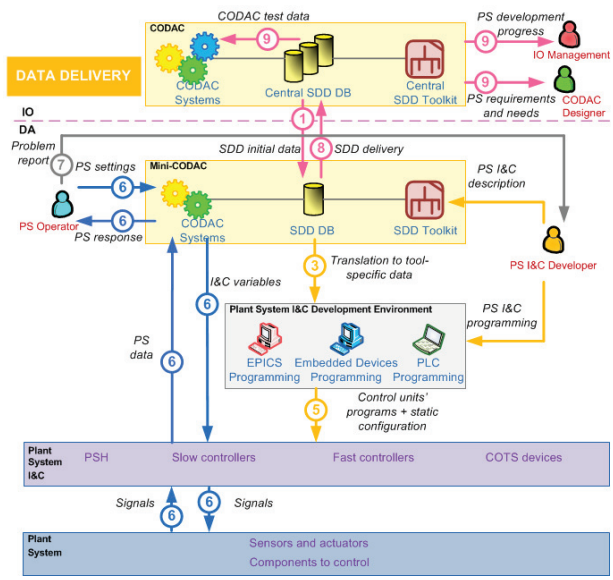

Figure 3: SDD Lifecycle.

COMPONENTS

SDD Webapp

We developed a web application to allow users to edit and to browse SDD information as shown in Fig. 4. It

ISBN 978-3-95450-139-7

uses Prime Faces [9] as a widget framework and Tomcat as servlet container.

The main advantage of web applications is it does not require any software to install, just a web browser.

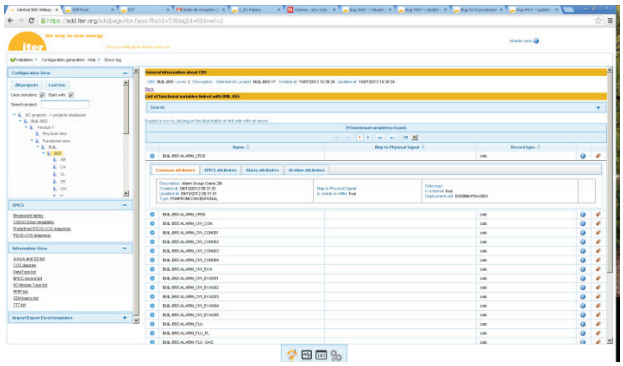

Figure 4: SDD web application: variable view.

SDD Editor

Core system is based on RHEL. Allowing an efficient framework for development is important. We developed SDD Editor (Fig. 5), based on Eclipse RCP technology. It is plug-in-based architecture – one can integrate third-party plug-in - so it is very modular. From CODAC Core system v4, we start to integrate the SNL plugin, developed by the CSS community, it allows developing SNL code. We also included the BOY editor to allow HMI development and eclipse CDT for developing C/C++ code.

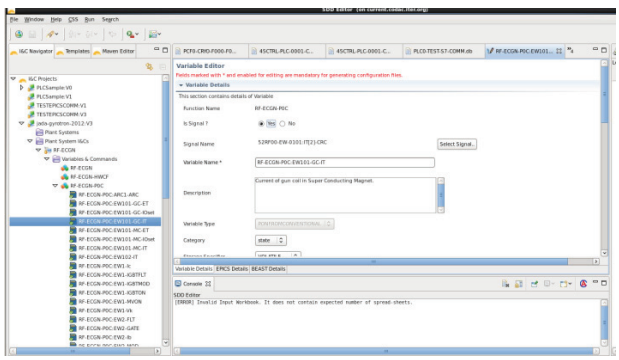

Figure 5: Example of SDD editor panel.

SDD Translator and Parser

Another key component of SDD toolkit is the translator which uses data in the database to generate configuration files for the different services. It is based on JAVA and uses Velocity [10] to define the templates for the different platforms (EPICS, alarm, archive, etc).

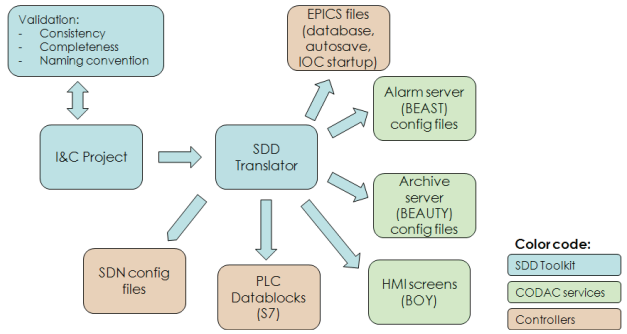

Figure 6: SDD generated files.

Using the information stored in SDD, the translator generates the following configuration files (Fig. 6):

- EPICS database files, start-up files
- Archive configuration
- Alarm configuration
- A set of HMI screens
- PLC datablocks (interfaces) and a VAT table for STEP 7 PLCs.
- SDN configuration files (SDN Topic headers, SDN topic multicast mapping, SDN configuration file for the monitoring node).

We also developed a parser, to retrofit EPICS files. It uses ANTLR [11] as a parser. It is a useful tool as it minimizes the development cycle: when debugging a project, it is often faster to modify the generated files and then integrate them in SDD DB.

Maven Editor

Maven framework is used in Core System for building and running the different services. The main advantage of maven is to hide the complexity of building heteregenous services (e.g. Makefile of SDN application is different as writing maven of EPICS application).

Also maven is used to enforce a consistent project structure and minimizing editing of makefiles.

In Core system v4, we developed a new Eclipse plug-in called maven editor (Fig. 7) and integrated it with SDD editor.

As a consequence, one can create and edit a project, generate configuration file, develop codes and then start, stop services.

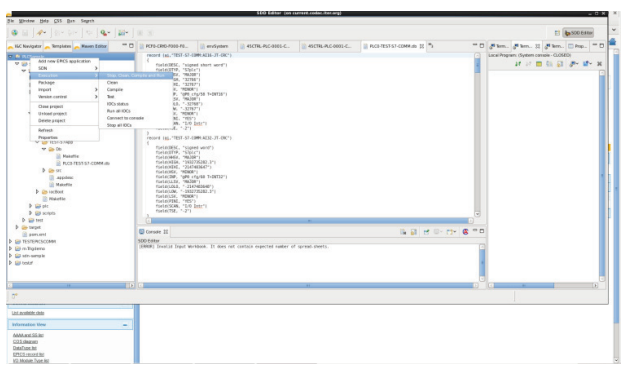

Figure 7: Maven Editor view.

ISBN 978-3-95450-139-7

Integration with SVN [12] has also been made. In recent version of Core system one can commit his/her project, check-out from SVN projects.

SDD-sync

The final component we describe is sdd-sync. SDD-sync has two main roles:

1) Create a snapshot of your project into XML
2) Communicate with Central SDD.

Sdd-sync is using JAXB [13] to parse XML files. We always encourage developers to submit their I&C projects to the central SDD DB, it allows sharing of information and structure definition. Also one can import/export other projects and integrate it in his/her local system as one can see in Fig. 8 and Fig. 9.

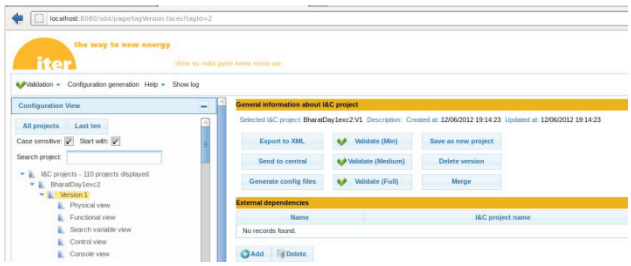

Figure 8: Export to XML and Send to central features.

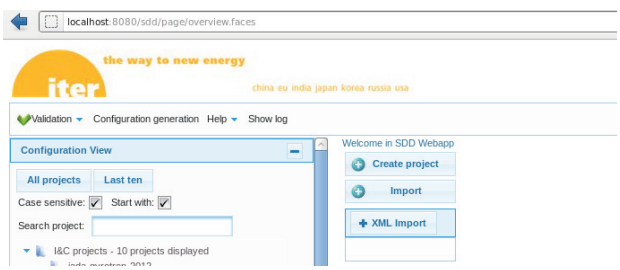

Figure 9: Import XML projects.

CONCLUSIONS

The development of SDD toolkit started three years ago from scratch. Now the SDD toolkit is becoming more stable and a mature product.

Current developments are now mainly focused on the support for remote execution especially when dealing with fast controllers. We are currently working on improving the validations based on users' feedback and starting to improve the central SDD DB to be ready to receive all the different systems and merge them.

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

REFERENCES

[1] CODAC Plant Control Design Handbook http://www.iter.org/org/team/chd/cid/codac/plantcontrolhandbook
[2] F. Di Maio et al., "CODAC Core System, the ITER software distribution for I&C", Proc. of ICALEPCS 2013, San Francisco, http://jacow.org
[3] Eclipse web site : http://www.eclipse.org
[4] Hibernate http://www.hibernate.org
[5] Spring http://www.springsource.org
[6] EPICS http://www.aps.anl.gov/epics
[7] CSS http://controlsystemstudio.github.io
[8] Postgresql http://www.postgresql.org
[9] Primefaces http://primefaces.org
[10] Velocity http://velocity.apache.org
[11] ANTLR http://www.antlr.org
[12] SVN http://subversion.apache.org
[13] JAXB https://jaxb.java.net

ISBN 978-3-95450-139-7