Report from Dagstuhl Seminar 19471

BOTse: Bots in Software Engineering

Edited by

Margaret-Anne Storey[1], Alexander Serebrenik[2], Carolyn Penstein Rosé[3], Thomas Zimmermann[4], and James D. Herbsleb[5]

1 **University of Victoria, CA,** `mstorey@uvic.ca`
2 **Eindhoven University of Technology, NL,** `a.serebrenik@tue.nl`
3 **Carnegie Mellon University – Pittsburgh, US,** `cprose@cs.cmu.edu`
4 **Microsoft Corporation – Redmond, US,** `tzimmer@microsoft.com`
5 **Carnegie Mellon University – Pittsburgh, US,** `jim.herbsleb@gmail.com`

Abstract

This report documents the program and the outcomes of the Dagstuhl Seminar 19471 “BOTse: Bots in Software Engineering”. This Dagstuhl seminar brought researchers and practitioners together from multiple research communities with disparate views of what bots are and what they can do for software engineering. The goals were to understand how bots are used today, how they could be used in innovative ways in the future, how the use of bots can be compared and synthesized, and to identify and share risks and challenges that may emerge from using bots in practice. The report briefly summarizes the goals and format of the seminar and provides selected insights and results collected during the seminar.

Seminar November 17–22, 2019 – http://www.dagstuhl.de/19471
2012 ACM Subject Classification Human-centered computing, Software and its engineering
Keywords and phrases automated software development, bots, chatbots, collaborative software development, cscw, devops, nlp, software engineering
Digital Object Identifier 10.4230/DagRep.9.11.84
Edited in cooperation with Nathan Cassee

1 Executive Summary

James D. Herbsleb
Carolyn Penstein Rosé
Alexander Serebrenik
Margaret-Anne Storey
Thomas Zimmermann

This Dagstuhl seminar brought researchers and practitioners together from multiple research communities with disparate views of what bots are and what they can do for software engineering. The goals were to understand how bots are used today, how they could be used in innovative ways in the future, how the use of bots can be compared and synthesized, and to identify and share risks and challenges that may emerge from using bots in practice.

Bots, often called chatbots, are considered by some to be computer programs that provide a conversational style interface for interacting with software services, while others consider bots to be any semi-autonomous software service that may or may not take on a human-like persona.

Regardless of the definition of what makes a bot a bot, bots are found in many domains such as shopping, entertainment, education, and personal productivity. In software development, bots are rapidly becoming a *de facto* interface for developers and end users to interact with software services in a myriad of ways: e.g., bots are used to fetch or share information, extract and analyze data, detect and monitor events and activities in communication and social media, connect developers with key stakeholders or with other tools, and provide feedback and recommendations on individual and collaborative tasks.

Through this Dagstuhl Seminar, we aimed to gain important insights on how bots may play a role in improving software development productivity and in enhancing collaborative software development. In particular we discussed how bots, with or without a conversational UI, may play a prominent role in software practice. We gathered literature and resources on how bots can have an impact on development processes, software quality, and on end users. The goal was to channel previously siloed communities and through this confluence forge a common vision and plot next steps that might leverage the variety of expertise and push forward both the research and the practices related to bots. The activities were meant to surface the difficult questions and tensions that arise when one looks beyond what at first blush appears to be a superficial distinction, but in fact touches upon core values and driving questions that define the boundaries between fields.

2 Table of Contents

3 Seminar Format

In this seminar, we brought together researchers and practitioners with diverse backgrounds in software engineering (engineering tools, empirical research), natural language processing (NLP), artificial intelligence (AI), data science, machine learning, human computer interaction (HCI), computer-supported cooperative work (CSCW), computer-supported collaborative learning (CSCL), social computing, affective computing and cognitive computing to discuss, explore and recommend how bots could be used in software engineering.

In advance of the seminar, we conducted a short survey to collect relevant questions to be addressed in the seminar and to form break out group topics. At the seminar, we arranged large and small group discussions, breakout activities, short and long talks and bot demos. These activities helped (a) to foster vibrant discussion, (b) to identify and address relevant questions on how bots are or could be used in software engineering, and to (c) foster interaction and collaborations between attendees.

In the next section, we summarize some of the selected insights discussed during the seminar concerning bots in software engineering. We conclude with a list of abstracts from the invited talks presented at the seminar.

4 Bots in Software Engineering: Insights and Outlook

4.1 Definitions: Software Bots and Software Bot Ecosystems

A discussion theme throughout the seminar involved defining what is or is not a bot. An agreed upon definition could assist researchers when designing, evaluating and conducting research that may involve or refer to bots.

From one breakout session, the following ideas emerged. First, it was seen as easier to define what is not a bot. Simple scripts and badges on GitHub were not considered as bots, whereas software that meets some or all of the following criteria may be seen as a bot:

- Automates one or more feature(s)
- Performs one or more function(s) that a human may do
- Interacts with a human or other agents

Furthermore, a bot may additionally exhibit these features:

- Acts autonomously as an independent actor or agent
- Appears intelligent (and may learn)
- Supports feedback loops
- Personifies a human or is human-like
- Has a name, and 2nd person pronoun
- Exhibits emotions and feelings

In a later breakout session, the concept of what is a **software bot** (in general terms, not just within the context of software engineering) was discussed once again but in the context of a faceted taxonomy developed by Carly Lebeuf as part of her Masters' thesis[1]. The definition of software bot given by Lebeuf is: "A software bot is an interface that connects users with software services and provides additional value to the user (by way of its interaction style, automation, and anthropomorphism)". Lebeuf claims it is the additional

[1] https://dspace.library.uvic.ca//handle/1828/10004

value that bots add on top of software's basic capabilities which distinguish bots from non-bot scripts and programs. Indeed, in this view, software bots are seen as special cases of software scripts and programs that bring this additional value in terms of automation, consolidation of multiple services, interaction mechanisms and anthropomorphic features. Chatbots and agents are more specific kinds of software bots, where chatbots are bots with natural language capabilities and agents are bots that can sense/act upon their environments and may be intelligent, autonomous and social.

The faceted taxonomy we discussed has three main dimensions: 1) Environment dimension; 2) Intrinsic dimension; and 3) Interaction dimension. To guide this discussion, we considered the Travis bot and how we could define Travis using this framework. This group discussion led to a number of suggested changes to the faceted taxonomy including refinement of some of the facets (such as scope, dynamism, and predictability) defined for the environment dimension. This framework was seen as a good start but would require much more input from a wider set of researchers with more diverse backgrounds (beyond a software engineering background) to agree on a new version of the framework. Revisiting the framework after the seminar is one of our future goals.

In another breakout we discussed the concept of **Bot Ecosystem**. We arrived at three different definitions as follows:

- A bot ecosystem is a set of bots working on the same or related projects
- A bot ecosystem is the set of APIs provided by a given platform supporting a set of bots using those APIs
- A bot ecosystem is a place where humans and bots can cooperate and communicate, enabled by conventions and tools

Throughout the seminar we also referred extensively to a list of software engineering bots being maintained by Mairieli Wessel[2].

4.2 Developer acceptance

A theme that emerged during the seminar as important was the acceptance of bots among developers, that is, what are important aspects that bots that must be satisfied for bots to be included in the daily work of developers. A breakout group identified a list of do's and don'ts:

- Don't: Repetitive notifications
- Don't: Wrong answers
- Don't: Hide the identity as a bot from the users
- Do: Automation tasks that users don't want to do
- Do: Provide actionable recommendations. Explain the reasoning.
- Do: Functionality for the user to perform the action (if possible).
- Do: Allow users to provide feedback and have bots learn over time.
- Do: Consistency in the task being done
- Do: Make bots more adaptive to user needs.
- Do: Be context aware.

[2] https://github.com/mairieli/awesome-se-bots

An important topic that the group discussed was **trust**. Bots need to build trust over time, for example, by focusing on trust initially rather than recall, not spamming the users with recommendations that are inaccurate. Context was mentioned as being important for trust. Making accurate recommendations can depend on individual developers, their experience and background, the current task, the rest of the team. This emphasizes the importance of adaptive bots that learn over time and are assessed with multi-dimensional benchmarks with respect to effectiveness. From an evaluation perspective, there are also many points of view, for example how effective bots are for individual developers, teams, the entire company, or even the society.

The group also identified several topics for follow-up discussions:

- Can bots eventually take over tasks from humans?
- What are the appropriate levels of intrusiveness?
- What about the maintenance cost of bots?
- What about exploration cost? Is there a tradeoff in the time that it takes to develop bots?
- How to integrate context and knowledge for bots?
- How to identify the right robot for the task?

4.3 To bot or not

Related to what is a bot or not, is the question of when to create a bot or not for a given use case. To explore this common design conundrum (and to also help understand more about what is or is not a bot) we brainstormed some use cases for which we could design bot and non-bot designs. This discussion set the stage for consideration of deeper challenges related to bringing multiple fields together.

The use cases we discussed in breakout groups were as follows: Potential aims of such an assistant included:

- Code review discussion support (to reduce noise in code reviews and related discussions)
- Privacy awareness support during development
- Pull request ranking support (provide support during pull request management)
- Contextual documentation (support for generating and assessing API documentation)
- Support to alleviate burnout and stress
- Transforming Data Science with Interactive Support for Feature Engineering and Model Adaptation
- Onboarding assistant (to lower the barriers to newcomers wishing to contribute to open source projects).

For example, in the breakout on a bot to support contextual documentation, discussion began by considering bots for documentation, and through this discussion specific challenges emerged about how software developers use documentation. Specifically, we focused on use cases from two perspectives: a) from a developer who is writing code to be used (API developers) and b) developers who are using or updating someone else's code (API users).

The functionalities of contextual documentation considered in the discussion included:

- Generating non-existing documentation from scratch
- Assessing documentation
- Modifying/improving/updating existing documentation (triggered by code updates)
- Generating of task-oriented documentation
- Asking refining questions

Throughout our documentation bot use case discussion we returned to two ideas. The first was identifying scenarios where bots were different from scripts. We resolved that this could be dictated by how they are triggered or even how they wait upon a user's reply. Waiting on a user's reply could be a function of communication styles of probing for information or waiting for another person or agent to prompt for dialogue.

The second open discussion we returned to several times was that the usefulness of the bot was something that mattered most, which begs the question of how bots are situated within a broader community and the roles and responsibilities of bots change the impact the roles and responsibilities of humans and their interactions with one another within that environment. More concretely, with the introduction of bots within the environment, who might be helped? Who might be displaced? How do individuals within the environment feel about their participation in the resulting sociotechnical system?

As a concrete example, we compared bot to non bot versions when trying to access documentation examples. In particular, we considered how getting relevant examples for how to use a new API has trade offs of copying an example from online vs. gaining offline social capital of discussing work in person.

When considering the big questions around what should be the roles and responsibilities of bots, and what should be the nature of their interaction with humans within the environment, difficult questions are raised. Some of these questions relate to ethical concerns, which are discussed in the next section. With the lack of clarity surrounding the extent of humanness desired, or what that humanness is meant to accomplish, or whether it is humanness itself that accomplishes those goals best, the difficult question was raised of what specific joint endeavors between the software engineering community and the NLP community would be mutually beneficial. A keynote talk and supplementary talk by David Traum offering an overview of the history of the field of dialogue systems and some pointers to available technologies provided some common ground to begin this substantive exploration. However, questions related to paths forward remain open, especially with regard to what new challenges for dialogue system technology would be interesting for researchers in NLP and would provide capabilities that would be valued within the field of software engineering going forward in the face of open questions.

4.4 Ethical concerns

One of the primary ethical concerns related to bots is whether the bots are allowed to impersonate humans. Such behavior might be considered unethical (as it might compromise privacy) and indeed it is explicitly prohibited by such platforms as Wikipedia. At the same time, previous studies (Murgia et al. 2016)[3] have shown the communities might be more likely to accept bots impersonating humans as opposed to bots disclosing that they are bots. During the seminar, participants identified additional advantages and concerns related to bots impersonating as humans:

- On the one hand, the primary advantage of designing bots impersonating humans is that humans know how to interact with other humans, so it might be easier for them to interact with human-impersonating bots. Moreover, by the same argument human-like communication might be more effective at communicating to developers and triggering

[3] Alessandro Murgia, Daan Janssens, Serge Demeyer, Bogdan Vasilescu: Among the Machines: Human-Bot Interaction on Social Q&A Websites. CHI Extended Abstracts 2016: 1272-1279

the desired action or response from them. Finally, the evolution from "humans using bots" to "egalitarian collaboration between humans and bots" is more likely to be achieved if the difference between humans and bots is less marked.

- On the other hand, bots posing as humans trigger a question of responsibility: who is to blame if the bot is "misbehaving"? Typically bots have owners that take responsibility of the bot's actions and can intervene in their behaviour if needed. If a bot is impersonating a human, there seems to be no obvious way to escalate or complain about poor behavior, or ultimately intervene. Furthermore, the question of impersonation is related to the ethics of lie: should the bots be allowed to pretend to be humans but be required to disclose their bot status if explicitly asked?

Another aspect related to ethics is the question of what cultural values are reflected in determining what kind of behaviour of bots should be considered ethical. Indeed, values of a software project that might be interested in adopting the bot, an ecosystem the project belongs to (e.g., "Python ecosystem") and the platform hosting the project or the ecosystem (e.g., GitHub/wikipedia) do not necessarily share the same cultural values.

Based on the aforementioned discussion, the seminar participants recommend that projects who use bots may adopt a *Manifesto of BOT ethics* that should include the following information:

- What behavior guidelines (e.g., contributor guidelines, code of conduct) are the bots required to follow?
- What rules pertain to bots impersonating humans:
  - Bots should not pose as humans
  - Or if asked directly it should respond
- What additional rules should the bots follow, e.g., what should the bots not do
  - Steal code
  - Modify with timeline/ project history
- What are the accountability rules for bots?
  - Who is accountable for the bot's actions?
  - Who is accountable/ have rights to materials created by Bots
  - When bots become autonomous and has a human persona, then what happens to its accountability?
- What cultural norms should the bots reflect in their behavior?

4.5 Diversity and inclusion

As diversity in software development teams is known to be beneficial for software development projects, the seminar participants discussed how bots can be used to encourage creation of a diverse and inclusive environment. During the discussion we identified several possible role for bots:

- Supporting newcomers. An example of a technical solution that can benefit newcomers would be a bot capable of identifying development tasks suited for newcomers or designing such tasks, e.g., by splitting more complex ones. Those microtasks might be better suited not only for novice developers but might also engage developers working on less powerful computing platforms such as mobile phones. Moreover, to facilitate onboarding bots can partner with newcomers working together on software development task: in this way bots can remove social stigma associated with asking for help. Finally, the bots can identify mentors.

- Support inclusive communication. Here bots can be used to implement a broad spectrum of actions, ranging from policing (e.g., checking for presence of toxic speech or code of conduct violations) to clarifying (e.g., flagging possible misunderstandings or clarifying communicative intentions), and monitoring the discussion to ensure that all voices are heard (e.g., reporting the percentage of the comments posted by the members of the dominant group).
- Support through personalisation: bots can be used to encourage self-reflection e.g., by applying mindfulness techniques, to deliver developers information in a way corresponding to their cognitive styles, or act as proxies of individual developers capable of communicating mental models to other bots/developers.

While the applications envisioned above focus on improving the software development process, bots can be also deployed to improve software products, e.g., by checking different facets of GenderMag[4].

At the same time, designers of bots should be aware that while aiming at encouraging diversity and inclusion, when not designed correctly bots might reinforce stereotypes and toxic cultures, appear patronizing or insincere, target minorities without changing community culture or increasing belonging, as well as shutting down communication channels by removing voices from the table that some people might find unsavory.

4.6 Bots to support collaboration

One important application of bots is to help developers collaborate effectively. Initially, several questions arose which made it difficult to understand the group's charter: What is a bot versus other sorts of tools, automation, scripts, etc.? Second, what do we mean by collaboration? For example, is the author of a library collaborating in some sense with a user who imports the library later, perhaps without the library author's knowledge? And third, is "collaboration support" too large a space to identify overarching research questions?

To make progress in the face of these threshold questions, the breakout identified three important dimensions along which such bots would likely deliver, and which might help distinguish importantly different types of bots and different research questions. There is a long and rich history in psychology about different forms of leadership, with two fundamentally different kinds of leaders providing very different ways of supporting group work. It was argued that while bots may not be group leaders, they could potentially help groups collaborate by supporting these two distinct functions. One dimension is Task-focused (helping the users complete the task faster/better) vs socio-emotional-focused (group well being, motivation, morale, cohesion). A second key dimension is collaboration in the large (e.g., Wikipedia, GitHub) versus collaboration in the small (people working at the same time and place). Finally, there seems to be an important distinction between "automation" agents that simply execute some standard task, versus dialog agents that interact with users to understand and be guided through some task, and support users through a natural language interface.

The group discussed how to understand the interactions between task focus versus socio-emotional focus for collaboration in the small versus collaboration in the large for dialog agents and for automation agents. The breakout concluded by identifying open questions and new ideas for bots.

[4] https://gendermag.org/

Open questions:

- Dialogue agents:
  - How to interpret user intent (opportunity: in SE this can be narrowed down to typical tasks in this area)
  - Bots may not be very generalizable (task-specific is too specific?)
  - Communication in the face of differences in vocabulary usage between subcommunities. That is, some commands / interactions may require specialized domain knowledge, that users may not have. How do bridge this gap?
- To what extent should we allow bots to act autonomously?
- To what extent do users trust bots?
- Rules of engagement – when should the bot jump in?
  - What's an appropriate role for a bot in a collaboration?
  - How to handle exceptions / unanticipated requests?
- Explainability / Transparency. Where are you now in the process? If you made an error, what were you trying?
- Bots are a socio-technical rather than technical system. Must be designed taking into account the human interaction
- How to coordinate code review checks (e.g., style guidelines) to maximize learning for PR submitter, rather than just providing a laundry list of things wrong with your contribution?
- How to **adapt to the user's background** (experience of the PR submitter)?

New Bot ideas:

- Canary releases: Bot goes through a checklist of things to check before releasing on a larger scale (building confidence)
- Ecosystem-level bots: Integrate information from the whole ecosystem (e.g., there is a new library available, how other people experienced the new library)
  - Which info to extract?
  - How to extract and aggregate it?
  - How to present it?
- Detect communication breakdown and intervene to prevent it. How can we design interventions to reduce the likelihood of these?

5 Follow up work

Following the vibrant discussions we had during the seminar, several collaborations were formed to continue research, such as the development of a software bot framework. Furthermore, a second edition of the BotSE workshop at ICSE is being planned for this May 2020 in South Korea[5]. Links to related documents from the seminar will be shared online. These links include more detailed notes from the breakout groups, bot design documents, a list of software bots to support software engineering and a bibliography.

[5] http://botse.org/

6 Overview of Invited Talks

6.1 Bots in Wikipedia – Brief review of selected research

Claudia Müller-Birn (Freie Universität Berlin, DE)

Over the last ten years, the Wikipedia community has gained manifold experiences in dealing with bots. They are primarily programmed to automate existing activities, for example, they inject data into Wikipedia content from public databases, monitor and curate Wikipedia content, extend Wikipedia's functionality, or protect from malicious activity. Wikipedia operates with a system of algorithmic governance that describes the interdependency between human user, bots and the technical infrastructure. From these experiences, we can provide a set of guidelines for the design or usage of bot in other settings, such as software engineering. Bots, for example, can be identified as such in Wikipedia. The community has developed a bot policy which is regularly adapted to the changing needs. Moreover, the community implemented a public approval procedure for bots. In summary, bots are always part of a social system, therefore should be treated as a socio-technical system and, therefore, designed as such.

6.2 Overview of Natural Language Dialogue Systems

David R. Traum (USC – Playa Vista, US)

Dialogue is defined as communication including multiple contributions, coherent interaction and multiple participants. An overview is presented of common types of automated dialogue systems that communicate with people in natural language. The most common types are task-oriented assistants and social chat, but also role-play of human roles and other types of systems have been built. A number of examples of different roles and systems filling these roles were presented as well as common dialogue system architectures. Finally research topics and resources for the area were introduced.

6.3 Highlights of the first International Workshop on Bots in Software Engineering (BotSE)

Emad Shihab (Concordia University – Montreal, CA) and Stefan Wagner (Universität Stuttgart, DE)

We organized the first workshop in the area of bots for software engineering as well as engineering bots this may in Montreal, Canada, co-located with ICSE. We reported on the workshop in general, the presentation topics as well as the discussions. We were impressed by the breadth of topics and noted that there was still an ongoing discussion about what a bot in SE is. There will be a BotSE again next year with ICSE.

6.4 Bots – The hidden side of software development

Bogdan Vasilescu (Carnegie Mellon University – Pittsburgh, US)

As automation agents, bots have become popular with the advent of the DevOps movement. In trying to maintain software quality and improve developer productivity while building software at a faster and faster pace, a myriad of automation agents have emerged; for continuous integration, testing, code coverage analysis, or dependency management, just to name a few. How do software engineers use such bots? Are the bots effective? Can we detect in archival data that bots are being used? And can we use such data to empirically study the effects of using bots?

In this talk I go over some recent results from my research group, focusing on how to detect bots in data from open-source software repositories, and what impact the introduction of automation agents has had on project outcomes. I also go over a recent experiment with a bit that answers programming questions automatically on Stack Overflow.

Participants

- Shivali Agarwal
IBM India – Bangalore, IN
- Ireti Amojo
Freie Universität Berlin, DE
- Ivan Beschastnikh
University of British Columbia – Vancouver, CA
- Kelly Blincoe
University of Auckland, NZ
- Nick Bradley
University of British Columbia – Vancouver, CA
- Fabio Calefato
University of Bari, IT
- Nathan Cassee
Eindhoven University of Technology, NL
- Jacek Czerwonka
Microsoft Research – GRedmond, US
- Antske Fokkens
Free University Amsterdam, NL
- Denae Ford
Microsoft Research – Redmond, US
- Thomas Fritz
Universität Zürich, CH
- Marco Gerosa
Northern Arizona University – Flagstaff, US
- Daniel Graziotin
Universität Stuttgart, DE
- Sonia Haiduc
Florida State University – Tallahassee, US
- James D. Herbsleb
Carnegie Mellon University – Pittsburgh, US
- Abram Hindle
University of Alberta – Edmonton, CA
- Akinori Ihara
Wakayama University, JP
- Minha Lee
Eindhoven University of Technology, NL
- Philipp Leitner
Chalmers University of Technology – Göteborg, SE
- Marin Litoiu
York University – Toronto, CA
- Walid Maalej
Universität Hamburg, DE
- Christoph Matthies
Hasso-Plattner-Institut – Potsdam, DE
- Marie-Francine Moens
KU Leuven, BE
- Martin Monperrus
KTH Royal Institute of Technology – Stockholm, SE
- Claudia Müller-Birn
Freie Universität Berlin, DE
- Nicole Novielli
University of Bari, IT
- Ayushi Rastogi
TU Delft, NL
- Paige Rodeghero
Clemson University, US
- Carolyn Penstein Rosé
Carnegie Mellon University – Pittsburgh, US
- Anita Sarma
Oregon State University – Corvallis, US
- Andreas Schreiber
German Aerospace Center – Köln, DE
- Alexander Serebrenik
Eindhoven University of Technology, NL
- Emad Shihab
Concordia University – Montreal, CA
- Arfon Smith
STSci – Baltimore, US
- Igor Steinmacher
Northern Arizona University – Flagstaff, US
- Margaret-Anne Storey
University of Victoria, CA
- David R. Traum
USC – Playa Vista, US
- Christoph Treude
University of Adelaide, AU
- Bogdan Vasilescu
Carnegie Mellon University – Pittsburgh, US
- Stefan Wagner
Universität Stuttgart, DE
- Mairieli Wessel
University of Sao Paulo, BR
- Jie Zhang
University College London, GB
- Thomas Zimmermann
Microsoft Corporation – Redmond, US