Continuous Adaptive Runtime Integration Testbed for Complex and Autonomous Systems

Christopher Stewart, The Ohio State University

I. Abstract

The integration of complex, distributed systems typically takes thousands of man-hours and years of detailed design and testing. Despite all this effort, the effectiveness of classical system integration becomes a major issue when one adds the runtime adaptive behavior that is increasingly vital for dynamic, autonomous systems influenced by their surroundings. Self-integrating systems proposed in recent research strive to autonomously integrate new components, reducing design and testing costs. However, these systems are challenging to validate, especially at scale. This paper describes CHARIOT, Continuous High-level Adaptive Runtime IntegratiOn Testbed, which allows for different approaches and systems to be dynamically deployed, assessed and compared on a shared common platform. CHARIOT uses self-flying drones and self-driving cars to validate autonomous signal integration. This paper will discuss our early work in designing the CHARIOT architecture and integration protocol.

II. Introduction

The integration of complex, distributed systems typically takes thousands of man-hours and years of detailed design and testing. Despite all this effort, the effectiveness of classical system integration becomes a major issue when one adds the runtime adaptive behavior that is increasingly vital in today's world of interconnected and relatively open networks of systems. Part of that desired adaptivity is the ability of integrated networked systems to use other systems over which they have little or no control and little knowledge.

Self-integrating systems proposed in recent research automate integration [1], but can be challenging to validate at scale. This validation occurs both intrinsically and extrinsically, where the systems must be able to assess and maintain their own integration status within the overall system composition. Intrinsic validation simultaneously facilitates and utilizes the reflection necessary for establishing self-awareness, while extrinsic validation informs the global integration management.

Common integration settings involve system-of-systems design in specific, mostly well-defined domains such as manufacturing [2] or healthcare [3]. These often do not involve advanced reflective capabilities and runtime adaptivity mechanisms which are necessary for more generic and open self-integration scenarios as, for instance, in cyber-physical or socio-technical systems with highly diverse constituents.

In order to facilitate the development of such self-integrating systems, we are building CHARIOT, the Continuous High-level Adaptive Runtime IntegratiOn Testbed, to allow for different approaches and systems to be dynamically deployed, assessed and compared on a shared common platform [4]. This paper will focus on one of the testbed use cases: autonomous unmanned aerial systems, outlining their structure and providing concrete examples of their early realization.

III. CHARIOT

The DARPA Active Networks program [5] focused on different technologies but had some similarities: it started by taking previously fixed and passive aspects of networking technology and explored new approaches to developing and managing a network of adaptive components. In the case of CHARIOT, we are taking what is usually fixed in system integration at design time and developing and managing approaches that shifts this integration to a runtime integration managed largely by the participating systems in an operational environment.

The testbed provides the middleware to accommodate different kinds of integration approaches for use with diverse subsystems and to explore the higher-order dynamics, as illustrated in Figure 2. An individual self-integrating system here can, e.g., be based on the Controller/Observer model [6]. The physical network layer shall either be simulated for full controllability, constitute an actual distribution, or be a combination of both.

The proposed testbed aims at allowing several self-* systems to plug into a shared environment, and experimenting with their integration, in various forms. Different kinds of integration will be supported: ad-hoc; purposeful; organised (cooperation, competition); etc.

Testbed Use-Case: Autonomous Unmanned Aerial Vehicles (AUAV): Unammend aerial systems have transformed cyberphysical systems. Quadrotor machines allow in-place take off, agile flight manuevers and sufficient battery capacity to complete real missions [7], [8]. However, they require human pilots— even if not in the cockpit— which inflates the cost of missions, as capable pilots are hard to find. Autonomous unmanned aerial systems reduce the cost of piloting by replacing humans with software [9], [10], [11]. Not only does software eschew costly pilot labor, it enables qualitatively different missions where aircraft sample flight paths and produce reports on partial data. Essentially, software piloted systems enable answer quality versus computation tradeoffs [12], [13]. Systems research has sought workloads with this property in multiple domains, e.g., Hadoop [14].

Figure 1 shows the software architecture of AUAV. At the lowest levels, traditional computer systems components

Layer	Function	Key Technology
hardware	execute workloads, sense & affect the physical world	Flight, compute & edge resources
runtime manager	map workloads to hetero-geneous & dynamic devices	java, containers, sdn, coap
device/ middleware api	provide interfaces to control UAV sensors and actuators	dji android api, rapl, opencV, etc.
application	workflow for sensing, pattern detection & actuation	java, python, scratch
use-case customization	customize sensed patterns to use cases	json, python, ML models

Fig. 1. Software architecture for AUAV

reside, such as, hardware, operating systems and networking components. Note, AUAV must balance the resource demands for computational hardware *and* aircraft and battery resources. While recent research has plotted paths to power efficient computer systems under tight budgets [15], [16], [17], it remains challenging to manage independent components with correlated access patterns. A key challenge for CHARIOT will be to devise autonomous systems that can collect arbitrary environmental signals and discover correlations that will aid resource management.

Figure 1 also depicts the key challenge facing self integration for autonomous systems: The autonomous workloads themselves. As shown, there are two layers. First, machine learning and feature extraction systems must execute on data sensed from the environment. In the figure, we label such standard libraries as traditional *applications*. However, machine learning and other data-intensive workloads present inherent quality versus resource demand trade offs [12]. Consider an AUAV attempting to integrate signals from another nearby AUAV. Assume the other AUAV is flying a few feet ahead and witnesses the same world as our primary AUAV but a few seconds earlier. Clearly, the leader AUAV can provide useful tips on resource management. For example, *you can shut off your camera and face detector because there are no faces upcoming in our mutual path*. Even if the follower AUAV can detect and interpret this signal, it must decide whether to trust it. The leader could have configured its application-level quality and resource demands differently. It may have degraded resolution sufficiently to hide faces that the follower would otherwise see. In early work, we have shown that such resolution effects are very possible [11].

Beyond the application level, each autonomous aircraft must move around in its environment. Due to battery constraints on flight time, we anticipate that such movement will be driven by contextual information. AUAV operators will always look for opportunities to land early to reduce battery consumption. Here again, CHARIOT will allow developers to test autonomic signal integration.

IV. Challenges

From an engineering perspective, the CHARIOT team [4] has great experience developing and releasing open source code. Team member Stewart produced a widely used Web 2.0 benchmark [18], a smart classroom platform for 1300 inner-city students [19] and a performance modeling product for Hewlett Packard [20]. In the DARPA Active Networks program [5], Bellman and Launder developed a virtual-world platform still in use today and created numerous designs that have been integrated into NASA space equipment. Building the autonomous systems is not the core challenge, but will require substantial engineering effort in the coming years.

The research challenge stems from the development of self-integrating systems. These systems must not only sense the world, but also detect patterns, determine if those patterns are communication and decide how to use such communication to improve their own outcomes. Clearly, self-integration exceeds the capabilities of modern compute systems. To build such systems requires, multi-layer and multi-disciplinary engineering, including experimental and assessment approaches which, in terms of complexity, are on a par with the systems they consider [21].

The remainder of this paper adapts challenges outlined in [4] and phrases them in the context of AUAV.

Challenge 1: Domain-specific and Domain-independent Modeling. Each AUAV has a specific task, e.g., scouting crops [9] or taking selfies [11]. Knowledge about the operating domain and goal-driven context can improve self integration. In the simplest case, AUAV would hard-code their task and goals and share with other devices over computer networks. Of course, too much coding defeats the notion of self integration. We don't encourage such hard-coded solutions as they are reflective of existing integration techniques are are both labor intensive and brittle. Instead, all AUAV have to integrate at task and process levels independent of domains, i.e., without any prior knowledge. CHARIOT provides a testbed to explore system abstractions that allow such self integration.

Challenge 2: Intention Modelling and Collaborative Learning. Self-integrating systems require knowledge-based models in order to express and share their intention, goals, plans, beliefs, and experience. Machine learning capabilities are a crucial element in SISSY systems [1]. Learning in dynamic environments requires continual knowledge acquisition processes, i.e., experiences are used to construct and incrementally reinforce or attenuate existing knowledge elements in order to steer the systems' behaviors toward (approximately) optimal (self-)adaptation policies. Collective self-improving systems in shared environments and with similar goals provide ideal conditions for successfully applying, e.g., efficient multi-agent reinforcement learning approaches. The transfer of shareable knowledge aspects to adjacent and mutually dependent systems needs thorough investigation. Shareable knowledge in itself constitutes another challenge, namely, how to represent the acquired knowledge.

Challenge 3: Online Detection of Dependencies. This paper has glossed over hardware for networking. In fact, autonomous systems have multiple channels for communication— unlike less complicated cloud computing systems or HPC systems. For example, nearby AUAV could communicate through TCP-

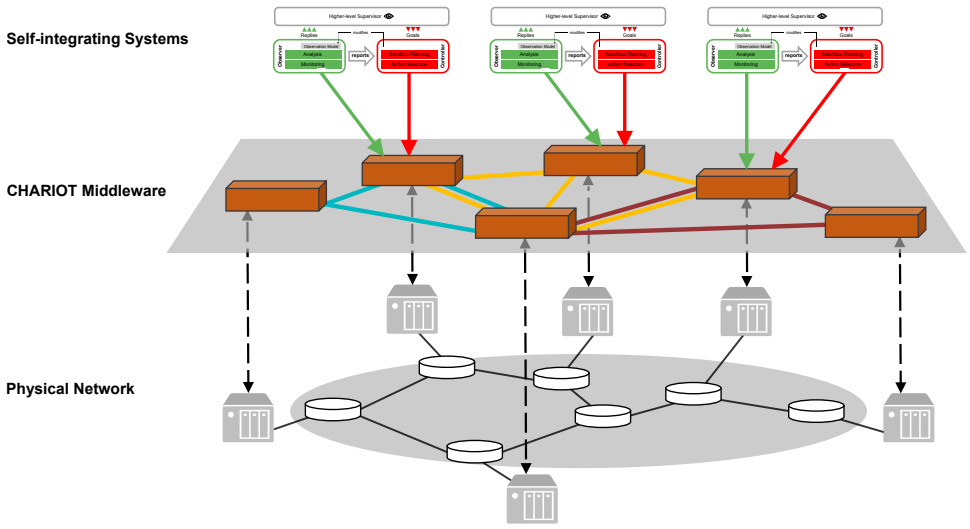

Fig. 2. Overall system structure including several interconnected self-integrating systems deployed on the CHARIOT middleware layer.

IP WI-FI, radar or via RGB cameras and flight actions. With so much flexibility, simply defining protocols is challenging. CHARIOT will allow system integration through a wide variety of channels. Two systems that share a channel are *adjacent*. Systems integrating into a larger collective of networked systems are characterized by adjacent dependencies. Adjacency does not not necessarily imply dependencies such as the necessity to cooperate for achieving the corresponding system goals. CHARIOT has to support the ability to detect and classify helpful dependencies and coincident adjacency.

References

[1] K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, A. Stein, S. Tomforde, and R. P. Würtz, "Self-Improving System Integration – Status and Challenges After Five Years of SISSY," in *Proc. 3rd IEEE Intl. Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 160–167, 2018.

[2] G. Morel, C. Pereira, and S. Nof, "Historical Survey and Emerging Challenges of Manufacturing Automation Modeling and Control: a Systems Architecting Perspective," *Annual Reviews in Control*, 2019.

[3] K. P. Stone, L. Huang, J. R. Reid, and E. S. Deutsch, "Systems Integration, Human Factors, and Simulation," in *Comprehensive Healthcare Simulation: Pediatrics* (V. J. Grant and A. Cheng, eds.), pp. 67–75, Springer International Publishing, 2016.

[4] C. Barnes, K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. Lewis, P. Nelson, A. Stein, C. Stewart, and S. Tomforde, "CHARIOT – Towards a Continuous High-level Adaptive Runtime Integration Testbed," in *International Workshop on Self-Improving System Integration (SISSY)*, 2019.

[5] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[6] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck, "Observation and Control of Organic Systems," in *Organic Computing – a Paradigm Shift for Complex Systems*, pp. 325–338, Birkhäuser Verlag, 2011.

[7] J. L. Sanchez-Lopez, R. A. S. Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy, "Aerostack: An architecture and open-source software framework for aerial robotics," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, 2016.

[8] J. Dunn, "Drones are growing rapidly, regardless of what the government does," 2017.

[9] J. Boubin, J. Chumley, C. Stewart, and S. Khanal, "Autonomic computing challenges in fully autonomous precision agriculture," in *IEEE International Conference on Autonomic Computing*, 2019.

[10] J. Boubin, S. Zhang, V. Mandadapu, and C. Stewart, "Characterizing computational workloads in uav applications," in *IEEE Internet-of-Things Design and Implementation*, 2018.

[11] Jayson Boubin, Naveen Tukmur Ramesh Babu, Christopher Stewart, Shiqi Zhang, John Chumley, "Managing edge resources for fully autonomous aerial systems," in *ACM Symposium on Edge Computing (SEC)*, 2019.

[12] J. Kelley, C. Stewart, D. Tiwari, Y. He, S. Elnikey, and N. Morris, "Measuring and managing answer quality for online data-intensive services," in *IEEE International Conference on Autonomic Computing*, 2015.

[13] J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, "Obtaining and managing answer quality for online data-intensive services," in *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 2017.

[14] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approxhadoop: Bringing approximations to mapreduce frameworks," in *ACM SIGARCH Computer Architecture News*, 2015.

[15] N. Morris, C. Stewart, L. Chen, R. Birke, and J. Kelley:, "Model-driven computational sprinting," in *ACM European Conference on Computer Systems*, 2018.

[16] N. Deng, C. Stewart, J. Kelley, D. Gmach, and M. Arlitt, "Adaptive green hosting," in *IEEE International Conference on Autonomic Computing*, 2012.

[17] C. Stewart and K. Shen, "Some joules are more precious than others: Managing renewable energy in the datacenter," in *ACM Workshop on Power Aware Computing and Systems*, 2009.

[18] C. Stewart, M. Leventi, and K. Shen, "Empirical examination of a collaborative web application," in *IEEE International Symposium on Workload Characterization (special session on Benchmark Innovation)*, 2008.

[19] S. Renganathan, C. Stewart, A. Perez, R. Rao, and B. Braaten, "Preliminary results on an interactive learning tool for early algebra education," in *IEEE Frontiers in Education*, 2017.

[20] C. Stewart, T. Kelly, and A. Zhang, "Exploiting nonstationarity for performance prediction," in *European Conference on Computer Systems*, 2007.

[21] S. R. Ray, "The Future of Software Integration: Self-Integrating Systems," in *Proc. SDP Workshop on New Visions for Software Design & Productivity: Research & Applications*, 2001.