Project information

Project title	Coordinating Optimisation of Complex Industrial Processes
Project acronym	COCOP
Project call	H2020-SPIRE-2016
Grant number	723661
Project duration	1.10.2016-31.3.2020 (42 months)

Document information

Deliverable number	D3.7
Deliverable title	Software architecture description for the runtime system (update)
Version	1.0
Dissemination level	Public
Work package	WP3
Authors	TUT
Contributing partners	TUT, BFI, IDE, VTT
Delivery date	26.9.2018
Planned delivery month	M24
Keywords	COCOP, system architecture, software architecture, software system, integration

This project has received European Union's Horizon 2020 research and innovation funding under grant agreement No 723661.

VERSION HISTORY

Version	Description	Organisation	Date
0.1	Document created based on previous D3.1	TUT	28.8.2018
0.2	Updates and figures	TUT	11.9.2018
0.3	Final preparation for internal review	TUT	14.9.2018
0.4	Improvements based on internal review	TUT	24.9.2018
1.0	Final document	TUT	25.9.2018

EXECUTIVE SUMMARY

COCOP aims for plant-wide monitoring and control of complex distributed processes. Control of complex industrial processes is typically dependent on computational models and having up-to-date data available to be used when optimising production and reducing the environmental impact. Many of these processes are distributed which is also the case for the control systems typically used at plants and at sub process levels. This makes it challenging to monitor and control the processes when information can not be easily communicated and direct integration of information systems is challenging. The developed architecture does not try to create a new control system but rather a concept for exchanging information and data in distributed processes, so that coordinating control applications can be created in a scalable and more flexible manner.

This document describes the COCOP run-time system architecture and details implementing integrations to existing systems as well as how control applications are created. This deliverable is the second updated version of two deliverables describing the software architecture, and builds on the initial architecture design of the first deliverable both for the internal composition as well as external integrations.

COCOP is implemented using a model-based, predictive, coordinating optimisation concept in integration with plant's automation systems. This means that existing local control systems are used to control sub processes, and the plant-wide control applications built using the COCOP architecture act as the coordinating layer transmitting events, restrictions, set points and targets for the plant-wide monitoring and control.

The general COCOP architecture is based on loose coupling of systems using a message bus architecture. The approach emphasizes a separation of concern of message semantics and communication protocols. Agreed message structures form the basis of information produced and consumed between system components. The communication architecture relies on the message bus as a broker. In addition to publish-subscribe communication, the architecture also considers request-response based communication to support integration to a variety of existing systems.

ABBREVIATIONS

Abbreviation	Full name
AMQP	Advanced Message Queuing Protocol
DCS	Distributed Control System
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
MES	Manufacturing Execution System
OPC UA	Open Platform Communications Unified Architecture
PLC	Programmable Logic Controller

TABLE OF CONTENTS

1 Architectural Requirements and Core Aspects

Plant-wide industrial process control and monitoring applications face challenges from system complexity as well as from multi-disciplinary networked system integrations. The distributed control systems used in production are typically vendor- and application-specific, which makes it challenging to integrate them into plant-wide control functions. Production processes may also span beyond one single plant, which introduces new requirements on system integrations in order to optimise production beyond the local processes. Further challenges may arise from the synchronisation of possibly conflicting data and events, e.g., combining estimated values with actual measurements. A traditional periodic control approach (that scans or queries for values and then decides its control actions) may prove complex, rigid and laborious to implement, especially if several point-to-point connections to other systems need to be implemented and maintained.

1.1 Decoupled Data-driven and Event-driven Architecture

The COCOP system architecture strives for scalability and enabling extensive utilisation of data either as refined information or as massive amounts of raw data. One of the main drivers is also decoupling the producers' information from their consumers through a centralised bus or - depending on the implementation - a pool of queues for data and events.

The event-driven approach comes from the requirement for reactive actions possibly also in real-time. In some cases, the end of a process step may trigger the start of another. For instance, when a batch of process finishes, it may enable the execution of another process step for that particular piece of material.

From a performance point of view, scalability is achieved by removing redundant queries to low-level systems and by having a centrally managed bus architecture in between producers and consumers of information and data. A centralised message bus allows for efficient scaling, caching and managing access independent of the consumers and producers. From a systems integration point of view, a centralised bus provides uniform access to the data thus reducing engineering effort for the plant-wide control applications. A centralised bus, however, can introduce a single point of failure, and additional preventive measures might be needed to ensure reliability, e.g., through redundancy or in restrictions how plant-wide monitoring and control applications are required to operate in case of manual intervention.

This kind of architecture benefits especially the development of new plant-wide monitoring and control applications in a platform-like development framework. From a development perspective, the burden is on integrating existing systems with a multitude of different communication interfaces to the bus and the event-driven approach. For this the development of adapters is needed to make heterogeneous interfaces compatible. With an adapter an existing system is wrapped behind a compatible interface as it is not intended to replace existing control systems but to integrate them into the new (COCOP) control environment.

Also, a full replacement would be expensive and create a significant hurdle towards implementation in plants, as it would imply the installation of new systems with new, unknown risks with potential production losses.

1.2 Enabling Distributed, Loosely Coupled Systems

Distribution is inevitable in industrial environments. The larger a production plant is, the more distribution there usually is. A production plant may even consist of several factories that have a dedicated task each.

Loose coupling is a fundamental design principle in COCOP. The goal is to enable an environment where various modules interact in a way that hides the internal details of each module. That is, the interfaces of modules should only expose their functionality or contents, not the underlying platform or implementation technologies. In addition, the integration technologies should be such that enable the modules to operate with minimal dependencies to one another, and their mutual integration should be easy.

Point-to-point integration should be avoided. Point-to-point means direct connections between systems, which often leads to a high number of direct dependencies between systems. Then, *scalability issues* would appear, as one of more systems are changed, updated or replaced in the future.

The downside of a bus approach is that producers of data and events do not know who and how information is utilised. As a result, additional information security measures may be needed in the message bus or even in the plant-wide control applications built on top.

COCOP focuses on the integration between systems rather than the systems themselves. Thus, COCOP does not provide engineering tools, run-time of controllers or similar ready-to-use applications. The COCOP project will provide complete control solutions for improving the operation of two pilot cases. Nonetheless, one of the main value elements generated in COCOP is the underlying concept focused on the integration between system. This concept enables an easy integration of the systems (as demonstrated in the project's use cases) by providing means for connecting events and data from the underlying systems, and building advanced plant-wide monitoring and control on top of this information.

1.3 Monitoring and Controlling Distributed Production

In production plants, a relaxing characteristic is that most systems are *static* in the sense that physical processing units do not change. In contrast, there are also environments where new network nodes may appear or leave the network at any time (i.e., the network is *dynamic*). Such environments include the fleets of mobile machinery or vehicles, for instance. Fortunately, the network nodes of a typical process control system do not appear or disappear arbitrarily. Naturally, any node may temporarily lose its connectivitity due to a failure, and any related network nodes should have a design that is robust enough to remain operable in such

occasions. Still, the set of available nodes is determined by the physical production equipment. That equipment is always installed, operated and uninstalled by plant personnel. Thus, the network of nodes does not change its structure very often, and the plant personnel commits those changes.

The information available and of value in control may change more frequently. This includes, for example, the deployment of new measurement systems or laboratory procedures, the use of improved models and algorithms more accurately estimating process values as well as other results available from processing vast amounts of data. One of the intentions with COCOP is to have a more flexible platform that can make use of external or processed information that can be used e.g. as a constraint in optimisation tasks.

1.4 Integration to Existing Systems

1.4.1 Integration Requirements

Multiple types of existing systems may be integrated with COCOP. The following table gives a few examples. Each example is explained after the table.

System	Examples of potential needs
DCS, PLCs, laboratory, quality control, other production-related resources	Measurement values from production processes
HTTP(S)	Hypertext Transfer Protocol (HTTP) over Transport Layer Security (TLS). Connecting to a wide range of services and information systems using the generic HTTP(S) protocol.
MES	Plant-level production coordination; plant-level production-related information, scheduling restrictions, availability of resources
ERP	Enterprise-level production coordination, resources and financial costs
Logistics	Constraints related to material flows, transport capabilities

DCS (Distributed Control System), PLC (Programmable Logic Controller) are systems that provide means for human workers to control complex, distributed production processes.

In some production plants, *laboratory systems* are utilised to provide information to help process control, e.g. adjust the process conditions. They analyse the substances that are involved in a production process. They may, for instance, estimate the concentration of specific substances, which may provide advice to reach closer-to-optimal operation. Laboratory systems are often also part of *Quality control* systems help reaching the desired quality of the

end product. With appropriate quality control the yield can be improved and also the amount of waste may be reduced, which increases productivity.

A *Manufacturing Execution System* (or *MES*) may provide production-related information at the plant level. Rather than controlling the individual low-level processes, MES systems coordinate plant operation as a whole. Concerning COCOP, MES systems may provide schedules or other coordinative information not available in the unit process level.

ERP (Enterprise Resource Planning) systems have business matters as the scope. From the production point of view, an ERP system may, for instance, receive production-related data to indicate production performance, but it may also coordinate production in the enterprise-wide scope.

Logistics is an important aspect in production optimisation. To optimise production, multiple logistics-related factors may be relevant, including the timely delivery of raw materials, intermediate products or an appropriate amount of materials in the storage.

1.5 Pilot Case Requirements

The COCOP architecture concept is developed for process industry in general. However, in the early stages of development, the design is focused on developing a solution to support the pilot cases in copper and steel production.

1.5.1 Copper

In a plant that refines copper from sulphide ores, there is a great degree of distribution and concurrency in production. The unit processes of such plant are operated locally, but the material flows between the unit processes create dependencies. The most important unit processes of a plant may be, e.g. (Schlesinger et al., 2011, pp. 1-12):

1. Flash smelt furnace (FSF)
2. Peirce-Smith converters (PSC)
3. Anode furnace and casting (AF)
4. Electrorefining
5. Melting, casting

Starting from unit process 1, each unit process provides the raw material for the next. Consequently, any processing-related shortcomings in a unit process may have adverse effects on the following phases.

Besides, efficient production requires further coordination. The waste (or slag) from each phase requires further processing, because it still contains some copper. Thus, a slag cleaning furnace (SCF) or a slag concentrator may exist, or the slag may also be circulated back to FSF. Furthermore, some of the end products are harmful to the environment. The ore being

processed can contain some heavy metals, and the gases released during each unit process can contain, for instance, sulphur dioxide. To process sulphur dioxide, the modern plants have off-gas handling and ventilation systems connected to an sulphuric acid plant to process the captured sulphur dioxide to sulphuric acid. As the capacity of the acid plant is typically limited and, on the other hand, the emissions of sulphur dioxide should be minimised, the operation of the acid plant may restrict the execution of other aforementioned unit processes. Within COCOP, the document *D2.3 System Requirements Specifications* (2017) explains the requirements that are considered in particular.

In the coordination of copper production, correct timing is essential. The execution of each task should be scheduled considering the entire plant. Event-driven operation is important, as changes in process states (such as start or end) may trigger or even hinder an operation. Still, due to slow process dynamics, the required resolution of response times is approximately one minute in most tasks. In comparison, some other production plants may require a timing accuracy of fractions of a second.

In summary, the operation of a copper plant requires coordination with a plant-wide perspective. Thus, for optimal operation, each unit process should be operated within plant-wide constraints.

1.5.2 Steel

The increase of the Chinese steel production has permanently changed the global market. The fierce competition has led several producers to close their production while the ones still operating have seen their margins drop. Even in this situation, European producers have showed higher resiliance due to the specialisation and the high quality of their products. Nonetheless, the European companies deal with the overcoming challenge of maintaining their competitiveness while reducing their emissions considerably and competing with countries with lower costs for energy and workforce.

In this framework, European producers are making great advances towards the optimisation of their production, targeting not only environmental improvements but also increased quality and resource use efficiency.

Two main process routes for steel production are Electric Arc Furnace (EAF) and Blast Oxygen Furnace (BOF). BOF and EAF processes both produce steel as the end product while having a different way of producing it. BOF plants use Iron ore, Coal and limestone as primary raw materials while the EAF use scrap steel as the main input. In the BOF, Sinter plant, Coke furnaces and blast furnaces are used for preparing the material which is then fed into the Blast oxygen Furnace. In the EAF route, the scrap metal is directly loaded into the electric arc furnace, thus having a much simpler route for the production. However, these two main routes represent only the first stage of the steel production, which is called primary metallurgy. From this point, the steel goes through a long set of processes before the end product is obtained. First, the steel goes to the secondary metallurgy where the composition of the steel is adjusted

to the specific needs of the client. After this, the steel is casted into subproducts (slabs, billets, ...) that are easier to work with. These subproducts are then used as input for the production of steel products.

As mentioned earlier, the main optimisation targets of steel plants are the more efficient raw material use (either scrap or primary materials) and the energy efficiency of the processes. Observing the European steel makers, a clear picture of the complexity of the production chain can be obtained. A regular steel plant can produce around 100 different grades of steel (different chemical compositions) and for each one of them, be able of producing several products (e.g. bars, sheets) in a wide variety of sizes. Accordingly, the steel plants can account for hundreds of thousands different end products being manufactured in their plants, including in this manufacturing process, several unit processes. With this scenario, it is clear that optimising the efficiency of the individual unit processes is not enough. Indeed, great efforts need to be made plant-wide for arranging optimal production planning and for ensuring final product quality.

These two aspects have a great impact in the overall efficiency of the plant, the first, by reducing the energy and time usage for sub-process adaptation for new orders and the second, reducing scrap generation and ensuring client satisfaction (high quality products). In the COCOP project, main focus will be in the optimisation of the final product quality (reducing rejection rate) through higher knowledge (and control) of the involved production steps and their parameters.

2 Platform Architecture Design

2.1 Message Bus Architecture for Scalable, Loosely Coupled Systems

The architecture design is illustrated in the following figure. The communication platform enables message exchange between various entities that do not have any direct mutual dependencies.

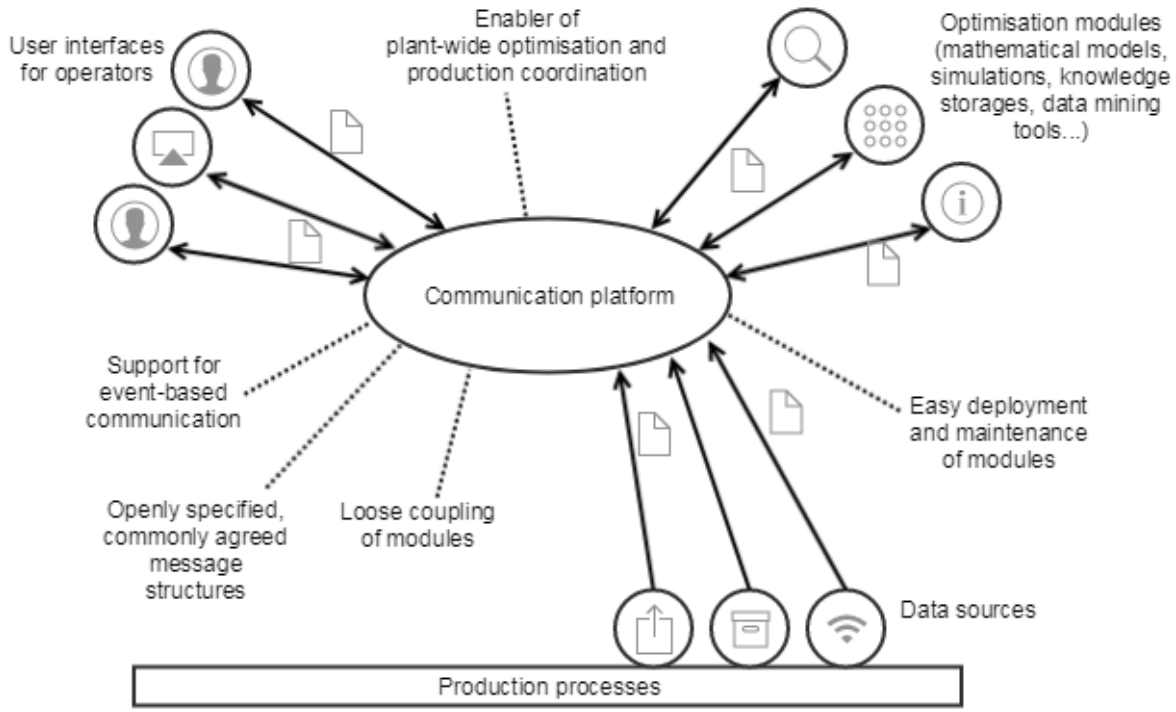

Figure 1 Architecture illustrated

COCOP aims for a message bus based approach mainly brokering data and event messages in contrast to Enterprise Service Bus (ESB). ESBs are typically used to deploy, connect and manage integrations between different information systems in the form of services, and they typically include support for several protocols including advanced transformation capabilities as well (Bhadoria et al. 2017). Including the necessary adapters from existing systems to the COCOP concept architecture the final technology stack can resemble that of common ESBs.

For COCOP the architecture is implemented using AMQP (Advanced Message Queuing Protocol, AMQP 0-9-1) as the messaging technology. As presented in *D3.5 Interface and protocol definitions,* AMQP provides good means for implementing the desired, loosely coupled architecture. The typical communication pattern in AMQP is *publish-subscribe*, which suits event-based scenarios that often occur in industrial production. In such scenarios, data sources generate events and data consumers react to them. In publish-subscribe, if any system (such as a supervisory system for human workers or an automated scheduler) wants to receive information of an event, the system would then subscribe for it. In such an event-based

approach, there is no need to query for state information, because the event reaches any party that is interested (in contrast to request-response, where each data delivery occurs on the request of the receiver). The event-based approach for message delivery is illustrated in the following figures. Once connected to a message bus, any node, such as a production system or an optimisation module, may receive events from other nodes. A single event may travel to multiple subscribers if appropriate.

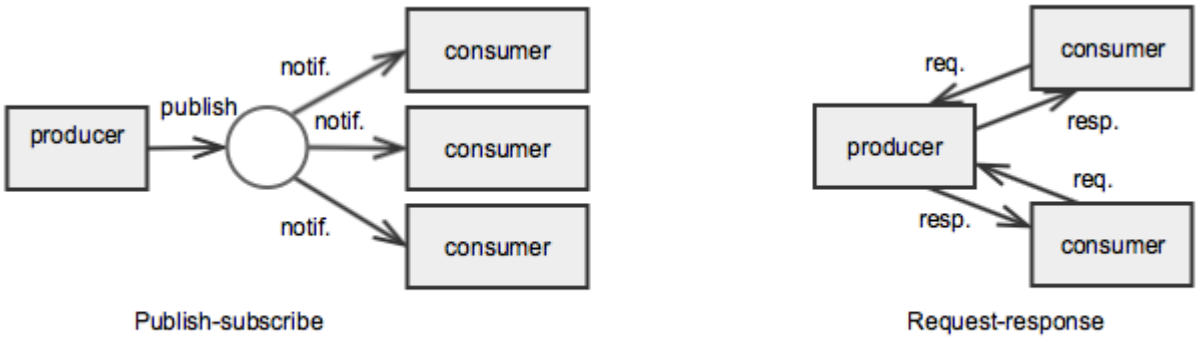

Figure 2 Publish-subscribe and request-response messaging.

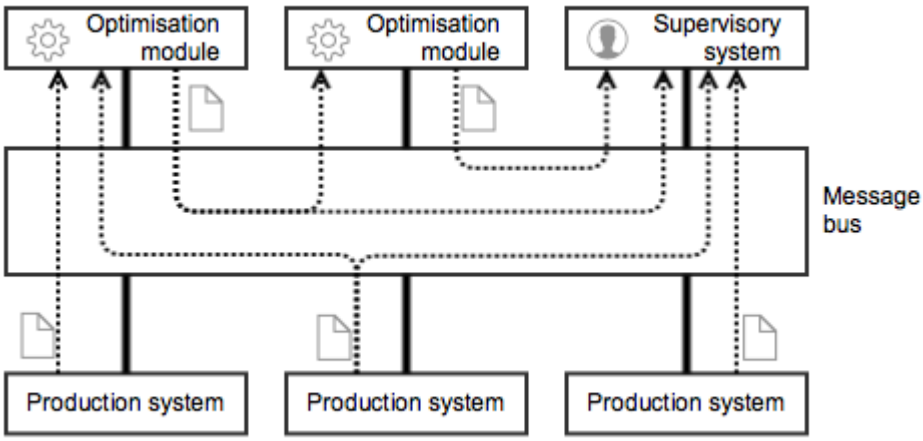

Figure 3 Example of production related entities communicating using a message bus.

AMQP is not the only message bus technology. For instance, ZeroMQ (2018) provides a message bus without any centralised node, which is simpler to implement but also lacks centralised coordination. Another alternative is Message Queuing Telemetry Transport (MQTT 2018), which is targeted to low-resource devices. For COCOP, AMQP was chosen due to its security features and centralised bus approach.

To implement the COCOP concept, application-specific customisation and specification is possible and even necessary. The COCOP architecture does not explicitly specify or limit what kind of messages are transferred between system components, although suitable message structures are presented in *D3.5 Interface and protocol definitions*. Similarly, COCOP does not enforce certain messaging patterns, although several patterns are supported.

2.2 Consumer Application State Considerations

A message bus communication can have different kinds of implementations and features. The most primitive ones transmit messages received from producers either to all consumers or only those that have registered for the particular messages. More advanced message bus implementations include advanced routing, caching and access management. Some implementations can even store messages for a certain duration and may, for example, retain the most recent messages for new consumers to receive. This is especially beneficial for new data or events that is seldom updated. From a plant-wide monitoring and control application development point of view, the state management responsibility is transferred to the particular applications and case implementations.

Implementing the event-driven approach will require either that 1) all operational data is periodically updated (even if no changes occur), 2) most recent messages are retained (as some events can happen very seldom), or 3) there are means to also query recent data and events. To enable integration with various systems, adapters are typically utilised to wrap existing systems to new interfaces. The first option of always updating values is straightforward to implement in adapters but increases the amount of data transferred unnecessarily. This also puts more pressure on consumer applications to know when and if to process new measurements. The second option is easier to implement in client applications but reduces the number of standard message buses and protocols available as many of such features are implementation specific. The third and the favourable option provides most flexibility and also enables request-response like behavior but incur additional functional requirements for adapters realising the query capabilities.

2.3 Supporting Client-Server Communication

It has been identified that many traditional systems currently in use operate on a polling or request-response basis. For example, Human Machine Interfaces (HMI) scan periodically for new values through queries and update the user interface displays accordingly. A message bus solution - depending on the implementation - may not store the last value, and therefore it is required for the COCOP concept to have request-response behavior available for client-server communication as well.

In industrial environments, there are also de facto protocols in use, such as OPC UA (OPC, 2015), that provide standard connectivity between systems. The standardisation may enable close to plug-and-play behavior when integrating components. Encouraging the use of standard solutions facilitates the interoperability of industrial control systems, and it is one of the COCOP architecture objectives. Interfaces and protocols suitable for implementing the COCOP system architecture are detailed in *D3.5 Interface and protocol definitions.*

To provide client-server communication - while using an asynchronous message bus as the core - requires new functionality for the adapters or systems directly integrated to the COCOP architecture. In addition to pushing new information onto the bus, the adapters need to listen

for specific requests from others and then be capable of returning the requested information onto the bus. The use of request-response wrappers is illustrated in the following figures.

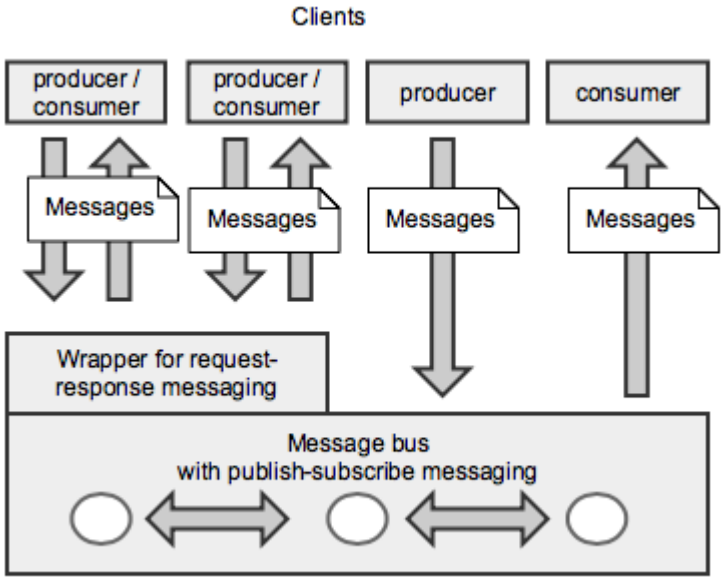

Figure 4 In order to realise request-response communication a wrapper to the message bus needs to be implemented.

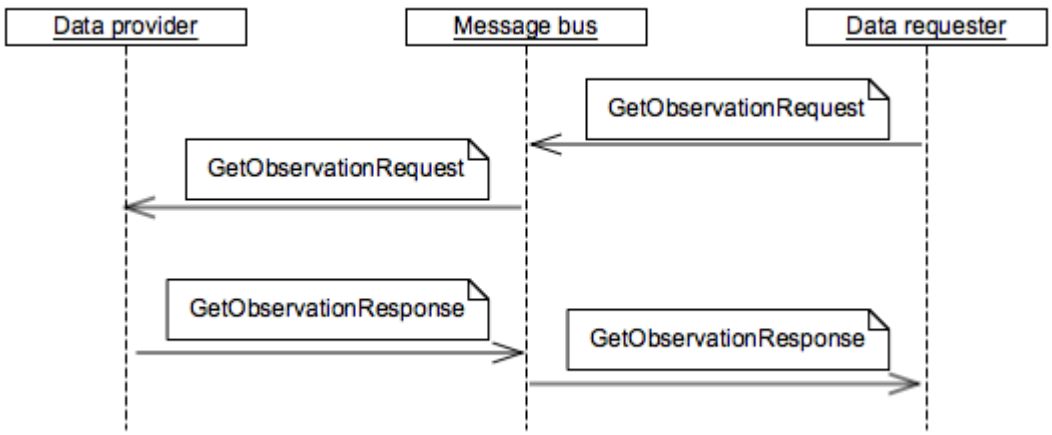

Figure 5 The message bus conveys request invocations between system components.

2.4 Integration Approach

To better support integration with various systems, adapters are utilised, i.e. an existing system is wrapped behind another interface (see the following figure). In principle, any existing interface may be wrapped with any other interface. Thus, to unify interfaces, adapters are a powerful approach; no matter how heterogeneous interfaces the existing systems have, it is possible to unify them. A similar approach has been documented as a generic design pattern in software design (Lasater, 2006, p. 206).

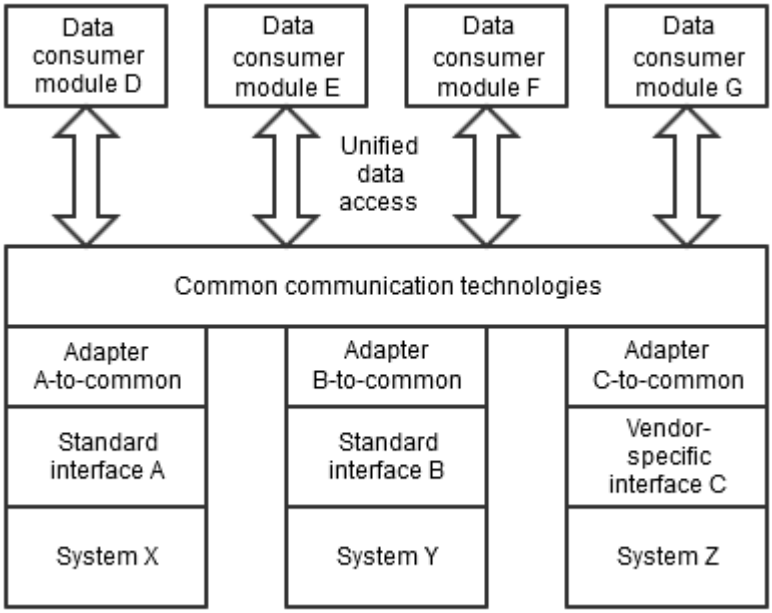

Figure 6 Adapter-based integration illustrated

To facilitate integration, and speed up the development of such adapters, client libraries have been developed that abstract the message bus. An API stack has been designed as shown in the following pictures to facilitate serialization of message structures, communication with the message bus as well as to enable request-response communication over the message bus. API SDK stacks have been successfully implemented in Java and C# for AMQP integration and the message structures presented in the following section.

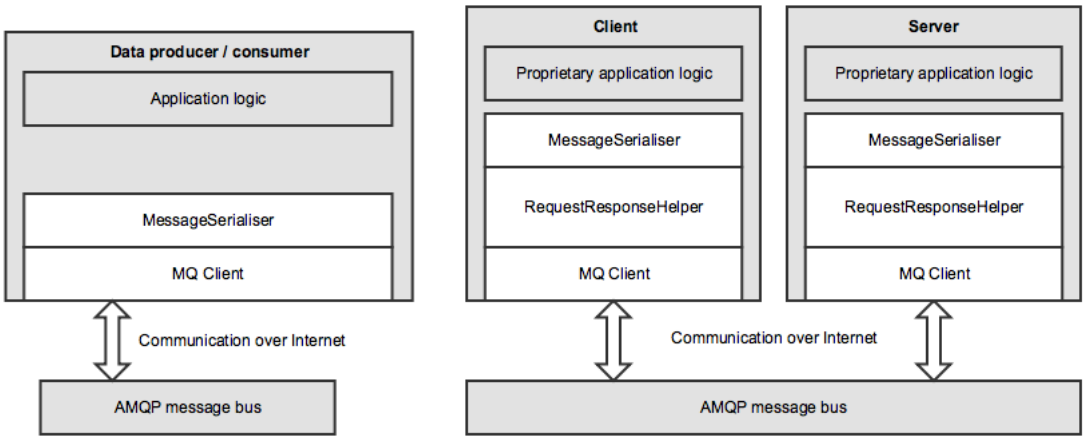

Figure 7 The SDK facilitates integration to the message bus and speeds up development. A request-response stack extension (as seen on the right) has also been designed to enable request-response communication over the message bus.

2.5 Entity Types

The COCOP system consists of *entities* (also called *components* or *modules*). The entities have various types as given in the following table. Each entity type is explained in the coming subsections.

Entity type	Purpose
Data source	To retrieve data from existing production-related information systems
Data mining tools	Data mining tools to discover information in production data
Models and optimisation	Modules that optimise production
Data output	Modules that provide information to external systems (e.g., operator interfaces)

2.5.1 Data Source Entities

Data sources provide production-related data from related information systems. The messages may cover, for instance, measurement values from the production process, both actual and historic data. Potentially, the data may also cover equipment state or process status information. For instance, the information about finishing a process step could trigger a scheduling operation for the next step. Also, condition monitoring and wear information is of importance as failing equipment can cause bad quality or complete disruption of the production proces.

2.5.2 Data Mining Tool Entities

Data mining and data analytics describe a group of known techniques used to extract information from data. Examples of these techniques are multivariate non-linear statistics, neural networks or decision trees. Beneath (offline) analytics, data-based or data-driven models can be developed that can be used, for example, as a soft sensor (regression model) or as decision support for the operators (classification models). Together with existing physical or first principle models, data analysis is often used for parameter estimation on the basis of historical data or in various combinations as a hybrid model.

Within the framework of COCOP, the Data Mining Tool Entity is used for the development of data-driven models. The necessary data is requested from the Data Source Entity via the message bus. All information collected is used indirectly during modeling in the Model Entity or in the design of the data preprocessing during the data request of the models.

2.5.3 Model and Optimisation Entities

Modelling and optimisation tools help the management of production processes. They may provide, for instance, production schedules or other assistance.

2.5.4 Data Output Entities

The data output entities refer to any items that expose information to higher-level systems, such as the graphical user interfaces of production operators. The actual data output is likely provided by computational models and other optimisation-related entities, but there may also be other output entities that gather or reformat the information supplied by other entities.

2.6 Message Structures

To enable communication between entities, appropriate message structures must be specified. It is possible to reuse a set of generic message structures to provide data from all entity types despite their varying requirements, provided that the coverage of the structures is sufficient.

In COCOP, message structures have been separated from the medium of transport by design, as shown in the following figure. On one hand, during design, separation helps to concentrate on one aspect only. In message design, that aspect is message structures, and in the design of communication functionality, the aspect is the actual communication mechanism. On the other hand, separation enables both flexibility and adaptability. This advantage realises whenever there is a need to change (or even replace) either the message formats or the communication protocol.

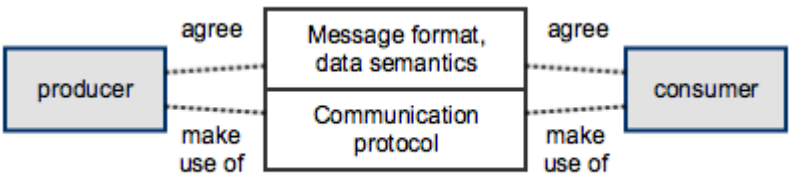

Figure 8 In COCOP the messaging semantics are decoupled from the communication protocol by design.

COCOP does not discriminate (there are no limitations) as to whether the message structures are utilised for internal or external communication. Due to the generic nature of the structures, there is no need for such limitations. However, to realize the integration and prove the conceptual architecture, some standards and message structures have been used in successfully implementing the communication.

As the format of messages, the standards in the following table are utilised. Each of the standards have been published by Open Geospatial Consortium (2018). Although the standards were designed for the geospatial domain, they suit well for the presentation of measurements and other industrial data. It is notable that these standards have been built on modules from other standards, so the actual number of related standards is higher. In addition, these

standards also have mutual relationships. For instance, the *Observations and Measurements* specification refers to the data structures of *SWE Common*.

Standard	Abbreviation	Purpose
Observations and Measurements (2013)	O&M	Delivery of measurements with related metadata
Sensor Observation Service (2012)	SOS	Request-response-based delivery of observations
Sensor Planning Service (2011)	SPS	Management of calculation tasks in the request-response manner
SWE Common Data Model Encoding Standard (2011)	SWE	Generic data structures
TimeseriesML (2016)	TSML	Data structures for time series

These message structure standards specify serialisation in XML (Extensible Markup Language). Although JSON (JavaScript Object Notation) would be a more lightweight alternative, XML has a wide tool support, so it is feasible for COCOP.

2.6.1 Messaging Examples

The following code provides an example of a measurement and the associated metadata (i.e., an *observation* based on Observations and Measurements (2013)). The message has been simplified to focus on the contents. For instance, namespace information has been omitted. *Phenomenon time* is the time when the measurement occurred, and *result time* is the time it became available. The actual measured value is 5.3 tons. This value is enclosed in the "result" element, which can hold a variety of structures depending on the need - here, it is a single measurement. Other data types, such as strings, enumeration values and timestamps are also supported. The elements other than *result* hold metadata, which has a similar structure regardless of the payload.

```
<OM_Observation>
 <description>The mass of a production batch</description>
 <name>Mass</name>
 <type href="http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement"/>
 <phenomenonTime>
 <TimeInstant>
 <timePosition>2018-02-05T12:10:13.00</timePosition>
 </TimeInstant>
 </phenomenonTime>
 <resultTime>
 <TimeInstant>
```

```
 <timePosition>2018-02-05T12:31:53.00</timePosition>
 </TimeInstant>
 </resultTime>
 <result xsi:type="MeasureType" uom="t">5.3</result>
</OM_Observation>
```

If the delivered payload is a composite element (i.e., holds multiple values), a type called *data record* would be utilised as the payload of an observation. This data record would be wrapped in an observation to enclose metadata. This data record structure comes from the SWE Common Data Model Encoding Standard (2011). It is notable that a data record can hold another data record, which enables even hierarchical data structures. Please see the following example that holds a sample ID, two measured concentration values and a nested data record.

```
<DataRecord>
 <field name="SampleId">
  <Count>
   <value>442</value>
  </Count>
 </field>
 <field name="Concentration1">
  <Quantity>
   <uom code="%"/>
   <value>5.6</value>
  </Quantity>
 </field>
 <field name="Concentration2">
  <Quantity>
   <uom code="%"/>
   <value>3.1</value>
  </Quantity>
 </field>
 <field name="NestedDataRecord">
  <DataRecord>
   <!-- Enclose another data record -->
  </DataRecord>
 </field>
</DataRecord>
```

The supported message structures are not limited to measurement data. As an example, there are also structures to remotely control long-running calculation tasks. The code below shows a request to update a calculation task related to the consumption of resources. The message

specifies new parameter values: energy input is 34 kW and raw material input is 0.3 tons per hour. The *targetTask* field identifies the calculation task. The message is based on the stardard Sensor Planning Service (2011).

```
<Update>
 <procedure>cocop/lca</procedure>
 <taskingParameters>
  <ParameterData>
   <values>
    <DataRecord>
     <field name="EnergyInput">
      <Quantity>
       <uom code="kW"/>
       <value>34</value>
      </Quantity>
     </field>
     <field name="RawMaterialInput">
      <Quantity>
       <uom code="t/h"/>
       <value>0.3</value>
      </Quantity>
     </field>
    </DataRecord>
   </values>
  </ParameterData>
 </taskingParameters>
 <targetTask>cocop/lca/tasks/314</targetTask>
</Update>
```

To illustrate the utilisation of messages, the following figure provides a scenario in copper production. In the scenario, a model for PSC (Peirce-Smith Converter) operation exchanges information with a coordinating scheduler and an online LCA (Life Cycle Assessment) module. Although PSCs operate autonomously, they receive coordinating schedules, as the operation of the copper plant is controlled in the plant level. Whenever the scheduler receives information about a state change in PSC, it refreshes the PSC schedule to consider the actual situation. The utilised message type is *Observation* (Observations and Measurements, 2013) for both schedules and state changes, although the actual payload varies and the observation type only holds metadata. The utilised messaging pattern is publish-subscribe, as the scheduler and PSC optimiser have previously signed up for certain messages from each other. The purpose of the online LCA module is to estimate environmental load. This LCA is run in the background. At the

start of each batch process, the PSC module submits a *TaskingRequest* (Sensor Planning Service, 2011) to the online LCA module to start LCA calculation, and the LCA module responds with a *TaskingResponse*. Later, whenever state changes occur in PSC, it informs the LCA module. For this LCA-related messaging, the request-response pattern is utilised. It is notable that, although not visible in the diagram, all messaging occurs via the message bus.

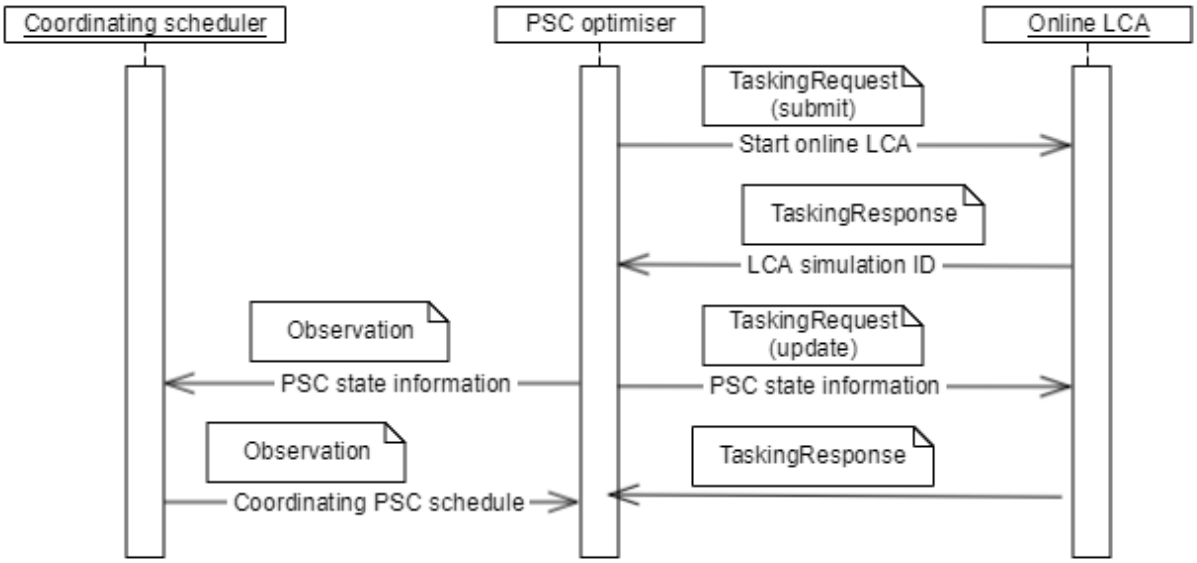

Figure 9 An example of message utilisation.

2.7 Information Security

User (or system) authentication should be required to access information transmitted using the COCOP platform. Using authentication, it is possible to prevent unauthorised access. Even in cases where the system is utilised within a single facility, *layered* security is desirable. Then, even if a malicious user were able to break into a system, authentication would form another obstacle. However, some production plants may actually run multiple facilities separated from each other. In such cases, some communication may occur via public network routes, which makes user authentication and encryption obligatory. In modern, distributed plants it is also increasingly common that several systems and system vendors communicate using the same networks.

Data encryption should also be applied in communication, and especially in multi-vendor or globally spanning networks. Then, there is no straightforward means to interpret any messages that could be captured. Furthermore, a suitable encryption technique also hides the traffic of user authentication. The importance of encryption increases in complex, geographically distributed production plants.

The following figure illustrates the security approach. Both user authentication and data encryption are required for secure communication.

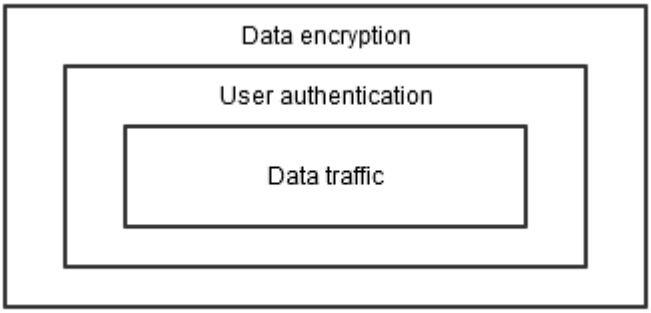

Figure 10 The security approach covers encryption and user authentication

Information security may be implemented either using the built-in features of the message bus (if available) or separately in an application specific manner. Using built-in features is obviously limited to the chosen message bus implementation; most implementations offer (application level) authentication and encryption, e.g. username/password or certificate and HTTPS. Application- or case-dependant authentication requires further implementation but increases the flexibility of the approach, as it can be integrated to any existing access rights management means being used. Embedding application-specific encryption of messages further improves the security, as not even the broker can interpret the message content.

2.8 Robust Module Design

The COCOP architecture does not specify the internal design of the modules that are connected to the system but only their external behavior. The modules may provide data to others or consume data. A module may also be both a consumer and a provider.

However, each module should be designed robust and independent (in accordance with industry practices). For instance, this means that the internal algorithms of a module must not get confused if some message is submitted multiple times for one reason or another. Although consistent messaging is a design goal as well, each module should process data in way not too fragile in terms of inconsistencies.

3 Conclusion

This updated architecture description document defines general COCOP architecture concepts, means for communication, integration requirements, and the identified internal entities of the COCOP system.

COCOP aims for plant-wide monitoring and control of industrial processes. These processes are typically dependent on computational models as well as having up-to-date data available when optimising production and the use of resources. Industrial processes are distributed - also globally - which makes it challenging to monitor and control the processes when information can not be easily communicated and the direct integration of multiple information systems is required.

COCOP does not implement a new control system but a model-based, predictive, coordinating optimisation concept that operates in integration with existing automation systems of a plant. The COCOP system architecture is developed as a platform for exchanging information and data in distributed processes, thus enabling coordinating control applications to be created in a scalable and more flexible manner.

The general COCOP architecture is based on loose coupling of systems using a message bus architecture in a data-driven and event-driven style. The arguments for this design are scalability, decoupling message producers and consumers, reducing direct system integrations, and facilitating building of new monitoring and control applications based on the conceptual architecture.

An imporant design principle is retaining flexibility in what kind of messages are transmitted in order to leave sufficient room for case specific applications to specify and build applications supporting the production. Therefore, messages and their structures are separated from the communication medium, i.e. allowing development to focus separately on semantics and the communication infrastructure or changing either one separately, if necessary.

The communication infrastructure of COCOP has been successfully implemented using AMQP.

References

"D2.3 System Requirements Specifications," https://cocop-spire.eu/content/deliverables, 2017.

"D3.1 Software architecture description for the runtime system," https://cocop-spire.eu/content/deliverables, 2018.

"D3.5 Interface and protocol definitions," https://cocop-spire.eu/content/deliverables, 2018.

Hästbacka D., Kannisto P., Vilkko M. Information Models and Information Exchange in Plant-wide Monitoring and Control of Industrial Processes. 10th International Conference on Knowledge Management and Information Sharing, September 18-20, 2018, Seville, Spain (KMIS 2018)

Hästbacka D., Kannisto P., Vilkko M. Data-driven and Event-driven Integration Architecture for Plant-wide Industrial Process Monitoring and Control. 44th Annual Conference of the IEEE Industrial Electronics Society. October 21-23, 2018, Washington DC, USA (IECON 2018)

Kannisto P., Hästbacka D. Asynchronous Communication Platform Concept to Coordinate Large-scale Industrial Processes. 16th IFAC Symposium on Information Control Problems in Manufacturing, June 11-13, 2018, Bergamo Italy (INCOM 2018)

Lasater, C. G., Design Patterns. Wordware Publishing Inc., 2006.

"OPC unified architecture specification part 1: Overview and concepts. Release 1.03," https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-1-overview-and-concepts/ (accessed 16.1.2018), OPC Foundation, 2015.

"Observations and Measurements. Version 2.0," http://www.opengeospatial.org/standards/om (accessed 3.9.2018), Open Geospatial Consortium, 2013.

F. Jammes *et al*., "Technologies for SOA-based distributed large scale process monitoring and control systems," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Montreal, QC, 2012, pp. 5799-5804. doi: 10.1109/IECON.2012.6389589

"MQTT," http://mqtt.org/ (accessed 14.9.2018), 2018.

Robin Singh Bhadoria, Narendra S. Chaudhari, Geetam Singh Tomar, The Performance Metric for Enterprise Service Bus (ESB) in SOA system: Theoretical underpinnings and empirical illustrations for information processing, Information Systems, Volume 65, 2017, Pages 158-171, ISSN 0306-4379, https://doi.org/10.1016/j.is.2016.12.005.

Schlesinger, M.E., King, M.J., Sole, K.C., and Davenport, W.G., "Extractive Metallurgy of Copper," Elsevier, 2011.

"Sensor Observation Service," http://www.opengeospatial.org/standards/sos (accessed 3.9.2018), Open Geospatial Consortium, 2012.

"Sensor Planning Service," http://www.opengeospatial.org/standards/sps (accessed 23.8.2018), Open Geospatial Consortium, 2011.

"SWE Common Data Model Encoding Standard," http://www.opengeospatial.org/standards/swecommon (accessed 3.9.2018), Open Geospatial Consortium, 2011.

"TimeseriesML," http://www.opengeospatial.org/standaropends/tsml (accessed 3.9.2018), Open Geospatial Consortium, 2016.

"ZeroMQ," http://zeromq.org/ (accessed 14.9.2018), 2018.