Learning-aided Adaptation - A Case Study from Wellness Ecosystem

Suman Roychoudhury, Mayur Selukar, Deepali Kholkar, Suraj, Namrata Choudhary, Vinay Kulkarni, and Sreedhar Reddy

Tata Consultancy Services Research, Pune, India
{suman.roychoudhury,mayur.selukar1,deepali.kholkar, suraj.39, namrata.choudhary, vinay.vkulkarni, sreedhar.reddy}@tcs.com

Abstract. Modern businesses are being subjected to an unprecedented variety of change drivers that cannot be predicted such as new regulations, emerging business models, and changing needs of stakeholders. This creates new demands on enterprises to meet stated goals in a dynamic and uncertain environment that translate to demands on the enterprise's software systems. In our earlier work, we proposed an architecture for dynamic adaptation under uncertainty. In this paper, we provide validation of our adaptation architecture through a case study of a wellness ecosystem comprising multiple stakeholders with evolving goals and behavior that need to be continuously learnt and adapted. We present experimentation results from our case study that serve as an initial step towards validating our approach.

Keywords: software adaptation, digital twin, reinforcement learning

1 Introduction

Enterprises today operate in a dynamic, uncertain environment where they are continually subjected to unpredictable changes to business, consumer needs, laws and regulations, and even technology. Enterprises need to be able to adapt to these changes and continue to meet their goals despite this dynamism and uncertainty. In current practice analysis of each change in the business environment and making decisions on the enterprise's response to change is done by human experts. On the other hand, modern businesses are driven by software systems that are now ubiquitous and serve as critical growth driver for enterprises. As businesses negotiate the uncertainty in their environment, systems must assist enterprises (i.e., human experts) in adapting to change and meeting their goals. The adaptation process needs to be dynamic and continual, i.e. systems must be capable of detecting, analyzing, and responding to changes in the environment at run-time.

To achieve dynamic adaptation, enterprise systems need to continually acquire knowledge about their environment, evaluate goals, suggest appropriate courses of action towards meeting the goals, and learn from previous experience.

However enterprise systems often have only partial visibility into their environment, also, available information is incomplete and uncertain. For example all usage scenarios of the system cannot be known beforehand. Behaviours and preferences of individual system users can span across a wide spectrum, which are unknown to the system when it begins operation. Moreover behaviour patterns and needs of system actors evolve with time, with life changes and unpredictable events.

In order to understand the adaptation triggers for modern business systems, enterprises need to be viewed in the context of the larger business ecosystem of which they are a part. Business ecosystems comprise multiple interacting stakeholders – business enterprises, suppliers, service, sales, and other partner organizations, consumers, regulators, and also products and processes. Participants in a business ecosystem come together to service the same set of consumers [15]. Working collaboratively enables them to deliver greater value to the customer than they could have individually while increasing their own market access and brand value. Collaboration between stakeholders needs to be managed by a central integrator [16] that orchestrates interactions in the ecosystem. Interactions among stakeholders leads to increased dynamism in the environment of individual systems necessitating their adaptation.

We look at the adaptation problem for businesses in the context of an ecosystem where the central integrator onboards consumers as well as service providers, acquiring the right information and orchestrating services from both so that goals of each stakeholder are met despite dynamism in the environment. The integrator's knowledge at the ecosystem level helps in making orchestration decisions and recommending courses of action to individual stakeholders.

In our earlier work [7] we proposed a learning-aided adaptive architecture for enterprise systems. In this paper, we provide validation of the adaptation learning component of the architecture through a real-life case study from the wellness domain. The key contribution of the paper is demonstrating the utility of adaptation learning to aid stakeholder decision making in the context of business ecosystems. Our approach uses digital-twin based simulation techniques to train a reinforcement learning (RL) agent that has no prior knowledge about stakeholder behaviour. The RL agent acts as the primary learning component and via interactions with various system stakeholders, learns their behavior over time and adapts the system accordingly.

The paper is organised as follows - in Section 2, we recap our learning-aided adaptive architecture. In Section 3, we detail our wellness ecosystem use case and provide experimental results to validate adaptation learning in enterprise systems. Section 4 summarises our work and outlines future directions.

2 Learning-aided Adaptive Architecture

In this section, we review our learning-aided adaptive architecture [7] that has four key components as shown in Figure 1.

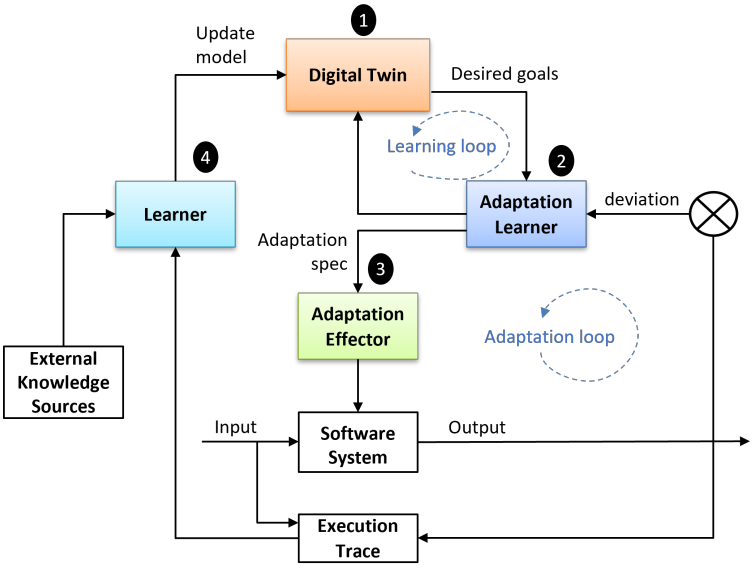

Fig. 1. Learning-aided Adaptive Architecture

1. A digital twin based decision-making system to model the enterprise system, its environment and goals
2. An in-built reinforcement learning-based adaptation learner system to control or adapt the enterprise model (i.e. the digital twin)
3. A sensor-actuator based adaptation effectation architecture to keep the enterprise model and system in sync
4. A continuous learning module to observe system interaction traces and update the digital twin based on new or updated knowledge

In this section, we review the first two components of the architecture, namely the enterprise model of the system (i.e its digital twin) and the adaptation learning component that are key to realizing the adaptive architecture.

2.1 Digital Twin - System, Environment and Goal Model

At the heart of our adaptation architecture is a System, Environment and Goal (SEG) model manifested as a Digital Twin that is a hi-fidelity machine-processable representation of the software that supports what-if scenario playing [8]. The digital twin not only represents our current understanding about the problem space, business domain and solution but is also continuously updated via behavioral learning and refinement throughout the software life cycle, as well as learning from external knowledge sources.

The SEG metamodel is depicted in Figure 2. The system interacts with the environment via actions and has specific goals that are predicates over its value

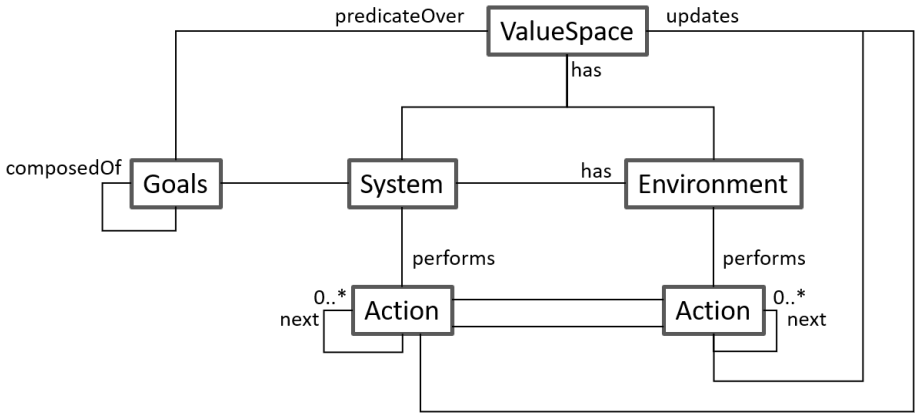

Fig. 2. Digital Twin - System, Environment and Goal Metamodel

space. Actions update the value space and can be composed of other primitive actions. An adaptation in the system can be triggered by either a change of actions in the environment or change in goals of the system.

2.2 Adaptation Learner

The Adaptation learner component in Figure 1 is responsible for

- (i) checking if the system is meeting stated goals when there are changes in the environment,
- (ii) identifying a strategy comprising a sequence of interventions to be introduced so as to meet stated goals, and
- (iii) getting assurance of efficacy of the strategy.

We propose a simulation-based (digital twin) reinforcement learning approach to achieve the three objectives, where the Digital Twin serves as an "experience generator" to train the reinforcement learning (RL) agent. The stated goals help define the reward function of RL agent and the candidate set of primitive actions constitute the action space. The RL agent comes up with a "good enough" policy i.e., a sequence of interventions to be introduced in order to achieve the stated goals in the light of partial knowledge, uncertainty and dynamic changes in the environment of business systems. The next section introduces the case study example from the wellness domain that is used to validate our adaptation learning approach.

3 Case Study - A Wellness Ecosystem

We illustrate adaptation learning using an example of a wellness ecosystem. Wellness needs of individual consumers encompass fitness, healthcare, nutrition,

leisure, as well as health insurance. The wellness ecosystem comprises service providers for each of these aspects with the common objective of fulfilling wellness goals of a consumer population of individuals as depicted in Figure 3. A central integrator in the form of a wellness provider facilitates and manages services offered by various service providers and acts as a centralized hub so that the consumer no longer needs to separately manage each of his wellness needs. The wellness provider creates a digital twin modeling individual service provider services and customer goals. Incomplete, uncertain, and partially visible information about the environment such as customer preferences and behaviour needs to be learnt in order to be able to meet customer goals. The goal of the wellness provider is to primarily satisfy needs of consumers while meeting goals of each of the stakeholders in the ecosystem. Although the wellness ecosystem of Fig-

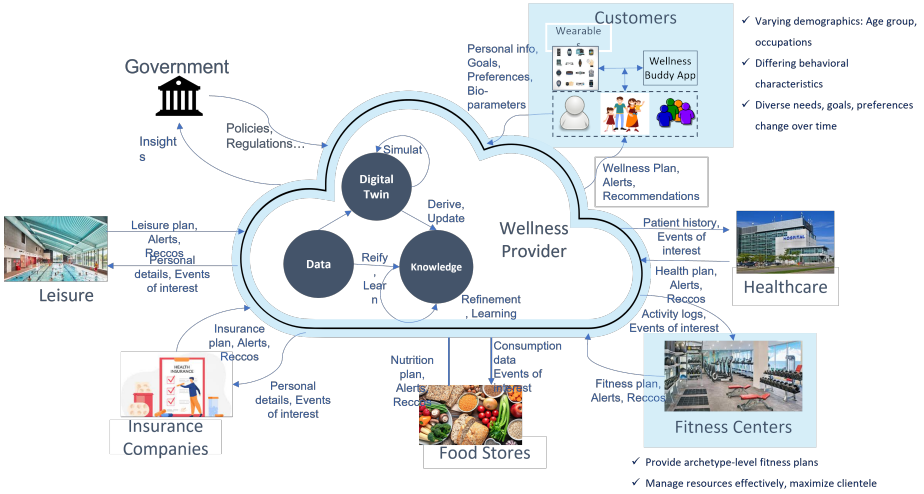

Fig. 3. Wellness Ecosystem

ure 3 consists of various stakeholders like "food stores", "insurance companies", "fitness centers" etc, for our initial experimentation, we only consider the fitness component of the wellness ecosystem, as highlighted in Figure 3. This is in tune with current trends with health management systems and smart health devices like mHealth apps that are becoming increasing popular with fitness aspirants [10], [11]. For our fitness case study, the stakeholders of interest are individual customers (i.e., fitness aspirants) and fitness centers or gyms that provide fitness facilities to the customers.

Once fitness aspirants register with a fitness center, the individuals are classified into archetypes based on age, gender and occupation. Fitness aspirants within the same archetype are initially given the same fitness plan as depicted in Figure 7. In this case study, we consider two such archetypes namely young professionals (Arch1) and senior management (Arch2). The Wellness provider

acts as a facilitator between the fitness aspirant and fitness centers and aims to personalize the generic archetype-level fitness plan for every individual by monitoring their behavior in the fitness center.

For our experimental setup, in the absence of real individual data, we simulate the behavior of individuals in a fitness center by constructing a digital twin of the gym environment as depicted in Figure 4. To replicate the behavioral

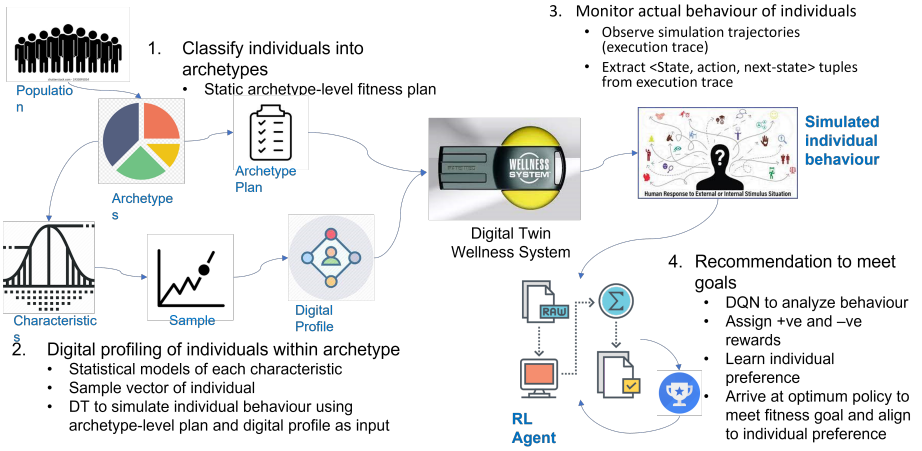

Fig. 4. Experimental Setup

characteristics and preferences of individuals in a fitness center we construct a 'digital profile' of fitness aspirants. We assume the 'fitness goal' of each fitness aspirant is dependent on certain variables (e.g., their age, weight, occupation, health condition, food habit, activity level) where each variable is governed by some probability distribution over the entire population. In addition, some variables have causal relationships with other variables (e.g., age, work schedule may determine activity level), hence we model the digital profile of an individual as a probabilistic model as shown in Figure 5, where each child node conditionally depends on parent nodes. Variables 'goal', 'workout type', and 'workout preferences' are deterministic in nature while all other variables are either discrete or continuous random variables.

Sample digital profiles (only limited variables shown) of two individuals are illustrated in Figure 6. To generate the given sample, we first sample 'age' from an exponential distribution and then categorize them as young (if age less than 30), mid-level (if age between 30 and 45), and senior (if age greater than 45). Similarly, discrete variable 'occupation' is sampled from a multinomial distribution with values ranging from 'professional', 'self-employed' or 'government'. Since both the above variables (i.e., 'age' and 'occupation') can influence the 'activity level' of an individual, hence 'activity level' is sampled from a con-

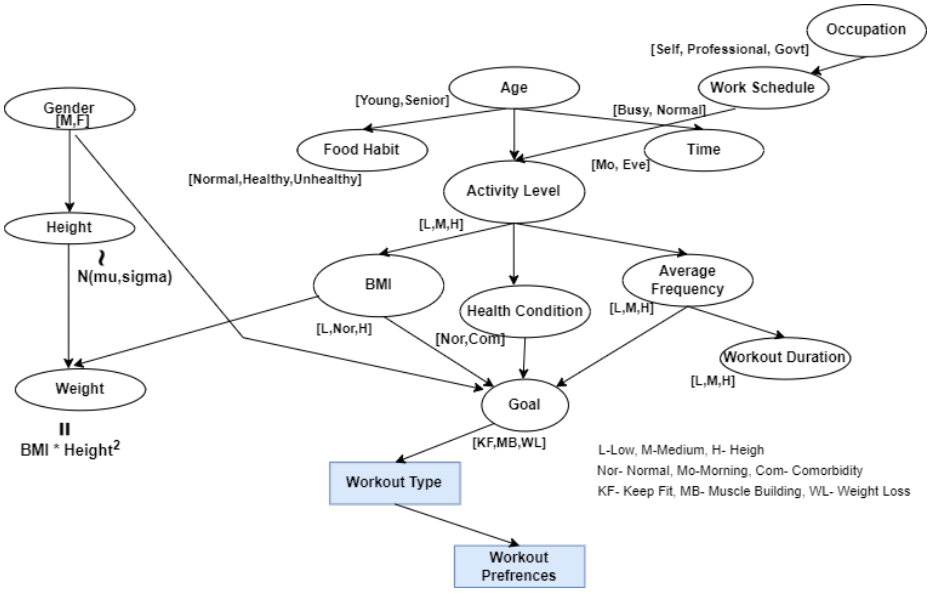

Fig. 5. Digital Profile

ditional multinomial distribution with condition on parent variables 'age' and 'occupation'.

Name	Age	Height (mt)	Weight	Activity Level	Occupation	Frequency per week	Health Condition	Fitness Goal	Workout Plan
Ind_1	29	1.65	63	High	Young Professional	5	Normal	Keep Fit	Default_ Plan1
Ind_2	54	1.7	75	Low	Senior Management	3	Comorbid	Keep Fit	Default_ Plan2

Fig. 6. Statistical Sampling of an Individual from Digital Profile

In this manner, once an individual is sampled from a given population, every individual belonging to an archetype is assigned a generic fitness plan as shown in Figure 7. The default fitness plan for senior management varies from that of young professional due to the difference in exercise intensity and calorie burn requirements. The digital twin encodes variations in fitness behaviour of an individual based upon characteristics recorded in the profile. We use reinforcement learning (RL) to learn about individual preferences by monitoring the actual execution trace of an individual in the simulated gym environment against a given workout. In this way the generic plan is continuously adapted to resemble not only individual preferences but also ensuring fitness goal (in terms of calories burnt) of an individual is met.

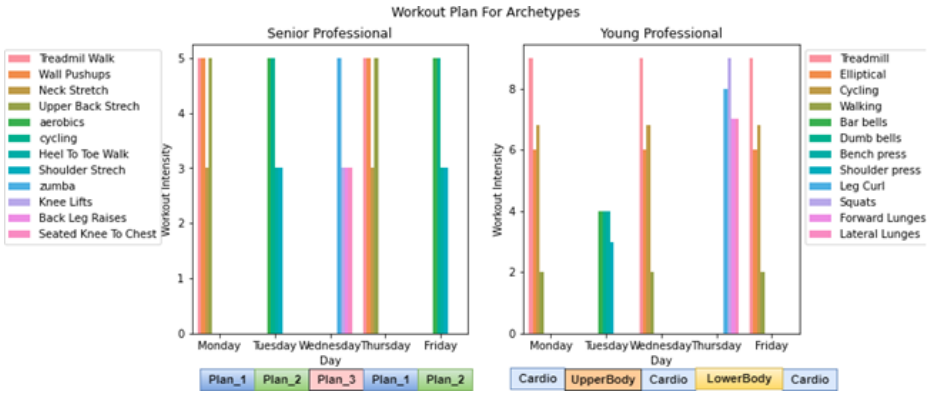

Fig. 7. Generic fitness plan

We summarize the execution steps of our learning-aided adaptation approach as follows:

1. An individual (fitness aspirant) registers with the wellness provider.
2. The individual is allocated a generic fitness plan by the wellness platform based on their archetype (i.e, young professional or senior management)
3. The digital twin of the fitness center simulates the individual performing exercises prescribed in the plan in accordance with the characteristics of their digital profile, providing feedback in terms of:
   - a. The distance between their exercise preferences and the allocated plan.
   - b. The distance between the level of fitness (calorific burn) and the desired fitness achieved by the plan.
4. Based on the feedback, the RL agent learns the behavior of the individual and then nudges the individual to follow the plan balancing their satisfaction level and also achieving the desired fitness goal (i.e., calorie burn)
5. Eventually the 'best' fitness plan for every individual in a given archetype is adapted from the generic fitness plan and personalized accordingly.

In the next subsection, we provide further technical details of the reinforcement learning solution.

3.1 RL Problem Conceptualization

The goal of the RL agent is to learn preferences of an individual and tailor its recommendations so that they not only help achieve the target calories burnt but are also preferred by the individual. In order to do this the system starts by recommending a generic fitness plan as illustrated in Figure 7 to a new client and observes their behavior in a simulated environment [8]. This allows us to generate a state that can encapsulate individual preferences, which the RL agent can then use to adapt and create a personalized fitness plan. The generic plan also lends us the basic structure in which the recommendations are to be made.

For example, if the generic plan consists of a 3:1:1 split between *cardio*, *upper body* and *lower body* workouts as in Figure 7 every week, then the consequent recommendation will also follow the same split. For a new individual there is no prior information available, therefore, in order to gather relevant information every individual goes through an exploratory phase at the beginning. During this exploratory phase a generic plan is recommended, this plan equally recommends each possible exercise. Based on the individual's behavior over this period a state is developed which in turn is used to recommend personalized plans. Note that this recommendation problem can be viewed as an Markov-Decision Process (MDP) [18] when

1. The RL agent is called for a recommendation after every exercise.
2. The state encapsulates the preferences of the individual through the activity trace and includes all the exercises performed in the current episode.

This formulation essentially encodes all the information of the individual in the state and thus follows the MDP dynamics.

3.2 Solution Methodology

We now proceed to define the reinforcement learning solution for the problem described. To deal with the diversity amongst the individuals, we map the individual to an appropriate archetype. We train a separate RL agent for each archetype as the goal changes amongst archetypes and even for same goal since the recommendation structure changes based on archetype, the *reward ranges* tend to vary making use of the same agent infeasible. Our approach breaks the larger problem into sub-problems, where the same algorithm can now be applied to each person in a diverse population based on their individual goals and attributes. This is a realistic situation in the wellness domain as people come in with varying expectations and timelines. This parallelization also keeps the agent simpler and speeds up the learning process.

States and Actions Based on the archetype we get a certain set of exercises that could be performed to achieve the goal. These form the action space for the archetype and based on the action space we get the corresponding state space.

Let's say a person can perform 3 exercises, A_1, A_2, A_3 respectively. Now our state must have enough information for the agent to estimate an individual preference for A_i. To capture this, we create a queue D_i of finite size (here 3) for A_i and initialize it with A_i. After this, whenever A_i is recommended the corresponding action performed by the individual is appended to D_i, and a concatenation of these queues forms the first part of the state. The second part is made of individual specific constants like body mass index (BMI) along with the exercises performed in the current episode. Therefore, for the example of 3 exercises the state size is $3 * 3 + 3 + 1 = 13 units$ We work with a much richer action space of 21 exercises so our state space is $21 * 3 + 21 + 1 = 85 units$.

Rewards The rewards are designed per archetype and in most cases a goal can be expressed using the metabolic equivalent (MET) values of the exercises. For example, for young professionals whose goal is to keep fit, the objective is to maximize calories burnt while sticking to a 3:1:1 split for a given week. To allow the agent to model individual preferences, we associate a penalty when the suggestion is not performed, this allows the agent to achieve the goal of recommending the exercise that the person likes without being biased towards recommending exercises that would burn maximum calories. However, this may lead to a situation where one exercise will be recommended always, so we add a diminishing factor to the preferences so that the reward for recommending the same exercise again is lower, this is in line with how the preferences of individuals are modeled. The reward function is modeled as below-

$$R = \alpha Penalty + \bar{\alpha}\beta^{n} Preference(e) * Met(e)$$

where $\alpha = 0$ if the recommendation was performed; β = preference diminishing factor; n = number of times the exercise is performed during current day; e = exercise recommended and performed.

Constraints and Action Mask The situation described so far ignores the various constraints that the wellness provider faces from its stakeholders. The wellness provider must deal with a range of constraints, from the limited gym equipment that remain constant across archetypes, to the personalized health related constraints that are unique to every individual. To deal with constraints, all action masks are collected from the stakeholders involved and we use a simple bitwise *and* operation to generate the final action mask [5]. The action mask is then used to mask out the invalid action by setting its predicted Q-Value to -∞. This relieves our RL agent from the burden of constraint handling as they are now handled externally. It also keeps the system modular, and any addition or deletion of constraints and stakeholders is streamlined.

Neural Network Architecture and Training We use Deep Q-Learning (DQN) [13] for the computation of individual recommendations given the current state of the individual. The policy network of Figure 8 is fully connected with layers of (85,512,512,512,21) neurons respectively with the first number indicating the input size and the last indicating the number of exercises that can be recommended. All the hidden neurons have *tanh* activation, and the output layer has linear activation.

The agent follows a Monte Carlo learning approach [18], which implies that we only have one policy network. The specifics of the learning approach followed are as described in [4]. The policy takes the 85 feature vectors described above and recommends appropriate action. The network is trained using the Pytorch library on Python 3.7 using Adam optimizer with a learning rate of 0.0005 and a batch size of 64, mean squared loss was used with a discount factor of 0.99. Finally, the training calls are only made after the memory buffer contains 1000

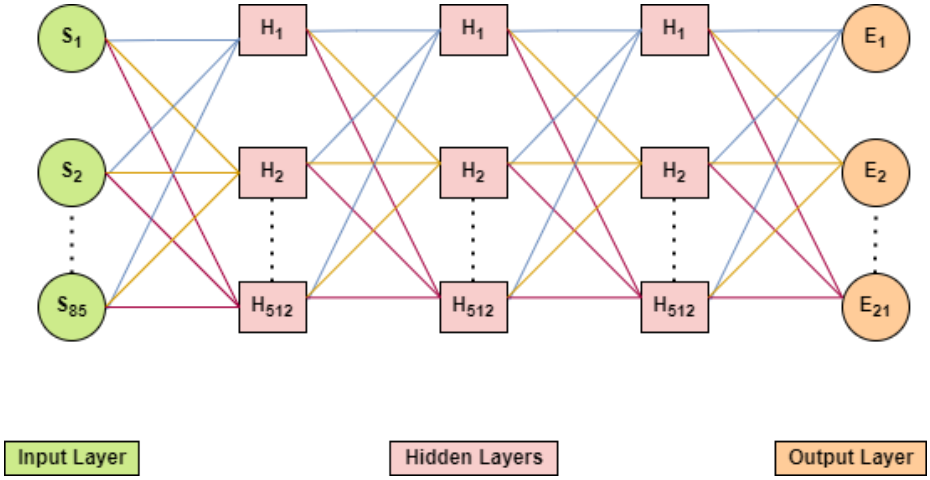

Fig. 8. Neural network architecture

training episodes. We use Boltzmann action selection [18] for exploration which implies that our agent is stochastic in nature.

Since the actual problem structure is that of an infinite horizon MDP [18], we terminate the episode at the end of every day. This is done because most of our constraints are either constant, daily or weekly in nature and keeping the length of episode as *daily* makes constraint management simpler. Each training episode begins after an initial exploratory period, which guarantees that the state has enough information regarding the person's preferences. The environment used simulates more than 200 unique individuals to avoid overfitting and to learn the mapping between the state and appropriate actions.

Training Results We train the agents for 20000 episodes and the episodes start at the beginning of a week. Based on the individual's archetype the recommendations are made following the structure of the generic plan. The episode length is set to a day and Figure 9 depicts the loss graph for the two archetypes under consideration.

3.3 Comparison with Baseline Algorithm

For comparison of our algorithm with a baseline, we propose a greedy upper bound solution as the baseline that can be obtained if we assume we have complete knowledge about individual's preferences. The digital twin provides us with the preferences of an individual at any point of time during the episode. Combining this with the MET values of the exercise, we can run an exhaustive search over an objective function and then take the action that maximizes the return. We define the objective function over reward as follows

$$O = P(e) * \beta^n Preference(e) * Met(e)$$

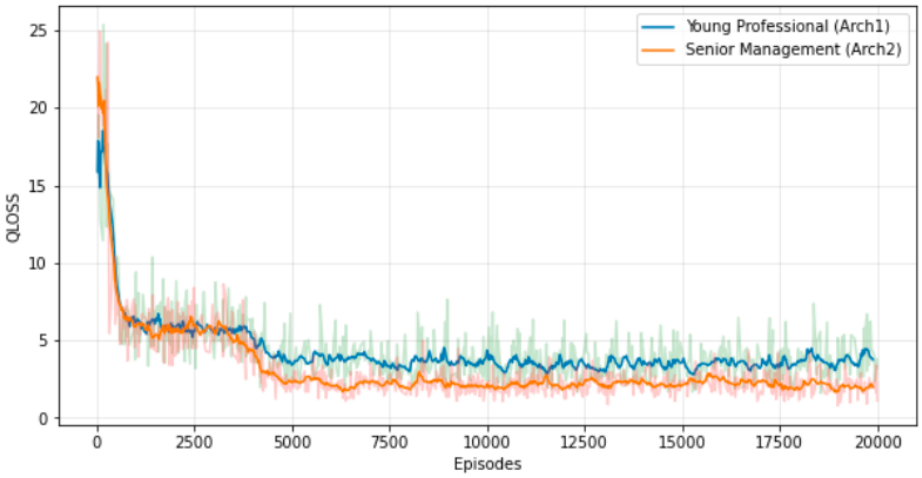

Fig. 9. Loss graph during RL training

where P(e) is the probability that exercise e will be performed after it is recommended. This is obtained from the execution trace of the digital twin and β is the diminishing factor defined earlier and preference and MET are the preference of the individual for the exercise and MET value of the exercise. Figure

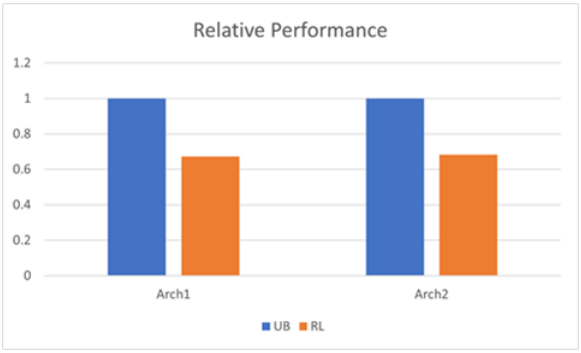

Fig. 10. Relative performance with baseline upper bound solution

10 represents the relative performance of the upper bound and the RL agent for the respective archetypes. We measure the performance in terms of the average return across 100 episodes. We see that the RL agent is able to achieve 70 percent of the returns generated by the upper bound solution by just observing the individual. We present a detailed exploration of the actions taken by both approaches and their effects in the subsequent section.

Comparison with Sample Profile Figure 11 depicts the recommendations made and their effect on the exercises performed by the same individual. The left column depicts the recommendations made by the greedy Upper bound algorithm and the right column depicts the recommendations from RL over a 120-day period excluding the initial exploratory period of 30 days.

Fig. 11. Results - nudging individual to meet fitness goal by learning their preference

The blue line depicts the probability of an exercise being performed without any external influence of recommendations and constraints whereas the green line depicts the probability with which the exercise was performed over the 120-day period. The underlying bar graph denotes the number of times the exercise was recommended. Here the prior is not visible to the RL agent and must be inferred from the person's behavior. We also note that the person's choices of exercise to perform are stochastic in nature and the objective function and reward are encoded in such a manner that when an optimal action is recommended and not performed, the same action must be recommended again. Our RL solution is able to map a similar policy where in each case of *Cardio*, *Upper Body* and

Lower Body, the most prominent recommendation is the same as that of the upper bound. The second and third recommendation do not match as strongly as the difference in upside is minimal. This can be explained as the RL approach is stochastic in nature as compared to the greedy nature of the upper bound solution.

The effect of the recommendations in nudging the user towards their goal is visible in both cases where the probability of the most prominent recommendations is drastically increased. This increase in the probability depicts the nudging behavior of the agent. Since all the probabilities must sum up to 1, the increase is only visible for the most prominent recommendation and in some cases a decrease in the probability of the second and third prominent recommendation is also seen. This is a probable outcome and in line with the expectations. Hence the results presented in this section demonstrate the effect of nudging an individual towards meeting their goal of calorie burn in addition to balancing their preferences for certain exercise routines.

4 Discussion and Related Work

Software adaptation has been an important area of research for some time [2]. Several adaptation architectures have been proposed in the literature [6], [17], [9] that principally comprise a sensing or monitoring step, followed by analysis and implementation of adaptation. Most approaches use statically built models to guide adaptation, while our approach uses a model built dynamically using deep reinforcement learning.

The adaptation learning scenario presented in this paper is similar to mHealth apps and mobile health management systems [11]. In these systems fitness aspirants log data using a multitude of models from fitness trackers, smart wearable devices to manual entry. These systems then use the data to encourage physical activity and reduce the risk of diseases related to physical inactivity. The intervention mechanism ranges from context aware motivational messages throughout the day [10], [12] to recommendation systems, delivering optimal set of activities [3].

In this paper, we have used RL agent as our adaptation-learning mechanism to constantly monitor user preferences and adjust the recommendations based on their dynamic behavior throughout the intervention period. In comparison, previous work in this space [10], [3] uses RL to encourage physical activity of an individual, where they use a generic model of an individual instead of learning the model from individuals' dynamic behavior.

The fitness-fatigue model proposed by [1] has been used to predict athletes' performance in a variety of endurance sports. According to the fitness-fatigue model, each practice session has a positive impact on performance by contributing to a "stock of fitness" and a negative impact by contributing to a "stock of fatigue." Fatigue can be measured in terms of calories burned which is proportional to "metabolic equivalents" (MET) of the exercise that we use in our experiment. Therefore, getting maximum fatigue or calorie burn during the ex-

ercise can bring maximum fitness for the individual provided individual does not fatigue himself beyond a limit, which we achieve by introducing constraints on calorie burn.

5 Conclusion

In this paper, we demonstrated how adaptation learning using RL can be achieved in a business ecosystem context through the example of a wellness ecosystem comprising multiple stakeholders without prior knowledge of stakeholder behavior. The wellness provider acts as central integrator and learns stakeholder behavior by suitable interactions via a digital twin such that stakeholder goals can be achieved without compromising stakeholder preference. Therefore, balancing stakeholder satisfaction along with meeting stakeholder goal is the key contribution of this paper, thus illustrating learning-aided dynamic adaptation. Use of a digital twin to model and simulate user behaviour helped us to circumvent the problem of unavailability of real users' data that classical learning approaches need for providing recommendations. Although in this paper, we considered goals of stakeholders only from the fitness perspective, in future, we intend to include other wellness perspectives like nutrition, healthcare, recreation, insurance etc that may require resolution of multiple stakeholder goal conflicts using Pareto optimal techniques [14].

Furthermore, the scope of this paper is limited to validating the Adaptation Learner component of Figure 1 using a business ecosystem, whereas we would like to extend the validation to other parts of the architecture [7], where adaptation policies learnt on simulated system behaviour are appropriately implemented in underlying software systems and execution traces of real users' behavior are used to update new knowledge resulting in further improvement of the digital twin model.

References

1. Banister, E.W., Calvert, T.W., Savage, M.V., Bach, T.: A systems model of training for athletic performance. Aust J Sports Med **7**(3), 57–61 (1975)
2. Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Serugendo, G.D.M., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H.M., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H.A., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., Whittle, J.: Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems [outcome of a Dagstuhl Seminar]. Lecture Notes in Computer Science, vol. 5525, pp. 1–26. Springer (2009). https://doi.org/10.1007/978-3-642-02161-91
3. Fang, J., Lee, V., Wang, H.: Dynamic physical activity recommendation on personalised mobile health information service: A deep reinforcement learning approach. arXiv preprint arXiv:2204.00961 (2022)

4. Hausknecht, M., Stone, P.: On-policy vs. off-policy updates for deep reinforcement learning. In: Deep Reinforcement Learning: Frontiers and Challenges, IJCAI 2016 Workshop. AAAI Press New York, NY, USA (2016)
5. Huang, S., Ontañón, S.: A closer look at invalid action masking in policy gradient algorithms. arXiv preprint arXiv:2006.14171 (2020)
6. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003). https://doi.org/10.1109/MC.2003.1160055, https://doi.org/10.1109/MC.2003.1160055
7. Kholkar, D., Roychoudhury, S., Kulkarni, V., Reddy, S.: Learning to adapt – software engineering for uncertainty. In: 15th Innovations in Software Engineering Conference. ISEC 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3511430.3511449, https://doi.org/10.1145/3511430.3511449
8. Kulkarni, V., Barat, S., Clark, T.: Towards adaptive enterprises using digital twins. In: 2019 Winter Simulation Conference (WSC). pp. 60–74 (2019). https://doi.org/10.1109/WSC40007.2019.9004956
9. de Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Göschka, K.M., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D.B., Sousa, J.P., Tahvildari, L., Wong, K., Wuttke, J.: Software Engineering for Self-Adaptive Systems: A Second Research Roadmap, pp. 1–32. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5-1
10. Liao, P., Greenewald, K., Klasnja, P., Murphy, S.: Personalized heartsteps: A reinforcement learning algorithm for optimizing physical activity. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **4**(1), 1–22 (2020)
11. Liu, Y., Avello, M.: Status of the research in fitness apps: A bibliometric analysis. Telematics and Informatics **57**, 101506 (2021)
12. Marcolino, M.S., Oliveira, J.A.Q., D'Agostino, M., Ribeiro, A.L., Alkmim, M.B.M., Novillo-Ortiz, D.: The impact of mhealth interventions: systematic review of systematic reviews. JMIR mHealth and uHealth **6**(1), e8873 (2018)
13. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
14. Moffaert, K.V., Nowé, A.: Multi-objective reinforcement learning using sets of pareto dominating policies. Journal of Machine Learning Research **15**(107), 3663–3692 (2014), http://jmlr.org/papers/v15/vanmoffaert14a.html
15. Moore, J.: Predators and prey: A new ecology of competition. Harvard business review **71**, 75–86 (05 1999)
16. Sarafin, G.: What business ecosystem means and why it matters (2021)
17. Shevtsov, S., Berekmeri, M., Weyns, D., Maggio, M.: Control-theoretical software adaptation: A systematic literature review. IEEE Trans. Software Eng. **44**(8), 784–810 (2018). https://doi.org/10.1109/TSE.2017.2704579
18. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA (1998), https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf