FLIGHT SOFTWARE DEVELOPMENT FOR THE CHEOPS INSTRUMENT WITH THE CORDET FRAMEWORK

V.Cechticky[1], R. Ottensamer[2], and A. Pasetti[3]

[1]*P&P Software GmbH, High Tech Center, Tägerwilen, 8274, Switzerland*
[2]*University of Vienna, Türkenschanzstr. 17, Vienna, 1180, Austria*
[3]*P&P Software GmbH, High Tech Center, Tägerwilen, 8274, Switzerland*

ABSTRACT

CHEOPS is an ESA S-class mission dedicated to the precise measurement of radii of already known exoplanets using ultra-high precision photometry. The instrument flight software controlling the instrument and handling the science data is developed by the University of Vienna using the CORDET Framework offered by P&P Software GmbH. The CORDET Framework provides a generic software infrastructure for PUS-based applications. This paper describes how the framework is used for the CHEOPS application software to provide a consistent solution for to the communication and control services, event handling and FDIR procedures. This approach is innovative in four respects: (a) it is a true third-party reuse; (b) re-use is done at specification, validation and code level; (c) the re-usable assets and their qualification data package are entirely open-source; (d) re-use is based on call-back with the application developer providing functions which are called by the reusable architecture.

Key words: Reusability; CHEOPS; CORDET.

1. INTRODUCTION

The CHaracterizing ExOPlanet Satellite (CHEOPS) spacecraft is a Swiss-led ESA mission. It is the first mission dedicated to determine the bulk density of already known exoplanets. This is achieved by ultra-high precision photometry of the transit events to measure the exoplanets' radii and detect atmosphere. The derived bulk density allows a first classification of the observed exoplanets. Launch is expected at the end of 2017. The CHEOPS spacecraft platform is developed by Airbus Defence and Space and the science payload instrument is under the responsibility of the University of Bern.

The CHEOPS satellite carries one single instrument which is a 32cm reflecting telescope designed to minimize any stray-light. It is equipped with a frame-transfer CCD operated at visual wavelengths. The instrument is controlled by a dedicated computer (the Digital Processing Unit or DPU). The DPU communicates with the central on-board computer (OBC) via a MIL-1553 link and it controls the front-end electronics (the Sensor Electronics Module or SEM[1]) through a SpaceWire link (see Figure 1). The measured science data are processed in the DPU with two main output data products, a compressed science data stream for downlink to ground and precise centroid measurements to support the fine guiding of the spacecraft. The software running on the DPU (the Instrument Flight Software or IFSW) has a layered structure as shown in Figure 2. Its bottom layer is the boot software which is provided by the DPU hardware supplier (IWF of Austria). The intermediate layer is the Instrument Basic Software (IBSW) which provides: (a) the operating system, (b) a middleware for accessing the communication links to the SEM and to the OBC, and (c) the DPU hardware management functions. The Instrument Application Software or IASW is the top layer which implements the high-level instrument control services and the data processing tasks. The IBSW and the IASW are developed by the University of Vienna.

The DPU hardware is based on an Aeroflex Gaisler GR712RC microprocessor, which is a system-on-chip ASIC providing two LEON3FT cores, SpaceWire and MIL-1553 interfaces. The redundant DPU boards and the boot software are developed by IWF Graz.

The IASW has two main tasks. The first one is to implement scientific algorithms for the processing of the raw data from the instrument. On-board processing of raw science data is essential to overcome bandwidth limitations and to maximize the value of data which are sent to the ground. The second task of the IASW relates to the fact that both the DPU and the SEM act as PUS terminals. Thus, if one counts the OBC, there are three PUS terminals on the CHEOPS instrument and the IASW implements the functions of:

- provider of PUS services to the ground
- provider of PUS services to the OBC

[1]The SEM is provided by DLR.

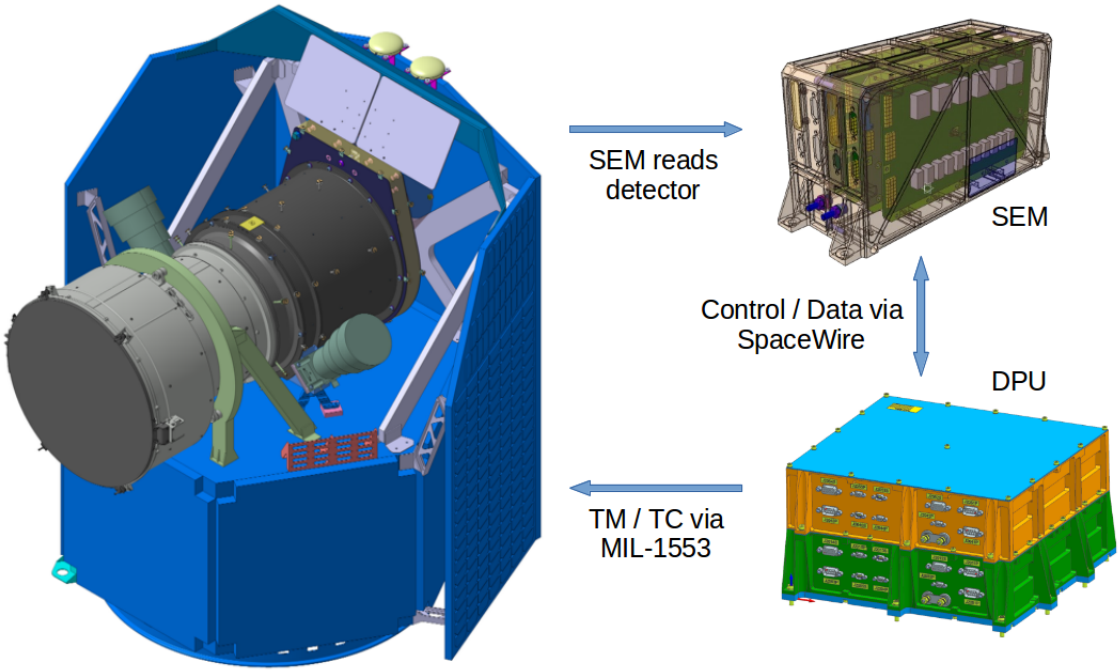

Figure 1. CHEOPS shown with payload instrument, indicating the data flow between the instrument electrical units.

- user of PUS services provided by the SEM
- gateway for PUS-based traffic between the OBC/Ground and the SEM

The PUS functionality of the IASW is therefore very complex. Rather than implementing this functionality from scratch, the University of Vienna decided to avail itself of the CORDET Framework. The next section describes the CORDET Framework in general terms and section 3 describes its application to the implementation of the IASW.

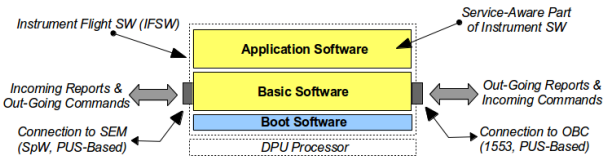

Figure 2. Structure of DPU Software

2. THE CORDET FRAMEWORK

The CORDET Framework is a software framework for service-oriented embedded applications. Its key concepts were introduced in the ASSERT Project and prototyped in the CORDET Project (ESA Contract 20463/06/NL/JD). The industrial-quality version of the framework was developed by P&P Software GmbH. A *software framework* is a repository of reusable and adaptable software components embedded within a pre-defined architecture that is optimized for applications in a certain domain (see Figure 3 and references [1] and [2]). The framework components are reusable in the sense that they encapsulate behaviour which is common to all applications within the frameworks domain.

To reuse a software component means to use it in different operational contexts. In practice, varying operational contexts impose different requirements. Hence, reuse requires that the reusable components be adaptable to different requirements. In this sense, adaptability is the key to reusability. For this reason, framework components offer *adaptation points* where their behaviour can be modified to match the needs of specific applications.

Framework components are embedded within a predefined architecture in the sense that the framework does not simply specify individual components but it also specifies their mutual relationships. Thus, the unit of reuse of a software framework is not a component but rather a group of cooperating components which, taken together, support the implementation of some functionality that is important in the framework domain.

The domain of the CORDET Framework is that of applications based on the Packet Utilization Standard or PUS[2] of reference [3]. The framework is defined at two levels. At ***specification level***, it specifies a generic architecture for applications which interact with each other by exchanging PUS commands and reports and it predefines components which implement the management

[2]In fact, the CORDET Framework is built on a service concept which is broader than that of the PUS. However, since this paper is addressed to users in the space domain, the discussion is restricted to the PUS.

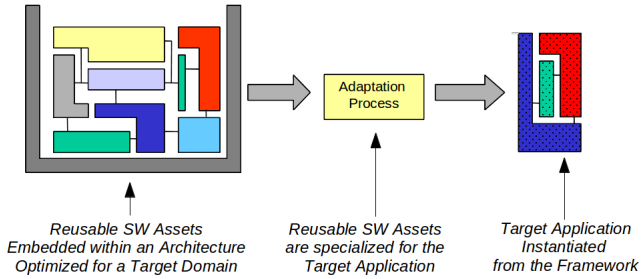

Figure 3. The Software Framework Concept

of these commands and reports. The pre-defined components are endowed with adaptation points where their behaviour can be modified to implement the specific services required by specific applications.

The CORDET Framework provides a complete and unambiguous behavioural model for these components. The behavioural model is built using the *FW Profile*. The FW Profile is a UML Profile which was first proposed in the ASSERT Project (see reference [4]) and was later extended by P&P Software GmbH (see reference [5]). This model-based approach allows users to analyze the behaviour of their applications statically and at specification level. Model checking techniques can be used to provide formal proof of correctness. The specification of the CORDET Framework is publicly available in reference [6].

Figure 6 shows an example of the models used to define the CORDET Framework. The state machine in the figure represents an incoming command. Its states (ACCEPTED, PROGRESS, TERMINATED and ABORTED) reflect the processing stages of incoming PUS commands which are first accepted, are then executed and remain ”in progress” for some time and are finally terminated; at any stage, telecommand processing can be aborted. Some of the elements in the state machine are prefixed by the UML stereotype <<AP>>. This stereotype identifies the *adaptation points* of the framework, namely the points where application developers can inject their application-specific behaviour to customize the framework to meet their requirements.

Two example of adaptation points in figure 6 are the Ready Check and the Start Action (both appear on the transition out of state ACCEPTED). The Ready Check represents the fact that incoming telecommands may remain pending for some time. A telecommand is only executed when it becomes ”ready” and the Ready Check encapsulate the conditions which makes the telecommand ready to start execution. This is an adaptation point because different applications and different telecommands within the same application have different readiness conditions. Thus, the framework states that all telecommand must define a readiness condition but it does not say what the readiness condition is for a specific telecommand. Similarly, the Start Action encapsulates a behaviour which is executed when the telecommand has become ready and has started execution. This is again an adaptation point because each application and each telecommand within an application may have a different Start Action. Thus, the framework states that all telecommand must define a Start Action but it does not say what that action is for a specific telecommand.

The CORDET Framework is provided with a list of Adaptation Points of which an excerpt is shown in table 1. This list exactly determines how the framework can be extended by an application developer. It can also be used as an instantiation guide since the instantiation process for the framework essentially consists in resolving each framework adaptation point. Note that, as shown in table 1, for most adaptation points, the framework pre-defines default actions which application developers are free to take over or override.

Table 1. Specification of Adaptation Points

AP-ID	**Adaptation Point**	**Default Value**
...	...	...
ICM-8	Start Action of InCommand	Set action outcome to ”success”
ICM-9	Progress Action of InCommand	Set action outcome to ”completed”
ICM-10	Termination Action of InCommand	Set action outcome to ”success”
ICM-11	Abort Action of InCommand	Do nothing
ICM-12	Operation to of Report Start Failed for InCommand	Generate command acknowledge report CMD_ACK_STR_FAIL with command’s identifier and with identifier of reason of failure as parameters
...	...	...

The use of the <<AP>> stereotype is constrained by the FW Profile which only allows certain elements of a UML model to be marked as adaptation points. This restriction is intended to allow the definition of *invariant properties*. Invariant properties are behavioural properties which are guaranteed to hold both on the framework model and on the model of all applications instantiated from the framework (provided that instantiation is done according to the rules, namely by overriding adaptation points). Examples of simple invariant properties defined on the model of figure 6 are:

- A telecommand only starts execution if its Ready Check returns ”ready”;

- When a telecommand is released for execution, it executes its Start Action.

The above properties can be easily proven on the framework-level model and are guaranteed to hold irrespective of how the adaptation points are filled in. They therefore represent aspects of the framework behaviour which the application developer can assume will hold on his application and which the application developers does not have to verify at application level.

The specification of the CORDET Framework is defined at UML-level and is therefore implementation-independent. P&P Software GmbH offer a C-language implementation of its specification with minimal CPU and memory requirements (about 20 kBytes) and excellent scalability in the sense that the framework overhead is independent of the number of services or of the number of telecommands or telemetry reports supported by an application. As already noted, a crucial feature of reusable components is their adaptation model. At C language level, the adaptation points of the CORDET Framework are mapped to the following adaptation mechanisms:

- *Define Constants*: framework components use `#DEFINE` constants whose value may be overridden by application developers.
- *Define Function*: framework components use function pointers and application developers must provide an implementation for the missing functions (or, if available, may choose to use the default implementation provided at framework level)
- *Implement Interfaces*: the framework defines interfaces as C header files and application developers must provide an implementation for them.
- *Define Types*: framework components use variables of a type defined as a `typedef` and application developers may override the default type definition.

The table of adaptation points of which an excerpt is shown in table 1 is extended to show how each specification-level adaptation points is mapped to an implementation adaptation-point using one of the four mechanisms listed above.

The selected adaptation mechanisms allow components to be adapted without changing their source code. This is extremely important because it has allowed P&P Software GmbH to provide a Qualification Data Package (QDP) for the C-language implementation of the CORDET framework. The QDP consists of:

- *User Manual* which discusses implementation issues which are relevant to end-users.
- *Software Requirements* which formally specify the implementation.
- An *Implementation Traceability Matrix* which shows how each requirement is implemented.
- A *Verification Traceability Matrix* which shows how the implementation of each requirement is verified.
- A *Validation Traceability Matrix* which justifies each requirement with respect to the intended use of the CORDET Framework.
- *Test Suite* with 100% statement, function, branch, and condition coverage.
- *Doxygen Documentation* for the entire framework code.

The implementation of the CORDET Framework is available as free software under a GPL from reference [7] and, on request, on less restrictive commercial licences. The Qualification Data Package is also freely available an can be downloaded together with the framework software.

The CORDET Framework is the end-point of a development effort which stretches back nearly 15 years and includes both ESA-funded projects (see reference [8]) and prototyping activities done by P&P Software GmbH within the space domain (see references [9] and [10]) and in other domains with similar reliability requirements.

3. USE OF CORDET FRAMEWORK FOR THE IASW

Like other PUS applications, the CHEOPS application software is specified in terms of the services it provides to others and of the services it requires from others. The CORDET Framework provides a specification-level behavioural model of PUS services. This model was used to express the requirements of the IASW. This resulted in a specification which was guaranteed to be both complete and unambiguous.

The transition from the specification to the implementation is fairly straightforward because the CORDET Framework User Manual explicitly lists all the adaptation points offered by the framework and, for each adaptation point, it identifies the adaptation mechanism (one of the four mechanisms listed in section 2) and it points to the part of the framework code where the adaptation point is implemented. The framework instantiation process therefore consists in going through this list and checking that each adaptation point is closed with the appropriate application-specific functionality.

With respect to the examples of adaptation points discussed in the previous section, the Ready Check and the Start Action are mapped to functions which must be implemented by the application developer and are plugged into the framework infrastructure as function pointers. The resulting structure of the application is rather like in figure 4. The CORDET Framework acts as a kind of

domain-specific operating system for PUS applications. User-provided code mostly consists of call-back functions which are registered with the framework and are called by the framework. An important point is that most of the logical complexity (branching logic) resides in the framework code. This code is already qualified at framework level and does not need to be re-qualified because the framework code is used without changes. The user code, by contrast, tends to be simple and often merely consists of linear sequences of statements. The use of the framework consequently has a significant and beneficial impact on qualification costs.

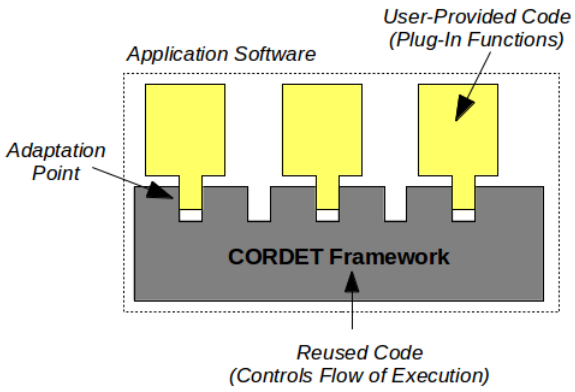

Figure 4. Application Software Structure

The most demanding part of the instantiation process is the provision of the functions which are plugged into the framework infrastructure as function pointers. These functions hold the behaviour associated to the IASW commands and reports. The IASW supports 16 services and well over 40 commands and reports. Each command and report defines several adaptation points corresponding to, for instance, the acceptance checks for the commands, the actions executed by the commands, the data collected by the reports, etc. The management of this variability proved daunting and eventually a code generator was built which automatically generates the headers and implementation stubs for all these function from an XML-based description of the PUS services provided by the IASW (see Figure 5). This auto-coding facility will probably be incorporated as a standard offering in future versions of the CORDET Framework.

4. CONCLUSIONS

At the time of writing, the development of the CHEOPS instrument software is between PDR and CDR. The experience from the use of the framework has so far extremely positive with positive impacts on the specification process (because the application was specified by customizing the pre-defined specification of the framework and because the framework behavioural models could be reused to build the application logical model) and at validation level (because the qualification data package of the framework could be directly imported into the application data package).

The re-use approach adopted for the CHEOPS IASW is distinctive in at least four respects:

- ***Third-party re-use***: the entity which developed the framework (P&P Software GmbH) is not the same as the entity which is instantiating it (University of Vienna).
- ***Multi-level re-use*** spanning specification (through the use of the behaviour model of the framework), implementation (through the deployment of the pre-defined framework components) and validation (through the incorporation of the framework qualification data package in the application's own qualification data package).
- ***Open-source re-use***: the framework and all its documentation (including in particular its qualification data package) are publicly available.
- ***Call-back form of re-use***: the IASW is built as a set of functions which are plugged into the framework infrastructure as function pointers and which, at runtime, are called by the framework which controls the flow of control for the handling of commands and reports.

REFERENCES

[1] Pasetti A., 2002, Software Frameworks and Embedded Control Systems, LNCS Series, Vol. 2231, Springer-Verlag

[2] Pasetti A., Schaufelberger W., 2009, Modeling and Software for Automation, in: S. Y. Nof(ed), Springer Handbook of Automation, Springer-Verlag

[3] ESA, 30 January 2003, Ground Systems and Operations Telemetry and Telecommand Packet Utilization Standard. ECSS-E-70-41A, ECSS Secretaria, ESA-Estec

[4] ESA, 2004, ASSERT Project ESA Web Site: http://www.esa.int/TEC/Software_engineering_and_-standardisation/TECJQ9UXBQE_0.html

[5] P&P Software GmbH, 2012, FW Profile Project Web Site: http://pnp-software.com/fwprofile

[6] P&P Software GmbH, 2014, The CORDET Framework Definition, PP-DF-COR-00002, Iss. 1.5, available from CORDET FW Web Site: http://pnp-software.com/cordetfw

[7] P&P Software GmbH, 2014, CORDET Framework Project Web Site: http://www.pnp-software.com/cordetfw

[8] Cechticky V., Pasetti A., Schaufelberger W., 2003, A Generative Approach to Framework Instantiation, in: F.Pfenning, Y. Smaragdakis (eds), Generative Programming and Component Engineering (GPCE), LNCS Series, Vol. 2830, Springer-Verlag

[9] Montalto G., Pasetti A., Salerno N., 2003, An Adaptable C++ On-Board Application, Proceedings of the 14-th Data Systems in Aerospace (DASIA) Conference, Prague, Czech Republic

[10] Cechticky V., Montalto G., Pasetti A., Salerno N., 2002, The AOCS Framework, Proceedings of the 5-th International ESA Conference on Spacecraft GNC, Frascati, Italy

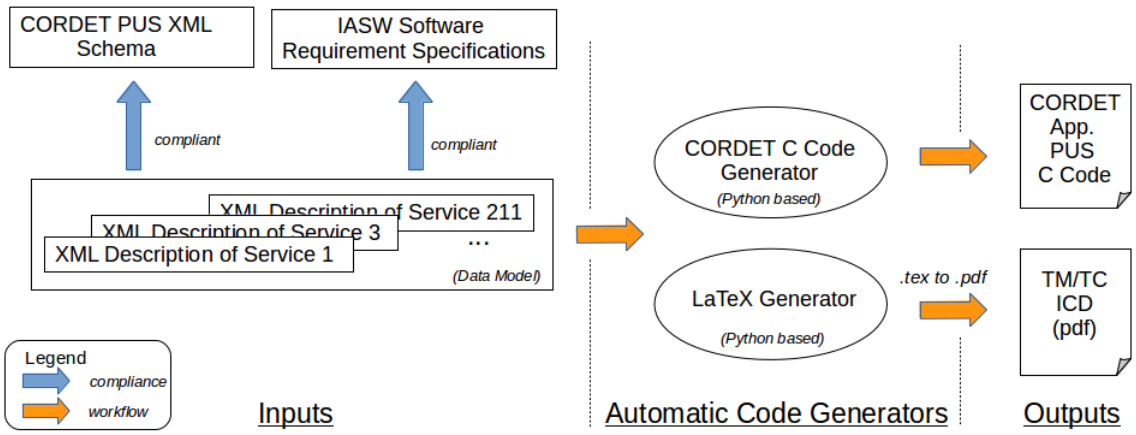

Figure 5. Automatic Generation of the Command and Report Functions

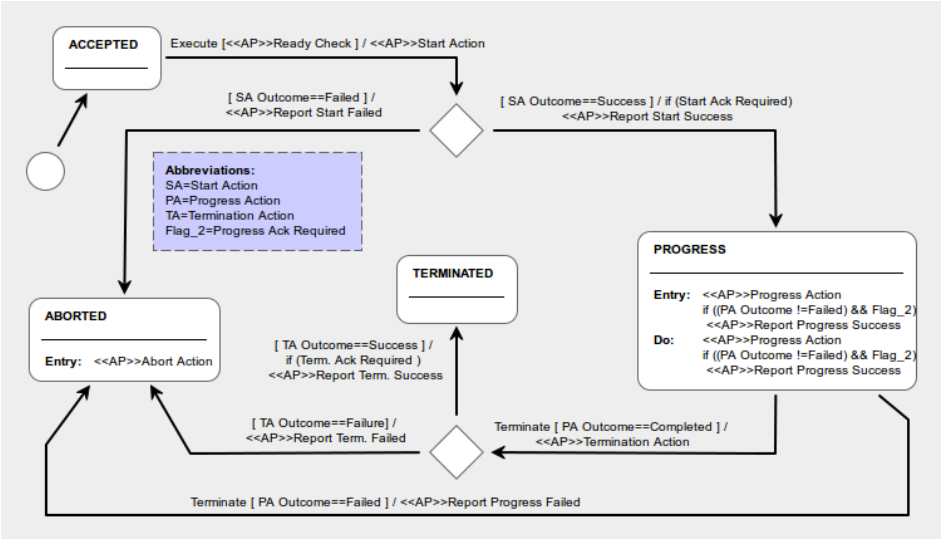

Figure 6. Behavioral Model for Incoming Commands