Evolutionary Architecture and Engineering Concepts for Very Large-scale Sensor-based Solutions

(Invited Paper)

Jerker Delsing
Lulea University of Technology
EISLAB
97187 Lulea, Sweden
Email: jerker.delsing@ltu.se

***Abstract*—The design and engineering of solutions based on sensor data is currently based on legacy low-level sensor communication. The transition to IoT-based sensors has started to provide IP-based sensor communication based on service concepts. This requires radically different solution architectures and the associated engineering process. Here, IoT sensors will provide services instead of a communication interface, as a means of integrating into application solutions. It is further anticipated that the size of the solution will reach well beyond that of the current legacy automation implementations.**

This paper addresses architectural concepts for very large-scale SoS- and IoT-based solutions. Architecture concepts and their associated engineering concepts are considered. IoT and SoS evolvability, run-time dynamics, scalability and segmentation, run-time engineering, self-engineering, self-mitigation, and machine-to-machine business are some examples of such concepts. These concepts must be coupled with the requirements such as the management of IoT/SoS solution functionality, security, safety, maintenance, and evolution.

I. Introduction

The design and engineering of solutions based on sensor data is currently based on legacy low-level sensor communication. The transition to IoT-based sensors has started to provide IP-based sensor communication based on service concepts. This imposes radically changing solution architectures and the associated engineering process. Here, IoT sensors will provide services instead of a communication interface as a means of integrating into application solutions.

The change to a service-based integration approach imposes changes to both the architecture and the engineering process. The move to services indicates an architecture change away from dedicated middleware toward IoT sensor configuration and SoS orchestration and management [1], [2].

The shift from legacy automation technology to Industry 4.0 and IoT- and SoS-based automation and digitalization large-scale solutions has been discussed for the last decade. New architectures beyond ISA-95 [3] have been proposed for production automation. The most widespread examples are RAMI4.0 and IIRA [4], [5]. The implementation of such architectures using SOA technology was pioneered by Jammes, Colombo and Karnouskos [6], [7], [8], [9]. Some recent interesting papers on SoS architectures are [10], [11] The engineering of such architectures using IoT-, SoS- and SOA-based technologies is still in early stages [12]. A recent interesting paper on

Architectures for very large-scale smart energy grids are being proposed [13], [14]. These architectures are clearly specific to the power distribution industry and are mostly focused on the communication infrastructure.

Interestingly, a thorough integration of the concepts, principles and properties from large-scale production automation using a very large-scale approach from the smart grid networks has not been addressed to any larger extent.

This paper addresses very large SoS/IoT-based solutions, their architecture concepts and the associated engineering procedures.

II. Conceptual approach

Conceptually, any sensor applications can be characterized as an interaction between the data providers and data consumers. In this case, the consumers perform some type of control/actuation based on the sensor data. An example of this is found in automotive wheel pressure sensors, where the data trigger an alarm to the driver. An emergent sensor application field is the integration of very large numbers ($10^5 - 10^{10}$) of simple sensor applications into large-scale distributed automation/digitalization solutions. A logistics system for emptying public and private trash and recycling cans and containers in a large city is one example of such applications. Such a system will enable improved emptying strategies such as on-demand instead of on-schedule. Trash/recycling can/container fill data will further provide an understanding of the actual trash/recycling can/container needs, thus enabling the dynamic re-positioning of trash/recycling cans/containers based on actual usage. To provide overall efficiency for such large-scale applications, functionalities such as self-reporting, self-mitigation, self-engineering, M2M business will be important.

This type of scenario can easily be applied to many applications. Each of these has its specific possible new business and operational scenarios providing benefits such as reduced operational costs, reduced environmental footprints, and autonomous functional evolution.

The above application scenarios can to some extent be readily realized using legacy sensor application engineering approaches. The largest legacy-based automation installation

ever built contains approximately 100.000 I/O points [15]. The high engineering costs are most likely one of the main reasons for this boundary.

The shift to the IoT-based sensor applications anticipates that the Internet scale of billions of connected IoTs and integrated into a system of systems, SoS, can be reachable. This scale of IoT interaction can further be expected to create new machine to machine (M2M), business models and M2M transaction technologies that currently are not manageable using traditional banking approaches. It is also further obvious that such a development will require that security and safety are fundamental parts of the SoS architectures and their implementation platforms.

Further comparison of the costs of the legacy engineering approaches and the IoT/SoS service-based integration approaches provides data that show significantly reduced engineering costs for the IoT/SoS service approach. Cost reductions on the order of 60-85% have been reported [16].

A. Very large-scale IoT and SoS characteristics

The application scenarios discussed above have IoT and SoS characteristics such as the following: a) Highly distributed solutions b) Very large-scale SoS, $10^5 - 10^{10}$ IoTs c) IoT error and maintenance reporting and mitigation d) SoS run-time dynamics e) SoS functionality evolution f) SoS scalability g) SoS segmentation for the protection of real-time operations, security, safety, ... h) SoS self-mitigation i) SoS self-engineering j) SoS self-management k) Machine to machine business models l) Machine to machine transactions - nano payments m) Multistakeholder autonomous integration and operations n) Management strategies and policies of SoS properties such as, e.g., 1) Operations 2) Functional evolution 3) Functional degradation and maintenance 4) Functional engineering 5) Security 6) Safety 7) Quality of service

For most very large-scale applications and their usage scenarios, it appears almost impossible to address and provide comprehensive and high-quality engineering solutions at design time. Thus, it becomes evident that we need architecture and engineering procedures that are feasible, efficient and economically viable at run time. Thus, the unforeseen and emergent behavior of SoS can be addressed at run time.

The current automation engineering standards such as IEC 81346 [17] are design-time-based and do not have a run-time engineering perspective.

Given the IoT/SoS characteristics described above, it is here argued that it is evident that very large-scale SoS will need to have an evolutionary behavior. This must be considered in the emerging IoT/SoS architectures and engineering approaches.

III. Architecture concepts for very large-scale SoS

As argued above, the anticipation of very large-scale SoS calls for new and additional architectural concept for engineering, management, maintenance, business models, transactions and cyber physical interaction. Below, we present a discussion on several of such new architectural components.

A. Distribution and run-time dynamics

One such architecture concepts is support for IoT and SoS run-time dynamics. In this case, the three **L**-properties of the service-oriented architecture provide the necessary support:

A) **L**ookup 1) Publish and register services to notify others about endpoints (how to reach me) 2) Discover others that I comply with (expected/wanted ServiceType)

B) **L**ate binding 1) Possible to use information at any time by connecting to the correct resource at a given time

C) **L**oose coupling 1) Autonomy - a service exchange is not supervised 2) Distributed - services are distributed over several devices 3) A system is responsible, owns the information, and can decide with who to share it

These fundamental mechanisms enable the run-time discovery of any changes to the IoTs of an SOA-based SoS. It further provides the run-time binding information on which IoT consumes which produced IoT service. Both properties are essential in addressing run-time engineering and SoS evolution. The autonomy property also provides fundamental support for segmentation and scalability. In recent years, we have found several SOA-based automation frameworks, e.g., the Arrowhead Framework, Eclipse BaSyx, IDS, and FiWare, that provide support for run-time dynamics in distributed environments [18], [19], [20], [21]

B. SoS Segmentation and scaleability

The segmentation of SoS is important for creating protection sensitive parts of an SoS operation such as critical real-time operations or safety critical operations. The concept of self-contained local clouds as introduced by the Arrowhead Framework appears to be one of the very few proposed to date to support such an architecture concept [2]. Arrowhead Framework local clouds provides support for internal run-time orchestration management, and authentication, authorization and audit (AAA) security and deployment security [22], [23].

SoS scalability must be addressed in the discussion of very large-scale solutions. The use of self-contained local clouds as SoS building blocks appears to be an attractive approach. The local cloud concept enables the 3**L** mechanisms to also be used in between local clouds. This also includes the AAA security mechanisms. This allows for the interaction between the local clouds at the expense of no real-time performance [24].

Scalability achieved in this manner clearly adds complexity to SoS management. Therefore, multidimensional management with time-dependent interdependencies must be considered.

C. Run-time engineering

To avoid the need for extensive re-testing of large SoS functionalities for security, safety and other issues, run-time engineering should be able to be executed in isolation from the other parts of the SoS. Thus, it must be possible to modify, test and deploy a component of an SoS without any engineering updates to the other parts of the full SoS. This indicates that centralization of any part of the architecture should be

avoided as much as possible. The local cloud approach of the Arrowhead Framework (see above) provides a clearly defined segmentation of SoS that may be beneficial here.

Engineering strategies for such segmentation will require the consideration of multiple management dimensions that will have dimensional dependencies that can also vary over time. Example of such dimensions and their dependencies are the following: 1) Operation functionality: f(system degradation, quality of service, supply chain optimization, ...) 2) Maintenance: f(system degradation, supply chain optimization, stakeholder policies, ...) 3) Quality of service: 4) System degradation: f(quality of service, supply chain optimization, operations, ...) 5) Supply chain optimization: f(operations, maintenance, quality of service, ...) 6) Security: f(safety, stakeholder policies, ...) 7) Safety: f(security, stakeholder policies, legal aspects, ...) 8) Stakeholder policies: f(business models, legal aspect, ...) 9) International and national policies It is clear that proper segmentation in such a multidimensional and dynamic environment will require numerous design and engineering considerations. The possibilities for dynamic re-segmentation in run time also must be considered.

D. Run-time management

From the above, it is further clear that the engineering process must create management policies, related strategies and associated information structures supporting the complexity of very large sensor-based automation and digitalization solutions. To a large extent, fundamental theory and model for such multidimensional systems with time-varying interdependencies are currently lacking.

E. Technology evolution

For the evolution of SoS and the individual IoTs, several aspects must be considered: A) IoT technology updates, e.g., 1) New SOA protocols - autonomous translation 2) New encodings, semantics, ontologers - autonomous translation B) IoT functionality updates, e.g., Updates of produced service - autonomous update of service consumers

C) SoS management: updates to orchestration, workflow, security, etc. based on higher-level policies, strategies, product policies, business models, deals, etc. D) IoT configuration: updates of IoT configuration based on updated IoT firmware and produced services

Here, an appropriate SoS architecture must provide support for the introduction of new or updated IoT properties regarding updated or new 1) SOA protocols 2) Payload encoding and compression 3) Payload semantics (ontologies) So-called service buses are a common architectural component for addressing the need for such translation support [25]. However, the service bus is a centralized approach that requires all communication to pass through the service bus. A distributed and dynamic mechanisms for autonomous translations of protocols, encodings and semantics can be found in [26]. The autonomous translation of semantics is a very complex process and will most likely be solved using distributed AI/ML technologies [27].

F. SoS self-engineering

The updates of produced IoT service may require updates to the consuming IoTs. Architectures should have components that provide mechanisms for autonomous updates of service consumers such that the updates produced service can be utilized. Early work on autonomous updates of IoT consumer code can be found in [28].

G. SoS self-mitigation

The operation of very large-scale SoS will clearly exhibit operational failures due to various types of malfunctions and mismanagement. Thus, mitigation mechanisms will be necessary architectural components. Approaches for such mitigation mechanisms were developed already in the early 2000s by IBM [29]. Various concepts for addressing mitigation of deviations from the desired SoS functionality and properties are currently being developed in several large EU projects. Some of the functionalities and properties addressed in these projects are re-orchestration due to component failure or supply chain re-optimization, SoS security [30] SoS standard compliancy[31], and SoS safety [32].

H. SoS M2M business models and nano-payments

The provision of business models and mechanisms to the very large-scale SoS discussed here appears inevitable. This will require means of creating trusted logs of actions and connecting such data to nano-payments between the machines. Early concepts and implementations of such approaches can be found in [33], [34].

IV. Conclusion

It is argued here that for very large automation and digitalization solutions, the current architecture and automation and digitalization engineering strategies are not sufficient. It is clear that several new architectural concepts/components are needed. These include the following: A) IoT: Policy-driven functional evolution, self-engineering and autonomous protocol and semantics translation. B) SoS: Run-time dynamics, segmentation, scalability, self-mitigation, self-engineering, policy-driven management, M2M business models, M2M nano-payment. C) Engineering support: evolution engineering, segmentation engineering, multidimensional SoS policy engineering and management strategies and policies

Such architectural and engineering component and concepts should to a very large extent be autonomous and automatically engineered through the implementation platforms and the involved machines.

Acknowledgment

The author would acknowledge the financial support from ECSEL-JU project Productive4.0 GAP-737459 and Arrowhead Tools GA 826452.

REFERENCES

[1] J. Delsing, "Local cloud internet of things automation: Technology and business model features of distributed internet of things automation solutions," *IEEE Industrial Electronics Magazine*, vol. 11, no. 4, pp. 8–21, Dec 2017.

[2] J. Delsing, J. Eliasson, J. van Deventer, H. Derhamy, and P. Varga, "Enabling IoT automation using local clouds," in *Proceedings World Forum - IoT 2016*. IEEE, Dec. 2016.

[3] *ISA95, Enterprise-Control System Integration*, ISA Std.

[4] P. Adolphs and et.al, "Status report: RAMI4.0," VDI/VDE-Gesellshact Mess- und Automatisierungstechnik, Tech. Rep., June 2015.

[5] S.-W. Lin, M. Crawford, and S. Mellor, "The industrial internet of things volume g1: Reference architecture," Industrial Internet Consortium, $http://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf$, Tech. Rep., 2016.

[6] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, Feb 2005.

[7] A. W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, and J. L. Lastra, "Industrial cloud-based cyber-physical systems," *The IMC-AESOP Approach*, 2014.

[8] S. Karnouskos and A. W. Colombo, "Architecting the next generation of service-based SCADA/DCS system of systems," in *Proceedings IECON 2011*, Melbourne, Nov. 2011, p. 6.

[9] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann, "Towards an architecture for service-oriented process monitoring and control," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010, pp. 1385–1391.

[10] L. Garcés, F. Oquendo, and E. Y. Nakagawa, "Software mediators as first-class entities of systems-of-systems software architectures," *Journal of the Brazilian Computer Society*, vol. 25, no. 1, p. 8, Aug 2019. [Online]. Available: https://doi.org/10.1186/s13173-019-0089-3

[11] C. E. d. B. Paes, V. V. G. Neto, T. Moreira, and E. Y. Nakagawa, "Conceptualization of a system-of-systems in the defense domain: An experience report in the brazilian scenario," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2098–2107, Sep. 2019.

[12] O. Carlsson, "Engineering of iot automation systems," Ph.D. dissertation, Lulea University of Technology, Lulea, Sweden, 2017.

[13] N. S. Nafi, K. Ahmed, M. A. Gregory, and M. Datta, "A survey of smart grid architectures, applications, benefits and standardization," *Journal of Network and Computer Applications*, vol. 76, pp. 23 – 36, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516302314

[14] M. Hashmi, S. Hänninen, and K. Mäki, "Survey of smart grid concepts, architectures, and technological demonstrations worldwide," in *2011 IEEE PES conference on innovative smart grid technologies, Latin Americe (ISGT LA)*, Oct 2011, pp. 1–7.

[15] J. Delsing, "Private communication with global autmation companies like ABB, Siemens, Schneider, Valmet, Honeywell."

[16] J. Lindström, A. Hermanson, F. Blomstedt, and P. Kyösti, "A multi-usable cloud service platform: a case study on improved development pace and efficiency," *Applied Science*, vol. 8, no. 2, 2018. [Online]. Available: http://www.mdpi.com/2076-3417/8/2/316

[17] (2019) IEC 81346:2019 industrial systems, installations and equipment and industrial products - structuring principles and reference designations. [Online]. Available: https://www.iso.org/standard/75265.html

[18] J. Delsing, Ed., *IoT Automation - Arrowhead Framework*. CRC Press, Feb. 2017, no. ISBN 9781498756754.

[19] Eclipse basyx. [Online]. Available: https://www.eclipse.org/basyx/

[20] B. Otto and at.al., "Reference architecture model for the industrial data space," Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V., Tech. Rep., 2017.

[21] Wikipedia, "FiWare — wikipedia, the free encyclopedia," 2016, [Online; accessed 21-April-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=FIWARE&oldid=711836994

[22] K. K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing, and R. J. DeLong, "An AAA solution for securing industrial IoT devices using next generation access control," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, May 2018, pp. 737–742.

[23] A. Bicaku, S. Maksuti, C. Hegedus, M. Tauber, J. Delsing, and J. Eliasson, "Interacting with the arrowhead local cloud: On-boarding procedure," in *Proc. IEEE ICPS 2018*, 2018.

[24] C. Hegedus, P. Varga, and A. Frankó, "Secure and trusted inter-cloud communications in the arrowhead framework," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, May 2018, pp. 755–760.

[25] M. . Schmidt, B. Hutchison, P. Lambros, and R. Phippen, "The enterprise service bus: Making service-oriented architecture real," *IBM Systems Journal*, vol. 44, no. 4, pp. 781–797, 2005.

[26] H. Derhamy, J. Eliasson, and J. Delsing, "Iot interoperability - on-demand and low latency transparent multi-protocol translator," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[27] J. Nilsson, F. Sandin, and J. Delsing, "Interoperability and machine-to-machine translation model with mappings to machine learning tasks," in *arXiv preprint arXiv*, vol. 1903.10735, 2019.

[28] C. Paniagua and J. Delsing, "Interoperability mismatch challenges in heterogeneous soa-based systems," in *Proceeding INDIN 2019*, 2019.

[29] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan 2003.

[30] S. Maksuti, M. Tauber, and J. Delsing, "Generic autonomic management as a service in a soa-based framework for industry 4.0," in *Proc. IECON 2019*, 2019.

[31] A. Bicaku, M. Tauber, and J. Delsing, "Standard compliance and continuous verification in cyber physical systems," *IEEE Communications Surveys & Tutorials*, vol. Submitted to, 2019.

[32] M. I. Rezabal, L. Etxeberria, X. Elkorobarrutia, J. M. Perez, F. Larrinaga, and G. Sagardui, "MDE based iot service to enhance the safety of controllers at runtime," in *Proc. STAF-2019 MDE@DeRun*, 2019.

[33] E. Palm, O. Schelen, U. Bodin, and R. Hedman, "The exchange network: An architecture for the negotiation of non-repudiable token exchanges," in *Proc. INDIN 2019*, Helsinki, 2019.

[34] E. Palm, O. Schelen, and U. Bodin, "Approaching non-disruptive distributed ledger technologies," *IEEE Access*, Submitted for publication 2019.