Industrial Internet:

Towards Interoperability and Composability

Authors:
Dr. Shi-Wan Lin
Co-Founder, Thingswise
Co-Chair, IIC Architecture Task Group
shiwanlin@thingswise.com

Bradford Miller
Sr. Cognition and Decision Scientist
GE Global Research
Co-Chair IIC Technology Working Group
millerb@ge.com

1. Overview

According to a 2015 survey by IoT Nexus[1], 77% of professionals surveyed believe that interoperability is the largest challenge facing the Industrial Internet. However, "Interoperability" (enabling interactions through common interfaces based on common conceptual models and shared context) is only a first step toward meeting the need. In addition, what is required is "Composability": a shared understanding between components of their behavior – necessary if the promise of the industrial Internet of Things – ad hoc applications composed by end users – is to be realized. This article explores the requirements for a connected, interoperable environment in the age of the Industrial Internet. It then reviews the approaches that have been used in addressing similar problems in the consumer/commercial Internet, identifies the foundational concepts and unique challenges for these problems in the Industrial Internet, and suggests a roadmap toward realizing the interoperability and composability the Industrial Internet will require.

2. Why Interoperability

To begin, it is useful to distinguish different levels of general interoperability: integrability, interoperability and composability. In the early days of software systems, functions often had names like I307F3; the actual computation done by the function would have to be looked up in documentation, along with the proper arguments and result types. Without such documentation, such a function or the capability exposed by this function could not be integrated into a system except by their original authors, as there was no way to determine the correct way to call the function. Later, explicit types and function signatures such as:

(Add Integer1 Integer2) -> Integer

helped us determine the correct way to call the function and gave us an expectation of the type of the result, making such function integrable or having **integrability**. However, we still could mistake the meaning of 'Add' – it evokes a common mathematical function, but the author may have meant it as shorthand for something else, such as adding the two numbers to a stack and returning the number of items on the stack. The result is integrability without **interoperability** – we can successfully call the function and understand the type of the result, without sharing the conceptual model of what it is that the function does.

But even if we do have such a shared understanding, that Add will perform addition of two integers and give us the resulting integer, there may be additional assumptions that differ between the user and the author of the Add function. For instance, the user may think that any

[1] http://www.theinternetofthings.eu/victoria-lloyd-iot-nexus-77-iot-professionals-saw-interoperability-biggest-challenge-internet-things

integer can be supplied, but the author only allows integers that can be represented in 15 bits, and furthermore that the result must fit into 15 bits or else it would not be correct. Worse still, the function might corrupt some part of the system leading to an undetermined behavior. Such a function would be interoperable but not composable – there is no shared expectation of how the function behaves. That is, composability requires that the interacting parties not only interoperate correctly by using correct protocols for information exchange and understanding what each other actually means, but also engage each other with the correct anticipation of each other's behavior resulting from the exchange of information so that no unintended or unexpected consequence would result from the information exchange.

The consequence of lacking composability in the above example might be, at worst, an incorrect sum undetected by the user or a frozen calculator, but in an industrial setting where system-to-machine and machine-to-machine interactions prevail, the consequence of the lack of composability could be much more severe.

As the Industrial Internet matures, more components are made of so-called Cyber-Physical Systems (CPS). In a CPS, logical/computational (cyber) and physical capabilities are co-designed and co-engineered to form a unified system. However, the introduction of cyber elements increases not only the behavior space of the new system, but also the variance of behavior for any particular observed interaction with the environment. That is, CPSs will at least appear to be less deterministic than their traditional counterparts, precisely because part of the state of the system and the interaction with other systems will appear to be hidden. However, its highly likely responses to external stimulus would be state-dependent, thus making its behavior less deterministic. Now imagine the case of two CPSs that interact with each other, resulting in a situation outside of their respected design or tested range; or the case when a CPS from one vendor is being replaced by that from another – how do we ensure the overall system behavior remains the same or at least safe? Because Industrial Internet systems are large-scale distributed systems assembled from multi-vendor heterogeneous building blocks, composability is required for safe, secure and resilient operation.

If we are to have a shared community of services, devices and operations across multiple communities of authors and users in the Industrial Internet, we must recognize that protocol, data models and even conceptual models are not enough. In addition, we need to have metadata models that support composability – allowing prediction of how the system components will act or interact in 'real world' situations, not simply for those test cases in the mind of the originating engineer when these artifacts were instantiated. Furthermore, we need this across multiple levels of abstraction, from understanding, for instance, how a flash memory will behave under low or high voltage conditions to how a system will behave when encountering a novel situation in the face of a deadline that may not have sufficient time for human intervention. This is necessary not only to drive risk out of our designs and deployments, but also to capture critical operational metaknowledge[2] that future automated systems will need to reason over; e.g., to determine the reliability of a subsystem, or a service outcome.

As we will discuss here, the good news is that while such metaknowledge might be generated manually at some expense and trouble, it need not be: We envision systems of the near future being able to create such metaknowledge based on observing their own operations and then having the ability to share that knowledge over the Industrial Internet with other peer systems with similar compositions and environments.

3. Interoperability in the Context of the Industrial Internet

The idea of the Industrial Internet is in a large part inspired by the consumer or commercial Internet. Additionally, the Industrial Internet may also be considered as a convergence of Information Technology (IT) and Operational Technology (OT). Therefore, it may be informative to take a brief review of the concept and practices of interoperability in the Internet and IT and to use that as a baseline to highlight the unique requirements and challenges in interoperability faced by the Industrial Internet.

The consideration of interoperability in the Internet and IT has been mainly focused on communication and information sharing between connected systems; e.g., among the client applications and browsers, the web and application servers and the databases across all the application and communication layers to make them work together. As IT systems undertake solving more complex problems, their applications become more complex as well. A task (e.g., order processing) usually requires concerted interactions of a number of subsystems (production, supply-chain, finance, delivery, customer-relation, etc.), each of which is a typically specialized system made by a different vendor. In these cases, interoperability focuses on how to ensure different systems or subsystems interact seamlessly to complete a large task.

Conventionally, there are two major approaches for achieving interoperability to enable communication or interaction between heterogeneous systems: 1) through the mediation of brokers and 2) through a common meta-model with agreed interfaces, as depicted in Figure 1.

[2] Metaknowledge: knowledge about our knowledge – how reliable it is, the provenance of the information, collectively referred to as epistemology – how we justify our beliefs of what we think is true.

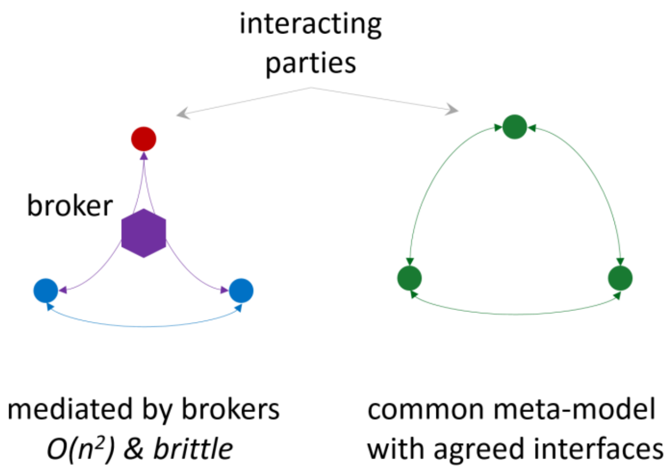


Figure 1: Main approaches for achieving interoperability

The broker approach, akin to using an interpreter between two speakers of different languages, works reasonably well with a small number of interacting parties (after tolerating the potential problem of 'lost in translation'). It is perhaps the only way to enable communication between parties 'after the fact' – that they were previously built with different specifications and now find the need to communicate. This is a task we are facing in connecting many legacy industrial systems that have previously been deployed. The broker approach clearly has its shortcomings in scalability both in term of the number of different 'species' of interacting parties requiring specific brokering (design time complexity) and the sheer number of parties in a deployment requiring brokering (runtime complexity). It also suffers in stability because a broker would require updating whenever any of the 'species' changes or new 'species' are added to the mix.

The common meta-model approach removes the scalability problem found in the broker approach but requires foresight – interoperability by design – as the interacting parties must be created with the common meta-model and agreed-upon interfaces.

There are a number of ways of achieving common meta-models and agreed interfaces. The prevailing approaches include:

1. *Common Specification*: The interacting systems are built from common specifications as a consensus of technical communities, often in the form of Open Standards available to all implementers free of charge or at a nominal fee. The modern communication systems serving as the foundation of the Internet (e.g., RFCs from IETF, specifications from IEEE and ETSI, etc.) and the application stacks in the IT systems (e.g., the SQL specification from

ANSI and ISO, the Web Services and SOAP stacks from W3C and OASIS[3], etc.) are examples of this approach.

2. *Common Components*: The interacting systems are built with common components, either from proprietary suppliers or, increasingly, from Open Source projects. All the systems built on the Linux OS, for example, share pretty much the same Internet Protocol (IP) stack (minus the Ethernet driver from individual device vendors). Another example is the Open SSL library or the SSH[4] program for Linux. There are countless other examples in this category.

3. *Open Source built on Standards*: The interacting systems are built with common Open Source components that are in turn built on standards. An excellent example of this approach is the highly successful J2EE[5] Java application development ecosystem.

4. *Closed Ecosystem Framework*: The interacting systems are built from proprietary specifications from dominant market players. In this approach, everyone has to build to the proprietary specifications or use proprietary components from the dominant players in order to play in the specific ecosystem or market. For example, anyone who wants to play in Apple's home automation ecosystem must abide by the Apple specification. A similar story exists for Google's Brillo platform for Android.

5. *Open Source Ecosystem Framework*: Increasing numbers of new generations of solutions, especially those of large-scale distributed systems from the Open Source arena, have established themselves as de facto open standards because of the dominant position they have established due to their success. Anyone who wants to use or interact with these systems often must adapt to the interfaces implemented by these solutions (often without a formal specification – the implementation being the specification). The Hadoop ecosystem and solutions in the application container area are good examples of this approach. In these systems, as long as you use the components provided from the ecosystem, interoperability is virtually ensured within these systems.

Clearly, all of these approaches have enabled interoperability in their respective environments or ecosystems and they continue to evolve. The Open Source movement is the latest force that

[3] RFC: Request for Comments, IETF: Internet Engineering Task Force; IEEE: Institute of Electrical and Electronics Engineers; ETSI: European Telecommunications Standards Institute; SQL: Structured Query Language; ANSI: American National Standards Institute; ISO: International Organization for Standardization; SOAP: Simple Object Access Protocol; W3C: World Wide Web Consortium; OASIS: Advancing open standards for the information society.

[4] SSL: Secure Sockets Layer; SSH: Secure Shell

[5] J2EE: Java 2 Platform, Enterprise Edition

is increasingly influential in how interoperability is achieved. All of these approaches are expected to play a role in achieving interoperability in the Industrial Internet.

The Industrial Internet, as an extension of the Internet to the physical world, including devices and machines, has its own unique challenges. Let us first examine its communication or interaction patterns.

Until now, the Internet communication pattern is predominantly a point-to-point client-server model where a client (as a service consumer) initiates a service request to a server (as an aggregated service point), which replies with a service response, as depicted in Figure 2. When two clients interact with each other (peer-to-peer interaction), they are likely to go through a server as an intermediary. In this way, the interoperability of peers is made possible through the client-server interaction.

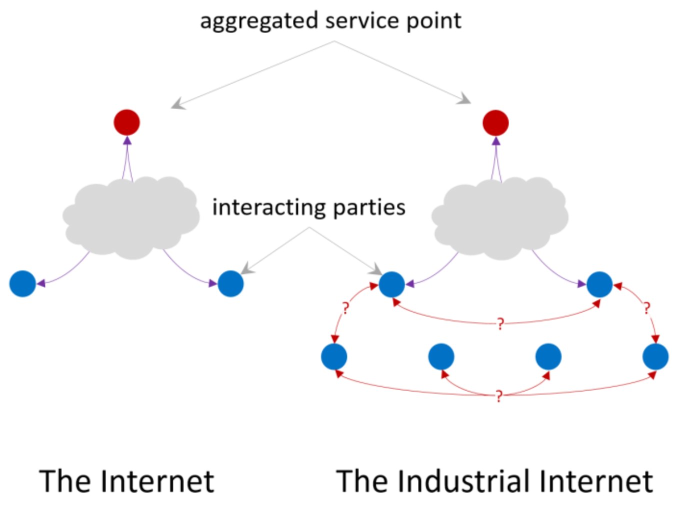


Figure 2: Interaction patterns between the Internet and the Industrial Internet

Initially, the Industrial Internet may adapt this pattern while it seeks to connect the industrial assets (industrial devices and machines) to a broader system (an aggregated service point). The immediate challenge lies in how to provide the connectivity to the industrial assets, some of which were built without any consideration of being connected to a broader network while some others were equipped with one version or another proprietary communication protocols. Moreover, many new sensors are being attached to the existing industrial assets to gain better insight on their operations. How to provide connectivity to the assortment of new sensors made by different vendors with proprietary specifications is another big challenge. Interoperability is an important factor in both cases. Before cross-industrial sector standards can be established and implemented and before the industrial assets are retrofitted or upgraded, the broker approach may be an effective tool to provide connectivity and ensure interoperability between the industrial assets and the broader system. Internet of Things (IoT) Gateways may be deployed as brokers (which can also be considered as an aggregated service point) to connect the industrial assets to the broader systems. Assets to assets interactions may be brokered by the brokers or by a higher level aggregated service points if the interacting assets are connected to different brokers. As the Industrial Internet matures, an increasing number of industrial assets will be upgraded to modern CPSs. With the increasing computational capability of these CPSs, the industrial assets will shift from automation to autonomy in their operations. Consequently, there is an increasing need for them to interconnect to and interact with each other directly to enable autonomous collaboration among themselves to solve problems locally. As a result, the existing point-to-point interconnect patterns will give way to a meshed many-to-many ad hoc interconnectivity. Given the large variations in the types of the CPSs expected to be involved in such interactions in a deployment, the interoperability problem multiplies. This multitude of interactions among the industrial assets raises a unique quantitative challenge in interoperability that we expect to face in the Industrial Internet. A new class of cross-industrial sector connectivity frameworks such as Data Distribution System (DDS)[6] and OPC-UA[7] are expected to be seen increasingly deployed not only providing the needed interoperability in communication, connectivity and data modeling but also scalability, performance and security as well.

Moreover, consider the many unique characteristics that are present in industrial systems, including:

- produce physical effects
- strong safety requirements
- often mission-critical
- subject to dynamic conditions and operate in environments that are not always foreseeable at the design time
- long lifecycle, being very costly or even impractical to replace on demand
- advancement towards autonomy, increasingly capable of learning and making decisions based on conditions without intervention

All of these pose a qualitative challenge in interoperability in how to ensure the correct outcomes in these interacting systems under changing conditions and dynamic environments. This challenge calls for a higher level of interoperability: composability, as highlighted at the introduction of this article.

Adding to these quantitative and qualitative interoperability challenges is the dynamism of the system within itself concerning how to sustain the required level of interoperability as the interacting components evolve over time in their respective lifecycles and as new components enter into the system. To address these issues, it may be beneficial to reference the IT/Cloud Computing world where the Service Oriented Architecture (SOA) paradigm has matured and is widely adopted in software design. Within this design paradigm, self-contained units of software capability are encapsulated as services and exposed through service interfaces, often in the form of APIs (Application Program Interfaces) that can be invoked remotely through a network. This allows software capabilities to be loosely coupled (allowing independent evolution, among other things) and large software capabilities can be assembled from a set of cooperating services

[6] http://portals.omg.org/dds/

[7] https://opcfoundation.org/about/opc-technologies/opc-ua/

through their APIs. If we consider a CPS, or an Industrial Internet system, as a collection of capabilities, albeit supported by physical components, the SOA paradigm seems to apply and may be beneficial to the Industrial Internet. Within this paradigm, the interoperability task would be to provide the common meta-models and agreed interfaces about specific services. However, in the Industrial Internet, the meta-models and interfaces are not static but expected to change dynamically. A temporal challenge in interoperability thus emerges: how to dynamically maintain the required level of interoperability beyond the time of initial deployment as existing services, including those embedded in the CPSs, independently evolve through their respective lifecycles (e.g., upgrading), and as new services/CPSs are joining the system. Therefore, the traditional SOA approach of statically composing capabilities is inadequate for the Industrial Internet for it leads to a brittle system nonresponsive to changes in its constituents or its environment.

If we consider that the services in the SOA paradigm are expressed by their respective APIs, then the question becomes how to ensure, automatically where possible, interoperability and composability as the API evolves over time. To address this class of issues, some forward-looking and innovative approaches are being actively discussed within the Industrial Internet Consortium (IIC). One bold approach[8] is to define various levels of API maturity with the highest level providing a mechanism:

1. to establish an a priori service contract between the interacting parties upon which an active service contract can be adapted to changing conditions upon mutual agreement between the parties,
2. to share service metadata about the service, potentially including a description of the meta model, precondition/constraints, expectant behavior,
3. to describe the service API providing the capability, and
4. a meta-service API for notifying service changes, updating the service metadata and negotiating adjustments to the service contact to adapt to changes in the service or usage condition.

This approach may provide a powerful mean to address not only the quantitative and qualitative but also the temporal interoperability challenges presented in the Industrial Internet.

4. Interoperability Evaluation Based on the Industrial Internet Reference Architecture

To meet the unique quantitative, qualitative and temporal interoperability challenges in the Industrial Internet, a systematic approach is needed. The starting point is to establish shared architectures and models for the Industrial Internet. Without a common architecture model, we

[8] http://blog.iiconsortium.org/2015/09/dynamic-apis-negotiating-change.html

cannot take on the challenges in interoperability because we would lack even a common understanding of what the interacting components are in such a system.

The Industrial Internet Reference Architecture (IIRA)[9] published by the IIC[10] is a first step toward such a shared architecture. The key intent of the IIRA is to establish common understanding in major architectural issues, structures and requirements for the Industrial Internet across industrial sectors, from which to build consensus to drive product interoperability and simplify development. Two key elements in the IIRA are most relevant to the discussion concerning interoperability. The first one is the Industrial Internet Functional Domain, as shown in Figure 3, and the other is the concept of composability in the Industrial Internet. We will discuss the latter point first and come back to the Functional Domain.

Taking into account the unique requirements in the Industrial Internet, the IIRA proposes a simplified interoperability model for building an Industrial Internet system from its various components, in relation to the different levels of understanding in communications, as summarized in Table 1. This model reflects the learning from the existing models of interoperability in the Internet, IT domain, industrial sectors and academic research[11]. In formulating this model, a reference to the human language model is made, anticipating that many of the Industrial Systems are increasingly designed to be more intelligent, mimicking human interaction patterns. Within an Industrial Internet system, the interactions among its components may be examined under this model to decide the required interoperability level. As Industrial Internet systems mature, more interactions are expected to move toward composability.

[9] www.iiconsortium.org/IIRA.htm

[10] www.iiconsortium.org

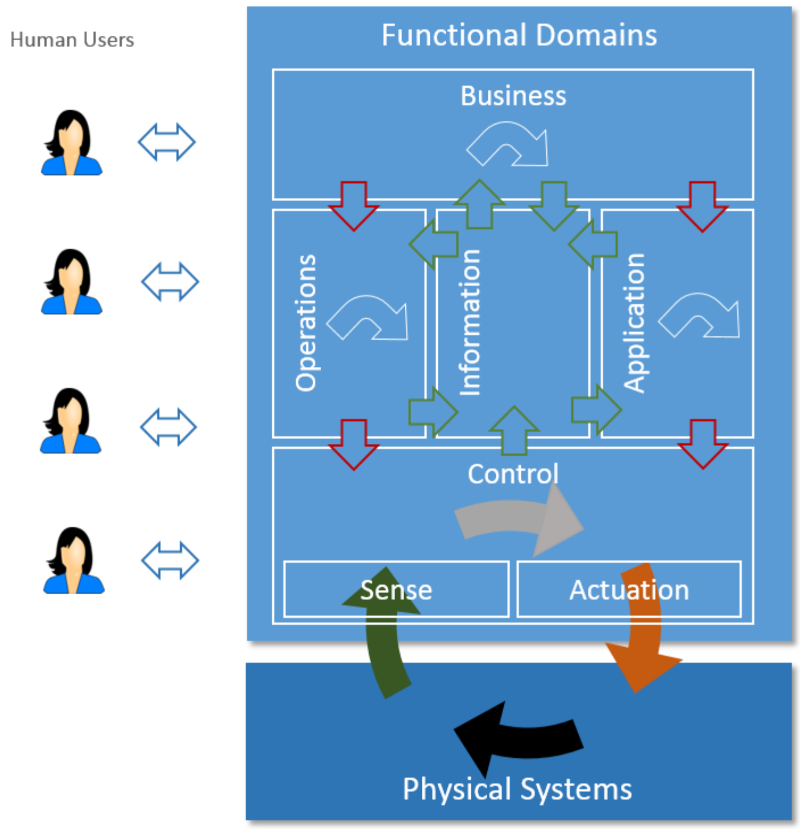
[11] Most noticeably the excellent work done by the GridWise project [https://www.smartgrid.gov/files/Framework_for_Addressing_Interoperability_Issues_200712.pdf, Ron Ambrosio] and that by Page *et al* [Page EH, Briggs R, Tufarolo JA. Toward a Family of Maturity Models for the Simulation Interconnection Problem. In: Proceedings of the Spring 2004 Simulation Interoperability Workshop, IEEE CS Press; 2004.].

IIRA Levels	**Requirements**	**Aspects**	**Human Language Model**
Integrability	Compatible signals and protocols	Communication	Syntax – how words can be arranged
Interoperability	Common conceptual models	Understanding	Semantics – the meaning of individual words (in context)
Composability	Mutually shared expectation in behaviors	Action	Pragmatics – the meaning of sentences and higher level structure (discourse)

Table 1: Industrial Internet Levels of Interoperability

The IIRA functionally decomposes a typical Industrial Internet system into five major Functional Domains, as shown in Figure 3. Briefly,

1. *The Control Domain* comprises a collection of functions performed by the industrial assets or controls systems involving sensing, control and actuation in the familiar 'closed-loop control.'
2. *The Operations Domain* comprises a collection of functions that are required to keep the industrial assets up and running efficiently. They include the typical maintenance functions with added intelligent capabilities such as predictive maintenance, etc.
3. *The Information Domain* comprises a collection of functions for collecting and managing data gathered from the industrial assets and for performing analytics on the data to gain insight into the operational states of these assets.
4. *The Application Domain* comprises a collection of functions specialized for specific usage scenarios that apply the analytic insights from the Information Domain to achieve specific business objectives such as optimizing the operations of a large fleet of assets.
5. *The Business Domain* comprises supportive business functions such as enterprise resource management (ERP), manufacturing execution system (MES), etc. needed to realize end-to-end Industrial Internet operations.



Green Arrows: Data/Information Flows; Grey/White Arrows: Decision Flows; Red Arrows: Command/Request Flows

Figure 1: Industrial Internet Functional Domains

The functional domains, Business, Information, Application and even Operations (which is similar to OT-type systems), are similar to the IT systems which are increasingly built on Enterprise/Cloud Computing platforms. So, conventional approaches in solving the interoperability problem may still be applicable. However, new interoperability requirements emerge. A few key new requirements are outlined below along with recommendations on some initial steps to address them for the immediate term.

1. In order to ensure that common capabilities in the Operations Domain can be platform-ized and used to manage and maintain various kinds of industrial assets from various vendors and across industrial sectors, it is highly desirable to establish interoperability between the industrial assets in the Control Domain and the Operations Domain. Establishing cross-industrial sector standards and providing Open Source implementations are considered important steps for establishing interoperability between these two domains, making it easy to connect to and effectively maintain the industrial assets.

2. The need for interoperability between the industrial assets in the Control Domain and the Information Domain. The most significant challenge in interoperability here lies in data

communication protocols, data syntax and semantics to enable effective and high quality analytics. It is highly desirable to establish standards at the lower stack providing common data communication protocols and abstract data syntactic and semantic models across industrial sectors, while at the same time allowing specific syntactic and semantic models to be instantiated based on the common abstract models for each industrial sector according to their unique requirements.

3. The need for interoperability between the Information and the Application Domains to concisely represent analytic results.

4. The need for interoperability between the Application Domain and the industrial assets in the Control Domain so that commonly understood semantics can be relied upon when the high level applications interact with the industrial assets to provide feedback to their operations. When providing feedback to the industrial assets, composability is generally required in that the Application must have a level of expectation on how a specific asset will react to the requests it issues.

The most demanding challenges in interoperability lie in the Control Domain where functional components from various assets interact with each other, as outlined previously. How to meet these challenges is a key to the success of the Industrial Internet in the long term and yet requires substantial research and experimentation to achieve.

5. COMPARING INDUSTRIAL INTERNET SYSTEMS

In this section, we focus on two different approaches to Industrial Internet systems and point out composability issues. The first approach is the path most of us are embarked upon, but it illustrates the danger of presuming composability can be solved by standardization or brute force at a global level – requiring buy-in to a particular vendor framework or a particular set of standards. The second approach instead encourages an open market where composability is addressed locally within communities of interest rather than globally, encouraging engineering automation and specialization. We do not yet have all of the tools we need for the second approach, but it may be time to invest!

5.1 Short-Term Approach: Analytics in the Cloud

A common approach toward an Industrial Internet system, today, is to collect data (as much as possible) from every asset, store it “in the cloud” for analysis, mine it to discover correlations and associations and then take the result and turn it into a new control or operating policy. This new control or policy can, for instance, optimize the life of the system, minimize downtime, improve asset power efficiency, etc. Such an approach is incremental over the kinds of things we have already been doing with assets. For instance, condition-based maintenance requires understanding the behavior of a component over its lifecycle. So that, given some sensor data, we can determine where it is in that lifecycle – nearly new or about time to replace, based on, for instance, wear indicators, deviations from normal within a tolerance range, etc. This requires enough data from similar parts to generate a model to which we can compare a given part, but it still does not track variations in the part. Through leveraging data on a massive scale, we can instead create 'digital twins'[12] which customize the model for the part in hand, rather than trying to fit the part to the generic model. However, the computation involved may exceed what is available locally – thus requiring non-local modeling.

Such a process essentially puts the control authority into the cloud as well – because it is the source of the control policy. This brings about two challenges. First, there is security – data about my process will not be under my company's immediate control and policies may be interfered with maliciously. Because my company is still responsible for the effect of the control policy, we have to make certain that any leakage of information (or control) does not lead to disastrous consequences, from existential threat to the company to safety issues that result in substantial casualty.

The second challenge is the introduction of a 'single point of failure.' The single point of failure is the connection between the machine and the cloud model. It creates a challenge that traditionally might be addressed with multiple models, each with different software (to avoid single points of failure due to the same software or operating system faults), as well as multiple connections to different cloud services (which may or may not be possible without local intermediation adding more latency). However, even given that some part of the model has failed, we would like the alternatives to be composable – we can smoothly substitute other (sub) models and resources to quickly recover from any issue as well as to cross check critical results between geographically and provenancially diverse systems.

Engineering processes today rely on whole-life practices that perform offline checks limiting what can be changed at runtime. Every combination must be tested in advance and because assets always do only what they are told, any remote operations team itself may become a 'single point of failure' (insider threat). Another challenge is the large amount of data that must be sent back and forth to cloud-based resources; the introduced latencies demand non-real time approaches limiting the response time to events that are not handled through prior local policies. To really take advantage of having multiple variations of model components, they must all make similar assumptions, have similar data formats and generate a similar singular point of view of the system. This will tend to lead to either large costs in bringing up new systems or a tendency to keep new systems as mere variations of old systems to avoid changing this language – how the system is described and understood.

[12] http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120008178.pdf

5.2 Long-Term Approach: Peer to Peer, Distributed, Ad Hoc, Amorphous Computing

There is, however, a different way to approach Industrial Internet systems that does not lessen the effort for an initial system, but does address some of the challenges that would press us toward having to keep duplicating our first copy (and the limitations inherent therein) for future systems. This approach requires us to change our systems engineering process: Rather than depending on an offline certification of requirements like safety, we move to an online approach that relies more on bounds checking and behavior analysis than Monte Carlo methods to show deontological adherence for a particular set of (control) functions. The key change is to move the responsibility for making decisions from the cloud to the asset – making the asset itself autonomous. Assets become responsible for their own analytic processes, which, if they are deployed locally, may well be part of a real-time response cycle. More importantly, assets can distribute problem solving between them as resources become available. Such a local set of ad hoc computational services can help with the limitations of embedded computing within a particular asset to take advantage of unused resources in nearby equipment, either stationed intentionally (like a set of local servers within a factory) or unintentionally (like the processing power of a plant visitor's cell phone). We can still take advantage of the "amorphous computing" concept that cloud gives us, without actually sending anything offsite!

By moving the responsibility of asset operation to the asset itself (or its analogue in the digital twin sense), we can filter any data necessary for broad-spectrum fleet level analytics, turning such approaches away from 'big data' problems where massive amounts of data need to be transmitted. Policy changes that are induced by such off-board analysis become suggestions, not commands – each individual asset makes the decision as to whether the new policy will improve its operation. Single points of failure introduced (e.g., by security operations) are mitigated by assets that can be suspicious of any attempts by privileged operators to effect changes – the asset, as autonomous, will have the final say as to if and when such requests will be acted upon.

Moving analytics to the asset addresses privacy (no more process information escaping the company's assets to non-owned resources) and resiliency (no single point of failure to cloud services can cause the asset to fail). Because network users cannot be expected to know all safety constraints, asset autonomy also enables an asset to refuse requests for safety reasons. Problems being solved in context may be simpler than those that are removed from the context. However, this does introduce challenges associated with requiring higher levels of intelligence on the device, including how situations are represented and understood, how to enable online machine learning (ML) rather than offline analytic ML approaches of today[13], etc.

[13] Promising advances in hardware, such as https://www.technologyreview.com/s/526506/neuromorphic-chips/ should lead to new applications of online learning in industrial settings.

Key to making this happen is to change from our top-down engineering approach to a bottom up, ad hoc construction of systems from parts, orchestrated to solve a particular problem at hand. This will require a description language for each software or hardware component that allows such composition. That language must include descriptions of how each part is expected to behave in various circumstances. Since we cannot know, a priori, how software will behave in all circumstances (except under very limited conditions) it is important that this task be augmented with machine learning and, leveraging the digital twin approach, can perform experiments in a simulated environment that can be validated in the real-world if the results appear promising. Not to understate the complexity of producing such a simulation, however, is it really any different than that required for an excellent digital twin? The complexity of a simulation depends on the questions we are trying to answer with it. If we build a simulation to understand how gears mesh, we may have left out the parts that would let us answer questions about gear wear. If we allow gear wear, we may still not understand how shavings from wear will interfere with the bearings, etc. As we approach perfection, we usually find it is simpler to just build the thingamabob and measure it! But nothing prevents us from doing exactly that – having physical experimental setups that allow such experimentation by the system prior to operational deployment.

Because a common operating picture[14] isn't needed – we don't require a single standard to which all software must be developed for interoperation because we can generate appropriate translation and adaptation software to insert between interoperable but not immediately composable pieces – we lessen the longer term disadvantages of being locked into a single model we have to select prior to broad application. The system should satisfy and adapt to circumstances similar to an insect colony or even a (non-congressional) committee.

6. CONCLUSION

In our 'analytics in the cloud' approach, we need a standard way to represent information about common processes. However, such approaches in the past (e.g., IEEE SUMO[15], Cyc[16], Ontolingua[17]) have generally only succeeded at the syntactic level. We think there is a common way to reason about anything, but the reality is that this is a trick of our own brain – we do not really reason using common methods, but have an ad hoc collection of hacks that work well in

[14] https://en.wikipedia.org/wiki/Common_operational_picture

[15] Pease, Adam; Niles, Ian; Li, John; "The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Applications," AAAI Technical report WS-02-11, 2002. https://www.aaai.org/Papers/Workshops/2002/WS-02-11/WS02-11-011.pdf

[16] http://opencyc.org

[17] http://www.csee.umbc.edu/csee/research/kse/ontology/

specific circumstances (i.e., case-based reasoning, analogical reasoning). Local compatibility is a simpler problem. While we have to tie descriptions to a common viewpoint, each component can work with whatever description they find best suited to their behavior. We do not have to represent 'how to tie a shoe' as a language problem, but as a sequence of visual steps, for instance. What other components need is metadata about such descriptions so they can reason generically about their content without having to understand them in detail. This is very similar to how humans work together in society: I do not have to understand how to assay gold, for instance, to be able to use it as a store of value. I can rely on some expert in the community to tell me how pure the gold is before I buy it. Similarly, we need to be able to represent what it is that the analytic will do or discover and the limitations thereof, without having to reason over how it gets done. Existing projects, such as DMDII's[18] Digital Manufacturing Commons is already creating such an ability to connect together disparate models (e.g., solid, mathematical continuous, discreet, XL, etc.) to help bring together communities around manufacturing. There is no reason an analogous approach, suitably generalized and upgraded, would not work within the Industrial Internet as well.

- Return to IIC Journal of Innovation 2nd Edition landing page for more articles

- Download the IIC Journal of Innovation 2nd Edition

- Visit the IIC Journal of Innovation 1st Edition landing page for more thought leadership

[18] http://dmdii.uilabs.org