Facebook's Cyber–Cyber and Cyber–Physical Digital Twins

John Ahlgren, Kinga Bojarczuk, Sophia Drossopoulou, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Maria Lomeli, Simon M Lucas, Erik Meijer, Steve Omohundro, Rubmary Rojas, Silvia Sapora, Jie M. Zhang, Norm Zhou*
FACEBOOK Inc.
USA

ABSTRACT

A cyber–*cyber* digital twin is a simulation of a software system. By contrast, a cyber–*physical* digital twin is a simulation of a non-software (physical) system. Although cyber–physical digital twins have received a lot of recent attention, their cyber–cyber counterparts have been comparatively overlooked. In this paper we show how the unique properties of cyber–cyber digital twins open up exciting opportunities for research and development. Like all digital twins, the cyber–cyber digital twin is both informed by and informs the behaviour of the twin it simulates. It is therefore a software system that simulates another software system, making it conceptually truly a *twin*, blurring the distinction between the simulated and the simulator. Cyber–cyber digital twins can be twins of other cyber–cyber digital twins, leading to a hierarchy of twins. As we shall see, these apparently philosophical observations have practical ramifications for the design, implementation and deployment of digital twins at Facebook.

CCS CONCEPTS

• **Computing methodologies → Intelligent agents**.

KEYWORDS

Web Enabled Simulation, Digital Twins

ACM Reference Format:
John Ahlgren, Kinga Bojarczuk, Sophia Drossopoulou, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Maria Lomeli, Simon M Lucas, Erik Meijer, Steve Omohundro, Rubmary Rojas, Silvia Sapora, Jie M. Zhang, Norm Zhou. 2021. Facebook's Cyber–Cyber and Cyber–Physical Digital Twins. In *Evaluation and Assessment in Software Engineering (EASE 2021), June 21–23, 2021, Trondheim, Norway*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3463274.3463275

*Author order is alphabetical. Correspondence to Mark Harman (markharman@fb.com).
Mark Harman's scientific work is part supported by European Research Council (ERC), Advanced Fellowship grant number 741278; Evolutionary Program Improvement (EPIC) which is run out of University College London, where he is part time professor. He is a full time Research Scientist at Facebook. Sophia Drossopoulou is a full time Engineering manager at Facebook, and also a part time professor at Imperial College, London. Simon Lucas is currently a full time Research Scientist at Facebook and also a part time professor at Queen Mary University of London

EASE 2021, June 21–23, 2021, Trondheim, Norway

ACM ISBN 978-1-4503-9053-8/21/06.
https://doi.org/10.1145/3463274.3463275

1 INTRODUCTION

Simulation is increasingly becoming one of the most important applications of software engineering, yet it has received comparatively little attention from the software engineering research community. The importance of simulation is hard to overstate: many, if not all, of the most important challenges we face as a species are being tackled by simulation-based solutions. For example, simulation is increasingly used for prediction and decision-making in economics [52], climate change and weather prediction [30], traffic safety [4], and, most recently, with the farthest possible impact, our response to the COVID19 pandemic [1].

In this paper we outline our on-going development of WW, a simulation of Facebook's WWW platform. The WW system can be thought of as a cyber–cyber digital twin of Facebook's WWW platform and infrastructure. That is, it is a simulation of the WWW platform that is both informed by and informs the behaviour of that platform. There has been much recent development of digital twins, with which the simulation process is extended so that the simulation and the simulated system automatically interact with one another. However, that research agenda is confined to cyber–physical digital twins; software systems that simulate real physical engineering systems and processes. Despite the fact that the digital twin is a software artefact, the potential for cyber–cyber digital twins, in which the simulated system is *also* a software system or process, remains relatively under-exploited and under-explored.

In this paper we argue that there is considerable potential for software engineering research to extend from cyber–physical digital twins to cyber–cyber digital twins. This is not merely 'yet another application of digital twins'; the fact that the two twins are constructed out of the same software engineering material makes cyber–cyber digital twins a *fundamentally different* research paradigm compared to cyber–physical digital twins. In particular,

(1) **Complete malleability**: The physical system in a cyber–physical digital twin can only partially adapt. Furthermore, when such a physical system does adapt, it is typically only the software components of the physical system that can be changed *automatically*. For example, the configuration parameters of an automobile engine controller can be automatically adapted in response to feedback from a cyber twin, but the bodywork of the automobile cannot. By contrast, the cyber system in a cyber–cyber digital twin can completely adapt in response to its twin; theoretically, no change is *unimplementable*.
(2) **True twins**: Cyber–cyber digital twins truly are *twins*; it may become hard to tell which is the simulator and which is the simulated, because either could inform or affect the

other, causing it to adapt any or all of its components, due to complete malleability.
(3) **Simulation Hierarchy**: Since a cyber–cyber digital twin can, itself, also have a digital twin, we can create recursive hierarchies of cyber^N digital twins.

A digital twin can predict a system's response to safety-critical events and uncover previously unknown issues before they become critical. Furthermore, it allows the replay of critical events that occurred in the past in order to assess whether prevention strategies could have mitigated these events. This is an example where the simulation paradigm goes beyond the traditional software engineering testing paradigm; unit and integration tests are designed to uncover known failure modes, whereas simulation allows the whole system to be tested.

This paper covers three principal aspects of cyber–cyber digital twin deployments:

(1) **Scalability**: How can we simulate salient behavioural properties of human–platform interactions in a large complex software platform? We consider two principal aspects of scalability:
   (a) **Execution Scalability**: can we compute results quickly enough to act on them using reasonable computational resources?
   (b) **Developmental Scalability**: can we develop new simulations quickly enough to take action in response to new behaviours?
(2) **Behaviour**: How can we simulate human behaviour realistically enough to make actionable predictions and recommendations?
(3) **Verification and Validation**: Cyber–cyber digital twins bring up many natural verification and validation questions. For example
   (a) **Verification**: To what degree can we *verify* our confidence that the simulation captures the desired salient properties with sufficient faithfulness for the intended digital twin applications?
   (b) **Validation**: How can we ensure that results of simulations are *valid* in order to address the questions, predictions and interventions in mind? This is especially challenging in situations where we are modelling counterfactual behaviours never previously witnessed

These three challenges lie at the heart of the technical and scientific problems we face in developing a scalable, high fidelity simulation able to produce actionable predictions and conclusions. We believe that tackling them will have a profound impact on both social media and on the wider application of cyber–cyber digital twins to general complex systems.

In this paper, we describe how we developed and deployed a hierarchy of digital twin simulations. Each twin occupies a position on a spectrum of engineering trade-offs allowing us to tackle execution scalability by trading simulation fidelity for speed. We describe the architecture of the online real-time simulation layer of the WW hierarchy, and how the other layers of hierarchy are related to it. Our experience is that this architecture is helping us to achieve reasonable developmental scalability. We also briefly outline our hybrid approach to semi-realistic user behaviour imitation and our approach to tackling the problem of verification and validation of simulation outcomes.

The paper concludes with a set of open challenges for both the scientific and software engineering communities. We describe applications to software engineering which promise many future benefits.

2 THE SIMULATION HIERARCHY

Figure 1 depicts the WW Simulation Hierarchy. Human users and the devices they use to interact with Facebook make up the physical aspects of the hierarchy. All other entities in the figure are digital entities, and relationships between them are thus cyber–cyber relationships.

The lower rectangle depicts the Facebook platform itself, considering it conceptually to be a cyber physical simulation of social interaction. In some ways, Facebook itself might be considered a digital twin of physical social interactions, but we leave further study of this perspective to future work. Our focus for this paper are the cyber–cyber digital twins that we have built on top of the Facebook platform.

The upper rectangle depicts the WW system, a simulation platform which rests on top of the Facebook platform itself. WW consists of the bots (simulating real users), and a simulation of the Facebook platform, chosen at one of several levels of precision; here online, offline/synthetic, and emulated. The simulations of the Facebook platform are constructed through a chain of digital twins, thereby forming a simulation hierarchy. In some ways, Facebook itself might be considered a digital twin of physical social interactions, but we leave further study of this perspective to future work.

As can be seen from Figure 1, the relationships between bots, real users, the Facebook platform and its various simulations in the simulation hierarchy collectively form a commutative diagram. In future work we hope to exploit this observation further, through rigorous informal and formal verification and validation obligations which are best formulated using this kind of commutative diagram. For example, suppose we have verified that the emulator correctly captures an important property, P, of the offline simulation. Now, when we have bots used in the offline simulation that maintain P, when transformed into bots in the emulator, we know that emulator interactions with the bots, and offline mode interactions with the bots should behave identically with respect to property P.

The chain of cyber–cyber simulations is represented in the right–hand–side of the diagram by a four-component pyramid, in which each component is a digital twin. The upper digital twins of the pyramid simulate the behaviour of those below, while the base of the pyramid is also a cyber–physical digital twin; the Facebook platform itself.

Online mode. Immediately above the Facebook platform itself, we find the WW online mode. This is our most faithful simulation of the Facebook WWW platform. It is 'faithful' in the sense that, although the behaviour of the bots is a simulation (of real user behaviours), the actions and observations in which these bots engage is executed directly on the real Facebook platform itself. In this way, online mode is a cyber–cyber digital twin, but it is also a web enabled simulation [2]. It differs crucially from conventional simulations

[30, 32, 52] because the simulated process can be digitally executed. By contrast, an engine simulation, for example, must execute a model of the physical engine.

Offline mode. The WW offline simulation is a cyber–cyber digital twin of the online simulation. In offline mode, WW does not necessarily execute all of the bots' actions, nor provide all of the observations to bots that are available on the Facebook platform itself. Rather, offline simulations either synthetically generate content, social graph topology, and bot state (simulated historical actions), or they recapture and cache online simulation results for subsequent offline reuse. These offline modes have many advantages:

(1) **Speed**: Offline simulations execute orders of magnitude faster than their online counterparts
(2) **Test–friendliness**: Offline execution supports verification and validation of simulation results, because it reduces test flakiness [3].
(3) **Counterfactual Support**: We can combine synthetic network topologies with simulated content and bot state information to create virtual worlds. These virtual worlds share *some* properties with the real world, but also allow us to explore behaviour in counterfactual worlds in which we mutate some or all of the social graph network topology, content and bot state.
(4) **Privacy–safety**: For simulations of sensitive aspects that would not be privacy-safe in online mode, the synthetic simulation mode allows us to perform simulations that are privacy-safe by construction.

These advantages may come at the expense of losing some fidelity of the simulation results, compared to those that would be experienced in reality. Therefore, the simulation hierarchy introduces classical engineering trade-offs between precision and speed.

Emulation. While offline mode improves simulation performance substantially, it is not sufficient for some situations. Many of the applications of WW digital twins aim to improve Facebook's infrastructure and apps in a way that balances protection of normal users and inhibition of malicious users. To tackle this technical challenge, we use an approach reminiscent of genetic improvement [43] that we call 'mechanism design' [2]. Mechanism design uses computational search [24] over the space of product changes, guided by fitness functions that capture the impact of the product change on normal and harmful behaviour.

Fitness Computation. Each fitness computation is, itself, an entire simulation story. Fitness assesses the effect of a product change on a particular kind of behaviour, as experienced by the bots. Effective computational search typically requires a large number of fitness evaluations [27]. This motivated us to develop faster simulations than can be achieved in either online or offline mode. In order to meet these high performance requirements, the top level of the simulation pyramid uses emulation. Emulation uses machine learning to create high performance black box predictive models of the behaviour of lower levels of the pyramid.

In summary. WW employs two strands of simulation: The first strand, to the left-hand side of the diagram, is a cyber–physical simulation of real users by bots. The second strand, to the right-hand side of the diagram, is a chain of cyber–cyber simulations, which simulate the Facebook platform. The interaction between the bots and the simulated Facebook platform is a cyber–cyber interaction which simulates the cyber–physical interaction between real Facebook users and the real Facebook platform.

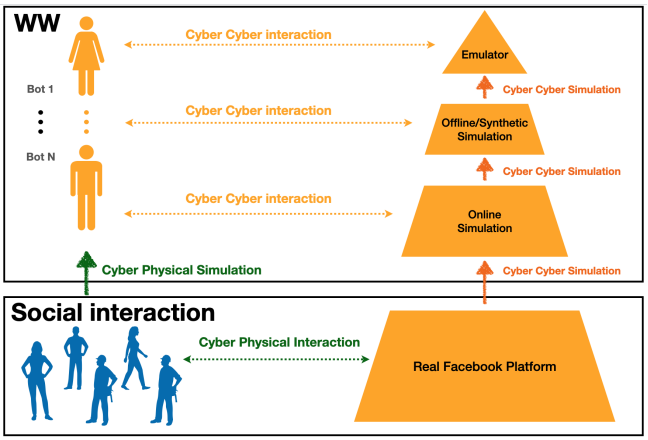

Figure 1: The WW Simulation Hierarchy: a recursive chain of digital twins that trade simulation precision for speed.

3 VERIFICATION AND VALIDATION OF SIMULATION OUTCOMES

Simulations need to be verified and validated before decisions can be based on them. The WW verification and validation framework is built around 4 properties: fidelity, regression, isolation and correctness (FRIC). We developed the MIA[3] testing framework to test the FRIC properties.

We tackle all four properties using the full panoply of software testing techniques, including mutation [29], regression [58], metamorphic [3], and property-based [14] techniques. MIA currently automates end–to–end, regression and metamorphic testing (end–to-end and regression testing are formulated, implemented and deployed as special cases of metamorphic testing [3]). We are currently extending MIA to handle other types of testing.

Fidelity: There are two aspects of fidelity that we test: simulation-simulation fidelity and reality-simulation fidelity.

Simulation-simulation fidelity can sometimes be constructed without real-world data, as a consistency check between different cyber–cyber digital twins. Reality–simulation fidelity measures the closeness of WW simulation results to reality. While simulation–simulation fidelity captures an internal consistency of the hierarchy, we cannot be sure that the results we are checking are valid (that they *matter*) without reality–simulation fidelity.

In some cases, measurement of reality-simulation fidelity proves to be straight forward. For example, when simulating an event sequence that previously occurred in reality, we have the ground truth from earlier observations. However, counterfactual simulations (which use newer versions of the Facebook platform not previously deployed) naturally raise a profound question about the meaning of fidelity: to what extent can we determine faithfulness

of simulation to reality when aspects of the simulated situations have never been previously witnessed?

Regression: At Facebook, any code change is called a 'diff'. Testing performed on individual diffs when they are submitted for code review is called 'per diff' testing. WW platform development occurs as a sequence of diffs and it is very useful to perform regression testing on each diff. Regression testing is clearly useful for per-diff mode testing of changes to WW code. When this kind of diff test fails, it indicates that the proposed code changes would alter simulation results.

Not only do we regression test the WW code, we also 'diff sniff' other Facebook code changes entirely unrelated to WW itself. We use the phrase 'diff sniff' to refer to a regression test on an arbitrary diff which is not necessarily a WW diff. A diff sniff regression test runs a short simulation on both the current code and on the proposed code change and compares them to 'sniff' for problematic changes. If the regression test fails, it means that the diff disrupted the behaviour of WW bots. This, in turn, means that the diff is likely to disrupt real users, and could indicate a serious bug.

This diff sniff mode exploits the fact that our test involves simulations and, as such, allows us to deploy social testing [2] on all diffs that might disrupt users' normal behaviours. For example, a simulation that involves a bot community sending and responding to messages can uncover potential social bugs in diffs that change Facebook's messaging infrastructure.

Isolation: Bots must not interact with normal users. The WW framework has systems in place to ensure that bots remain isolated from real users [2]. Specifically, we built WW to have inherent isolation of bots by design and also encase the bots in a privacy layer that further traps any potential to bleed through from bot behaviour to production. However, isolation is clearly highly important, so we also need to adopt a 'trust but verify' policy with regard to our own isolation-by-construction approach, thereby continually testing that our systems are, indeed, separate.

Correctness: In a WW simulation, we can only execute actions that real users are able to execute on the platform. Based on a policy and the observations received, a bot can execute or ignore a specific type of action. WW code errors might break the simulation framework, might alter the behaviour of the platform, or might cause incorrect bot actions. The current WW framework uses end–to–end, regression and metamorphic testing to assess whether the simulation terminates without failure [3]. We are currently working on property-based testing [14] that will test a wider variety of properties of a simulation including the number and types of bot actions and the number of simulated bots.

4 REALISTIC BOTS

We refer to bots' distinguishing attributes, such as the demographic they simulate, as a *persona*. Personas specify salient characteristics of bots such as their age, gender, and platform activity level. They also specify aspects of a bot's goals such as whether it is malicious or benign.

A simulation is typically set up with a set of bot personas that capture the particular scenario of interest. For example, if we are interested in understanding the ability for systems to detect and impede scamming attacks on our users, then we would set up a simulation with scammers and benign users.

In the scamming scenario, we might implement a set of relatively sophisticated scamming behaviours, captured by imitation learning, together with a set of more simple rule–based behaviours that capture the benign users that might become victims to such scamming attacks.

Our bot behaviour modelling techniques range from simple random actions to more advanced rule-based and ML-learnt behaviours. Different techniques can coexist in the same simulation, as different bots use different methods based on their role and persona.

The set of available personas is defined at the beginning of a simulation, roles are then assigned to each bot according to a predefined set of rules yielding a distribution comparable to the real environment.

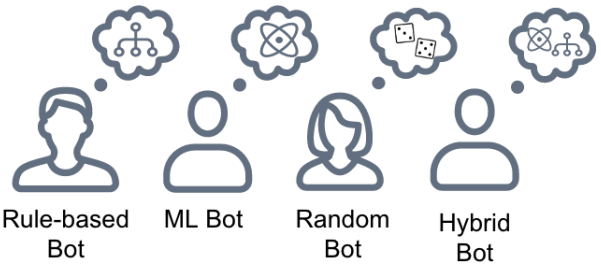

Figure 2: WW simulations can include bots following different strategies. The phrase 'Hybrid bot' represents a mix of strategies in a single agent.

Figure 2 depicts the four principal techniques we currently use to train bots' behaviours in WW, each of which we describe in more detail below.

Random: The simplest behavioural model causes bots to execute actions randomly from a pool of available actions. This model can be useful in cases such as testing, where the developer may care less about realistic behaviour than they do about code coverage. Random behaviour also provides a useful baseline against which to compare more intelligent and purposeful behaviours.

Rule-based: Given some basic assumptions about the goal of a simulation, it is often possible to express the desired user behaviour by a set of parameterised rules. The rules serve as a backbone algorithm that captures the core of a bot's behaviour. The agents in a simulation often do not need to be very sophisticated for their aggregate behaviour to lead to interesting conclusions.

Using the rule-based approach allows us to focus on one specific aspect of behaviour. Rule–based bots are also more likely to behave deterministically. Determinism tends to help with interpretability, and is particularly useful for exploring extreme behaviour. Rule-based behaviours also allow us to capture the distilled essence of harmful behaviour, and thereby check that our systems are robust even in the presence of such extremes.

Even if we have not witnessed such extreme behaviour, this remains a useful integrity stress test for the cyber-physical platform.

Although no specific user may behave in this way, if we suitably capture a specific behaviour of interest, this can be used to explore mechanisms designed to either inhibit or allow that behaviour.

Rule–based bots are configured according to parameters, which capture the specific behaviours, outlined by the overall backbone algorithm. This allows us to parameter sweep over the specific aspects of behaviour, under certain conditions and constraints. The parameters in the rule-based behaviour descriptions allow us to investigate whole areas of counterfactuals: by sweeping over a range of parameters we can explore how well harmful behaviour of various levels of severity is handled.

ML: Clearly, rule-based strategies alone are insufficient to capture all of the nuanced complexities and emergent properties of social interaction. Where more realistic behaviour is required, we have employed Machine Learning techniques to imitate decisions made by classes of real users. We only simulate whole classes of user of behaviour, thereby allowing us to train a bot to imitate specific personas, but not specific individuals.

We use sequences of actions from de–identified logs to train imitation learners. We encode each action as a vector using Word2Vec [39]. Positive samples are collected from the logs of executed actions, together with information about the context in which the action was carried out. The context is then used to infer other actions that could have been performed, but were not.

The model is trained to predict the next action given the context, one positive sample (the real action) and one negative sample (the available action that was not executed in the log). During the simulation, at each step, pairwise comparison is performed between available choices, using scores from the ML model. The action with the highest score according to this ranking is then executed.

Each type of action has a defined set of features that are collected for accurate prediction. These features are sets of de–identified features which, like all supervised training based on labelled features, are used to train the bot to generalise to particular kinds of decision responses. For example, a bot may need to decide which group to join next. Groups features include when the group was created, the number of members, the number of posts per day, etc. From these features we can train a bot that simulates scamming behaviours by training it to join groups with similar features to those features that scammers, on aggregate, have tended to favour in the past.

ML–Rul–based hybrids: Our imitation learners can be substituted in a plug–and–play manner into our backbone rule-based algorithms. This supports us in deploying hybrid approaches which combine a role-based backbone, with different imitation learners to capture different kinds of personas and different kinds of behaviour patterns. Such a hybrid typically uses the rule-based algorithm for its overall strategy, such as crawling over the platform. However, at each step in the crawl, a number of options are available. When a bot has a range of decisions available to it, it uses the imitation learner in order to determine the next step.

5 FACEBOOK'S WW MICRO-SERVICE ARCHITECTURE FOR USER SIMULATION

In the WW architecture, we describe the set of actions and responses that a bot can perform its 'simulation story'. Simulation stories can be composed. Larger simulations can be constructed from smaller ones.

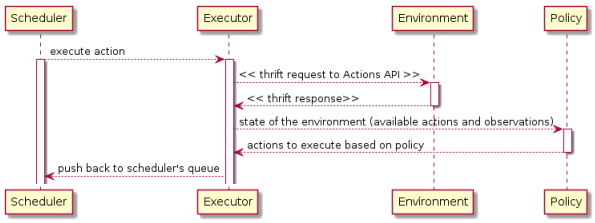

Figure 3: The Top-Level WW Micro Services Architecture for Real-Time Simulation

In addition to simulating bot actions, the WW architecture supports monitoring, statistical analysis, and realtime scheduling for high fidelity online simulations.

Figure 3 shows the top level architecture of WW. There is a scheduler and an executor which interact with the Facebook environment according to a policy.

The scheduler dispatches events at the appropriate simulation time and is responsible for terminating simulations. The platform policy is the 'brain' of the simulation and determines how bots behave. A WW 'simulation story' is defined by an initial action which triggers a cascade of responses. Each response can trigger additional actions which can trigger additional responses. We use a 'plug and play' approach whereby different policies and environments can be used in the same WW simulation story.

The platform policy is responsible for selecting agent actions and for maintaining the state of the execution. Agent actions can range from hard-coded stochastic rules to adaptive actions based on Reinforcement Learning (RL) similar to that used in game AIs [61]. The policy maintains bots' 'memory' and is also responsible for content replacement since environment calls are stateless. Each action has a content placeholder like '<comment>'.

The interface to the Facebook environment is provided by an API for action execution which is implemented as a WWW Thrift service [49]. The environment service can be (re)implemented for different platforms or as a synthetic environment, where all actions are executed offline.

This architecture provides several advantages:

(1) **Plug and play**: Policies can be reused in any simulation and different platform policies can be applied to the same WW simulation story.
(2) **Flexibility**: The environment is implemented as a service which provides the Actions API. The Environment Service can be implemented for any platform, allowing a simulation story to be run on different platforms.
(3) **Scalability**: The Environment Service is deployed on multiple machines and can be scaled when needed.

6 OPEN CHALLENGES IN APPLYING CYBER–CYBER DIGITAL TWINS TO WEB ENABLED SIMULATION

Modelling Users: Modelling the behaviour of real users is a natural challenge in any simulation of social behaviours. We have found

a combination of rule–based approaches and decision procedures guided by imitation learning to be effective for many use cases. However, our work has only scratched the surface of what is required in order to simulate behaviours and complex community interactions. Much more research is needed to accurately simulate situations involving complex human behaviours..

We have found that, although realism is a natural goal, it is not essential in all application areas. For example, when using mechanisms designed to tackle harmful behaviours, we often want to simulate the 'distilled essence' of the harmful behaviour. This distilled essence is particularly useful in the early phases of the search process to find mechanisms to counteract such behaviour. Any mechanisms which can't handle the most egregious examples of a harmful behaviour can be discarded early in the search process.

However, finding interventions to tackle harmful behaviour is inherently a multi–objective search process: while we want to add friction to reduce the prevalence and effectiveness of harmful behaviours, we do *not* want to impede normal user behaviour. In order to incorporate this into the search for optimised mechanisms, we are naturally led to a multi-objective formulation. In such a multi objective formulation, we need to balance the impact of mechanisms on harmful behaviours against their impact on normal behaviour. This, in turn, requires a realistic model of normal behaviour, wherein lies the greatest challenge. How does one best capture the normal behaviour of users in such a way that interventions that might disrupt their normal behaviour can be dismissed early in the search process?

Balancing Speed and Precision: Cyber–cyber digital twins will increasingly need to produce simulation results in real-time. Furthermore, predictions which alter the real platform have to be made ahead of real-time. These tight time constraints, coupled with the potential need for large amounts of training data, present us with a scalability challenge: Cyber–cyber digital twins will increasingly need to produce simulation results in real-time. And predictions which alter the real platform have to be made ahead of real-time?

WW addresses this challenge with the simulation hierarchy which provides different precision and speed trade-offs for different use cases. When optimising the outcome of multiple agents, this approach allows us to formulate search in terms of a range of precision-speed fitness computation choices. In the early exploration phase of an optimisation algorithm, a more coarse-grained, faster fitness can be used. This corresponds to more abstract levels of the simulation hierarchy. As the search for mechanisms enters the exploitation phases of the search, the algorithm can gradually switch to fitness functions which have higher precision and higher computation cost [35].

Validation, Verification and Testing: We need techniques to verify and test properties of simulation systems that are specifically tailored to interactive multi-agent scenarios. Techniques from AI assisted gameplay [56] and multi-agent system testing [31, 42] may be relevant here, but much more work is needed to determine their usefulness.

For example, we need to deal with unknowable oracles, and widespread non-determinism, and consequently flaky tests [3]. Although challenging, it will also be extremely beneficial to have automated verification for cyber–cyber digital twins. This may be possible, for example, where properties of models of a digital twin can be proved correct. Such a property-preserving model can be used as a surrogate for more computationally expensive alternatives.

Finally, and perhaps most important of all, we need techniques to help understand, check and optimise validity of simulation results. This is challenging because we are often dealing with counterfactual scenarios for which, even when ground truth is known, it can be only partially relevant to validity.

7 FURTHER SOFTWARE ENGINEERING APPLICATIONS OF CYBER–CYBER DIGITAL TWINS

In this section we outline ways in which cyber–cyber digital twins could be applied in four areas of current active software engineering research. Our list is not intended to be exhaustive, but merely illustrative.

Testing: Currently we tend to think of a testing tool as a separate system that automatically tests the system under test [6, 9]. The test system might report tests automatically into a continuous integration system [5]. However, we typically do not think of testing tools as cyber–cyber digital twins that run in production alongside the system being tested. Nevertheless, with cyber–physical digital twins, it is increasingly common to see continuous real-time deployment of the digital twin, running alongside the system under test. In some regards, therefore, software testing research is behind other engineering testing research in the use of digital twins as part of an overall automated continuous testing deployment.

Automated Repair: Automated repair is gaining increasing attention from both software engineering researchers [17] and practitioners [38] . However, like testing, automated repair systems are often considered to be executed by the provider, rather than automatically executed in production, and therefore implicitly invoked by the user. Consider what a cyber–cyber digital repair twin might look like. Such a system would automatically run in production, and test repairs to the system under test. It would model execution of the system under test and its repairs. It would automatically respond to unwanted behaviour witnessed in production. It would deploy patches to the system under test as the system executes. In this way, the twin is to the system under test, as a personal surgeon is to a patient; continually monitoring vital signs, and responding with cures (fixes) when problems are uncovered.

Adaptive and Self-Managing Systems and Autonomous Systems: Adaptive autonomous systems need to be able to respond quickly to their environment, making predictions and taking actions to respond to the problems in real time [33]. This is a perfect setting in which to deploy a cyber–cyber digital twin, able to maintain and adapt its own model of the system, and to respond to changes by adapting that system, autonomously, as the two twins execute in parallel. The twins respond to the world together in a technological embrace that enables one twin to simulate in order to explore counterfactual variations of previously witnessed behaviour, during downtime when its twin would otherwise be relatively inactive [26].

This has been referred to as the 'dreaming phones' [26], because the simulation of smart phone optimisations to be deployed the next day could take place overnight, while the phone is charging, reusing the phone's computational resources for the simulation

overnight; the phone is essentially 'dreaming' about the day that just occurred.

Modern Code Review: It is increasingly common to think of tools that assist code review as bots [51] that, like engineers, comment on code using the same continuous integration interfaces used by human engineers [5, 13]. Taking this a step further, we could imagine a cyber–cyber digital twin that plays the role of a maintainers' assistant [54].

Such an assistant would be a full member of the team, in the sense that the assistant would be able to upload code for review and to respond to code review feedback from humans and other bots, as well as commenting on code as a reviewer. The assistant would maintain a mental model of the system being developed, and would use this to take measurements and recommend changes, and also comment on other engineers' changes. Like any other member of the team, this assistant would have particular skills and attributes specific to its background and training. This system might lack a lot of the context that human engineers would bring to the development and review process. In compensation, however, the bot would be able to perform meticulous repeated experimental analyses in order to provide comprehensive scientific evidence to back up its claims.

8 RELATED WORK

We have previously discussed the wide range of research related to Web Enabled Simulation [2] including Multi-Agent Reinforcement Learning [12], Search-Based Software Engineering [25], and Automated Mechanism Design [48]. These topics are also relevant here, but we focus on the work most related to digital-twins [44].

While cyber–cyber digital twins are a new concept[1], many of the cyber–physical digital twin insights are also relevant to them. Digital Twins were first described by Grieves in 2002 in the context of Product Lifecycle Management [7] [18] [19]. This model included a real space, a virtual space, and two-way communication between them to keep them synchronized.

Digital twins have already been deployed in healthcare, meteorology, manufacturing, education, smart cities, transport and the energy sector [44]. Singapore and other 'smart cities' have created digital twins to manage roadways, pedestrian traffic, energy use and other city functions [36]. The ultimate digital twin is an EU plan for a digital twin of the entire earth [8].

For social media, successful prediction enables features to be chosen that maximize social good and user satisfaction. This kind of optimization may require robust models of normal human users. To test the system's response to extreme circumstances, it is also useful[2] to model aberrant human behaviours, or to test against increasingly smart malicious bot AI.

8.1 Digital twins in healthcare

Digital twin technology for healthcare could fulfil the goals of personalised medicine by enabling predictive medicine, where disease can be predicted and stopped before it happens [11].

Data-driven and rule–based mathematical models have been used in conjunction to determine effective interventions [23, 34, 37, 46, 59]. There are similarities between digital twins from engineering paradigms and those applied to personalised healthcare. There are also some marked differences with respect to the engineering paradigm. For example, patients' health can be continuously monitored via wearable devices [22]. However, direct intervention on a patient is clearly more problematic than it would be in the purely engineering paradigm. Instead, healthcare digital twins tend to aim at supporting the engineering of a healthy status via recommendations and early identification of upcoming disease states [10, 11]. Such healthcare–facing digital twins are not fully automated.

8.2 Simulation for Recommender Systems

In 2004, MySpace[3] became the first social media site to reach a million users. Facebook first became available to the general public in 2006 and now has 2.8 billion[4] monthly active users. There are now wide range of social media sites[5], yet none has yet built a full cyber–cyber digital twin. Nevertheless, many sites now use online experiments and simulations to improve their services[6].

Recommender systems allow social media platforms to select the pages, news stories, ads, groups and friend suggestions to show to users [15]. Simulation has been used to design and test recommender systems and to model their social impact [57].

The first recommender systems were based on 'collaborative filtering' [16]. Currently, recommender systems use advanced neural networks and graph algorithms [55] and incorporate a richer recommendation context. However, the current generation of recommender systems remain relatively 'myopic'; focusing on individual recommendations rather than on optimizing for impact over time. The opportunities for better recommendations are well understood, though the field remains wide open for improvement [50].

The 'Reco-gym' system [45] is a simulation test environment for reinforcement-learning based recommender systems. The creators hope that it will stimulate the kind of advancement for recommenders that the OpenAI Gym did for general reinforcement learning. Two other recent examples include a simulator for evaluating conversational recommender systems [60] and the first-generation 'RecSim' platform for testing reinforcement learning based recommender systems [28].

Recommender systems have also been blamed for various kinds of negative social impact: radicalization, polarization, addiction, and the prevalence of 'click-bait' headlines [40] [53]. Simulation is being used to understand these phenomena and to create advanced recommender systems that address the issues. For example, the recent 'RecSim NG' system [41] simulates content providers in addition to users and studies the impact of their incentives under different recommender policies.

[1] They are mentioned as cyber-digital twins in this interview: https://siliconangle.com/2020/02/26/qa-accenture-creates-cyber-digital-twins-simulate-potential-attack-scenarios-rsac/ but we are not aware of any other previous publications on the topic. We prefer the term 'cyber–cyber' over 'cyber-digital' because it emphasises the composability that naturally leads to the the many unique properties of cyber–cyber digital twins, such as the recursive simulation hierarchy and the twins' inherent maximal adaptivity.

[2] https://datadome.co/bot-management-protection/bot-detection-how-to-identify-bot-traffic-to-your-website/

[3] https://en.wikipedia.org/wiki/Myspace

[4] https://en.wikipedia.org/wiki/Facebook

[5] https://en.wikipedia.org/wiki/List_of_social_networking_services

[6] https://hbr.org/2017/09/the-surprising-power-of-online-experiments

8.3 Game Simulations and Learned Models

In the case of a cyber–cyber digital twin, we already *have* the true model implemented in software, so one may question the relevance of learning real–world models. However, for many use cases, the software platform has many restrictions which make experimentation difficult or impossible and indeed create the need for a twin: these are mostly related to speed, security (it can be hard to decouple some aspects of the platform from real users) and flexibility.

Hence, learning a model of the platform may provide a useful alternative to writing the software for a high-level model. Impressive progress has been made in learning game models [20] [21] [47]. Whether these methods can scale to the challenge of learning digital twins remains an open question. When learning a game model, the agent is in control of its interactions with the environment, and may try risky actions without bad consequences. This allows it to quickly learn circumstances in which specific actions are impossible; a luxury we often cannot afford when learning models of real-world systems.

However, social media platforms have vast inflows of data from which to train bots, thereby facilitating sample efficiency. Another very practical approach is to start with a hand-programmed high-level model and learn the parameters of the model by probing the system. This has the advantage of being well understood, and still open to direct counterfactual experimentation and relatively efficient adversarial learning (e.g. to develop platform variants that limit bad actors). Hence an engineering approach that mixes hand-designed and learned components offers great promise, and this is already a significant line of research[7].

REFERENCES

[1] David Adam. 2020. Special report: The simulations driving the world's response to COVID-19. *Nature* (April 2020).

[2] John Ahlgren, Maria Eugenia Berezin, Kinga Bojarczuk, Elena Dulskyte, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Ralf Laemmel, Erik Meijer, Silvia Sapora, and Justin Spahr-Summers. 2020. WES: Agent-based User Interaction Simulation on Real Infrastructure. In *GI @ ICSE 2020*, Shin Yoo, Justyna Petke, Westley Weimer, and Bobby R. Bruce (Eds.). ACM, 276–284. https://doi.org/doi:10.1145/3387940.3392089 Invited Keynote.

[3] John Ahlgren, Maria Eugenia Berezin, Kinga Bojarczuk, Elena Dulskyte, Inna Dvortsova, Johann George, Natalija Gucevska, Mark Harman, Maria Lomeli, Erik Meijer, Silvia Sapora, and Justin Spahr-Summers. 2021. Testing Web Enabled Simulation at Scale Using Metamorphic Testing. In *International Conference on Software Engineering (ICSE) Software Engineering in Practice (SEIP) track*. Virtual.

[4] Saif Al-Sultan, Moath M. Al-Doori, Ali H. Al-Bayatti, and Hussien Zedan. 2014. A comprehensive survey on vehicular Ad Hoc network. *Journal of Network and Computer Applications* 37 (2014), 380 – 392.

[5] Nadia Alshahwan, Xinbo Gao, Mark Harman, Yue Jia, Ke Mao, Alexander Mols, Taijin Tei, and Ilya Zorin. 2018. Deploying Search Based Software Engineering with Sapienz at Facebook (keynote paper). In 10^{th} *International Symposium on Search Based Software Engineering (SSBSE 2018)*. Montpellier, France, 3–45. Springer LNCS 11036.

[6] Saswat Anand, Antonia Bertolino, Edmund Burke, Tsong Yueh Chen, John Clark, Myra B. Cohen, Wolfgang Grieskamp, Mark Harman, Mary Jean Harrold, Jenny Li, Phil McMinn, and Hong Zhu. 2013. An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software* 86, 8 (August 2013), 1978–2001.

[7] B. R. Barricelli, E. Casiraghi, and D. Fogli. 2019. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access* 7 (2019), 167653–167671. https://doi.org/10.1109/ACCESS.2019.2953499

[8] Peter Bauer, Bjorn Stevens, and Wilco Hazeleger. 2021. A digital twin of Earth for the green transition. *Nature Climate Change* 11 (2021), 80 – 83.

[9] Antonia Bertolino. 2007. Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE'07)*. IEEE, 85–103.

[10] Saul Blecker, Stuart Katz, LI Horwitz, Gilad Kuperman, H Park, A Gold, and David Sontag. 2016. Comparison of approaches for heart failure case identification from electronic health record data. *JAMA Cardiology* 1, 9 (2016), 1014–1020.

[11] Koen Bruynseels, Filippo Santoni de Sio, and Jeroen van den Hoven. 2018. Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm. *Frontiers in Genetics* 9 (2018), 31. https://doi.org/10.3389/fgene.2018.00031

[12] L. Busoniu, R. Babuska, and B. De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

[13] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. W. O'Hearn, I. Papakonstantinou, J. Purbrick, and D. Rodriguez. 2015. Moving Fast with Software Verification. In *NASA Formal Methods - 7th International Symposium*. 3–11.

[14] Koen Claessen and John Hughes. 2002. Testing monadic code with QuickCheck. *ACM SIGPLAN Notices* 37, 12 (2002), 47–59.

[15] Magdalini Eirinaki, Jerry Gao, Iraklis Varlamis, and Konstantinos Tserpes. 2018. Recommender Systems for Large-Scale Social Networks: A review of challenges and solutions. *Future Generation Computer Systems* 78 (2018), 413–418. https://doi.org/10.1016/j.future.2017.09.015

[16] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* 35, 12 (Dec. 1992), 61–70. https://doi.org/10.1145/138859.138867

[17] Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. 2019. Automated program repair. *Commun. ACM* 62, 12 (2019), 56–65.

[18] Michael Grieves. 2015. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. (2015).

[19] Michael Grieves and John Vickers. 2017. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, Franz-Josef Kahlen, Shannon Flumerfelt, and Anabela Alves (Eds.). Springer International Publishing, 85–113. https://doi.org/10.1007/978-3-319-38756-7_4

[20] David Ha and Jürgen Schmidhuber. 2018. World Models. *CoRR* abs/1803.10122 (2018). arXiv:1803.10122 http://arxiv.org/abs/1803.10122

[21] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to Control: Learning Behaviors by Latent Imagination. *CoRR* abs/1912.01603 (2019). arXiv:1912.01603 http://arxiv.org/abs/1912.01603

[22] Thurow K Haghi M and Stoll R. 2017. Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices. *Healthc Inform Res.* 1 (2017). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5334130/

[23] Yoni Halpern, Steven Horng, and David Sontag. 2016. Clinical Tagging with Joint Probabilistic Models. In *Proceedings of the 1st Machine Learning for Healthcare Conference (Proceedings of Machine Learning Research, Vol. 56)*, Finale Doshi-Velez, Jim Fackler, David Kale, Byron Wallace, and Jenna Wiens (Eds.). 209–225.

[24] Mark Harman. 2007. The current state and future of Search Based Software Engineering. In *Future of Software Engineering 2007*, Lionel Briand and Alexander Wolf (Eds.). IEEE Computer Society Press, Los Alamitos, California, USA. This volume.

[25] Mark Harman, Yue Jia, Jens Krinke, Bill Langdon, Justyna Petke, and Yuanyuan Zhang. 2014. Search based software engineering for software product line engineering: a survey and directions for future work (Keynote Paper). In 18^{th} *International Software Product Line Conference (SPLC 14)*. Florence, Italy, 5–18.

[26] Mark Harman, Yue Jia, William B. Langdon, Justyna Petke, Iman Hemati Moghadam, Shin Yoo, and Fan Wu. 2014. Genetic Improvement for Adaptive Software Engineering (Keynote Paper). In 9^{th} *International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2014)* (Hyderabad, India). ACM, New York, NY, USA, 1–4. https://doi.org/10.1145/2593929.2600116

[27] Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza, and Shin Yoo. 2012. Search Based Software Engineering: Techniques, Taxonomy, Tutorial. In *Empirical software engineering and verification: LASER 2009-2010*, Bertrand Meyer and Martin Nordio (Eds.). Springer, 1–59. LNCS 7007.

[28] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. RecSim: A Configurable Simulation Platform for Recommender Systems. *arXiv e-prints* (Sep 2019), arXiv:1909.04847.

[29] Yue Jia and Mark Harman. 2011. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering* 37, 5 (September–October 2011), 649 – 678.

[30] Gregory L Johnson, Clayton L Hanson, Stuart P Hardegree, and Edward B Ballard. 1996. Stochastic weather simulation: Overview and analysis of two commonly used models. *Journal of Applied Meteorology* 35, 10 (1996), 1878–1896.

[31] Sabrine Kalboussi, Slim Bechikh, Marouane Kessentini, and Lamjed Ben Said. 2013. On the Influence of the Number of Objectives in Evolutionary Autonomous Software Agent Testing. In *25th International Conference on Tools with Artificial Intelligence (ICTAI '13)*. IEEE, Herndon, VA, USA, 229–234.

[7] Also see JuliaCon 2020 tutorial: Doing Scientific Machine Learning (SciML) with Julia: https://www.youtube.com/watch?v=QwVO0Xh2Hbg

[32] Jack PC Kleijnen. 2005. Supply chain simulation tools and techniques: a survey. *International journal of simulation and process modelling* 1, 1-2 (2005), 82–89.

[33] Christian Krupitzer, Felix Maximilian Roth, Sebastian Van Syckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive Mobile Computing* 17 (2015), 184–206.

[34] Trent Kyono, Fiona J. Gilbert, and Mihaela van der Schaar. 2019. Multi-view Multi-task Learning for Improving Autonomous Mammogram Diagnosis. In *Proceedings of the 4th Machine Learning for Healthcare Conference*. PMLR, 571–591. http://proceedings.mlr.press/v106/kyono19a.html

[35] Benjamin Letham and Eytan Bakshy. 2019. Bayesian Optimization for Policy Search via Online-Offline Experimentation. *Journal of Machine Learning Research* 20 (2019), 145:1–145:30.

[36] Patricia Liceras. 2019. *Singapore experiments with its digital twin to improve city life.* https://www.smartcitylab.com/blog/digital-transformation/singapore-experiments-with-its-digital-twin-to-improve-city-life/

[37] Bryan Lim and Mihaela van der Schaar. 2018. Disease-Atlas: Navigating Disease Trajectories with Deep Learning. arXiv:1803.10254 [stat.ML]

[38] Alexandru Marginean, Johannes Bader, Satish Chandra, Mark Harman, Yue Jia, Ke Mao, Alexander Mols, and Andrew Scott. 2019. SapFix: Automated End-to-End Repair at Scale. In *International Conference on Software Engineering (ICSE) Software Engineering in Practice (SEIP) track*. Montreal, Canada.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[40] Silvia Milano, Mariarosaria Taddeo, and Luciano Floridi. 2021. Recommender systems and their ethical challenges. *AI and Society* 35 (2021), 957 – 967.

[41] Martin Mladenov, Chih-Wei Hsu, Vihan Jain, Eugene Ie, Christopher Colby, Nicolas Mayoraz, Hubert Pham, Dustin Tran, Ivan Vendrov, and Craig Boutilier. 2021. RecSim NG: Toward Principled Uncertainty Modeling for Recommender Ecosystems. *arXiv:2103.08057 [cs]* (March 2021). http://arxiv.org/abs/2103.08057 arXiv: 2103.08057.

[42] Cu Nguyen, Anna Perini, Paolo Tonella, Simon Miles, Mark Harman, and Michael Luck. 2009. Evolutionary Testing of Autonomous Software Agents. In 8^{th} *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*. Budapest, Hungary, 521–528.

[43] Justyna Petke, Saemundur O. Haraldsson, Mark Harman, William B. Langdon, David R. White, and John R. Woodward. 2018. Genetic Improvement of Software: a Comprehensive Survey. *IEEE Transactions on Evolutionary Computation* 22, 3 (June 2018), 415–432. https://doi.org/doi:10.1109/TEVC.2017.2693219

[44] A. Rasheed, O. San, and T. Kvamsdal. 2020. Digital Twin: Values, Challenges and Enablers From a Modeling Perspective. 8 (2020), 21980–22012. https://doi.org/10.1109/ACCESS.2020.2970143 Conference Name: IEEE Access.

[45] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising. *arXiv:1808.00720 [cs]* (Sept. 2018). http://arxiv.org/abs/1808.00720 arXiv: 1808.00720.

[46] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. 2017. Learning a Health Knowledge Graph from Electronic Medical Records. *Nature Scientific Reports* 7, 1 (2017), 5994.

[47] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2019. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *CoRR* abs/1911.08265 (2019). arXiv:1911.08265 http://arxiv.org/abs/1911.08265

[48] Weiran Shen, Pingzhong Tang, and Song Zuo. 2019. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 215–223. https://arxiv.org/pdf/1805.03382.pdf

[49] Mark Slee, Aditya Agarwal, and Marc Kwiatkowski. 2007. Thrift:Scalable cross-language services implementation. *Facebook white paper* 5, 8 (2007), 127.

[50] B. Smith and G. Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 03 (may 2017), 12–18. https://doi.org/10.1109/MIC.2017.72

[51] Margaret-Anne D. Storey and Alexey Zagalsky. 2016. Disrupting developer productivity one bot at a time. In *Proceedings of the 24th International Symposium on Foundations of Software Engineering (FSE 2016), Seattle, WA, USA, November 13-18, 2016*. ACM, 928–931.

[52] Sergio Terzi and Sergio Cavalieri. 2004. Simulation in the supply chain context: a survey. *Computers in Industry* 53, 1 (2004), 3–16.

[53] Wenjie Wang, Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2020. "Click" Is Not Equal to "Like": Counterfactual Recommendation for Mitigating Clickbait Issue. arXiv:2009.09945 [cs.IR]

[54] Martin Ward. 1999. Assembler to C Migration using the FermaT Transformation System. In *IEEE International Conference on Software Maintenance (ICSM'99)* (Oxford, UK). IEEE Computer Society Press, Los Alamitos, California, USA.

[55] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. arXiv:2011.02260 [cs.IR]

[56] Geogios N Yannakakis. 2012. Game AI revisited. In *Proceedings of the 9th conference on Computing Frontiers*. 285–292.

[57] Sirui Yao, Yoni Halpern, Nithum Thain, Xuezhi Wang, Kang Lee, Flavien Prost, Ed H. Chi, Jilin Chen, and Alex Beutel. 2021. Measuring Recommender System Effects with Simulated Users. arXiv:2101.04526 [cs.LG]

[58] Shin Yoo and Mark Harman. 2012. Regression Testing Minimisation, Selection and Prioritisation: A Survey. *Journal of Software Testing, Verification and Reliability* 22, 2 (2012), 67–120.

[59] Jinsung Yoon, Ahmed Alaa, Scott Hu, and Mihaela Schaar. 2016. ForecastICU: A Prognostic Decision Support System for Timely Prediction of Intensive Care Unit Admission. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, 1680–1689. http://proceedings.mlr.press/v48/yoon16.html

[60] Shuo Zhang and Krisztian Balog. 2020. Evaluating Conversational Recommender Systems via User Simulation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Aug. 2020), 1512–1520. https://doi.org/10.1145/3394486.3403202 arXiv: 2006.08732.

[61] Yan Zheng, Changjie Fan, Xiaofei Xie, Ting Su, Lei Ma, Jianye Hao, Zhaopeng Meng, Yang Liu, Ruimin Shen, and Yingfeng Chen. 2019. Wuji: Automatic Online Combat Game Testing Using Evolutionary Deep Reinforcement Learning. In *Automated software engineering (ASE)*. IEEE, 772–784.