Design considerations introducing analytics as a “dual use” in complex industrial embedded systems

Daniel Hallmans
Hitachi-ABB Power Grids
Ludvika, Sweden
daniel.hallmans@hitachi-powergrids.com

Kristian Sandström, Stig Larsson, Niclas Ericsson
RISE Research Institutes of Sweden
Västerås, Sweden
{kristian.sandstrom, stig.larsson, niclas.ericsson}@ri.se

Thomas Nolte
Mälardalen University
Västerås, Sweden
thomas.nolte@mdh.se

***Abstract*—Embedded systems are today often self-sufficient with limited and predefined communication. However, this traditional view of embedded systems is changing through advancements in technologies such as, communication, cloud technologies, and advanced analytics including machine learning. These advancements have increased the benefits of building Systems of Systems (SoS) that can provide a functionality with unique capabilities that none of the included subsystems can accomplish separately. By this gain of functionality the embedded system is evolving towards a ”dual use” purpose. The use is dual in the sense that the system still needs to handle its original task, e.g., control and protect of an asset, and it must provide information for creating the SoS. Larger installations, e.g., industry plants, power systems and generation, have in most cases a long expected life-cycle, some up to 30-40 years without significant updates, compared to analytical functions that evolve and change much faster, i.e., requiring new types of data sets from the subsystems, not know at its first deployment. This difference in development cycles calls for new solutions supporting updates related to new requirements inherent in analytical functions.**

In this paper, within the context of ”dual usage” of systems and subsystems, we analyze the impact on an embedded system, new or legacy, when it is required to provide analytic data with high quality. We compare a reference system, implementing all functions in one CPU core, to three other alternative solutions: a) a multi-core system where we are using a separate core for analytics, b) using a separate analytics CPU and c) analytics functionality located in a separate subsystem. Our conclusion is that the choice of analytics information collection method should to be based on intended usage, along with resulting complexity and cost of updates compared to hardware cost.

***Index Terms*—embedded systems, systems-of-systems, analytics, data gathering, data collection, long life time**

I. Introduction

Historically, most embedded systems have been designed as stand-alone self-sufficient systems with a limited and predefined connectivity to higher level systems, e.g., connection to a Human Man Interface (HMI) or dispatch center via a Gate Way Station (GWS). These types of systems have been engineered, verified and validated to Control and Protect (C&P) assets in a factory, vehicle, power plant, energy transmission etc. This often creates a complex installation containing several subsystems closely engineered and verified together over long time, sometimes with several years from first design until the system is taken in to use. New technologies, e.g., communication and cloud technologies, have given us a possibility to create a next generation of systems, i.e., Systems of Systems (SoS), that enable new functionality relying on advanced analytics using Machine Learning (ML) and Digital Twins, to be added to already existing installations. A SoS is defined as creating new functionality enabling unique capability that none of the subsystems can accomplish on its own, ISO/IEC/IEEE 21839 [1]. Several different initiatives around us are driving suppliers in the direction of using the information available in subsystems, e.g., Industry 4.0 [2], Smart manufacturing, and Smart grids [3], to increase, for example, flexibility in manufacturing, revenue, business opportunities, new business models, and other competitive advantages.

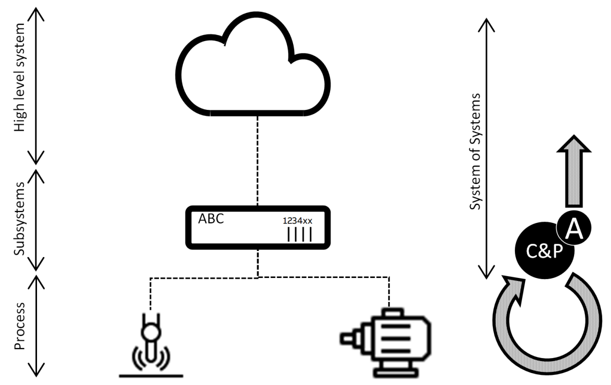


Fig. 1. A subsystem controlling and protecting devices in a process at the same time as providing data to a system on a higher level.

Subsystems in an SoS setup have a dual purpose; to control and protect an asset, and to provide information to the higher level system, Fig. 1, e.g., to provide information to an analytics service. An example of such a usage could be a sensor value, e.g., flow measurement. Such a sensor value is used by the subsystem control loop to control the output to a water pump motor, and the same sensor value could also be used in an advanced analytics function in the higher layer SoS, e.g., to predict the need for a future higher flow. A prediction like this is probably not only based on the single flow or voltage measurement from one object, but is rather derived using information from several different sources that are using the water or energy further down in the system, or other parameters that could affect usage. The challenge for the subsystem architecture will be to provide the information with sufficient quality, since the quality of the analytic results is direct proportional the the quality of the measurements. The demand from the customer to support collecting additional information from the subsystem will arise when the added value of analytics generates a value that may be similar to that of the subsystem or higher. This can for example be achieved by using thousands of sensors in ML calculations, in order to offer new and innovative insights in to the whole SoS, and by that giving a competitive advantage or new business opportunities.

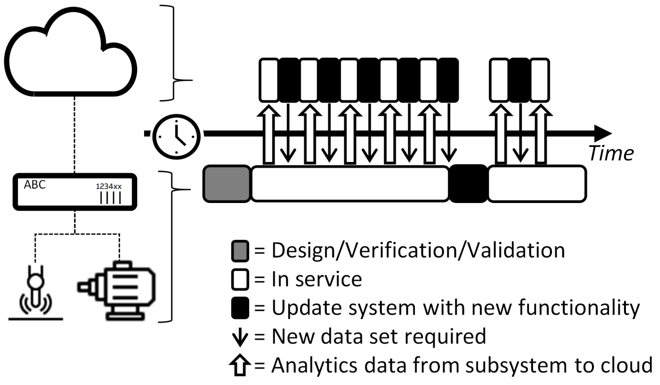

Time
ABC
1234xx
= Design/Verification/Validation
= In service
= Update system with new functionality
= New data set required
= Analytics data from subsystem to cloud

Fig. 2. The SoS system has a different life cycle, shorter, vs. the subsystem, longer. Each update of the SoS analytic will require a new set of data.

The complexity increases since the life-cycle and update frequency of the high-level system will not fully match the connected subsystems, consisting of a few to thousands of subsystems. Fig. 2 shows an example where the analytics requirements of new types of data sets are much higher than what is provided by the update frequency of the subsystem. The life-cycle difference becomes even more significant in the case of systems that have an expected long life cycle up to 30-40 years with only a few planned updated/upgrades during its life-time. Such a system typically also require a long design time, including verification and validation that can take years until the system is deployed and commissioned at a customers' site before put into use. Only once the system is operational, years after its initial design, the higher levels of SoS analytics will be initiated and start to evolve, e.g., results from different data sets are evaluated and suggesting additional needs for new data sets to be collected in order to improve or enable the desired result. A typical data science framework [4] is CRISP-DM (*CR*oss *I*ndustry *S*tandard *P*rocess for *D*ata *M*ining). CRISP-DM includes six phases (business understanding, data understanding, data preparation, modeling, evaluation and deployment) along with a life cycle that is repeated iterative until the expected result is archived. Furthermore, since information from one subsystem can be used in several other systems, the life cycle becomes even more complex.

A large extent of future systems, and also present, will not be able to avoid handling a "dual usage" of both the original and local C&P functionality together with becoming a future enabler of, at first design time unknown, analytics in a SoS. As a consequence careful considerations must be made concerning and preparation for dual use already at first design time, to avoid future expensive redesigns and costly verification of the complete systems. Therefore, in this paper, we have investigated the challenge of the dual purpose use of an embedded system and how different design solutions will effect the result.

The outline of the paper is as follows: Section II introduce related work. Section III describes major challenges with designing a subsystem that should be able to handle future analytics requirements. Section IV describes the concepts, data gathering and pre-processing. Section V analyzes four different solutions, and Section VI includes a discussion of the results. Finally, Section VII concludes the paper and point out future work.

II. Related work

The field of SoS and related topics is large and includes a number of different research areas, and several of them touches the different challenges connected to "dual use". On a higher level the ISO/IEC/IEEE 21839 [1] describes the life cycle considerations for a subsystem, i.e., System of Interest, when it is part of a SoS. Focus on its functionality, different life cycle stages and the concern that need to be taken to the SoS during the stages. [5] combines the utilization, support and retirement stage, from ISO/IEC15288 [6], to a constant evolution that better describes the handling of larger systems that has a 30-40 years lifetime with an evolution stage. This stage then also need to handle changes in the "dual usage" in parallel to the evolving SoS systems. In [7] a Survey is presented on Concepts, Applications, and Challenges in Cyber-Physical Systems (CPS) and they conclude that existing legacy systems have limited awareness of the CPS requirements, and that revolutionary design approaches are necessary to achieve the overall system objectives. The importance of real-time data, volatile, and nonvolatile, e.g., relevant layouts, specifications of machinery, for digital twins are presented in [8], vs. using common tools for process improvements. Similar examples can be found in [9] where access of information is a key challenge concerning external control loop, system predictability to be able to handle real-time tasks and adaptability. [10] also highlights the retrieval of information as a challenge, in this case for a Cyber-physical production system.

In [11] the subject of time series data analysis is discussed and pointing out that it is used in almost every scientific field, i.e., measurements are performed over time. Embedded, and the challenges that exists with time series data. Embedded systems and its resource constraints and the need for user interaction to allow for dynamic exploration and refinement of the solutions.

From a subsystem implementation point of view real-time containers could be a promising solution to being able to change functionality during run time. [12] presents a survey of different alternatives, from the first container virtualization on Linux. Identified challenges and future research that is needed in the areas is real-time behavior, tools and container to container communication. [13] evaluates the usage of Docker containerization in a real-time application and, based on experimental results, can show that it meets more soft real-time requirements with an added processing delay. Service oriented architectures (SOA) could also be part of a solution to allow for offering functionality to other system, [14] extends the traditional SOA to a Real-Time SOA to better be able to meet timing constraints. Micro services [15] extends the SOA concept even further with smaller sizes giving high flexibility, modularity and possibility to evolve.

To achieve the ”dual usage” we need to combine the results from the different related fields, e.g., cloud analytics, system evolvability, implementation advantages with virtualization of containers, and at the same time accomplish with out effecting the real-time and deterministic behavior of the control and protection system.

III. Problem description

In systems with long expected life cycles we have identified two major challenges when it comes to analytics and data gathering, often also refereed to as data collecting. In such systems all requirements are not fully known at design time but must be anticipated. The challenges are:

1) High-quality analytics require high quality data. Sampling rate, measurement accuracy, timing, and time stamping are examples of parameters that must be selected to allow future processing, both local pre-processing and remote processing by an analytics service. High quality analytics will require data gathering close to, and in parallel with, processing in the sub-system. Thus, analytics functions will compete with the original C&P functionality for the same hardware and software resources.
2) Life cycle of the analytics’ requirements will differ compared to the original systems’ life cycle that is significant longer. Embedded systems in a larger complex installation, e.g., factories, power transmission systems, and power plants, have undergone several stages of design, verification and validation before it has been taken to use. These stages are sometime lasting for years, involving several subsystems that need to co-exist and will be in use for 10-15 years until any major update is scheduled. Analytics on the other hand starts to provide useful information once the system is taken in to use, and will then evolve by applying different analytics solutions the following 10-15 years, requiring different sets of data during this time. New sets of data will then also require updates to the data gathering or pre-processing functionality in the embedded system at a rate that is much higher than the update frequency of the embedded system core functionality. Providing all types of data from the start, i.e., every input, every state of the control, and outputs, will not be possible in a larger system or not even in a smaller system, so a configurability of what information to gather is needed.

IV. Data gathering and pre-processing

Simplified, there exist three types of ”live” information in an embedded system that is valuable to gather for further analysis; *Inputs* from sensors, e.g., fluid speed, internal *processed* information generated from different states and calculated values in the C&P logic, and *outputs* to actuators, e.g., control signals to an electrical motor, or trip signals to a high voltage circuit breaker.

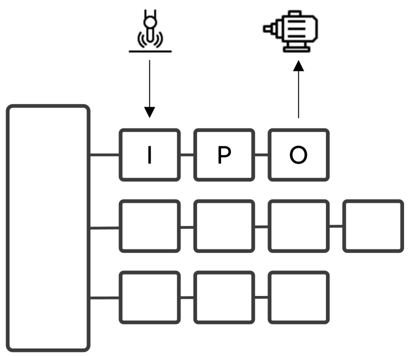


Fig. 3. A three level scheduler with different tasks connected. Task ”I” takes inputs from a sensor, ”P” processes the information and ”O” sends the processed information to an actuator.

In Fig. 3 we are representing these three different types as different tasks (I, P and O) in a scheduler with three priority levels. We have placed them on the fastest level, i.e., shortest cycle time, but in reality the valuable information is located in all tasks. In addition, pure static information is of interest to fully understand the system configuration, e.g., used hardware and software versions, in order to compare system information against each other. A use case for data gathering that should not be forgotten is advanced fault tracing or as input for verification and validation of the system. The use case including gathering of parameters on a embedded system oriented level, e.g., cache misses, scheduling jitter, missed deadlines, that would be to a high value for an embedded engineer but probably not to a system engineer, e.g., a power systems designer that are more interested in energy flow in the transmission network rather than the scheduling of tasks in the embedded system.

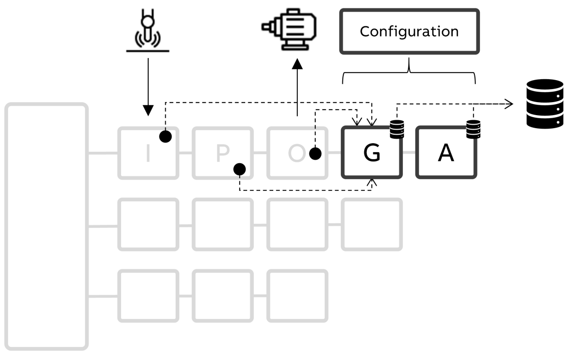


Fig. 4. Additional to the ”IPO” tasks a data gathering task,”G”, and pre-analytics,”A”, has been added with configuration and external storage.

In Fig. 4 we have added separate tasks for data gathering, G, to the same scheduler and A for pre-processing of analytic data, to reduce the amount of data that is needed to be sent from the subsystem. G and A are both equipped with local memory where information can be temporally stored before transmitted with a lower priority to an external storage. Additionally, a configuration part is added to allow for changing what information that is collected, from I, P and O, and also pre-processed. Adding the gathering tasks directly to the scheduler enables that all priority signals can be sampled at the same speed as they are generated. If there are more, slower or lower priority signals, they could also be gathered by tasks that are running slower or on idle priority but then without guarantee that all data is captured.

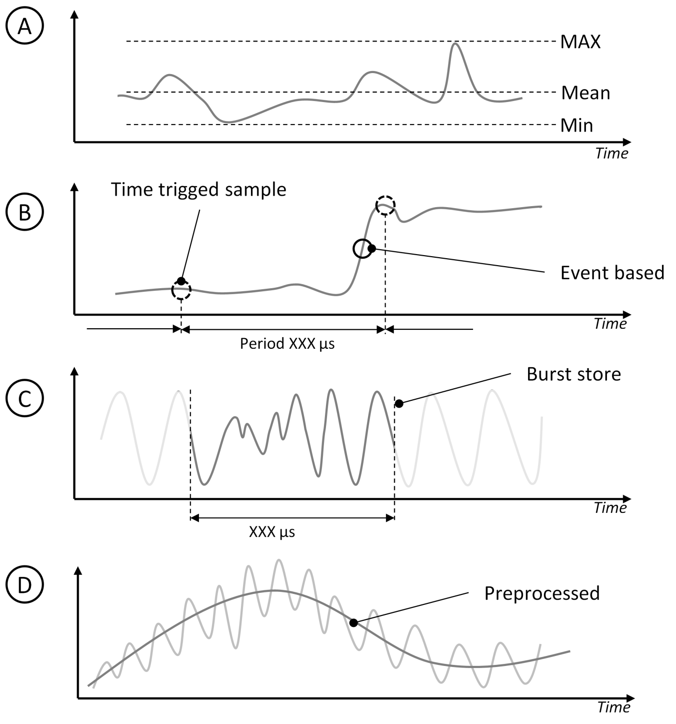


Fig. 5. Different examples of methods for data gathering to balance the accuracy need and at the same time reduce performance requirements, e.g., storage space.

To improve the quality and to reduce the size of the gathered information, different types of gathering functions must be implemented. If not, all samples, all the time, are streamed or stored with the highest possible frequency, sometimes resulting in an extreme amount of data to handle. Fig. 5 provides a few examples of possible pre-analytics functions, e.g., only storing the maximum (MAX), mean and minimum (MIN) value for a specific signal during a time interval, e.g., during an hour. This would give more information about the signal behavior, e.g., noise level, compared to only store a single value each hour. Storing a burst of data with a high frequency, and not continuously, would still allow for frequency analyses, but with a limited scope in a later stage. The same analyses can be configured to be executed in the subsystem by utilizing pre-processing, Fig 5 alternative D, and by that reducing the amount of information that is sent and stored. Configurability of what to store, i.e., from I, P and O tasks, and how to store it, i.e., the amount of possible pre-processing functions, will be a key function going forward to handle future use cases.

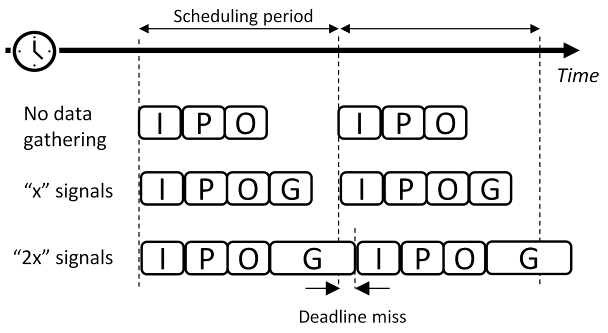


Fig. 6. Deadline miss, i.e., overload, when to many signals are gathered at each scheduled cycle. Task A is not included in the example.

Each of the different gathering methods, and the number of signals collected per time unit, will consume CPU resources, e.g., clock cycles and memory, to handle and by that interfere with the C&P functionality since it will be executed with the same priority. Fig. 6 shows the risk of "stalling" the system, in the example when two times, "2x", the number of signals are handled, giving deadline misses and risk of critical failures in the C&P part. In a system where the priority of the analytics' requirements are lower than the corresponding C&P requirements the "overload" would probably be dropped and reported to the SoS as "missing samples", but if the priority is equal the loss of value in the SoS level would be significant. This makes us to conclude that the amount of CPU resources for the gathering process must be predefined, e.g., the number of data storage buffers, already at the design of the system and by that it will also be included in the initial verification and validation process of the complete system to avoid any unexpected behaviors when analytics is enabled later on in the process.

A. Evaluated systems, criteria and method

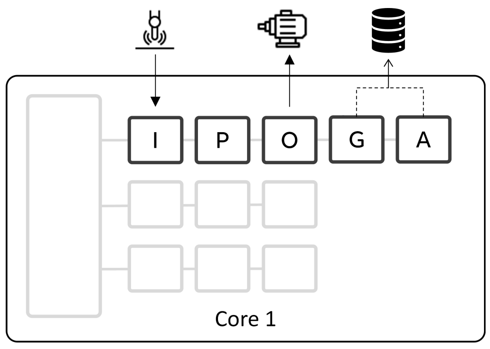


Fig. 7. One core reference system used in the evaluation.

As a reference for our evaluation we are using a setup where the C&P functionality and analytics are located in the same core, independent of if it is a single core or multi-core CPU, see Fig. 7. It is the most fundamental setup from a hardware point of view, since no additional hardware is needed, but with a high impact on the C&P scheduling from the analytics functionality. We will compare the reference system to three other alternatives: a multi-core system where we are using a separate core for analytics, using a separate analytics CPU located on the same hardware and last one where the analytics functionality is located in a completely separate subsystem.

To be able to compare the different alternatives we will use the Pugh method [16], i.e., we will give a relative score for the different alternatives in comparison to the reference architecture. The scoring will be S = same/equal, + = better and - = worse than the reference system. An alternative to using the relevant scoring in the Pugh method could be using more specific measurements for each system, e.g., differences in number of CPU cycles, memory bandwidth, and jitter. The problem with such an approach is that it will be much more hardware dependent since the exact figures will differ between different systems, e.g., some having a better memory bandwidth utilization vs. number of CPU cycles per function, since the CPUs in the embedded systems would range from, e.g., simple ARM M0 to high performance many-core systems based on the Intel Xeon architecture. Basing the comparison on a higher level of abstraction the Pugh method will give a direction and more detailed investigations can then later be used to find implementation details or further needed studies for a specific case. For the same reason we have not used details of specific implementations' positive or negative effect on, e.g., cache utilization, board space requirements, and power requirements.

G	Effect on C&P due to the needed gathering of information/data
A	Effect on C&P due to resources consumed by pre analytics functionality
LG	Possibility to handle different life cycles for the data gathering
LA	Possibility to handle different life cycles for the pre analytics functionality
C	Additional hardware cost for building the data gathering and pre analytics system
U	Possibility to update an already existing legacy systems with the proposed analytics.

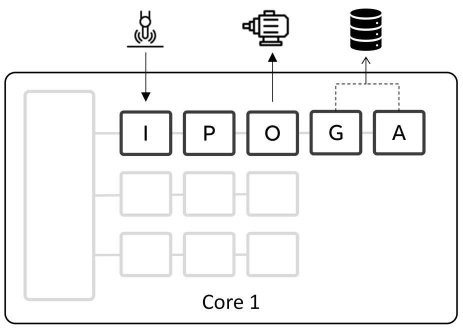
Fig. 8. Table with the different criteria that we have used for the evaluation.

In Fig. 8 we have presented the six different criteria that we have used to evaluate the different alternatives.

The availability of I and O, i.e., sensor data and actuator outputs, in different cores or external hardware will differ depending on the system setup. An example with using EtherCAT [17] for IO system data transfer in a multi-core system, with the EtherCAT master located in one core would have the I and O data available in that core and not in the other, or on the other hand if a core is used for communication and the other for processing the information will available to all if shared via a shared memory. The processed information, P, on the other hand will only be available in the core where it is produced and used. If access is needed in other cores the information must be transferred, or as an alternative, that will be complex and contain a lot of uncertainty, would be to replicate the complete C&P logic in another core based on I and by that produce P. For simplicity, in our reference system we assume that I, O and P are only available in one core, due to that for example the EtherCAT master relies in that core. Still in our comparison it does not change the evaluation since P, most likely, will only be available in one core and must be transferred.

V. Analysis

A. Reference system



A: Single core reference system

G	A	LG	LA	C	U
S	S	S	S	S	S

Fig. 9. Single core reference system.

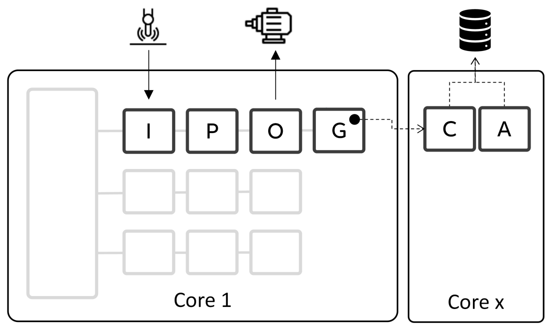
In the reference system, acting as the base line, all functions are executed on the same core. All information extracted from the I-, P- and O-tasks will then be fully accessible, and any functionality executed in the gathering task and pre-processing (task G and A) will have a direct impact on the available resources for the C&P part.

The life cycle for G and A will follow the rest of the system, i.e., in a simple system G and A can not be reloaded or restarted without affecting the C&P, instead any functional changes to G and A must be able to be handled online. Changes could then be done via, for example, different configurations of already implemented functionality. A more advanced implementation could include using of containers, e.g., Dockers, that could be modified during run time, or a real-time hypervisor allowing for parallel execution. Different scripting languages, e.g., µ-phyton, LUA, could also give a possibility to update the functionality independent on the C&P part, since the code then is interpreted in run time. Since the functionality is located on the same CPU and competing for the same resources, caution has to be taken during design on the potential effects on the C&P functionality. Some real-time code, e.g., docker containers, must be handled in such a way that it will not interfere with other real-time tasks in an unpredictable manner. What is technically possible will also be dependent on the subsystems' performance. A real-time hypervisor on an ARM M4 will probably not be as effective as implementing possible scripting features or predefined configurations.

No additional hardware is needed for the G and A tasks giving no additional hardware cost (C) and a possibility to ”only” change the software when a system need to be updated in the future, U, i.e., updating a legacy system with data collection functionality. Since no new hardware is added, additional effort must be added to handle the complexity of the software solution, but do not scale with the amount of updated units. A cost that also is valid for the other solutions on different levels.

In the summary table in Fig. 9 all evaluated criteria are “S” (Same) since it is our reference system.

B. Multi-core system with analytics on a separate core



B: Multi core system with analytics on a separate core

G	A	LG	LA	C	U
+	+	S	S	S	-

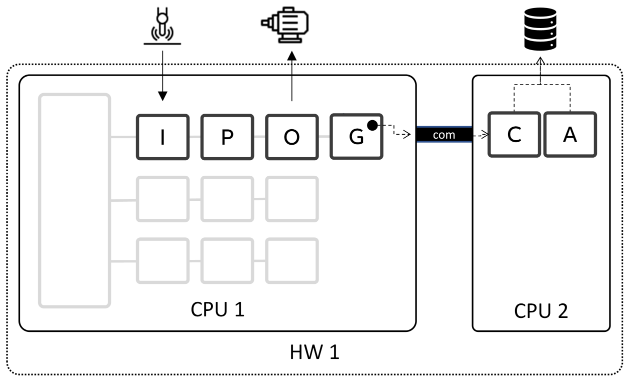
Fig. 10. Alternative B: Multi core system with analytics on a separate core.

Allowing the analytics functionality to use a separate core will give us a possibility to move part of the gathering of information, G, from the C&P and by that allow for a simpler configuration of what data to be streamed to the analytics core. Any preprocessing, e.g., calculation of maximum or minimum values, could then be handled in the analytics core without affecting the C&P system. Any limitation in the number of signals that can be transferred, e.g., over a common cache, and by that impact on cache misses, must be evaluated during design time for each type of system. Similar limitations are also valid for the other types of system setups, e.g., for the reference system when a value is stored by the CPU for later usage, since a set of information always must be transferred or stored in memory.

Since all functionality is still in the same CPU, but on different cores, the life-cycle handling of the analytics will be the same as for the reference system, e.g., the analytics life-cycle can be improved by using, e.g., a real-time hypervisor or containers, with the main difference vs. the reference system being that the C&P functionality is impacted, performance vise, less by the analytics part, A.

Hardware cost, C, will be more or less the same since no additional components are added to the system, rather choosing a CPU with more available cores to a slightly higher price. If this decision has not been made during design time, an upgrade of a legacy system, U, with this method is not feasible since a core is seldom left completely unused in a system.

C. Separate CPU in the same system



C: Analytics on a separate CPU in the same system

G	A	LG	LA	C	U
-	+	S	+	-	-

Fig. 11. Alternative C using a separate CPU for handling the analyzing tasks.

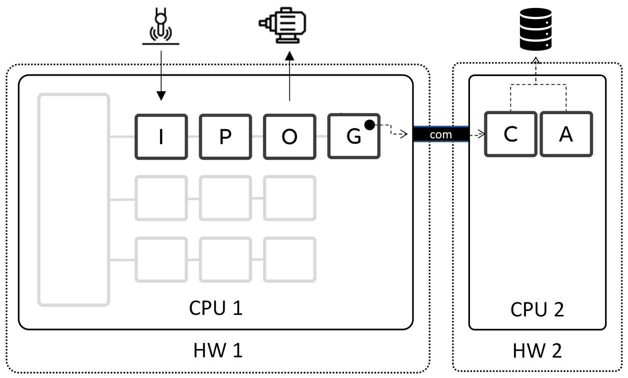
Separating the analytics part to another CPU located on the same hardware will enable better possibilities for a different life cycle, LA, i.e., analytics functions can be updated and restarted independent of the C&P CPU. The data gathering, G, is still within the C&P and by that requiring a more complicated external communication, e.g., CAN-FD, Ethernet, or an external dual port memory is also a possibility if the CPU is equipped with a communication bus that can handle the amount of data to transfer. Note that it will be more complex/slower than an internal memory transfer giving it a lower score for G.

Hardware cost, C, will increase due to the additional hardware devices needed, e.g., CPU, memory, and power supply, and at the same time making an upgrade, U, of a legacy system impossible since the hardware is not available.

D. Separate system for analytics

Placing the analytics CPU on a completely separate hardware has similar advantages and disadvantages as with alternative C, with the exception that the upgrade possibility becomes easier since all additional functionality, except for data gathering (G), are located on the external hardware.

Hardware cost, C, will be higher but could also be shared between different subsystems if one analytics hardware is serving several C&P systems, but still higher than the reference system. Communication bus performance, e.g., using EtherCAT, OPC-UA, between C&P and the analytic hardware will be important and could be a bottleneck pushing more or less of the processing to be done locally, affecting the performance of the gathering part G negative, or as a limitation to the amount of gathered signals and by that making all types of signals not available.



D: Separate system for analytics

G	A	LG	LA	C	U
-	+	S	+	-	+

Fig. 12. Alternative D where the analytic functionality is located in a separate system, i.e., outside the subsystem.

VI. Discussion

Collecting information that is intendend to be used for analytics need to be done with a data quality that is not limiting the evaluation of the results. In most cases that will imply adding the gathering functions already in the embedded systems next to the C&P functionality. All ”raw” signals and data cannot be collected but instead different methods need to be introduced to reduce the amount of data, e.g., by storing MAX and MIN values or by preprocessing the data such that a compact representation can be achieved, all in a way that future changes to the data gathering will not affect the C&P logic.

G	A	LG	LA	C	U	∑ S	∑ +	∑ -
A: Single core reference system								
S	S	S	S	S	S	6	-	-
B: Multi core system with analytics on a separate core								
+	+	S	S	S	-	3	2	1
C: Analytics on a separate CPU in the same system								
-	+	S	+	-	-	1	2	3
D: Separate system for analytics								
-	+	S	+	-	+	1	3	2

G = Effect on C&P due to the needed gathering of information/data
A = Effect on C&P due to resources consumed by pre analytics functionality
LG = Possibility to handle different life cycles for the data gathering
LA = Possibility to handle different life cycles for the pre analytics functionality
C = Additional hardware cost for building the data gathering and pre analytics system
U = Possibility to update an already existing legacy systems with the proposed analytics.

Pugh Score:
S = Same
+ = Better
- = Worse
Compare to the reference system

Fig. 13. Summary of the Pugh scores for the evaluated systems.

In Fig. 13 our evaluation result from the four different cases are presented along with their corresponding Pugh-scores. In summary the choice of solution will depend on the intended future use together with the stage that your system is in today, i.e., still at design stage or already in service. In addition it is important which functionality that will be prioritized, i.e., by adding weights to the different results as presented. An example could be a system that already is in service for several years and now needs to be updated/upgraded, U, with new analytic functions. The system will then not be able to add new CPUs to handle the analytics on the same hardware. Alternative C, or alternative B, is also not feasible since there are most likely no free/unused cores available. The most viable solution is then alternative A or D. The choice between A and D will then depend on what functionality that is prioritized: is it cost for hardware, i.e., alternative A with only implementing the gathering of data as a software function, or is it the higher flexibility and additional compute resources in alternative D that is most interesting. A limiting factor will also be the amount of hardware resources, e.g., CPU or memory, that is available in the legacy system for the ”dual usage”, setting a firm constraint to the amount of signals that is possible to handle.

If the system is still at design stage we have more options to tailor the solution to allow for future flexibility. Alternative B with a separate core for the analytic pre-processing has the advantage of reducing the complexity for the C&P part but the life-cycle issues are still the same as of a single core system. Technologies such as virtualization, containers, interpreted languages, could improve the flexibility by adding and changing functionality – independent on being a single- or multi-core system. Adding an additional CPU already at design time will improve the life cycle updates by, for example, allowing for restart/reload of that part but at a higher cost for hardware, and also required a larger physical space. Having a complete external Hardware, alternative D, will allow for even higher flexibility when it comes to improving local processing of analytics and future changes in functionality. This is also an option that is independent on the system being a brown field installation, i.e., the system is already available, or a green field system, i.e., a new installation.

All alternatives are still dependent on that the core C&P functionality is modified to provide the “raw” data samples and that it is implemented in a way such that it will not risk any C&P functionality, today or in any future update of analytics needs. Due to this, when the system is designed, caution must be taken to set the limits of number of maximum allowed samples, CPU resources, memory bandwidth etc., i.e., not overloading the system, but at the same time allow for configurability of what data to collect. We are today not aware of the exact set of signals needed by future analytics, and at the same time we are not able to store all possible signals. Instead the flexibility, future possible evolvability and configurability, will be important, but system limitations will be critical aspects to consider.

VII. Conclusion and future work

A majority of future embedded system will not be stand alone. Instead these embedded systems will be connected as subsystems to a higher level of systems and by that creating a SoS. The information provided from the subsystems will allow for the SoS to deliver new sets of functionalities that enable functionality larger than the sum of all installed subsystems. Advances in new technologies, e.g., Machine Learning, are pushing the limits and possible usage of a SoS to a level where the value that they will introduce will be equally or more important compared to the functionality of the original subsystems. To prepare for this, embedded systems created today need to consider this ”dual usage” already at design time to avoid expensive and time consuming modifications and verification later on. This is especially true for systems in a complex environment and/or for systems that have an expected long life cycle, up to 30-40 years. Even if it may be difficult to fully understand what data and information that may be needed from a subsystem in the future, precautions must be taken s.t. it is possible to make as much as possible of such information available for a potential future usage.

In this paper we have analyzed challenges when it comes to this “dual usage”, i.e., to handle it original intentions of C&P at the same time as gathering information for coming installations in a SoS. In this context we also analyze the effect of different life-cycle requirements for the C&P and analytics. To evaluate the impact of different implementations, and by that give us directions going forward with an implementation, we have investigated four different CPU configurations; A) where C&P and analytics are located in the same core, B) where they are separated on diffident cores in one CPU, C) where we divide them on two different CPUs on the same hardware, and D) where they are located in two separated systems. A multicore implementation would most likely be the best solution if the upgrade ability is not calculated or if it is prepared for already at design time.

In summary we cannot avoid connecting our systems together in the coming future, small or large, if we want to be able to fully benefit from future advanced features relying on, e.g., analytics. Such features are important and have the potential to provide a competitive edge of systems that are installed and allowed to evolve over time. To enable the dual functionality of subsystems discussed in this paper we need to prepare our system already at design time, i.e., to handle the local system functionalities at the same time as we provide a higher level system with information, to avoid costly and complex updates and verification later in the system life cycle.

A. *Future work*

In the context of the local system we will continue our research when it comes to how to efficiently share resources between different functions without having them to cause unpredictable interference among each other. Here we investigate the usage of hypervisors, containers, and different scheduling algorithms that are providing such guarantees. Since the amount of idle time in a system vary due to external factors inherent in usage shared resources, a flexible approach to pre-processing that allows for temporary overloads and catch-up in between seems to offer an interesting solution, i.e., a solution that changes between storage of ”raw” information vs. pre-processing of information, all dependent on the current situation of available resources.

From a system point of view also other parts of the system will be affected, e.g., network, storage, cyber-security, data modeling and configuration, since the understanding of the provided information will be a challenge that most likely also grows with the age of the system. A technology solution that would be interesting to look into, but not limited to, is Time Sensitive Networking (TSN), which is a promising technology allowing for priority sharing on the same network. In this case analytics could potentially be added at a later stage without disturbing the legacy traffic, if free resources are made available for such usage already at design time. Hence, a similar thinking as for the local data gathering, i.e., that it must be prepared already at design time.

References

[1] “Iso/iec/ieee international standard – systems and software engineering – system of systems (sos) considerations in life cycle stages of a system,” *ISO/IEC/IEEE 21839:2019(E)*, pp. 1–40, 2019.

[2] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[3] X. Yu and Y. Xue, “Smart grids: A cyber–physical systems perspective,” *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1058–1070, 2016.

[4] R. Wirth and J. Hipp, “Crisp-dm: Towards a standard process model for data mining,” in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, vol. 1. Springer-Verlag London, UK, 2000.

[5] D. Hallmans, M. Jägemar, S. Larsson, and T. Nolte, “Identifying evolution problems for large long term industrial evolution systems,” in *2014 IEEE 38th International Computer Software and Applications Conference Workshops*. IEEE, 2014, pp. 384–389.

[6] I. ISO, “Iso/iec 15288: Systems and software engineering—system life cycle processes,” *ISO, IEC*, pp. 24 748–1, 2008.

[7] V. Gunes, S. Peter, T. Givargis, and F. Vahid, “A survey on concepts, applications, and challenges in cyber-physical systems.” *KSII Transactions on Internet & Information Systems*, vol. 8, no. 12, 2014.

[8] T. H.-J. Uhlemann, C. Schock, C. Lehmann, S. Freiberger, and R. Steinhilper, “The digital twin: Demonstrating the potential of real time data acquisition in production systems,” *Procedia Manufacturing*, vol. 9, pp. 113–120, 2017.

[9] J. Lee, B. Bagheri, and H.-A. Kao, “A cyber-physical systems architecture for industry 4.0-based manufacturing systems,” *Manufacturing letters*, vol. 3, pp. 18–23, 2015.

[10] L. Monostori, “Cyber-physical production systems: Roots, expectations and r&d challenges,” *Procedia Cirp*, vol. 17, pp. 9–13, 2014.

[11] P. Esling and C. Agon, “Time-series data mining,” *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–34, 2012.

[12] V. Struhár, M. Behnam, M. Ashjaei, and A. V. Papadopoulos, “Real-time containers: A survey,” in *2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[13] M. Sollfrank, F. Loch, S. Denteneer, and B. Vogel-Heuser, “Evaluating docker for lightweight virtualization of distributed and time-sensitive applications in industrial automation,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3566–3576, 2020.

[14] W. Tsai, Y.-H. Lee, Z. Cao, Y. Chen, and B. Xiao, “Rtsoa: real-time service-oriented architecture,” in *2006 Second IEEE International Symposium on Service-Oriented System Engineering (SOSE’06)*. IEEE, 2006, pp. 49–56.

[15] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, “Microservices: yesterday, today, and tomorrow,” *Present and ulterior software engineering*, pp. 195–216, 2017.

[16] S. Pugh, “The systems engineering tool box,” 2009.

[17] M. Rostan, J. E. Stubbs, and D. Dzilno, “Ethercat enabled advanced control architecture,” in *2010 IEEE/SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. IEEE, 2010, pp. 39–44.