Universität Augsburg

A Taxonomy for Organic Computing Systems regarding Mutual Influences

Stefan Rudolph, Sven Tomforde

Report 2016-03 April 2016

Institut für Informatik
D-86135 Augsburg

Abstract

In Organic Computing (OC) applications, we often face mutual influences between the entities of the system. These influences can be either explicit, i.e., directly visible for the designer, or implicit, i.e., they are not visible on first sight. In previous work, we developed a methodology to make these implicit influences measurable. In this work, we present a taxonomy that classifies OC applications regarding their nature of influence within the system. This taxonomy is helpful for the selection of suitable methods for the detection of hidden mutual influences.

1 Introduction

In recent years, we observe a significant increase of the complexity in technical systems due to growth of interconnection between devices. Due to problems that occur when such a large networks of devices is controlled by a central unit, the Organic Computing (OC) initiative [1] uses the concepts of *self-adaption* and *self-organization* for mastering the resulting complexity issues. In this context, we face systems that consist of several to huge numbers of entities that have interdependencies. We call this class of systems *Interwoven Systems* [2, 3, 4]. Such interdependencies can be either explicit, which means that they are easy to observe or they can be implicit, meaning that they do not reveal to the observer at first sight. To resolve this issue, a methodology to make mutual influences explicit has been presented in [5, 6].

The main contribution of this work is to define a taxonomy that allows us to find classes of OC systems that have different characteristics regarding their mutual influences. This is the first step in the direction of a *guideline* that allows to give general rules on how the influences can be detected best in the different classes.

The remainder of this article organized as follows: in Section 2, the general notion of mutual influences and the methodology for the measurement of them is outlined. Afterwards, in Section 3, the taxonomy is introduced. Furthermore, in Section 4, we see two example applications that are classified regarding the taxonomy. Finally, the article is concluded in Section 5.

2 Mutual Influences

In the previous section, we briefly scratched our notion of the detection of mutual influences in distributed and self-organized systems. In order to formalize what is meant with this brief description, this section outlines our utilized system model which is inspired from standard machine learning notions. Despite the background in machine learning, the methodology is assumed to be applicable to all systems covering the basic system model. Afterwards, we continue this

section with the presentation of the mutual influence detection algorithm. The system model and algorithms have been originally proposed in [5]. However, we briefly describe them here for the purpose of better readability of the following sections. For further details please see [5].

2.1 System Model

We start with a set of entities $\{E_1, \ldots, E_n\}$, where each entity can assume different configurations. Such a configuration typically consists of different parts. Consider a router as a simple example: The router can take varying configurations into account, such as the processed network protocol or parameter settings (i.e. for time-out intervals, buffer sizes, etc.). We define the whole configuration space of an entity E_i as cartesian product $C_i = c_{i1} \times \ldots \times c_{im}$, where c_{ij} are the parts of the configuration. A further assumption is that the particular configurations of individual entities are non-overlapping, meaning each entity has its own set of configurations, $c_{ij} \neq c_{kl}$ for all defined $i \neq k, j, l$. This does not mean that the configuration parts have to be completely disjoint in structure and values of the contained variables. For instance, two routers might have the possibility to configure the time-out interval, which would lead to the same set of possible configurations in these attributes, but on different devices. Such a relation is explicitly allowed within the model.
Besides the configuration space, we need to consider a further element: the *local performance measurement*. In order to apply the proposed method, each entity has to estimate the success of its decisions at runtime – as a response to actions taken before. This is realized based on a feedback mechanism – with feedback possibly stemming from the environment of the entity (i.e. *direct* feedback) or from manual assignments (i.e. *indirect* feedback). This resembles the classic reinforcement model, where the existence of such a performance measurement (mostly called *reward*) is one of the basic assumptions, cf., e.g., [7].

2.2 Measurement

Given the described system model, we continue with the actual methodology for the measurement of mutual influences. The goal is to identify those other entities that have influence on the entity itself.

The basic idea of the algorithm is to make use of stochastic dependency measures that estimate associations between the performance of an entity A and the configuration parts of a second entity B. These dependency measures are designed to find correlations between two random variables X and Y. We therefore identify the performance of A with a random variable X and the performance of entity B with a random variable Y. This mapping implies that if the association between X and Y is high, we also have a high influence of B on A since it reflects that the configurations of B *matter* for the performance of A. Vice versa, if the association is low then, we do not see an influence.

For the measurement of the dependencies, we have so far considered the following (stochastic) dependency measures: the Pearson Correlation Coefficient [8], the Spearman Rank Correlation [9], the Kendall Rank Correlation [10], the Distance Correlation [11], the Mutual Information [12] and the Maximal Information Coefficient [13]. A comparison between these measures can be found in [6].

3 Taxonomy

In order to find suitable solutions for the dependency detection in as many application domains as possible, we identified the important characteristics of OC systems that can be used to give a guideline on how to measure the influences in the system.

3.1 Entities

Obviously, the number of entities is an interesting characteristic for the influence detection. If we look at OC systems, we can roughly identify three categories of systems:

(i) *small systems*, i.e., systems with few entities.

(ii) *middle-size* systems, i.e., systems with less than few hundred entities.

(iii) *large-scale systems*, i.e., systems with more than a few hundred entities.

Another characteristic that is related to the entities is the configuration space of them. As highlighted before the system model allows for multiple configuration parts (cf. Sec. refsec:system-model) that can have different forms. One interesting criteria is simply the number of configuration parts. Furthermore, we identified the following types of configuration parts:

- nominal: the different values can be categorized, but there is no order for the categories. For instance, categories like left and right.
- ordinal: the categories can be ordered. For instance, categories like low, medium and high, or 1, 2, 3.
- infinite real-valued: an infinity number of values can be assumed. For instance, this could be an interval $[0, 1]$.

For the types nominal and ordinal there is although another characteristic for classification which is the number of categories. In contrast, for the infinite real-valued class, we always assume that the set of values is infinite.

3.2 Communication

Regarding the detection of mutual influences, there are different system types regarding the communication that is possible between the entities and the associated costs. Within the context of detection, we face two border cases. The first one is the case where communication is free with all entities in the system. For instance, this can happen if the system is composed of virtual entities that utilized the same hardware, which would lead to negotiable communication costs. The second border case is that the communication is strictly limited to the neighbors. This is the border case since a limitation to no communication at all makes no sense because a potential influence cannot be detected. Between these two cases there is a variety of possibilities that reach from low to high costs for multi-hop communication.

3.3 Influence

A first characteristic that is especially interesting in the context of high communication costs is that the influence could originate from an entity that is in the neighborhood or one that can only be contacted over multiple hops. This leads to different detection possibilities regarding the possible communication.

Another characteristic of the influences is the possibility that multiple entities have to act jointly in order to reveal there influence. For example, two robotic arms have to hold a workpiece in place while another drills a hole in it. Either of the holding arms does not reveal its influence as long as they are observed separately.

Naturally, also the strength of influence is important. This can be classified regarding two aspects. The first is the type of dependency regarding the power of the dependency measures. Some of them are limited to linear or monotone dependencies, but the more powerful can measure stochastic dependencies which is the most general class of dependencies. The second aspect is the strength of the reflection of dependency in the joint distribution; it can be very distinctive or rather not distinctive.

One last important aspect regarding the influences in the system is that it can include temporal aspects. This means that in some cases the influence is imitate meaning that the results are reflected right after the configuration is assumed. However, there are also cases where the entity is affected with a delay.

4 Examples

We consider two examples here that are interesting in the context of mutual influences and in our current research focus. The first one are *smart camera networks* and the second is an *industry 4.0* application. We chose these two since they are applications with real-world background, show strong mutual influences and and have quite different characteristics.

4.1 Smart Camera Networks

Smart cameras are cameras with a build-in computation unit that can be utilized for several tasks, such as, image processing, object localization and object tracking. Also, most smart cameras have pan, tilt and zoom (PTZ) capabilities and the computation unit is used to determine beneficial alignments for the camera. Beyond, the smart cameras are equipped with wired or wireless communication devices that allow communication with neighbored cameras.

Such smart cameras can be utilized to achieve different goals. Exemplary, this can be the tracking of objects, the identification of new objects or the construction of 3D-models of objects. Of course, the performance measure, that is obligatory to fulfill the before presented system model, heavily depends on the chosen goal. For the purpose of this article, we assume a performance that is based on the detection of new objects in the area and additionally on the construction of a 3D-model of such. The second task needs at least two cameras observing the object at the same time. This implies that the performance of the cameras is strongly influenced by the other nearby cameras. Regarding the configuration as depicted in the system model, we consider the three adaptable parameters of the alignment, which are the pan, the tilt and the zoom of the cameras.

In the following, we classify the depicted smart camera networks regarding the taxonomy. Starting with the *entity* characteristics. If we look at the first characteristic, that is the number of entities, we do not have a clear classification since SCN kann have different sizes from few cameras to few hundred cameras. Large-scale systems are also possible, but, looking at the current development, this is not expected for the near future. The number of configuration parts in this domain is three, i.e., the pan tilt and the zoom. Each of the configuration parts is infinite real-valued.

Regarding the communication, we can face different instances of SCN. As mentioned before, smart cameras can be connected wired or wireless. In the case of a wireless ad-hoc network, we face high communication costs at least for multi-hop communication. If we face wired connections, the situation is not as demanding as in wireless communication, but, still limited.

Considering the influences in such system, we find that the influences are limited to the spatial neighborhood. This is since we the cameras can only be influenced by other cameras that share the potential field of view due to the nature of the performance measure. But we can not suspect that the influences can be detected by linear or monotonic measures, therefore we categorize them as stochastic. Moreover, we cannot find instances in which the influence only reveals if several neighbors act in common. Furthermore, we do not see a temporal influence in the system since previous configurations of the camera do not play a role for the other cameras. This is because the cameras only get higher

rewarded if the observe the same camera at the same time.

4.2 Industry 4.0

Behind the general term industry 4.0 one finds manifold attempts for interconnection and the further atomization of industrial facilities (cf. e.g. [14]). They reach from a machine that can be analyzed and operated via a tablet computer up to a fully autonomous, so called smart factory. The industry 4.0 application presented here is inspired by a smart factory demonstrator of the company Beckhoff[1]. Even though this basic demonstrator only consists of two autonomous parts, i.e., a transport system and a work station, it is already intended to use several of such autonomous systems [15]. Therefore, we pick this expected development up and create scenarios with multiple entities that are based on Beckhoff's demonstrator, but exceed the possibilities of it in some facets in order to reflect the expected developments in the area of smart factories. The two main components we focus this analysis on, are the *flexible transport system* and the work stations. The work stations are able to use different tools, such as different drills, saws, or die cutters. They have the possibility to configure which of the different tools are used during runtime and form the points in the smart factory where the workpieces are actually processed. The flexible transport system has several *mover*, i.e., trolleys, that hold workpieces and can move at individual speeds on a designated track. The track connects two work stations and the pick-and-place robot, i.e., a robotic arm that is able to pick up work pieces and put them in other places, e.g., another work station.

Regarding the entity characteristics, we can assume that such smart factory consists of a rather small system with few entities. This reflects the current state-of-the-art in this area. However, in principle, a smart factory could also be a large-scale system. The configuration space of the work station is composed of only a single configuration part that is nominal with a small number of categories.

For the communication, one can assume that the communication is rather cheap since the workstations are stationary. Moreover, a designated area can easily be connected by wire and could also be controlled by virtual agents that are emulated on a central computer. The situation changes if the system consists of several closed up areas that might not be connectible that easy.

Since in systems with a smaller number of nodes we can assume that all entities are neighbored, for instance, as virtual agents in a central device, we do not have to consider multi-hop influences here. In contrast, the situation is more complex regarding the influences that only reveal if several entities act in common, e.g., two robotic arms have to hold a workpiece in place while another drills a hole in it. Since the configuration space of the different entities in a

[1] https://www.beckhoff.com/ (accessed on 19.04.2016)

smart factory is rather small, it might be possible to use linear or monotonic dependency measures for this purpose. What can be clearly stated here is that the influences in the system will be temporal since the workpieces move through the system and will cause a delay in the occurrence of influences.

5 Conclusion

Concluding the work, we have seen a taxonomy regarding the detection of mutual influences in OC systems. The classification is illustrated by two exemplary application studies, i.e. a smart camera and an industry 4.0 example. Comparing the classification of these two systems, we see that they are quite different in some key characteristics, e.g., in the type of influences that occur.

In future work, we will create a guide to help designers of systems to find suitable methods from a mutual influence detection tool kit in order to ensure the best results regarding the different application classes.

6 Acknowledgment

Research support was provided by the DFG (German Research Foundation) under grants HA 5480/3-1, SI 674/9-1 and WA 2828/1-1.

References

[1] C. Müller-Schloer, H. Schmeck, and T. Ungerer, Eds., *Organic Computing - A Paradigm Shift for Complex Systems.* Birkhäuser, 2011.

[2] K. L. Bellman, S. Tomforde, and R. P. Würtz, "Interwoven systems: Self-improving systems integration," in *Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, September 8-12, 2014*, 2014, pp. 123–127.

[3] S. Tomforde, J. Hähner, and B. Sick, "Interwoven Systems," *Informatik-Spektrum*, vol. 37, no. 5, pp. 483–487, 2014, Aktuelles Schlagwort. [Online]. Available: http://dx.doi.org/10.1007/s00287-014-0827-z

[4] S. Tomforde, J. Hähner, H. Seebach, W. Reif, B. Sick, A. Wacker, and I. Scholtes, "Engineering and Mastering Interwoven Systems," in *Proc. of ARCS 2014 Workshops*, 2014, pp. 1–8.

[5] S. Rudolph, S. Tomforde, B. Sick, and J. Hähner, "A mutual influence detection algorithm for systems with local performance measurement," in *Proceedings of the 9th IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO).* IEEE Press, to appear, 2015.

[6] S. Rudolph, R. Hihn, S. Tomforde, and J. Hähner, *Architecture of Computing Systems – ARCS 2016: 29th International Conference, Nuremberg,*

Germany, April 4-7, 2016, Proceedings. Cham: Springer International Publishing, 2016, ch. Comparison of Dependency Measures for the Detection of Mutual Influences in Organic Computing Systems, pp. 334–347.

[7] M. Wiering and M. van Otterlo, Eds., *Reinforcement Learning: State-of-the-Art (Adaptation, Learning, and Optimization)*. Berlin / Heidelberg, Germany: Springer Verlag, 2012, iSBN-13: 978-3642276446.

[8] K. Pearson, "Notes on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 1, pp. 240 – 242, 1895.

[9] C. Spearman, "The proof and measurement of association between two things," *American Journal of Psychology*, vol. 15, pp. 88–103, 1904.

[10] M. Kendall, *Rank correlation methods*. London: Griffin, 1948.

[11] G. J. Szkely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *Ann. Statist.*, vol. 35, no. 6, pp. 2769–2794, 12 2007.

[12] C. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[13] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *Science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[14] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, "How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective," *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 8, no. 1, pp. 37 – 44, 2014. [Online]. Available: http://waset.org/Publications?p=85

[15] B. A. GmbH. (2014) Industrie 4.0 Forum: pc-based control concepts as core technology for the smart factory. [Online]. Available: http://ftp.beckhoff.com/download/document/catalog/Flyer_Industry_40_forum_2014.pdf