A Data-Centric Approach for Modular Assurance

Gabriela F. Ciocarlie, Heidi Schubert and Rose Wahlin
Real-Time Innovations, Inc.
{gabriela, heidi, rose}@rti.com

Abstract. A mixed-criticality system is one that must meet multiple assurance requirements, in particular safety and security requirements. Ideally, such a system could be constructed with components that have been individually certified to meet safety or security requirements relevant to their function. The challenge is that the components need to communicate with each other. In this case, the safety and security properties of a certified component can be affected by changes in other components that it is linked to. In this paper we propose a method to alleviate this problem, by moving from a *component-interaction model* to a *data-centric model*. A data-centric system is based on definitions and characterization of the data that flows through it, which are in turn used as a foundation for individual components. For instance, in an airplane, airspeed could be safety-critical data, while images from an on-board camera could be security-critical data, and aircraft position could be both safety and security-critical. Once the data is defined, system components and their functions are defined as well. Each component has to specify what data it requires and produces, and align its safety and security properties with the data it produces. With this approach, components are decoupled from each other – they are data originator agnostic, as long as data is delivered on time and meets the assurance requirements. This paper outlines a data-centric approach for modular assurance, and a specific instantiation of it, based on the RTI implementation of the Object Management Group (OMG) Data Distribution Services (DDS) [3,4] standard middleware, which defines and disseminates data, and the Wind River VxWorks MILS platform [2], which ensures component separation.

Keywords: real-time middleware, data distribution service, data-centric model, modular assurance

1 Introduction

Mixed-criticality systems must manage complex functions executed by applications with different security and safety requirements. For instance, onboard functions for Unmanned Aerial Systems (UAS) include safety-critical flight control and navigation, and many other functions such as communications, sensor control, and planning. While each function is important to the success of the mission, some are critical to either the physical safety of the system, the security of the exchanged data or both.

Most mixed-criticality systems must be flexible to adapt to changing needs and evolving technology. Such systems must support quick "plug in" components, such as sensors, while still fulfilling the safety and security requirements. One key concern is unwanted interactions between components at different levels of criticality.

Unfortunately, it is currently impractical to certify "mixed criticality" or reconfigurable systems. Current certification methodology requires an expensive whole-system verification and validation process. This process exhaustively tests all components and their interactions to confirm that the system meets safety and security requirements. All components that coexist or interface within a system face similar testing frameworks, regardless of their importance to safety and security. The cost of recertifying the entire platform prohibits plugging in new components for new missions.

On the other hand, strict separation is impractical; critical software and less-critical mission software must communicate. For example, the mission planner must send commands to the flight computer to direct an aircraft. Sensor information must flow to communication systems and possibly affect the mission plan.

A typical component-interaction approach for creating a distributed, composed system is to first define each of the components of the system. Then the safety-critical and security-critical components are defined. Finally, the interaction of the components is defined and specified through messages that will pass between them.

In this paper we will present a *data-centric* approach for developing a distributed, composed system. In a data centric approach, a data model for the system is defined first. Once the data model is defined, the safety-critical and security-critical properties of the data are defined. Next, components are identified, including the data produced and consumed by each component. At run-time, the safety and security assurance policies are applied to the data movements.

2 Data-Centric Architecture

A data-centric architecture specifies the data that exists in the system first. Consequently, applications focus only on the data that they intend to consume or produce, and on the assurance level they require for the data exchange. Moreover, a data-centric infrastructure understands the data that flows through it, the data schemas that are used, which data items are distinct from each other, their lifecycles, and the attached behavior (*e.g.*, filters, Quality of Service) of individual data items. These capabilities enable the decoupling of components that communicate with each other, thus allowing for reconfigurable systems that can scale and evolve.

A data-centric approach avoids complex, hidden interactions by using well-defined interfaces. The interfaces between components and the data model are specified through a strong contract of the data name, data type, and Quality of Service (QoS) and assurance level requirements. Interfaces for any one component are not dependent on any characteristics of the other components with which it must interact. This provides a modular design approach that requires only a common data model. As a result, components are data originator agnostic.

2.1 Modular Architecture Properties

A data-centric approach is based on a "smart data bus" design. With crisp QoS specifications, components can be developed independently and then reused in many applications without regard to interfaces with other components. This is the major differentiation from the old "connection oriented" approach that requires special interfaces for every module (see Figure 1).

As a consequence, a data-centric architecture facilitates a loosely coupled data design, which is essential for mixed-criticality systems, which require explicit data types and dependencies, as well as location independence. A component declares the type of data it needs and the type of data it sends. This fulfills the requirement for mixed-criticality data design that the data dependencies need to be explicit.

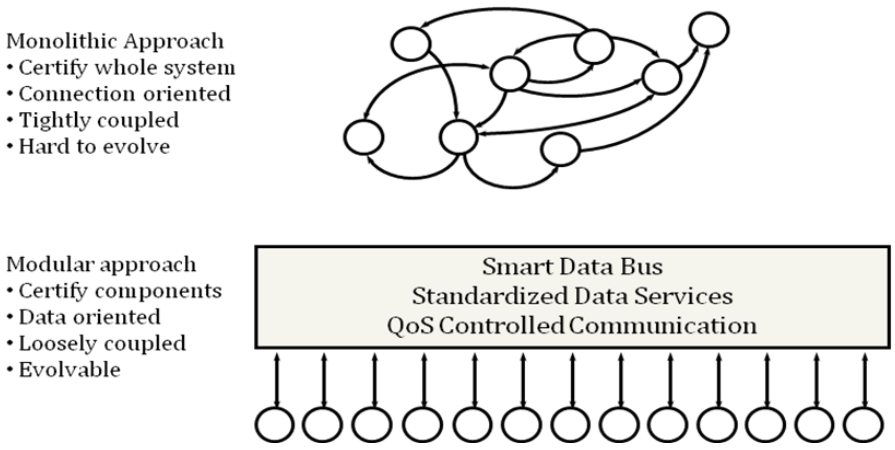

Figure 1: A data-centric approach creates a loosely coupled architecture that allows components to be upgraded or added without modifications to other components. This architecture lends itself to modular certification as the data exchange is decoupled from applications' function.

While data dependencies and characteristics must be explicit, the location of data senders and receivers should be implicit, meaning that the data model must be decoupled from the physical location of the components. A data-centric approach provides such a capability, as components are data originator agnostic. Assurance level enforcement is achieved through producers publishing only data that meets the level required by the data model. For example, if data is required at a high assurance level, it could be created by a high assurance component. It could also be created by a high assurance component that combines data from multiple, redundant lower assurance components.

3 Realization of a Mixed-Criticality Data-Centric Architecture

A mixed-criticality, data-centric architecture can be achieved through the use of a separation kernel to securely isolate applications and information flow, and a publish-subscribe middleware to distribute the data safely and securely.

3.1 Separation kernels

Separation kernels, and specifically the ones that comply with the U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness (SKPP) [6] with stringent requirements for high-assurance separation, provide part of the solution for mixed-criticality systems certification. A separation kernel supports isolated partitions such that each guest operating system (OS) runs in its own partition. Each guest OS is isolated from the other guest OSs over both time and space. The information flows are tightly controlled such that only the authorized flows are allowed in the system. Moreover, fault effects can be minimized with a combination of information flow control, data isolation and the principle of least privilege. In such an environment, testing numerous smaller components is much faster than testing the entire system, and just as safe if their interactions are well controlled. With guaranteed separation, these components can be pre-certified and composed quickly into new configurations.

3.2 Anonymous publish-subscribe

A publish-subscribe middleware can be used to build a composed, distributed, certifiable system. Publish-subscribe is an effective communication architecture where, at a fundamental level, applications simply publish what they know and subscribe to what they need. Networking middleware provides the functionality for discovering publishers and subscribers, handling network traffic and errors, and delivering the data.

Anonymous publish/subscribe is well suited for real-time, distributed, heterogeneous systems. The middleware handles the data delivery and reduces the dependency of the applications on each other. The benefit of this approach is useful for easing integration of applications running on multiple different platforms or partitions.

Effort is required to migrate from a point-to-point component interaction model to a data-centric model. Systems with a small number of computers will see fewer benefits from an anonymous publish/subscribe approach. An example of such a system is one where the flight control is done on a single operating system with minimal communication outside of it. Migration to a data-centric approach is useful if the system is planned to scale up, or needs to enable extensibility.

Object Management Group (OMG) Data Distribution Service (DDS) standard [3,4] is a data-centric publish-subscribe middleware for real-time communication, with proven performance and strong data typing. DDS is further distinguished from previous communication standards by its strong Quality of Service (QoS) parameters. DDS QoS parameters characterize the data contracts between participants, the properties of the overall data model, and real-time communication and delivery requirements on a per-data-stream basis. DDS service control includes deadlines for message delivery, bandwidth control, reliability model control, failover and backup specification, data filtering, and more. It has the flexibility

to adapt to embedded environments, high-performance server integration, and can be used over both extremely fast and slow, lossy communication paths such as satellite links. Moreover, DDS is designed for scalability and can handle many data types and nodes [8] (more details on DDS can be found in [1]).

Combining the capabilities of the DDS middleware with those of separation kernels we can build a composed, certifiable system that allows applications of different levels of criticality to interact, while incurring low complexity.

3.3 Data Flow Control Assurance

For a data centric architecture, the assurance level for data is encoded in the data model itself. Normally, when a system is integrated, its data flows must be controlled by all the following layers: application, middleware, and operating system. With our approach, the middleware and the operating system must ensure the safe and secure delivery of the data. In order to achieve this goal, the separation kernel is configured to only allow the correct data flows, while the middleware is configured to link applications to the correct interfaces on the separation kernel. The middleware QoS features can be leveraged to monitor data flows; as a result, the middleware, for example, can warn the applications when data samples are not received on time.

However, a system can include applications on the same partition, on different partitions, and on different computers. DDS can communicate data between applications regardless of their location, using different transport mechanisms. For example, DDS could use an inter-partition mechanism provided by the separation kernel for inter-partition communication, and UDP/IP to communicate across machines. Moreover, given the pluggable transports support that DDS provides, it could also use a safety-critical transport, such as Time-Triggered Ethernet (TTE).

4 Feasibility Study: RTI Data Distribution Service and VxWorks MILS

To demonstrate the feasibility of integrating a separation kernel platform with a data-centric middleware, we integrated the Wind River VxWorks MILS Platform [2] and Real-Time Innovations (RTI) Data Distribution Service [1].

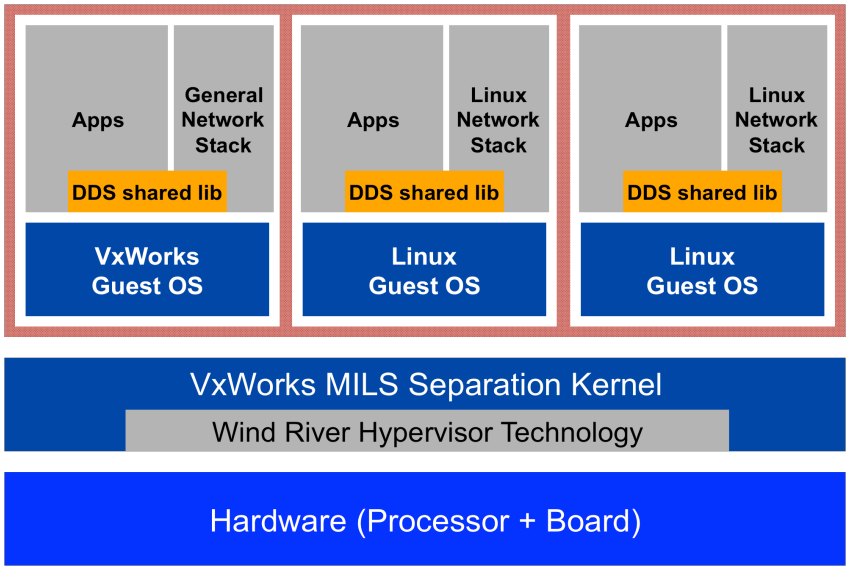

Figure 2. VxWorks MILS provides strict separation between applications running on different partitions. Using DDS for communication between applications enables the data-centric modular architecture

The VxWorks MILS Platform includes a separation kernel that complies with SKPP. The technology was designed for real-time embedded systems, and it can be extended to a PC through support of a standard operating system as a guest OS. RTI Data Distribution Service [1] is RTI's leading implementation of the OMG DDS standard.

Figure 2 presents the high-level integration architecture. A single processor is partitioned among multiple software components, with time and space resource allocation, information flow, and fault isolation, all enforced to conform to security policies that the system requires. The VxWorks MILS separation kernel includes hypervisor technology that provides virtualization capabilities to enable guest operating systems to run in the MILS user mode partitions. A single partition can run multiple DDS applications, and multiple applications can contain multiple publishing components. The DDS middleware requires accurate time in order to maintain the state of its components. The guest OS provides the "real" time tick information for each DDS component regardless of the actual time slice allowed for that partition. This enables the middleware to function correctly inside a partition.

The intrinsic characteristics of DDS offer explicit data types, explicit data dependencies, and location independence. Another important capability that is required by a mixed-criticality system is a high-degree of fault-tolerance [7]. As part of this feasibility study, we simulated different failure scenarios in a mission-critical system, and analyzed their effect on a safety-critical system. Our experiments show that when forcing failures in the mission-critical components, the proposed integration architecture successfully blocks any failure propagation to the safety-critical components.

5 Conclusions

We introduced a novel approach for mixed-criticality systems certification. A data-centric architecture facilitates a loosely coupled data design and can be combined with a separation kernel in order to define clear interactions between components with different levels of certification. Our solution addresses all the requirements that mixed-criticality systems impose: modular integration, evolvability and fault tolerance. Moreover, we have performed a feasibility study for the integration of RTI Data Distribution Service and Wind River VxWorks MILS platform. As future work, we intend to further identify and analyze the characteristics of the data models that the DDS middleware can provide in order to formally demonstrate the applicability to mixed-criticality systems.

Acknowledgements

We would like to thank Wind River for providing their MILS platform as well as for valuable feedback. This material is based on research sponsored by the United States Air Force under contract FA8650-10-M-3025. The content of this work is the responsibility of the authors and should not be taken to represent the views or practices of the U.S. Government or its agencies.

References

1. "RTI Data Distribution Service", http://www.rti.com/products/dds/index.html
2. "Wind River VxWorks MILS Platform", http://www.windriver.com/products/platforms/vxworks-mils/
3. "The Data Distribution Service specification", http://www.omg.org/spec/DDS/1.2/
4. "The Real-Time Publish Subscribe DDS Wire Protocol", http://www.omg.org/spec/DDSI/2.1/
5. G. Derrick Hinton, "Multiple Independent Levels of Security: The Changing Face of Range Information Management in the 21st Century, http://www.itea.org/files/2006/2006_Tech_Notes_Jun_Jul.pdf
6. "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness", http://www.niap-ccevs.org/pp/pp_skpp_hr_v1.03/
7. James Barhors et al., "A Research Agenda for Mixed-Criticality Systems", Joint MCAR White Paper, RBO-09-130, 07 Apr 2009
8. Stan Schneider and Bert Farabaugh, "Is DDS for You?", http://www.rti.com/mk/DDS.html