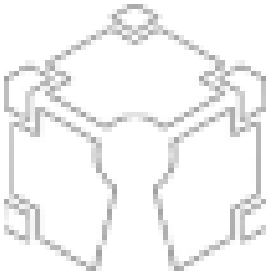

Contribution ID: **13** Type: **Talk**

Finding an Open Least Common Denominator for Live Integration of Non-space-system-standard Components within Constraint Budgets

Tuesday, 25 September 2018 10:20 (20 minutes)

The practical results presented in this talk were gained with a range of Open Source Software (OSS) solutions. The most important employed OSS is "bowerick" [1], which is a Message-oriented Middleware (MoM) wrapper around Apache ActiveMQ [2] and various supporting libraries such as Eclipse Jetty [3] and Spring Messaging/Websocket [4] for WebSocket-based [5] transport or Eclipse Paho [6] for MQTT-based [7] transport. bowerick and its dependencies are available under commercially friendly OSS licenses such as the Eclipse Public License, the Apache License, or the MIT License.

Space system standards such as PUS [8] or CCSDS-MO Services [9] aim on enabling interoperability between different systems or components. Yet, there are still situations that require the integration of components that do not support these space system standards. Furthermore, these standards are sometimes comparably heavy-weight with respect to the implementation effort such that, when no pre-existing libraries can be used, their adoption would require significant budget.

Small projects such as studies, prototypes, or small satellite operations may suffer from strongly constraint budgets. Hence, it may not be possible to implement heavy-weight standards when no pre-existing libraries are available. Our focus is explicitly on implementation effort and interoperability and not on resources like CPU, memory, or network throughput because, usually, the largest cost factors for small projects are the actual development and integration effort and not computing infrastructure related hardware resources.

In this talk, we report our experiences with other more light-weight implementation OSS alternatives for easing and reducing the cost for the live integration of non-standard components. The presented results are partly based on experiences gained during two studies regarding the integration of ESA software systems with Virtual Reality (VR) [10] and Augmented Reality (AR) devices.

For enabling live integration at run-time, we consider the aspects serialization format and communication infrastructure protocol. Popular serialization formats are, e.g., XML, Google Protocol Buffers, JSON, and to some degree CORBA. Popular communication infrastructure protocols are, e.g., TCP/IP Sockets, CORBA, or MoM-based solutions based on protocols such as STOMP, OpenWire, MQTT, or Web-sockets.

In the talk, we will first contrast key aspects of these formats and protocols. Based on a conceptual assessment and practical results, we selected the combination of JSON UTF-8 String byte arrays for serialization and the MoM protocols OpenWire [11], STOMP [12], MQTT [7], and STOMP over WebSockets [6] as communication infrastructure.

In our scenario, all protocols can be used equivalently as they are all supported by the ActiveMQ broker and the broker forwards between them. However, while the broker supports automatic exchange of messages between these protocols it showed that even simple combinations suffered from conversion problems.

As simple example we consider the exchange of String messages between OpenWire and STOMP. The broker falsely converted STOMP StringMessages to OpenWire ByteMessages because of misinterpreting a header field, while the opposite direction worked. More complex conversions pose even more potential pitfalls.

By using byte-array-based messages only, all so far encountered conversion problems could be avoided and messages can be transparently exchanged between the supported protocols. Thus, the selection of the MoM protocol only depends on the availability of libraries and implementation effort.

In addition, JSON-based serialization is well supported in many programming languages. Moreover, even if no JSON de-/serializer implementation is available it can typically be easily implemented.

In the VR- and AR-related studies, it showed that these capabilities, having a well-supported light-weight implementation serialization format and transparent support and exchange between MoM protocols, significantly eased development and integration. E.g., the development of a key component was done in Python with STOMP as protocol because of the availability of well-featured libraries. The VR application was based on the Unity 3D/VR framework and thus was implemented in C#. Because of the availability of a well-featured C# library, OpenWire was used as protocol for the VR application. Furthermore, we performed experiments with web-based VR, which used STOMP over WebSockets for connecting to our integration infrastructure.

On the transition from VR to AR, the OpenWire library had to be replaced with an MQTT library due to implications of the employed computing platforms, a standard gaming computer for the VR application and the Microsoft HoloLens for the AR application. This transition was easily possible simply by replacing the OpenWire library with an MQTT library, adjusting the MoM-related code in the AR application to use the new MQTT library, and enabling MQTT in the bowerick broker by adjusting a simple command line option. All other parts and components, such as exchanged messages, data format, de-/serialization, etc. could remain the same.

Furthermore, our development was eased by the functionality of bowerick to act as interactive Command Line Interface (CLI) client. The interactive CLI allows, e.g., to send and receive messages with all supported protocols, which helped in run-time testing and debugging of the system.

From our experience gained during the AR and VR studies, we consider the discussed approach very helpful. The development was not hindered by unnecessary limitations, conversions, or adapters. Instead, we could select the best tools, such as programming languages or libraries, for the particular use case, reduce boiler plate code, and focus on the actual functionality.

While we focused on development and integration effort, performance may still be an important factor. However, performance aspects should be considered on a case-by-case basis. For assessing performance aspects, bowerick offers benchmark functionality such as customizable message generators or a configurable benchmark consumer.

In future, we consider that MoM-based infrastructures will be increasingly used, e.g., EGS-CC uses Apache ServiceMix [13], which includes Apache ActiveMQ, as part of its infrastructure [14]. New developments at the German Space Operations Center (GSOC) consider the selection of the MQTT MoM protocol for integration [15]. CNES ISIS employs ZeroMQ, another MoM [16].

[1] Gad, R., “GitHub - ruedigergad/bowerick: Easing simple Message-oriented Middleware (MoM) Tasks with Clojure (and Java),” Online: https://github.com/ruedigergad/bowerick, last accessed 2018-06-13.
[2] The Apache Software Foundation, “Apache ActiveMQ – Index,” Online: http://activemq.apache.org/, last accessed 2018-06-13.
[3] The Eclipse Foundation, “Jetty - Servlet Engine and Http Server,” Online: https://www.eclipse.org/jetty/, last accessed 2018-06-13.
[4] Pivotal Software, “Getting Started · Using WebSocket to build an interactive web application,” Online: https://spring.io/guides/gs/messaging-stomp-websocket/, last accessed 2018-06-13.
[5] Lombardi, A., “WebSocket, Chapter 4. STOMP over WebSocket,” O’Reilly Media, Inc., Sept. 2015, Online: https://www.safaribooksonline.com/library/view/websocket/9781449369262/ch04.html, last accessed 2018-06-13.
[6] The Eclipse Foundation, “Eclipse Paho - MQTT and MQTT-SN software,” Online: https://www.eclipse.org/paho/, last accessed 2018-06-13.
[7] MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.
[8] European Cooperation for Space Standardization, “Ground systems and operations - Telemetry and telecommand packet utilization,” Online: http://www.ecss.nl/wp-content/uploads/standards/ecss-e/ECSS-E-70-41A30Jan2003.pdf, last accessed 2018-06-13.
[9] The Consultative Committee for Space Data Systems (CCSDS) “MISSION OPERATIONS SERVICES CONCEPT,” Online: https://public.ccsds.org/Pubs/520x0g3.pdf, last accessed 2018-06-13.
[10] Gad, R., Valadas, M., Graf, H., Olbrich, M., Keil, J., Sarkarati, M., Bamfaste, S., Nicolini, F., Cowley, A., “Assessing the Potential of Virtual Reality for Improving ESA’s Space Operations,” SpaceOps 2018, May 2018.
[11] The Apache Software Foundation, “Apache ActiveMQ – OpenWire,” Online: http://activemq.apache.org/openwire.html, last accessed: 2018-06-13.
[12] STOMP Spec Group, “STOMP - The Simple Text Oriented Messaging Protocol,” Online: https://stomp.github.io/, last accessed: 2018-06-13.

[13] The Apache Software Foundation, “Welcome to Apache ServiceMix!,” Online: http://servicemix.apache.org/, last accessed 2018-06-13.
[14] Schiller, T. M., “EKSE: A Command Line Interface for EGS-CC based Systems,” SpaceOps 2018, May 2018.
[15] Gaertner, S. A., et al., “Rethinking Ground Systems: Supporting New Mission Types through Modularity and Standardization,” SpaceOps 2018, May 2018.
[16] Gélie, P., Pasquier, H., Labrune, Y., “CNES Mission Operations Systems roadmap: towards rationalisation and efficiency with ISIS,” SpaceOps 2018, May 2018.

Primary author: GAD, Ruediger (Terma GmbH, Germany)

Presenter: GAD, Ruediger (Terma GmbH, Germany)

Session Classification: Talks

Track Classification: Ground Networks, Launchers, and Operations