CARMENES. IV: instrument control software

Josep Guàrdia[a], Josep Colomé[*a], Ignasi Ribas[a], Hans-Jürgen Hagen[b], Rafael Morales[c], Miguel Abril[c], David Galadí-Enríquez[d], Walter Seifert[e], Miguel A. Sánchez Carrasco[c], Andreas Quirrenbach[e], Pedro J. Amado[c], Jose A. Caballero[f], Holger Mandel[e], and the CARMENES Consortium

[a]Institut de Ciències de l'Espai (IEEC-CSIC), Campus UAB, Facultat de Ciències, Torre C5-parell, 2a pl., E-08193 Bellaterra, Spain; [b]Hamburger Sternwarte (HS), Gojenbergsweg 112, D-21029 Hamburg, Germany; [c]Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía s/n, E-18008 Granada, Spain; [d]Calar Alto Observatory, Centro Astronómico Hispano-Alemán (CAHA), C/Jesús Durbán Remón, 2-2, E-04004 Almería, Spain; [e]Landessternwarte (ZAH), Königstuhl 12, D-69117 Heidelberg, Germany; [f]Centro de Astrobiología (CSIC-INTA), ESAC PO Box 78, E-28691 Villanueva de la Cañada, Madrid, Spain

ABSTRACT

The overall purpose of the CARMENES instrument is to perform high-precision measurements of radial velocities of late-type stars with long-term stability. CARMENES will be installed in 2014 at the 3.5 m telescope in the German-Spanish Astronomical Center at Calar Alto observatory (CAHA, Spain) and will be equipped with two spectrographs in the near-infrared and visible windows. The technology involved in such instrument represents a challenge at all levels. The instrument coordination and management is handled by the Instrument Control System (ICS), which is responsible of carrying out the operations of the different subsystems and providing a tool to operate the instrument from low to high user interaction level. The main goal of the ICS and the CARMENES control layer architecture is to maximize the instrument efficiency by reducing time overheads and by operating it in an integrated manner. The ICS implements the CARMENES operational design. A description of the ICS architecture and the application programming interfaces for low- and high-level communication is given. Internet Communications Engine is the technology selected to implement most of the interface protocols.

Keywords: CARMENES, spectrograph, instrument operation, control system, software, Internet Communications Engine, EPICS

1. INTRODUCTION

CARMENES[**] will perform high-precision measurements of stellar radial velocities with long-term stability[1, 2]. To carry out its purpose, CARMENES is based on two spectroscopic channels, optimized in the near-infrared (NIR) and visible (VIS) windows, and multiple subsystems that have to work in a coordinated manner: the front-end, calibration units and exposure meters of each spectroscopic channel, acquisition and guiding module, interfaces with the telescope and the dome, and, finally, the software subsystems for task scheduling, data processing and data archiving.

The Instrument Control System (ICS) is the main software component of the system in charge of coordinating and managing the subsystems, providing completely automatic control and high level of reliability and performance. Its main purpose is to permit scientists to operate the instrument without having to interact directly with the low-level Application Programming Interfaces (APIs) that control each subsystem and to provide an integrated control of the entire instrument. Scientists should only define astronomical observations and the ICS will setup, manage and control all subsystems to perform the observations and to obtain scientific results. This approach is intended to maximize the system operation

[*] colome@ieec.cat, http://www.ice.cat/view_staff.php?MID=25

[**] http://carmenes.caha.es/

Software and Cyberinfrastructure for Astronomy II, edited by Nicole M. Radziwill, Gianluca Chiozzi,
Proc. of SPIE Vol. 8451, 84512S · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.926981

time and efficiency. These statements and the instrument operational design compose the high level requirements that drove the design of the ICS.

2. OPERATIONAL DESIGN

The strategy to complete the CARMENES survey translates into a list of targets and their observation patterns aimed at carrying out an intensive monitoring to achieve the envisioned scientific goals. The instrument will have to execute the tasks involved in the data acquisition process. Complementary tasks will also be necessary for the instrument commissioning and its routine calibration. All these tasks (science observations, calibration and commissioning) are supported and managed by the ICS, which handles the operation of the CARMENES instrument by taking into account several control, operation and scheduling modes.

The routine operation will be configured to use simultaneously the two channels, with NIR being the master and VIS being the slave (see Section 2.2), under the automatic control mode.

2.1. Control Modes

Four different control modes are defined:

- **Interactive/Operator:** This mode allows controlling all ICS functionalities while carrying out the acquisition of data. The system always starts in this control mode and then it can be changed to a different one.
- **Observer:** This mode allows users to take observations using a basic ICS interface.
- **Engineering:** This mode introduces some additional functionality for controlling any subsystem or configuring the ICS behavior.
- **Automatic:** The system runs autonomously in this control mode, with the supervision of an operator.

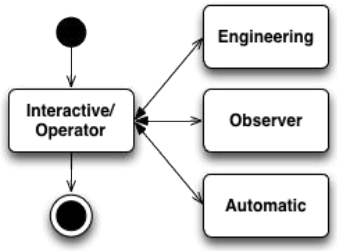

Figure 1. State transition diagram for the different control modes.

2.2. Operation Modes

Three different operation cases are considered to provide higher flexibility during the instrument commissioning and, also, maximum efficiency during the nominal operation:

- **Single channel:** Just one channel (NIR or VIS) is used to perform any of the operational tasks under any of the control modes.
- **Multiple channels:** Both NIR and VIS channels are used, but none of them takes the role of master. Observation and data processing execution workflow runs independently. This operation mode is used under the nominal control mode.
- **Multiple channels, one of them acting as master:** Both NIR and VIS channels are used and one of them acts as a master regarding observation and data processing control.
    - **CARMENES mode:** The nominal operation mode (or CARMENES mode) for the instrument is defined using both channels with the NIR one working as master. The VIS channel operation is subordinated to the execution time and overheads of the NIR channel that leads the workflow.

2.3. Scheduler

The scheduler[3] selects the most suitable task/s to be performed by the instrument according to predefined criteria and with the goal of maximizing the efficiency of science operations. The selection is done by executing a global rating of all

the active tasks (science operations and calibrations) and creating a ranked list according to their global merit. Long- and short-term plannings are considered to compute the task prioritization. In the automatic control mode, the ICS executes the task(s) with the highest priority.

2.4. Nominal run workflow

Nominal operations run under automatic mode to perform science operations or calibration tasks. The workflow for these operations is predefined and handled by the ICS with the design of suitable procedures. These procedures are based on a set of input and output data and on subsystem actions. Each action takes a series of preconditions for execution. The steps in a nominal science data acquisition workflow are grouped into those processes required for: task selection, performed by the scheduler application; change of the system configuration; acquisition of the spectra with both channels; and, finally, data processing. Operation idle time is minimized in the workflow design.

3. INSTRUMENT CONTROL SYSTEM

The ICS is the central software application in the CARMENES control layer. It is based on a modular architecture and a high level of abstraction design motivated by the heterogeneity of the different subsystems.

A master/slave model architecture is used to build the control layer, where the ICS acts as a master and almost every other subsystem is a slave. The ICS acts as a slave only for the User Interface subsystem, which controls and monitors the ICS functionalities.

The subsystems managed by the ICS are seen as a “black box” and there is no dependence with their internal architecture thanks to the high abstraction level used. A well-defined API, providing a connection bus and a protocol, is the basis to intercommunicate them. This approach ensures the separation in functionality and increases the modularity of the full system. The system topology is shown in Figure 2.

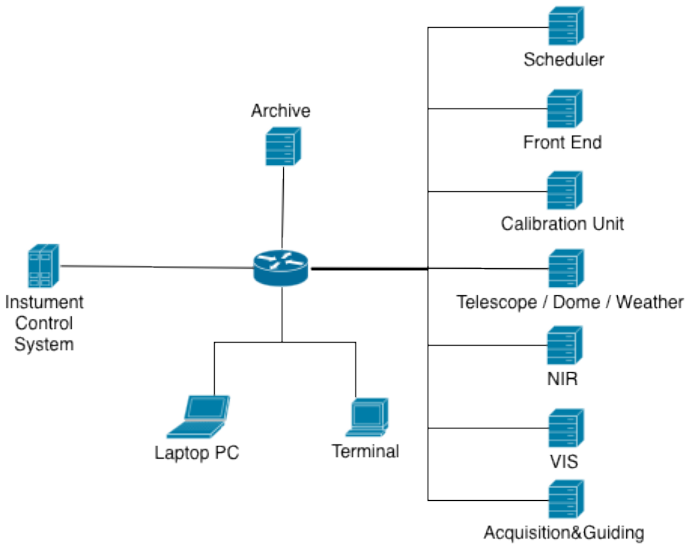

Figure 2. CARMENES hardware architecture

The instrument subsystems are abstracted into the ICS with a logic representation (see Figure 3). Each subsystem is divided into specific functionalities, such as data representation, actions and events.

The subsystem data (e.g., temperatures, encoder positions) are reported periodically to the ICS using its communication protocol and are stored into a central database that conforms a pool of updated data for all the parameters of the system (like a snapshot).

Events (or contingencies) can arise during a nominal subsystem execution. An event is something that happens to the subsystem that generates a notification to the user or the execution of actions.

The designed actions are encapsulated in a common API and the ICS can trigger them using a common interface and without any dependence with the communication protocol. Actions can be stored in an action manager and executed when necessary following a predefined sequence. The system status is checked previous to the action execution to prevent possible erroneous commands.

The overall system operation is, finally, handled by events that trigger predefined actions. These actions change the status of one or more components to reach the required configuration or response. Actions can be also triggered by an operator using any of the interface options (graphical or simple scripting) when the system is not running in the automatic control mode.

The heterogeneity of the CARMENES subsystems imposed different interaction methods and protocols. Therefore, the ICS supports the most common communication protocols used in astronomical observatories.

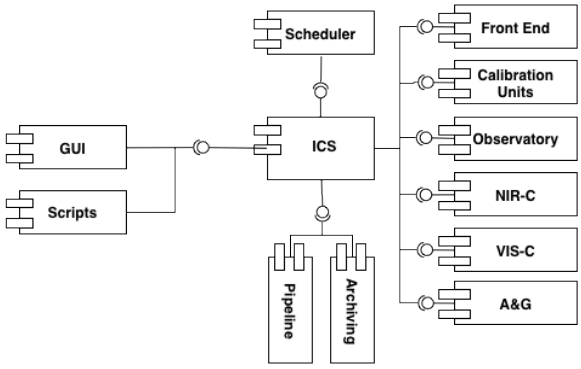

Figure 3. CARMENES logic components architecture.

4. MODULAR DESIGN

The ICS is designed using the Layers design pattern, which separates application functionalities into distinct levels of abstractions by decomposing complex problems into smaller and more manageable ones. The architecture (see Figure 4) is split in the following layers:

- **Operating System Layer:** This is the abstract layer that interacts with the operating system. It provides the necessary functionalities to manage threads, semaphores, mutex, file systems, etc.
- **Third-Party Libraries Layer:** It contains the libraries used from third party developers.
- **Modules Layer:** This layer does not process any data. Its aim is to manage a large amount of information by encapsulating it in data structures, which are grouped into modules.
- **Procedures Layer:** This layer defines all processes necessary to manage data and actions.
- **Subsystems Layer:** It contains the subsystems abstraction.
- **Communication Layer:** It contains all the protocols to communicate with the subsystems.
- **Interface Layer:** It defines all communication APIs to interact with the subsystems, modules and procedures.

Most of these layers, except the Operating System and the Third Party Libraries, are specifically designed for the CARMENES instrument and are described in the following sections.

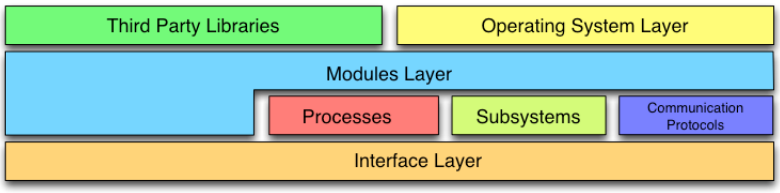

Figure 4. Layer design for the ICS.

4.1. Modules Layer

The modules defined in this layer represent the ICS core functionalities. Each module is designed to accomplish only one functionality and contains everything necessary to fulfill its purpose. This design reduces coupling and increases readability and maintainability.

- **System Data Pool** (SDP): The SDP is a centralized database, where all information is stored in a standardized way. All services (other modules and subsystems) can access the data in a common, simple, fast and accurate manner in order to update it or to get updated values. Communication among subsystems is based on this module, which also provides synchronization capabilities.
- **On-Board Monitoring Function** (OBMF): The OBMF contents a set of checks to be performed to SDP items. It defines which SDP item must be monitored and the kind of checks to be performed. It provides the conditions to create events as a function of the result obtained from every check.
- **Events:** All services can generate events. Events are generated, for instance, by the OBMF service each time an SDP item is not correctly checked, by the NIR subsystem when an image is not correctly created, or by the Telescope subsystem when it is correctly pointed. This module centralizes the temporal storage of all events.
- **Actions:** All subsystems are controlled using actions. An action encapsulates the command that will be sent to the subsystem, its custom data checks, and the method to save its responses. Actions can be blocked following different criteria: all responses must be received, some events must be received, etc.
- **Procedures:** This module stores a set of actions for a sequential execution. Procedures can be modified by adding, removing or modifying actions. A procedure can be stopped when: an action does not terminate as expected, an event is received, etc.
- **Event-Action:** The system is event-responsive: an event is generated when a predefined or anomalous situation arises, and it subsequently produces logs, triggers actions, etc. Links between events and actions are specified in this module.
- **Housekeeping:** All system information is stored into the SDP and must be reported to the user interface in concordance with a set of reporting definitions previously defined. Structure identification (SID) is associated with each distinct reporting definition and associated housekeeping parameter report. The SID uniquely identifies the housekeeping parameter report and is used to interpret its contents. Each SID has a subset of SDP items that will be the ones reported in each call to Housekeeping. Each reporting configuration has an associated collection interval, which determines the data sampling.
- **Processes:** This module defines an abstract class that provides access to all processes that are running in the ICS. A process running in the ICS is executed periodically at a predefined rate and with a soft deadline. This guarantees the service quality for each process running. The process module also provides continuous performance data describing its own execution. It is useful for starting, stopping, controlling, and monitoring processes inside the ICS.
- **Subsystem:** It defines an abstract class to implement all the subsystems. This class provides an implementation of the Façade design pattern. The Façade design pattern simplifies the interface with a complex system. It also decouples the subsystem from its low-level implementation.
- **Statistics:** This module computes statistic parameters for predefined SDP items that are necessary to monitor the system behavior. The obtained values are also stored in specific SDP items.

4.2. Processes

The ICS has several processes to carry out the internal actions. These processes take information from different modules.

- **Statistics:** This process calculates statistics over some SDP items.
- **OBMF:** This process performs a periodic check of the OBMF items. It compares them with the corresponding SDP items and generates events according to programmed conditions.
- **Action:** It is in charge of executing actions when the system requires them and is ready to do so. The actions check the preconditions before the execution.
- **Housekeeping:** This process is in charge of obtaining information from the SDP and generates the Housekeeping messages that are reported to the User Interface.
- **Event:** This process is in charge of reporting the events generated (asynchronously) by different modules or procedures. It can execute actions associated to an event (following the configuration given in the Event-Action module), report events, etc.

4.3. Subsystems

This layer contains a class for each CARMENES subsystem. Each subsystem inherits from the subsystem abstract class defined in the modules layer and implements the Façade design pattern. This pattern hides the details of each subsystem action and the subsystem communication protocol (see Figure 5).

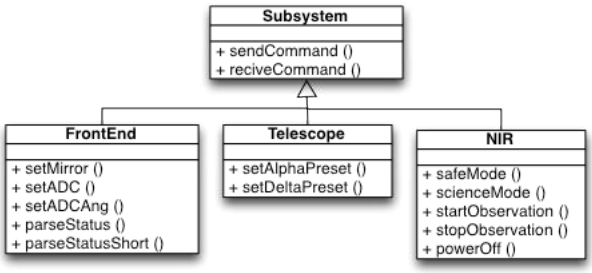

Figure 5. Use of the Façade design pattern to hide the details of the actions in each subsystem.

The ICS is highly customizable: each command can be executed in an analogous manner for all the subsystems. A command pattern is used for this purpose. Commands can be stored and executed, providing the capability to customize procedures. Figure 6 illustrates how the command pattern encapsulates the action requests. It is implemented in the action module of the modules layer.

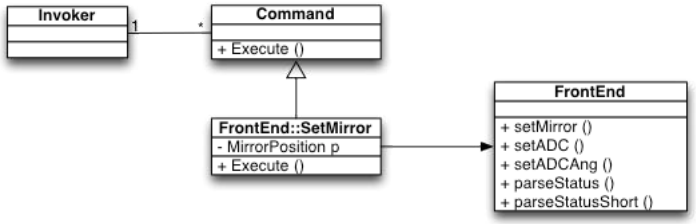

Figure 6. Actions are encapsulated using the command pattern design.

Each action checks some SDP items to confirm that the conditions for its execution are met. In case they are, the action is executed and sent to the corresponding subsystem. When the action is finished, some SDP items are updated and the command finishes. Figure 7 shows the flux diagram of an action.

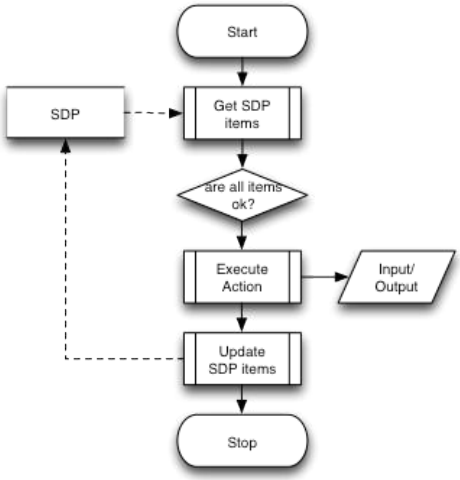

Figure 7. Action flux diagram.

4.4. Protocols

The CARMENES control layer is composed by a heterogeneous set of subsystems at both physical and logical levels. Different physical implementations are used to run the low-level control software (i.e., microcontrollers, PCs) and some of them do not support using communication protocols that require high processing capability. This lack of homogeneity motivated the design of the ICS to support any communication protocol using a communication layer that hides the protocol implementation constrains.

The ICS uses the Internet Communication Engine[*] (ICE) middleware as the base communication framework. ICE provides a complete solution to communicate different distributed subsystems that span multiple operating systems and programming languages.

In addition to ICE, two more protocols are also implemented: Epics[**], used to interface with the telescope and dome control software; and a TCP/IP-based custom protocol (called CARMENES protocol), used to interface with the NIR and VIS channels, the front-end, and the calibration units.

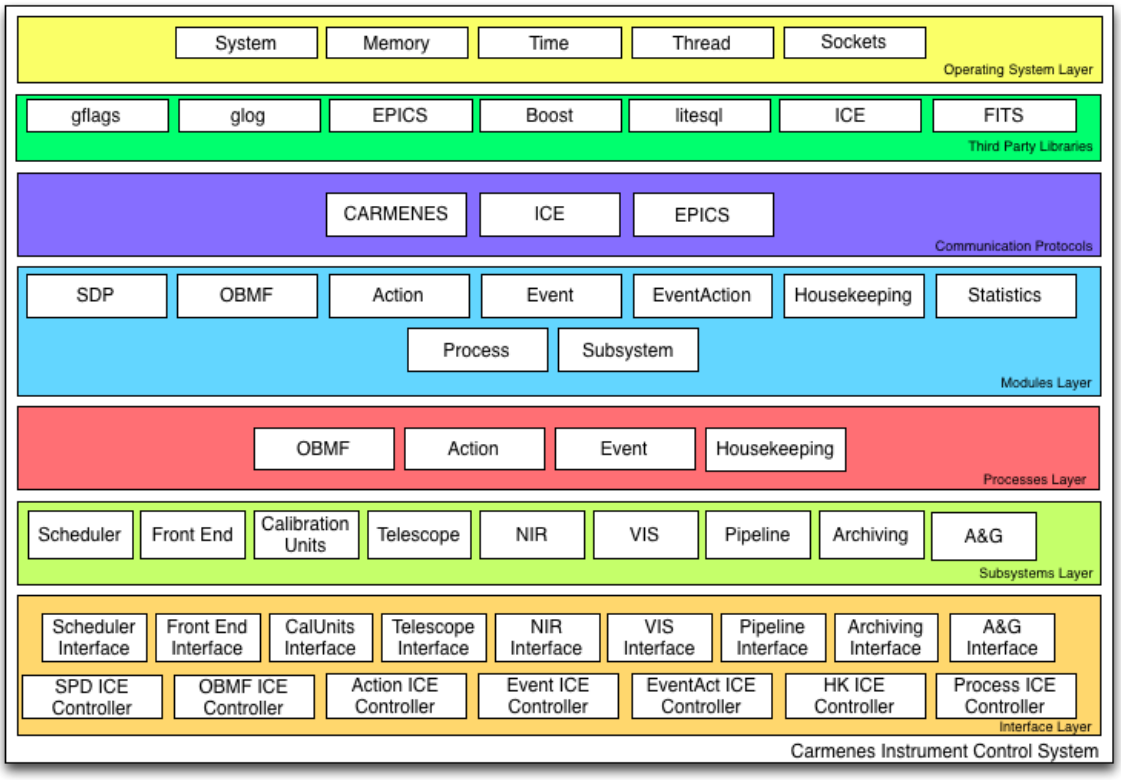

Figure 8. Layers and main modules of the CARMENES ICS.

[*] http://www.zeroc.com

[**] http://www.aps.anl.gov/epics/

5. CONCLUSIONS

CARMENES is a new generation instrument that will provide stellar radial velocities of unprecedented accuracy (1 m/s). It will be installed at the 3.5-m telescope in the CAHA Observatory and will start operations in 2014. It will be equipped with two channels for spectroscopic data collection in the NIR and VIS windows. The scientific requirements specified impose stringent conditions on stability and performance of the instrument so that the technology used in the development represents a challenge at all levels. The efficient and reliable operation is also specified as a requirement to maximize the scientific return. Such requirements on operation translate into strong constraints to define the control layer, at hardware and software levels, and all the subsystems that compose it. The operational design is the next step to start specifying the requirements for the ICS, the central software application responsible of managing the overall instrument operations. The routine operation, according to this operational design, is based on the simultaneous use of the two channels running under automatic control mode, in order to maximize the efficiency by minimizing the operation idle time.

The main goal of the ICS and the CARMENES control layer is to fulfill the operation requirements, which impose constraints on the ICS design, but also on the design of the low level control of each subsystem. A modular architecture and a high level of abstraction were the basis to design this software application that has to handle a heterogeneous group of subsystems in a coordinated manner. Several layers and modules compose it (see Figure 8) and have been described. They implement the different functionalities in a highly customizable way.

The control system presented here can be considered as a solution to control any complex instrument composed by a large variety of subsystems, connectivity (i.e., RS-232, CAN, USB, Bluetooth) and top-level communication protocols.

The detailed design phase of this software is reaching its final stage to be ready for the Final Design Review of the project that will be held in November 2012. The development will end with the AIV process, and final delivery is expected in early 2014, just after the instrument commissioning.

ACKNOWLEDGEMENTS

CARMENES is an instrument for the Centro Astronómico Hispano Alemán de Calar Alto (Almería, Spain). The observatory is operated jointly by the Max-Planck-Institut für Astronomie of the Max-Planck-Gesellschaft and the Instituto de Astrofísica de Andalucía of the Consejo Superior de Investigaciones Científicas. Partial financial support for the work presented was provided by the Spanish Ministry of Economy and Competitiveness under grant AYA2009-06934 of the Programa Nacional de Astronomía y Astrofísica.

REFERENCES

[1] Quirrenbach et al., “CARMENES project: Calar Alto high-resolution search for M dwarfs with exo-earths with a near-infrared Echelle spectrograph”, Proc. SPIE 7735, E37 (2010)

[2] Quirrenbach, A., Amado, P. J., and the CARMENES Consortium, “CARMENES. I: instrument and survey overview”, Proc. SPIE 8446, these proceedings (2012)

[3] Colomé et al., “Research on schedulers for astronomical observatories”, Proc. SPIE 8448, these proceedings (2012)