

Position Paper: Beyond SCADA

New methods in developing large-scale distributed high-confidence real-time systems.

We present a new paradigm to design, build, test, deploy and modify distributed high confidence real-time systems that monitor, sense, detect, analyze, and control. Systems are no longer monolithic and fixed, but are modular and adaptable. Relationships between software entities are no longer master/slave. Components are now autonomous entities that interact as peers, yielding more powerful solutions that are more easily created, managed and adapted. Our system development platform and tools lower entry level barrier for novice developers and enables domain experts to assemble systems without depending on software engineering resources. Our technology utilizes reusable components that can be validated and verified. Sub-systems could be tested, simulated and deployed one at a time while system is operational. Our unique approach to I/O integration allows creation of real-time, deterministic processing sub-systems in overall non-deterministic, event-driven system. Availability, throughput and extensibility of any modern infrastructure in any industry, municipality, or country is dependent on the ability of the control systems to accommodate ever-changing environment. Such changes include not just growth, but also new decision making processes, new elements, new technologies, new functional requirements, and previously unforeseen interdependencies.

To date application and system development tools and technologies, and middleware offerings are separated. Such separation leads to separation between design, development, deployment, testing, and integration domains. In turn, the results are high degree of complexity, narrow specialization and long learning curve of the technical personnel, inefficient knowledge transfer processes, and very expansive cost-to-benefit ratios of upgrade and modification initiatives for already deployed systems. Component architectures of today target developers and produce monolithic applications. As systems complexities increase, businesses and developers recognize inherent limits of these architectures and approaches. The Internet, globalization and proliferation of “smart” devices and RF technologies dictate distributed system environments that exacerbate the problems and limitations normally encountered in typical large-scale system scenarios. Methods and tools used today in system design and deployment are not flexible enough to satisfy the following:

- Large-scale, widely distributed systems require a decoupled, asynchronous interaction model.
- Large-scale, heterogeneous systems with many autonomous entities and parallel activities require a new programming model.
- Future computing environments require software systems that support customization, adaptation and dynamic modifications.

Even if an application is built using components, the resulting code creates deep interdependencies between individual components as well as between components and the “glue” code. Problems escalate when the application has to be distributed, adding inherent problems of RPC-related proxies and stubs, network latencies, error handling, and synchronous access to information. Developers have to address these “technical” problems rather than concentrate on the business domain problems. Another problem comes from the encapsulation and method-based access to data that objects and components provide. Application developers need to know the internal implementation of components to identify a sequence of calls necessary to invoke functionality. Because this information is not present in an interface specification or IDL it becomes implementation specific. It also means that a client that expects a specific interface from the component may not work with all components that implement that interface, resulting in application code that is rigid and implementation specific. Such systems, once deployed, are difficult to change, maintain, and integrate with other applications. They also have an inherent tight coupling with the underlying deployment schema.

A new development paradigm, one that supports numerous levels of abstraction and is inherently programmable and configurable, is needed to enable economical and timely design, deployment, testing and integration of distributed high confidence real-time systems. IQB platform is one example of such paradigm. IQB technology has been tested in numerous industrial and lab installations. Currently, the deployments include automotive,

building and environment, chemical process control, manufacturing, electrical and utilities systems, track and trace and RFID systems. Independent study has shown minimum of 30% time savings for a systems engineer previously unfamiliar with the IQB technology in comparison with traditional methods. Our experience shows that time savings grow beyond 60% for experienced personnel and beyond 80% for deployment of additional functionality and system components.

Packaged as components, libraries of agents can be created to target various application domains. Agent collaboration is configurable and separated from agent implementation. This modular paradigm allows large complex systems to be developed incrementally. By identifying system components that can be modeled as stand-alone tasks, developers can design, build, and test them one by one or in small groups, eventually getting a complete system up and running.

From the developer's point of view the agent paradigm simplifies the amount of work needed to get a job done. Components are designed, built, and tested before system assembly has begun, simplifying overall system debugging and troubleshooting. Protocols provide well-defined collaboration abstraction, and provide a decoupling mechanism. Developers are also presented with a much simpler set of run-time requirements, allowing them to concentrate on the actual code related to the application's domain. Developers have freedom to design agents for their systems as simple or as complex as necessary, as well as present to end-users, if necessary, a configuration/development view that models that user's domain as close as possible. This ability to adapt on multiple levels makes our framework simple to integrate into existing products as well as a compelling platform for the new product development. It provides the following benefits, not only to products developers but also to the end-users:

- Plug-and-play of agent components
- Software reuse
- A natural means of controlling concurrency
- Evolving behavior
- A closer mapping of software implementation to the enterprise
- Improved performance through distribution of processing
- Improved fault tolerance through distribution of processing
- Inherent programmability

Michael Feldman

CTO

As One Technologies Inc.

Michael has 17 years of industry experience designing and implementing large scale control and SCADA systems, as well as development tools and frameworks for these systems. He is a patent holder for IQB technology. Michael has a master degree in Control Systems and Robotics.

Alex Fiksel, PhD

Senior Software Architect

As One Technologies Inc.

Alex holds PhD in Physics and has more than 20 years experience in developing software solutions for various industries including telecommunication, medical, and energy utilities.

Alex Portnoy

Vice President

As One Technologies, Inc.

Alex holds BS in engineering and MS in Management of Technologies from the University of Minnesota. Mr. Portnoy has over 15 years of experience in large-scale systems, new product development and technology marketing.