Digital Twins Composition via Markov Decision Processes

Giuseppe De Giacomo, Marco Favorito, Francesco Leotta, Massimo Mecella, Luciana Silo

Department of Computer, Control and Management Engineering,
Sapienza, University of Rome,
Via Ariosto, 25, 00185 Rome RM, Italy
{degiacomo,favorito,leotta,mecella,silo}@diag.uniroma1.it

- **Unità Lab AIIS:**
  Sapienza Università di Roma
- **Titolo del contributo:**
  Digital Twins Composition
  via Markov Decision Processess
- **Persone coinvolte:**
  Giuseppe De Giacomo
  Marco Favorito
  Francesco Leotta
  Massimo Mecella
  Luciana Silo
- **Temi di ricerca:**
  Artificial Intelligence
  Planning and Program Synthesis
  Reasoning under Uncertainty
  Reinforcement Learning
  Smart Manufacturing
  Digital Twins
- **Progetti:**
  ERC Advanced WhiteMech <https://whitemech.github.io>
  EU ICT-48 TAILOR <https://tailor-network.eu>
  EU DESTINI <https://destini2020.eu>
  EU FIRST <https://www.h2020first.eu>
- **Applicazioni in sviluppo/sviluppate/operative:**
  Smart Manufacturing
  Digital Twins
- **Challenges/prospettive:**
  See the main text
- **Riferimenti a paper e risorse su web:**
  See the main text

Abstract

The project we are going to exhibit is: Digital Twins Composition via Markov Decision Processes. The research group, which deals with Industry 4.0 through AI methods, worked on this project at the Sapienza University. Our research project focuses on the application of service composition techniques for Smart Manufacturing and IoT, considering Digital Twins as services to be orchestrated to implement complex industrial processes. In particular, we enrich the robustness of these processes through AI techniques such as Markov Decision Processes.

1 Introduction

In recent years we have been witnessing the continuous evolution of technologies in the fields of communication, networking, storage and computing, applied to the more traditional world of industrial automation. In order to increase productivity and quality, to ease workers' lives, and to define new business opportunities, smart manufacturing has emerged as the core of the Industry 4.0 revolution. DTs are up-to-date digital descriptions of physical objects and their operating status. Modern information systems and industrial machines may natively come out with their digital twin; in other cases, especially when the approach is applied to already established factories and production processes, digital twins are obtained by wrapping actors that are already in place. The main goal is to establish a tight integration between the physical world and the virtual world, in order to make production more efficient, reliable, flexible and faster. DTs are ideal tools to accomplish the purposes of Industry 4.0, since they enable a massive exchange of data that can be interpreted by analytical tools, in order to improve decision making.

Inspired by the research about automatic orchestration and composition of software artifacts, such as Web services, in [Catarci *et al.*, 2019] it has been argued that an important step towards the development of new automation techniques in smart manufacturing is the modeling of DT services and data as software artifacts, and that the principles and techniques for composition of artifacts in the digital world can be leveraged to improve automation in the physical one. In particular, starting from the Roman model for service composition [Berardi *et al.*, 2005], they consider smart manufacturing scenarios where DTs of physical systems provide stateful services wrapping the functionalities of machines and tasks of human operators.

Nevertheless, the proposed approach, suffer of an inherent limitation of the classical Roman model, which requires that the available services, i.e., the services that can be used to realize the target service, behave deterministically. This assumption is often unrealistic, because in practice the underlying physical system modeled as a set of services might show a non-deterministic behavior due to the complexity of the domain, or due to an inherent uncertainty on the dynamics of such system.

Moreover, the above-mentioned techniques work only when the target is fully realizable, i.e., the specification can either be satisfied or not, with no middle ground. In the context of Industry 4.0 this might be seldom the case, and instead it would be preferred a technique that, rather than returning no answer, returns the "best-possible" solution under the actual circumstances. The work [Brafman *et al.*, 2017] contributes to this direction by providing a solution technique, based on Markov Decision Processes (MDPs) that coincides with the exact solution if a composition exists; otherwise, it provides an approximate solution that maximizes the expected sum of values of the target service's requests. Unfortunately, such model is not expressive enough to capture the non-determinsitic behaviour of the available services which, as argued above, is a must-have in our setting.

In our work, we marry the vision of employing service composition techniques to orchestrate digital twins. We propose a generalization to the service composition in stochastic setting proposed in [Brafman *et al.*, 2017], in which not only the target but also the services are allowed to behave stochastically. Moreover, we allow the services to be taken into account in the optimization problem by associating a reward to each service's transition, besides the target's rewards.

2 Preliminaries results

Before stating the problem, we give preliminary definitions.

2.1 Background

MDPs. A Markov Decision Process (MDP) $\mathcal{M} = \langle S, A, T, R\rangle$ contains a set S of states, a set A of actions, a transition function $T : S \times A \rightarrow Prob(S)$ that returns for every state s and action a a distribution over the next state, and a reward function $R : S \times A \rightarrow \mathbb{R}$ that specifies the reward (a real value) received by the agent when transitioning from state s to state s' by applying action a. A solution to an MDP is a function, called a *policy*, assigning an action to each state, possibly with a dependency on past states and actions. The *value* of a policy ρ at state s, denoted $v_\rho(s)$, is the expected sum of (possibly discounted by a factor λ, with $0 \leq \lambda < 1$) rewards when starting at state s and selecting actions based on ρ. Typically, the MDP is assumed to start in an initial state s_0, so policy optimality is evaluated w.r.t. $v_\rho(s_0)$. Every MDP has an optimal policy ρ^*. In discounted cumulative settings, there exists an optimal policy that is Markovian $\rho : S \rightarrow A$, i.e., ρ depends only on the current state, and deterministic [Puterman, 1994]. Among techniques for finding an optimal policy of an MDP, there are *value iteration* and *policy iteration* [Sutton e Barto, 2018].

The Roman Model in stochastic settings. The problem of service composition, i.e. the ability to generate new, more useful services from existing ones, has been considered in the literature for over a decade [Hull, 2008; Medjahed e Bouguettaya, 2011; De Giacomo *et al.*, 2014]. The goal is, given a specification of the behavior of the target service, to build a controller, known as an *orchestrator*, that uses existing services to satisfy the requirements of the target service. Here we concentrate on the approach known in literature as the "Roman model" [Berardi *et al.*, 2003; Berardi *et al.*, 2005]: each available service is modeled as a finite-state machines (FSM), in which at each state, the service offers a certain set of actions, where each action changes the state of the service in some way. The designer is interested in generating a new service (referred to as target) from the set of existing services. The required service (the requirement) is specified using a FSM, too.

Unfortunately, it is not always possible to synthesize a service that fully conforms with the requirement specification. This zero-one situation, where we can either synthesize a perfect solution or fail, often is not very applicable. Rather than returning no answer, we may want notion of the "best-possible" solution. A model with this notion has been developed in [Brafman *et al.*, 2017], where the authors discuss and elaborate upon a probabilistic model for the service composition problem, first presented in [Yadav e Sardina, 2011]. In this model, an optimal solution can be found by solving an appropriate probabilistic planning problem (e.g. an MDP) derived from the services and requirement specifications. Due to lack of space, we do not report the details of such technique.

2.2 Problem

A *stochastic service* is a tuple $\tilde{\mathcal{S}} = \langle \Sigma_s, \sigma_{s0}, F_s, A, P_s, R_s\rangle$, where Σ_s is the finite set of service states, $\sigma_{s0} \in \Sigma$ is the initial state, $F_s \subseteq \Sigma_s$ is the set of the service's final state, A is the finite set of services' actions, $P_s : \Sigma_s \times A \rightarrow Prob(\Sigma_s)$ is the transition function, and $R_s : \Sigma_s \times A \rightarrow \mathbb{R}$ is the reward function. In short words, the stochastic service is the stochastic variant of the service defined in the classical Roman model, and it can be seen as an MDP itself.

A *target service*, as defined in [Brafman *et al.*, 2017], is $\mathcal{T} = \langle \Sigma_t, \sigma_{t0}, F_t, A, \delta_t, P_t, R_t\rangle$, where Σ_t is the finite set of service states, $\sigma_{t0} \in \Sigma$ is the initial state, $F_t \subseteq \Sigma$ is the set of the service's final state, A is the finite set of services' actions, $\delta_t : \Sigma \times A \rightarrow \Sigma$ is the service's deterministic and partial transition function, $P_t : \Sigma_t \rightarrow \pi(A) \cup \emptyset$ is the action distribution function, $R_t : \Sigma_t \times A \rightarrow \mathbb{R}$ is the reward function.

A *stochastic system service* $\tilde{\mathcal{Z}}$ of a community of stochastic services $\tilde{\mathcal{C}} = \{\tilde{\mathcal{S}}_1, \dots, \tilde{\mathcal{S}}_n\}$ is a stochastic service where $\tilde{\mathcal{Z}} = \langle \Sigma_z, \sigma_{z0}, F_z, A, P_z, R_z\rangle$ are defined as follows: $\Sigma_z = \Sigma_1 \times \dots \times \Sigma_n$, $\sigma_{z_0} = (\sigma_{10}, \dots, \sigma_{n0})$, $F_z = \{(\sigma_1, \dots, \sigma_n) \mid \sigma_i \in F_i, 1 \leq i \leq n\}$, $A_z = A \times \{1, \dots n\}$ is the set of pairs (a, i) formed by a shared action a and the index i of the service that executes it, $P_z(\boldsymbol{\sigma'} \mid \boldsymbol{\sigma}, (a, i)) = P(\sigma'_i \mid \sigma_i, a)$, for $\boldsymbol{\sigma} = (\sigma_1 \dots \sigma_n)$, $\boldsymbol{\sigma'} = (\sigma'_1 \dots \sigma'_n)$ and $a \in A_i(\sigma_i)$, with $\sigma_i \in \Sigma_i$ and $\sigma_j = \sigma'_j$ for $j \neq i$, $R_z(\boldsymbol{\sigma}, (a, i)) = R_i(\sigma_i, a)$ for $\boldsymbol{\sigma} \in \Sigma_z$, $a \in A_i(\sigma_i)$.

We define the set of joint histories of the target and the system service as $H_{t,z} = \Sigma_t \times \Sigma_z \times (A \times \Sigma_t \times \Sigma_z)^*$. A joint history $h_{t,z} = \sigma_{t,0}\sigma_{z,0}a_1\sigma_{t,1}\sigma_{z,1}a_2\dots$ is an element of $H_{t,z}$. The projection of $h_{t,z}$ over the target (system) actions is $\pi_t(h_{t,z}) = h_t$ ($\pi_z(h_{t,z}) = h_z$). An orchestrator $\gamma : \Sigma_t \times \Sigma_z \times A \rightarrow \{1, \dots, n\}$, is a mapping from a state of the target-system service and user action $(\sigma_t, \sigma_z, a) \in \Sigma_t \times \Sigma_z \times A$ to the index $j \in \{1, \dots, n\}$ of the service that must handle it. Crucially, since the stochasticity comes also from the services, the orchestrator *does* affect the probability of an history $h_{t,z}$. Moreover, in general, there are *several* system histories associated to a given target history.

Let $P_\gamma(h) = \prod_{i=0}^{|h|} P_t(\sigma_{t,i}, a_{i+1}) P_z(\sigma_{z,i+1} \mid \sigma_{z,i}, \langle a_{i+1}, \gamma(\sigma_{t,i}, \sigma_{z,i}, a_{i+1})\rangle)$ be the probability of a (joint) history $h = \sigma_{t0}\sigma_{z0}\langle a_1, j_1\rangle\sigma_{t1}\sigma_{z1}\langle a_2, j_2\rangle \dots$ under orchestrator γ. Intuitively, at every step, we take into account the probability, determined by P_t, that the user does action a_{i+1} in the target state $\sigma_{t,i}$, in conjunction with the probability, determined by P_z, that the system service does the transition $\sigma_{z,i} \xrightarrow{(a_{i+1},j)} \sigma'_{z,i+1}$, where j is the choice of the orchestrator at step i under orchestrator γ, i.e. $j = \gamma(\sigma_{t,i}, \sigma_{z,i}, a_{i+1})$.

The value of a joint history under orchestrator γ is the sum of discounted rewards, both from the target and the system services: $v_\gamma(h) = \sum_{i=0}^{|h|} \lambda^i \Big(R_t\big(\sigma_{t,i}, a_{i+1}\big) + R_z\big(\sigma_{z,i}, \langle a_{i+1}, \gamma(\sigma_{t,i}, \sigma_{z,i}, a_{i+1})\rangle\big)\Big)$

Intuitively, we take into account both the reward that comes from the execution of action a_{i+1} in the target service, but also the reward associated to the execution of that action in service j chosen by orchestrator γ. Now we can define the expected value of an orchestrator to be: $v(\gamma) = \mathbb{E}_{h_{t,z}\sim P_\gamma}\big[v_\gamma(h_{t,z}) \cdot realizable(\gamma, \pi_t(h_{t,z}))\big]$ where $realizable(\gamma, \pi_t(h_{t,z}))$ is 1 if $h_t = \pi_t(h_{t,z})$ is realizable in γ (i.e. all the possible target histories are processed correctly), and 0 otherwise. That is, $v(\gamma)$ is the expected value of histories realizable in γ. Finally, we define an optimal orchestrator to be $\gamma = \arg\max_{\gamma'} v(\gamma')$.

It can be shown that, under certain assumptions (i.e. target is realizable, every history has strictly positive value, and the target's rewards are always greater than services' rewards), optimality of the orchestrator implies that the target is realized by the orchestrator.

2.3 Solution technique

The solution technique is based on finding an optimal policy for the *composition MDP*. The composition MDP is a function of the system service and the target service as follows: $\tilde{\mathcal{M}}(\tilde{\mathcal{Z}}, \tilde{T}) = \langle S_{\tilde{\mathcal{M}}}, A_{\tilde{\mathcal{M}}}, T_{\tilde{\mathcal{M}}}, R_{\tilde{\mathcal{M}}}\rangle$, where $S_{\tilde{\mathcal{M}}} = \Sigma_{\tilde{\mathcal{Z}}} \times \Sigma_{\tilde{\mathcal{T}}} \times A \cup \{s_{\mathcal{M}0}\}$, $A_{\tilde{\mathcal{M}}} = \{a_{\mathcal{M}0}, 1, \dots, n\}$, $T_{\tilde{\mathcal{M}}}(s_{\mathcal{M}0}, a_{\mathcal{M}0}, (\sigma_{z0}, \sigma_{t0}, a)) = P_t(\sigma_{t0}, a)$, $T_{\tilde{\mathcal{M}}}((\sigma_z, \sigma_t, a), i, (\sigma'_z, \sigma'_t, a')) = P_t(\sigma'_t, a') \cdot P_z(\sigma'_z \mid \sigma_z, \langle a, i\rangle)$, if $P_z(\sigma'_z \mid \sigma_z, \langle a, i\rangle) > 0$ and $\sigma_t \xrightarrow{a} \sigma'_t$ and 0 otherwise, $R_{\tilde{\mathcal{M}}}((\sigma_z, \sigma_t, a), i) = R_t(\sigma_t, a) + R_z(\sigma_z, \langle a, i\rangle)$, if $(a, i) \in A(\sigma_z)$ and 0 otherwise.

This definition is pretty similar to the construction proposed in [Brafman *et al.*, 2017], with the difference that now, in the transition function, we need to take into account also the probability of transitioning to the system successor state σ'_z from σ_z doing the system action $\langle a, i\rangle$, i.e. $P_z(\sigma'_z \mid \sigma_z, \langle a, i\rangle)$. Moreover, in the reward function, we need to take into account also the reward observed from doing system action $\langle a, i\rangle$ in σ_z, and sum it to the reward signal coming from the target. By construction, if ρ is an optimal policy, then the orchestrator γ such that $\gamma(\sigma_z, \sigma_t, a) = \rho(\langle \sigma_z, \sigma_t, a\rangle$ is an optimal orchestrator.

To summarize, given the specifications of the set of stochastic services and the target service, first compute the composition MDP, then find an optimal policy for it, and then deploy the policy in an orchestration setting and dispatch the request to the chosen service according to the computed policy,

2.4 Use case

The scenario proposed is the following: there is an industrial process of ceramics production in which a product must be processed sequentially in different ways. Each sub-task can be completed by a set of *available services*. The tasks to be carried out in order to complete the industrial process are: provisioning, moulding, drying, first baking, enamelling, painting, second baking and shipping. Such tasks can be accomplished by different types of machines or human workers. Each available service that can perform the task can be seen as finite state machines with a probability and a reward associated to each action. There could be multiple services for the same task, e.g. multiple version of a machine (new one and old one) and a human that can perform the task required, and so on. When an available service is being assigned a task, this has a *task cost* in terms of time taken and resources needed for the completion of the operation on that specific service. Usually, in terms of task cost, machines are cheaper than human workers, because they can perform their task much faster. However, the machines have a certain probability to *break* when they perform their job. In such a case, the machine must be repaired as soon as the operation has been carried out, that incurs in a *repair cost* for that specific machine.

From the above description of the use case scenario, it is clear that the composition technique must be able to handle the stochasticity of the available services' transitions, as well as their reward/cost. Indeed, an optimal orchestration depends on several parameters, like the task costs, the breaking probabilities and the repair costs, one for each candidate service for accomplish a certain task. Therefore, it is not straightforward to determine a priori which service a certain task must be assigned to. For example, it might be the case that despite the task cost of a machine is low, its breaking probability might be high, and considering the repair cost it might let us to prefer a human worker for that task. We argue that our model can fit very well our use case. Indeed, we can reduce the problem to an instance of stochastic service composition suggested above in which a service can capture the task cost, the breaking probability, and the repair cost.

3 Conclusion

In this work we have proposed a stochastic service composition, in which also the services are allowed to have stochastic behaviour and rewards on the state transitions. We formally specified the problem and proposed a solution based on a reduction to MDPs, showing how it is well-suited for a realistic Industry 4.0 scenario.

Besides the development of this work we considered several future research directions, aiming to enrich the theoretical framework with interesting features such as: exception handling, modularity of the target specification, high-level specification of the target service based on temporal logic formalisms, possibility to specify safety constraints. Moreover, other interesting directions to explore are the integration with learning techniques in order to achieve greater scalability and investigation on how to achieve the resilience of the system. The expected theoretical contributions from this project are potentially impacting many other research areas. The reason is that the service composition models and techniques have different application areas, e.g. Web Service composition, and the theoretical advances are virtually applicable without much effort in other contexts that the service composition fits in. We aim to unlock the full potential of Digital Twins, and letting businesses take advantage of the new trends in AI and enabling technologies to achieve more significant productivity goals, improving the efficiency and the sustainability of the production processes.

Riferimenti bibliografici

[Berardi *et al.*, 2003] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, e Massimo Mecella. Automatic composition of e-services that export their behavior. In *International conference on service-oriented computing*, pages 43–58. Springer, 2003.

[Berardi *et al.*, 2005] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, e Massimo Mecella. Automatic service composition based on behavioral descriptions. *International Journal of Cooperative Information Systems*, 14(04):333–376, 2005.

[Brafman *et al.*, 2017] Ronen I Brafman, Giuseppe De Giacomo, Massimo Mecella, e Sebastian Sardina. Service composition in stochastic settings. In *Conference of the Italian Association for Artificial Intelligence*, pages 159–171. Springer, 2017.

[Catarci *et al.*, 2019] Tiziana Catarci, Donatella Firmani, Francesco Leotta, Federica Mandreoli, Massimo Mecella, e Francesco Sapio. A conceptual architecture and model for smart manufacturing relying on service-based digital twins. In *2019 IEEE international conference on web services (ICWS)*, pages 229–236. IEEE, 2019.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Massimo Mecella, e Fabio Patrizi. Automated service composition based on behaviors: The roman model. In *Web services foundations*. 2014.

[Hull, 2008] Richard Hull. Artifact-centric business process models: Brief survey of research results and challenges. In *OTM Confederated International Conferences*, pages 1152–1163. Springer, 2008.

[Medjahed e Bouguettaya, 2011] Brahim Medjahed e Athman Bouguettaya. *Service composition for the Semantic Web*. 2011.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes*. 1994.

[Sutton e Barto, 2018] Richard S Sutton e Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[Yadav e Sardina, 2011] Nitin Yadav e Sebastian Sardina. Decision theoretic behavior composition. In *AAMAS*, 2011.