Self-organization of Cyber-physical systems using a Goal-Oriented approach

Gabriel Rodrigues, Genaina Rodrigues
Departamento de Ciłncia da Computao
Universidade de Braslia (UnB)
Campus Universitrio Darcy Ribeiro - Asa Norte
Braslia, DF, Brazil 70910-900
Email: gabrielsr@gmail.com, genaina@unb.br

***Abstract*—Cyber-physical systems (CPS) integrate computational and physical components in large scale systems. By interacting with the physical world a CPS can potentially provide impacting solutions in a number of domains such as healthcare, transportation, manufacturing and agriculture. CPS is commonly supposed to be executed in a large scale network of different and unreliable devices. Develop a system for such complex environment require new software engineering approaches. We present in this work the state of our ongoing research on a goal-oriented self-organizing approach for deployment of CPS in a distributed, dynamic and heterogeneous environments.**

I. Introduction

Cyber-physical systems (CPS) integrate computational and physical components in large systems[]. CPS can have a great impact in society. Internet of Things (IoT) [**?**], Assisted Living [**?**], Smart Cities, Industry 4.0 are examples of domains in which CPS can be applied. Although CPS has a great potential for impact, it presents challenges for application developers.

CPS are supposed to be executed in distributed heterogeneous and dynamic computing environments largely unknown at design-time. The computing environment is distributed and heterogeneous if is composed of different devices with different sets of resources. Besides computing resources (CPU, memory, storage, networking), these devices can have different sets of sensors and actuators that allows it to interact with the physical environment. The computing environment is dynamic when devices can enter and leave the environment. In CPS dynamicity is common as many devices can be mobile, powered by battery and/or susceptible to failure. Additionally, a CPS can be intended to a computing environment that is largely unknown at design-time as (a) different users can have different sets of devices and (b) the deployment environment evolves over time as new devices can be added to the environment, while old devices are decommissioned. All these attributes of the environment add complexity to the deployment process of a CPS, which can be limiting to cyber-physical application providers.

II. Background

Software deployment is the process of getting a software ready for use in its operational environment [6]. It includes selecting suitable artifacts to deploy, moving them to the target environment, configuring the environment, and starting the execution. Continuous Integration/Continuous Delivery (CI/CD) pipelines are frequently used by software engineer teams to automate the process of deploying software.

Self-adaptive systems (SAS) are systems that are capable of monitor their operation and manage it-self according. SAS can improve a system execution while avoiding the need for human intervention. SAS typically use some kind of loop to monitor, analyse and decide on adaptation actions. Component Based adaptation approaches are the ones where the adaptation actions are related to selection of software components.

Goal Requirements Engineering (GORE) was first proposed as a model to represent software requirements. More recently GORE paved the way for solutions that can leverage self-adaptive configurations, behaviors and strategies. In particular, goal-oriented modeling captures what stakeholders want to achieve in terms of the goals and alternative strategies they can adopt. Such strategies can be expressed through AND / OR refinements, embodied as alternative sets of executable tasks [16]. Goal modeling can be useful in context of self-adaptive software as it allows the designers to represent variability at design in the form of OR refinements. Contextual Goal Models (CGMs) extend objective models [2] by adding contextual conditions on the need for goals and the appropriateness of such alternatives. GORE models were used to drive and design software architecture and as a basis for engineering self-adaptive systems [17]. The CGM would be a suitable base model for our highly heterogeneous environments, both by accommodating variability across alternative strategies to achieve goals and by being able to capture the contextual restrictions in the applicability of a given strategy.

Goal-oriented Adaptive Systems are SAS that uses a Goal model as a high level reference model to drive the adaptation. This means that in a Goal-oriented adaptive system a model that represents the system Goals is kept at runtime. This model is constantly analysed in order to make adaptation decisions. Close-adaptive systems are the ones that have a fixed set of possible adaptive actions, and no new behaviors and alternatives can be introduced at runtime. On the other hand, open-adaptive systems are the ones that can be extended at runtime by adding new alternatives to the adaptation mechanism.

In previous work we have proposed Goalp, an approach for automatic goal-oriented deployment planning in heterogeneous environments. In Goalp, software components should be annotated with information about the (a) goals that the component can achieve; (b) the required runtime resources; (c) components requirements in terms of sub-goal or tasks and (d) level of service quality that the component could provide as a function of available resources. Goalp we introduced the idea of goal-oriented deployment planning. In a later work we proposed GoalD, an autonomous extension of Goalp. GoalD extended Goalp to efficiently adapt the deployment in response to changes in the environment. GoalD introduced a runtime model that keeps the system goals, alternatives to achieve that goals and algorithms to device adaptation plans in response to changes. As a limitation, both works are centralized approaches and the deployment can be planned for a single device. Centralized adaptation approaches are known to have limitations. More specifically in a CPS, a centralized approach can have a limited use as a single node can not have all required physical resources, as sensors and actuators are commonly distributed across different devices.

Multi-agent systems (MAS) are systems composed of multiple autonomous components that can autonomously decide on its actions. MAS can be used in distributed environments as an alternative to centralized controlled systems. As stated before, centralized adaptation processes are well known to have limitations, such as (a) single point of failures; (b) scalability issues related to move all the data needed for adaptation decision to the central decision unity. MAS approaches can avoid these problems. Multi-agent reinforcement learning (MARL) are MAS where agents use reinforcement learning in order to create policies for choosing actions.

Holonic systems are inspired by nature. The main idea is that complex systems can evolve faster if based in an underlying more stable structures (e.g. a complex animal evolved based on an underlying cell structure). In Holonic systems the holon is the compositional organizational unit. The role of the holon is twofold: part and whole. As part, it can receive commands and emit notification events. As whole, it can issue commands to its constituent members and be influenced by notification events of such components. Diaconescu at all [] proposed a Goal-oriented holonic approach for self-integration of systems in which each holon has a model of its goals.

III. The Approach

IV. Preliminary Results

V. Conclusion

In summary, CPS can have a great impact in society but in order to ease the development of practical innovations in CPS an approach to handle complexity at the computing environment is required. The complex computing environment in a CPS have the following characteristics: distributed, dynamic, heterogeneous environments and partially unknown at design-time. While there is existing promising goal-oriented self-adaptive approaches that can handle some of challenges in a CPS, to the best of our knowledge there is no proposed challenge that handle all these challenges.

VI. Conclusion

The conclusion goes here.

Acknowledgment

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.