

Better Software Reuse by State of the Art Design Approaches

H. Wulf¹, C. Stöcker¹

¹VCS AG, Bochum, Germany

Introduction

Complex information and communication infrastructures are composed of a variety of hardware platforms, operating systems and application software packages. Due to decreasing hardware prices, investment costs into infrastructures are lowering continuously. The standardization in software applications and operations for commercial ITC infrastructures is an ongoing effort of the industry. In order to decrease the operations costs in satellite and manned flight mission ground segments accordingly, the diversity of products and procedures to operate the various platforms has to be limited by standardization and software reuse.

Software Reuse Problems

Software reuse is not a novel idea in the IT-world in general and in ground segment software development in particular. However, due to the high performance requirements of ground segments, the use of highly specialized software has long been the design approach of choice. Not only the high level of specialization but also the risk factors of software reuse often cause software reuse projects being unsatisfactory for supplier and customer. Lack of conceptual documentation increases time and cost to understand the existing software solution and to assess the applicability for the new environment. A good error record in the original environment does not prove for robustness and stability in new runtime environment. Hidden errors in the reused software are distributed to the new field of application and may cause malfunctions. Many problems also arise from a lack of reuse management that is definitely needed to avoid underestimate of qualification efforts for the new environment, to cope with the complexity of error correction and release planning and to avoid conceptual entropy by repeated reuse (lossy reuse). Another problem is the unawareness of the human factor, there is a common aversion to reuse existing code.

Approaches to Software Cost Efficiency

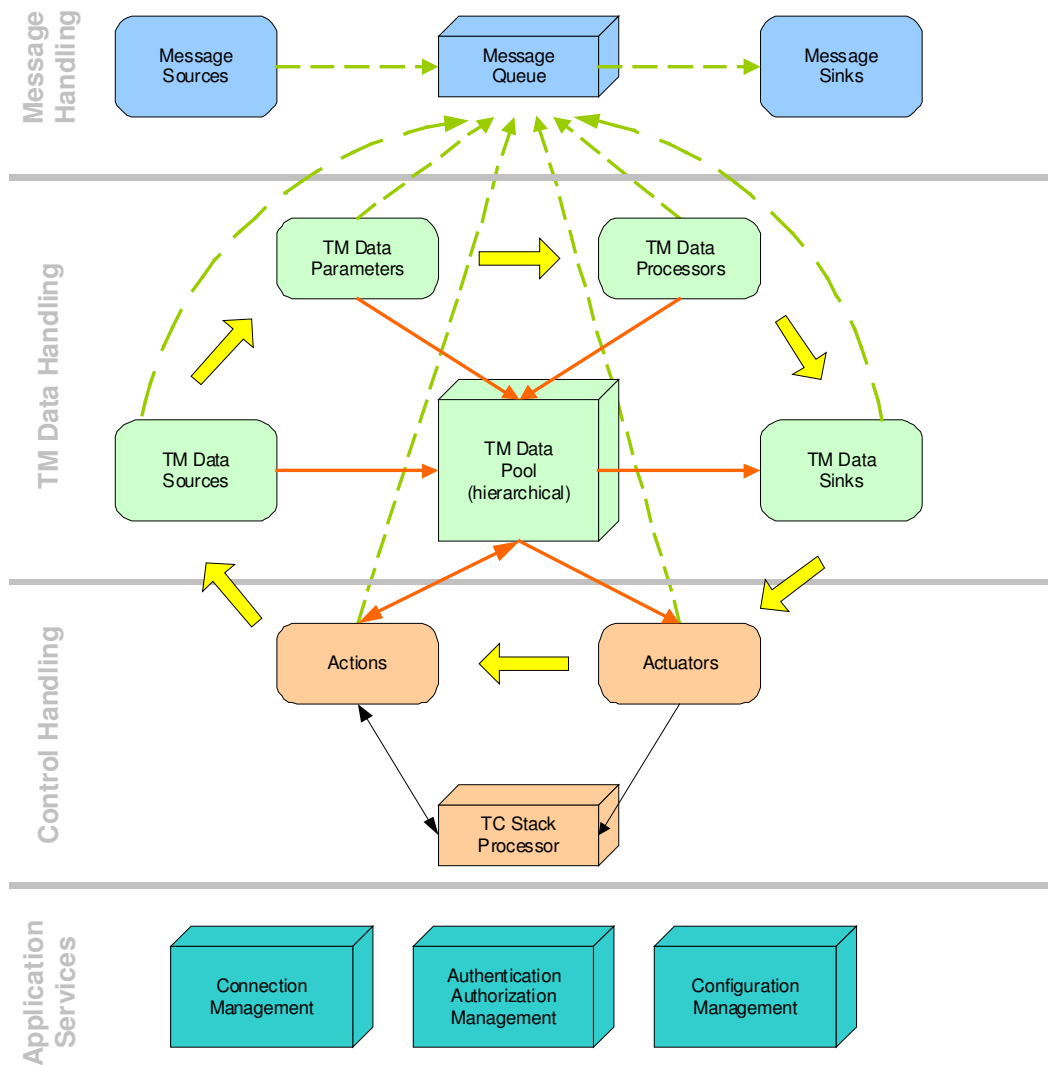
Today, the availability of very powerful hardware at moderate costs in conjunction with improvements in high-level software development tools enables the creation of generic applications for ground segments. One approach to improve cost efficiency of software solutions is to rationalize solutions and resources via the standardization of functions and interfaces and via harmonization of user requirements. This approach finally results in productisation and finally to software products. Another approach is go for conceptual variability and adaptability via abstraction of functions and interfaces and via definition of concepts and patterns. This approach results in software frameworks that can be reused for a wide range of application.

Software products achieve cost reduction through communality. They are based on common specifications and requirements, implement common functions and interfaces provide common operations concepts and are built from common libraries and languages. Software frameworks achieve cost reduction through flexibility. They are based on a universal approach while providing full conceptual integrity, they provide adaptability, scalability and expandability through high level abstraction of functions and interfaces.

Software Framework for Monitoring, Control and Configuration

The use of application cores with surrounding helper applications and integrated modules is an emerging approach to address at the same time the universal scope and the reusability of software. This approach is in this paper referred to as software frameworks, though the term is also seen to refer to mere software libraries. VCS provides a software framework for monitoring control and configuration, named **EGMC²**, which targets to ease operation of complex IT-infrastructures at affordable price. Being a framework, it allows extensive reuse of approved and reliable software components for communication and processing by providing sufficient flexibility and scalability to adapt to individual requirements. It is planned from scratch for reuse and variability. By concept it fulfills many high level requirements that are often needed of ground-segments but are rarely met by supplied applications. Zero coding for most deployments and support for future extensions allow flexibility and adaptability. The framework scales to small and large systems, it support standard interfaces and protocols to improve interoperability. It is generally independent of operating system, database system and vendors, it provides an intuitive, manageable configuration and conceptual documentation.

Monitoring, Control and Configuration is provided by 4 conceptual layers. Basically there is an application service layer providing connection services, authentication and authorization as well as configuration management. The other 3 layers are related to message handling, parameter processing and command execution. Messages represent system events, all messages are received via message sources (external, and internal - the green arrows), processed through message queues and delivered to message sinks. Message sinks can be alarm displays, emails, pager alarms, database entries, etc. Parameters are all types of measurements, status flags, data that are regularly received by the system. They are stored and processed via central parameter storage implemented as generic tree structure (the TM data pool). Parameters can be just single leafs or even complex subtrees. Access to the TM data pool is denoted as red arrows. Parameters are received via TM data source, process in internal or external data processors and distributed via TM data sinks. Control is handled via actuators and actions. While actuators define conditions when actions are required, actions can be all type of sending commands, executing procedures, updating parameters, sending messages etc. The yellow arrows denote an abstract flow for automatic service management (ASMA). This means that the system can be configured to automatically react on specific system status, thresholds, system dynamic or race conditions.



Universal Tool for Ground Segment Applications

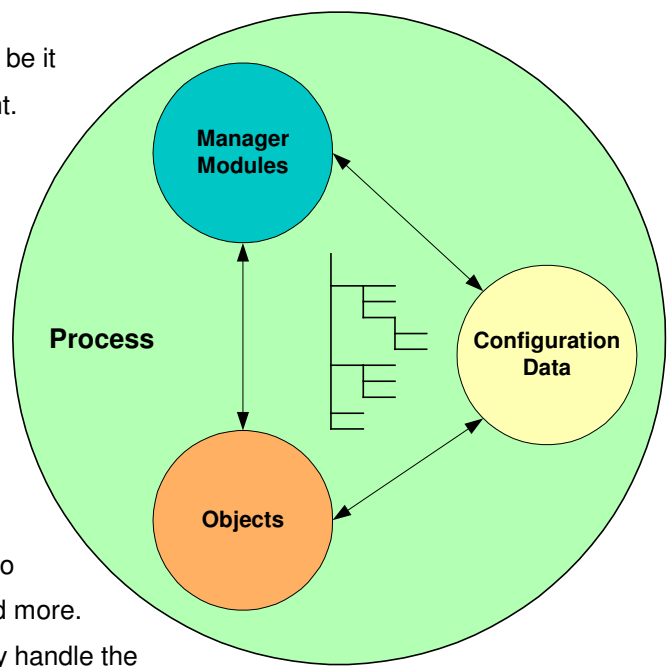
In the scope of several projects, the **EGMC²** framework has grown to what is now a universal tool for the fast and – in terms of costs – predictable development of ground segment monitoring and control infrastructures. It bridges the gap between network management and satellite control functions provided by frameworks such as SCOS-2000 and CGS. Through the use of just-in-time compiled languages such as Java and Python, the framework offers a flexibility that cannot be reached by traditional OO-Languages such as C++. The framework applications can play various roles in the ground segment, from data collection agents to data aggregating gateways and servers to data displaying clients or archive processes. New applications conforming to specific subsystem requirements are not manually coded, but generated from existing building blocks in GUI-centered Application Design Tools. Here, both GUI aspects in client applications and server-side aspects such as data acquisition, data processing and data distribution can be configured using the Application Design Tools. With the built-in flexibility **EGMC²** can be used to rapidly establish operational systems but also to act simulator for a variety of test scenarios.

All processes within **EGMC²** have identical structure be it configured as agent, gateway, server, proxy, or client.

The generic process hold the data tree and hosts manager to delegate the work, objects to perform the work and configuration data to define the functionality. By application of this principle, the behavior of an **EGMC²** process can usually be configured at zero coding.

There are several type of managers e.g. to support input interfaces, to handle parameters in the tree, to monitor parameters and trigger actions, to execute actions, to manage MMI views, to writes data trees to relational database, to handle client connections and more.

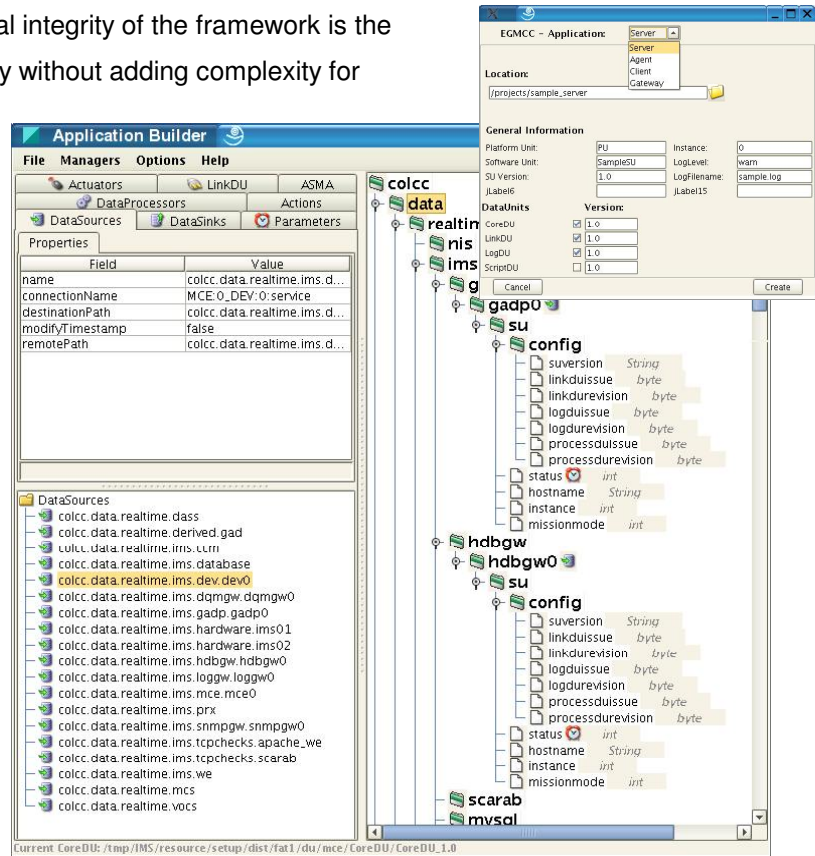
The managers never execute any functions, but they handle the interfaces, the nominal working scenario and the exceptions via objects to e.g. receive data, process data, detect and trigger, execute or display MMI views of parameter collections.



Distributed and Multi-Mission Architectures

The local flexibility and the conceptual integrity of the framework is the basis to build-up higher level flexibility without adding complexity for configuration and management of

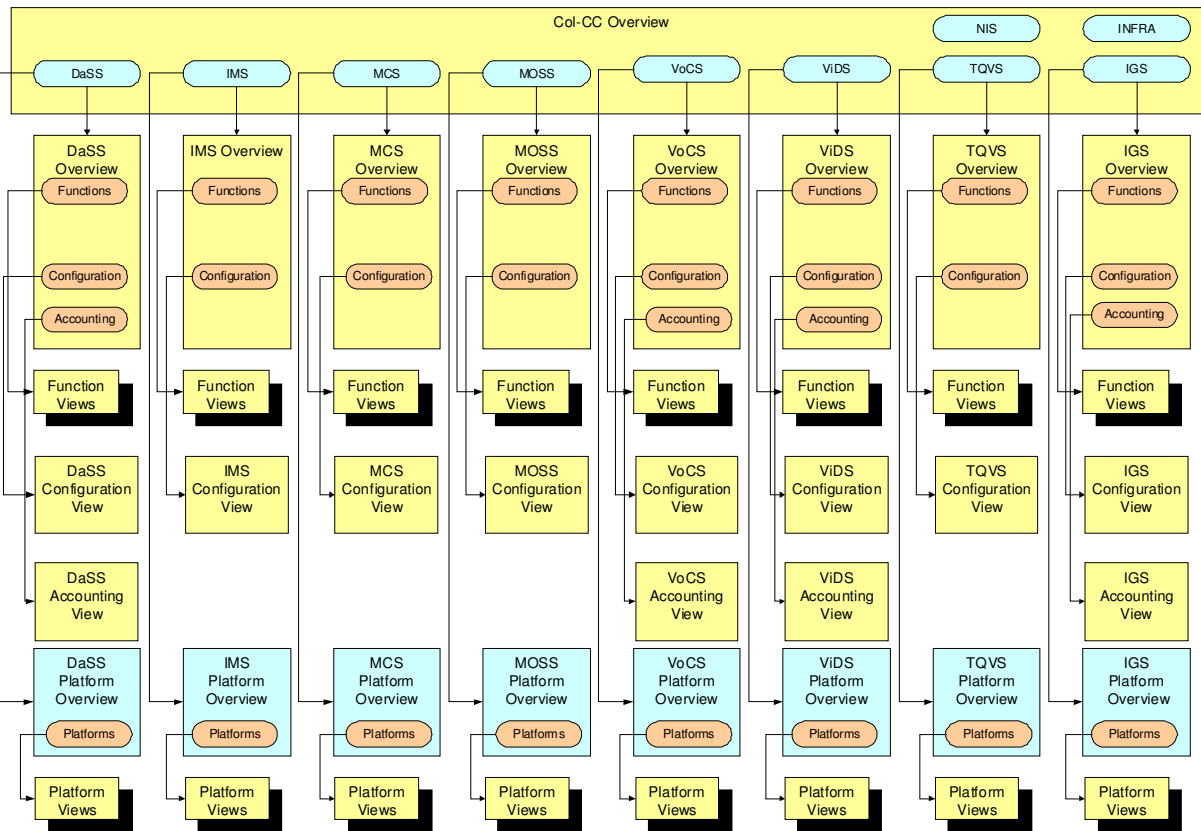
the application. In vertical direction this enables the **EGMC²** framework to build multi-tier architectures to solve complex ground segment monitoring and control tasks. In horizontal direction, the framework supports building of multi-mission applications. The System Design Tools provided with the framework support the generation of such higher level applications. Through the use of these tools, a network of applications can be deployed to provide the telemetry data and telecommand handling required in a ground segment monitoring and



control system.

The application builder allows building up complex **EGMC²** applications by configuration of nodes to be integrated in the system architecture, links and interfaces to connect the nodes and build the network as well as sinks and sources to define data flows. Data processors are configured to generate derived parameters or to compute complex algorithms, actuators to check limits or thresholds, to detect complex system status and to trigger actions and finally activate abstract actions with system commands, procedures, messages etc. The automated service management allows defining actions responding to limits exceeded, discontinuities, extrapolations and derived status.

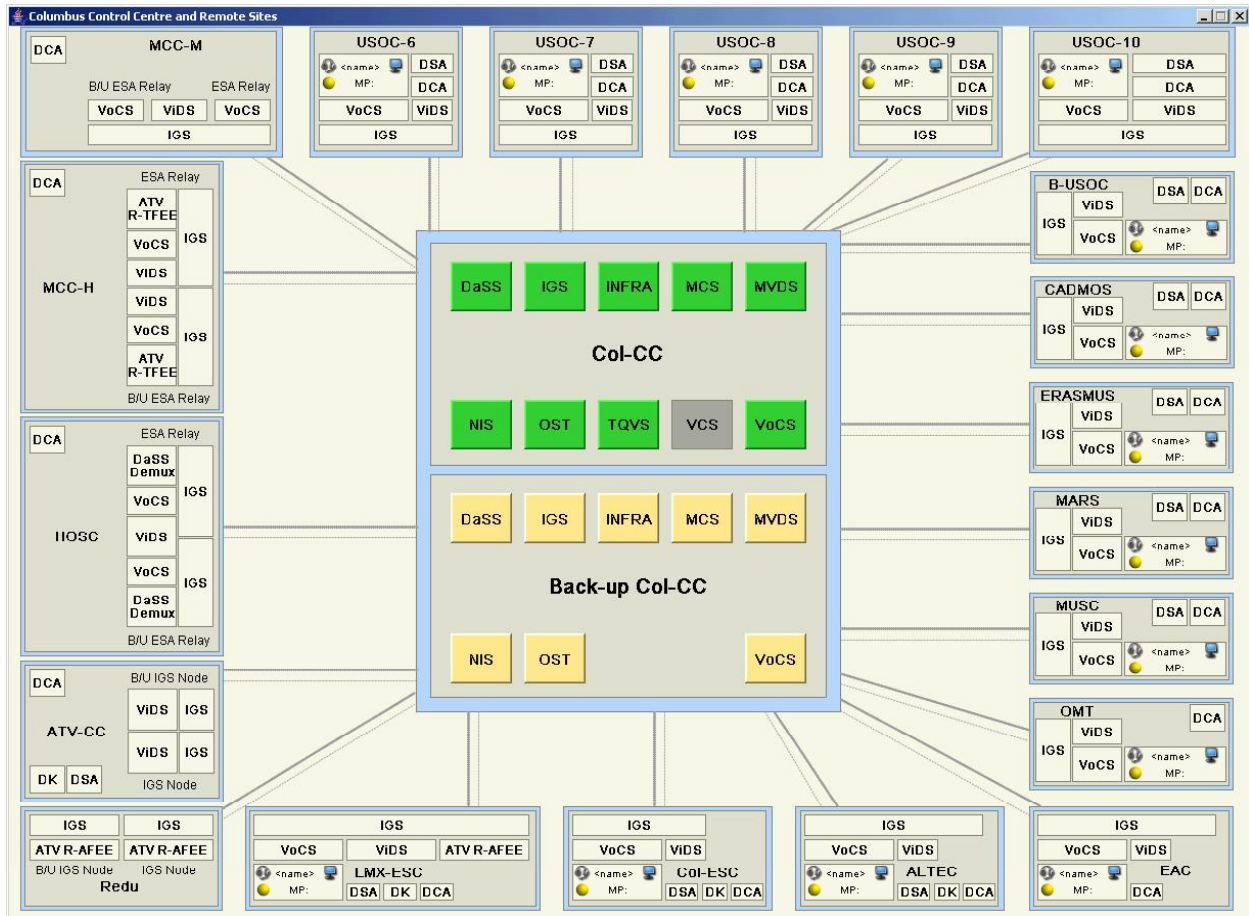
The MMI builder allows defining MMI views of parameter collections from the central data tree. It allows basic drawing features like line, circle, etc., allows grouping, adjustment and inclusion of external symbols and graphic. The advanced features allow configuring built-in thresholds, color schemes, tree-map structures to represent a full sub-tree and binding of external libraries.



Reference Systems

One of the **EGMC²** reference systems is the Integrated Management System (IMS) of the Columbus ground segment, which has been procured by ESA and is being implemented by DLR at the German Space Operations Center in Oberpfaffenhofen. The IMS is designed to monitor, control and configure the whole Columbus ground segment with distributed main sites in the Germany, the US, France and Russia

and user operation centers spread all over Europe. The IMS provides a strictly hierarchical Col-CC overview MMI, allowing to easily spread into subsystem overviews and from there into details represented by functional, configuration and platform views. The Col-CC Overview Monitoring Screen is shown below as an example of a configured MMI view built with the MMI builder mentioned above.



Conclusion

VCS provides a software framework for monitoring control and configuration, named **EGMC²**, which targets to ease operation of complex IT-infrastructures at affordable price. Being a framework, it allows extensive reuse of approved and reliable software components for communication and processing by providing sufficient flexibility and scalability to adapt to individual requirements. It is planned from scratch for reuse and variability.