Editor: **Christof Ebert**
Vector Consulting Services
christof.ebert@vector.com

Software Bots

Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky

From the Editor

Bots have become a common user interface for software services. Many people prefer bots to real persons owing to bots' perceivable passionate "personality." The Turing test obviously has been passed. However, bots' involvement in social networks and fake emails in elections exposed major risks. A lot still must be done, including some indication that "you're now talking with a bot." Here, Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky discuss current bot technology and present a practical case study on how to use bots in software engineering. I look forward to hearing from both readers and prospective authors about advances in software technologies. —*Christof Ebert*

FROM COMPUTER PROGRAMS' earliest days, people have dreamed about programs that act, talk, and think like humans. Such programs could not only automate tasks that humans perform but also work with humans to solve intellectual tasks that can't be entirely automated. Even as far back as 1966, the hope was for these programs to pass the Turing test,[1] in which humans are fooled into believing they're interacting with an intelligent human rather than a mere program.

The terms "chatbot," "chatterbot," and "bot" were interchangeably used to describe the realization of this vision quite early on. But now, they refer mostly to a conversational-style UI, an anthropomorphized script, or an agent that automates rote and tedious tasks. Bots typically aren't intended to fool users into believing they're interacting with a real person, but many bots do have a pleasant, engaging personality.

Bots typically reside on platforms on which users work or play with other users. They also frequently integrate secondary services into communication channels, providing a conduit between users and other tools. They might fetch or share information, extract and analyze data, detect and monitor events and activities in communication and social media, connect users with each other or with other tools, or provide feedback and recommendations on individual and collaborative tasks.

Here, we discuss types of bots, describe some platforms for working with them, and offer guidelines on how to create and use them.

Bots and Software Development

Bots are rapidly becoming a de facto interface for interacting with software services. This is due partly to the widespread adoption of messaging platforms (for example, Facebook Messenger for social networking and Slack for developer communication), and partly to the advancement of natural-language processing, which many bots leverage. But another driver is the prevalence of big data, along with machine-learning algorithms for analyzing data across many domains. Bots provide a convenient way for developers to generate a UI for interacting with these algorithms and data.

Major software companies are recognizing the value bots bring in terms of integrating services, users,

 0740-7459/18/$33.00 © 2018 IEEE

and communication channels. Facebook aims to "replace apps" one bot at a time in its messaging platform,[2] while Microsoft claims that "conversation as a platform" is the OS of the future.[3]

Alexa, Siri, IBM Watson, and Google Now all support this shift toward bots. There are also many bots in the platforms software developers use to connect with other developers and services, such as Slack, Microsoft Teams, and HipChat.

The transition from command-line interfaces to interacting with bots through messaging tools feels intuitive to most developers. We see new examples of sophisticated and innovative bots stemming from developer's needs;[4] these examples are paving the way for bots in other domains.

This situation is inspiring the development of bots for users, who are spending increased time in messaging applications and are embracing bots as an alternative to installing and relying on external apps. However, bot developers must carefully consider not only where to host bots and how to create them but also when not to use them.

Botology: Understanding the Bot Landscape

Although the development and widespread adoption of bots have occurred in just a few years, what's truly surprising are the diverse tasks and roles bots have taken on. Instead of trying to narrowly define bots or chatbots, we embrace their diversity and look for ways to characterize their distinct features.

One way to characterize bots is through their interaction model. Some bots support a domain-specific language in which users interact with bots using specific commands in a command-line interface. Other bots might parse natural language through text or speech. These bots might also embed rich UI controls in a platform that lets users respond quickly.

Bots can support a pull-based approach in which users initiate interaction with them (for example, a user invokes the bot using commands such as "Hey Siri"). They can also support a push-based approach in which they initiate the interaction on the basis of some system or user context.

Another way to characterize bots is in terms of their intelligence:

- *Adaptation*. Some bots are context-aware and might use that context to change how they interact with users.
- *Reasoning*. Some bots follow simple logic rules; others use more advanced AI to drive their behavior.
- *Autonomy*. Some bots are entirely autonomous, some rely on human input before acting, and others use a mixed approach.

Finally, we can characterize bots according to their purpose:

- *Generalist bots* such as Siri or Cortana support a range of simple tasks and direct users to appropriate external resources when deeper knowledge is required.
- *Transactional bots* work on the users' behalf, automatically executing transactions with external systems (for example, automatically making a purchase when a price level is reached).
- *Informational bots* fetch information for users (for example, gathering insights about stocks or providing weather updates).
- *Productivity bots* improve user or team productivity by automating rote or tedious tasks (for example, updating calendars or silencing notifications).
- *Collaboration bots* help users communicate, coordinate, and collaborate (for example, connecting the right people at the right time).

The sidebar provides examples of how developers use bots to support their work.

Creating and Hosting Bots

Although simple bots can be built from scratch and self-hosted, many developers leverage third-party frameworks to streamline creation and distribution. With the explosion of new tools for bot development, we need to distinguish between the tools for building bots (creation platforms) and the platforms on which the bots dwell (distribution platforms).[5]

Companies such as Microsoft and Facebook offer comprehensive tooling to support bot creation and distribution. Other companies provide customized resources for specific creation and distribution tasks. Table 1 lists some common bot platforms and the creation and distribution services they use, along with other bot technologies.

Creation Platforms

Creation platforms provide a variety of software foundations, frameworks, toolkits, APIs, and other advanced features (for example, natural-language processing, search, and image processing). These platforms can be distribution-platform-specific or produce bots that are deployable across multiple platforms, such as the

DEVELOPMENT BOTS IN ACTION

Software developers have been early adopters and proponents of bots because they've recognized bots' potential for both enhancing individual and team productivity and significantly improving software quality. Chatbots bring awareness and transparency to the communication channel and enable nontechnical team members to engage with operations without needing domain expertise (for example, by using a bot for deployment).

For example, Sendwithus (sendwithus.com), a company that provides transactional email services, has adopted bots as part of software development. With 25 employees and two office locations (Victoria, Canada, and San Francisco), Sendwithus uses Slack as the central hub for its communications and DevOps, and bots fulfill several development tasks. For example, the bots

- set reminders for team members (using Slackbot);
- manage and coordinate customer support and help-desk tickets (using Help Scout; www.helpscout.net);
- facilitate real-time communication with users visiting the Sendwithus website (using Olark piped into Help Scout; www.olark.com);
- manage physical communication (for example, through short message service and phone) and send messages to the company's communications hub (using Twilio; www.twilio.com);
- accommodate custom tasks (for example, ordering a team's lunch via a message on Slack) (using Hubot; hubot.github.com);
- automatically share resources and documents (for example, screenshots) within the team (using Dropbox and Google Drive);
- manage and track code migration and deployment from within Slack, raising team awareness when this happens, because this might cause errors during migration (using a customized bot with Heroku integration);
- monitor and index runtime warnings, errors, and exceptions (using Papertrail; papertrailapp.com);
- notify the appropriate team members when errors and exceptions occur (using Sentry; sentry.io/welcome); and
- track and aggregate the service status of other service providers the company depends on (using StatusPage; statuspage.io).

Developers can expect to see more and more bots being introduced to support their workflow and development-related activities. In addition, companies such as Sendwithus are recognizing their dependence on Slack and the bots that perform many of their daily tasks, which have become crucial to the running of the company.

Microsoft Bot Framework, Botkit, and Pandorabots.

The provided services range from documentation and code templates to no-code-required bot-building interfaces such as Chatfuel. Many popular creation platforms also have vibrant developer ecosystems—developers can connect with these online communities to obtain expertise in the form of tutorials, articles, discussions, and support. Other general bot development communities, such as Botmaker's Slack group and the *Chatbot Magazine* community, are hotbeds for discussions on a variety of bot-related topics.

Distribution Platforms

Distribution platforms dictate where and how users access bots; many center around messaging or social networking (for example, Messenger, Skype, and WeChat). Other platforms are domain-specific channels for developers (for example, Slack, Teams, and HipChat). These platforms support human–bot, bot–bot, or even system–bot interactions.

Selecting the right distribution platform can benefit developers in a variety of ways. Some platforms provide access to a user base. Launching a bot on an existing platform gives developers a head start in overcoming the cold-start user problem many new applications face. Developers should consider not only the size of the platform's user base but also the general user demographics and the access costs for users. These platforms define and standardize how users interact with bots and have built-in support for commands, natural language, speech, and even rich UI controls. The method of interacting with a bot strongly influences the user's experience and the types of tasks users can perform.

Many distribution platforms offer mechanisms for users to discover and try out new bots. Like Apple's

Table 1. Features of popular bot creation and distribution services.*

Platform		Distribution services: User base: Size	Distribution services: User base: Cost for users	Distribution services: Interaction mechanisms	Distribution services: Discovery	Distribution services: Monetization (bots collecting payment from users)	Creation services: Software foundations: Cost or license	Creation services: Software foundations: Tools or technologies	Creation services: Software foundations: Platforms	Creation services: Developer community
Slack (slack.com)		6 M daily & 2 M paid users (2017)	• Free • Paid	• Text • Commands • GUI	App directory	No	Free	• Web API • RTM API • App blueprints	Slack	Yes
Microsoft	Teams (teams.microsoft.com	125 K teams (2017)	Paid	• Text • Commands • GUI	Bot directory	No	Microsoft Bot Framework			
							OSS	• Bot Builder • Bot Connector • Service integrations • Azure Bot Service • .NET (C#) SDK • Node.js REST SDK	Multiple (Bot Framework Portal)	Yes
	Skype (www.skype.com)	3 M monthly users (2017)	• Free • Paid	• Speech • Text • Commands • GUI	Bot directory	Yes				
HipChat (www.hipchat.com)		90 K paid users (2017)	Paid	• Text • Commands • GUI	Marketplace	?	No	No	No	No
Messenger (www.messenger.com)		1.2 B monthly users (2017)	Free	• Text • Commands • GUI	• Facebook pages • Bot Explorer • QR codes	Yes	Free	• WebHooks • APIs • Design Kit • Wit.ai • RTM	Facebook	No
WeChat/Weixin (web.wechat.com)		889 M monthly users (2016)	Free	• Speech • Text • Commands • GUI	QR codes	Yes	No	No	No	No
Telegram (telegram.com)		100 M monthly users (2017)	Free	• Text • Commands • GUI	Bot store	Yes	OSS	• Bot API • Telegram API • BotFather	Telegram	No
Kik (www.kik.com)		15 M monthly users (2017)	Free	• Text • Commands • GUI	• Bot shop • QR codes	No	Free	API	Kik	No
Other technologies		**Bot-hosting platforms** • Alexa • Echo • Cortana • Line • Android • Discord • Cisco Spark • Messages • Viber • Intercom • Google Allo • Twilio • SMS • Web		**Extending interactions** • Google Cloud • Watson Conversation • Alexa Voice • Luis.ai • MindMeld	**Bot directories** • BotList • ChatBottle • Botwiki	**Payment SDKs** • PayPal • Stripe	**Bot-building tools** • Api.ai • Pandorabots • Chatfuel • Rebotify • Botkit • Gupshup • OnSequel • Flow XO • Botsify • BotMock • BotMan			**Online communities** • Botmakers • Chatbots.org • chatbotsmagazine.com • Stack Overflow

**The table's bottom row lists additional bot development technologies, by category. This list isn't comprehensive; it's just a sampling of popular technologies for bot builders to leverage. RTM = Real Time Messaging, OSS = open source software, REST = Representational State Transfer, SDK = software development kit, and SMS = Short Message Service.*

App Store, some platforms offer virtual "bot stores" where users browse for bots. Third-party sites (for example, BotList and ChatBottle) also provide online catalogs of bots for many popular distribution platforms, making it easier for developers to promote and market their bots. Mature distribution channels have monetization features that let bots safely collect payments from users, which is particularly useful for people developing transactional-style bots.

Insights on Creating and Using Bots

Bots are rapidly becoming pervasive: we interact with them in cars, at home, in entertainment devices, and at work. And, as we discuss in the sidebar, bots play a sophisticated and increasingly significant role in software development projects. We need to learn from these early adoption experiences to find out not only what works well but also what might go wrong. Here are insights we've gained from our research that developers should consider when creating and using bots.

Amplification Doesn't Replace Collaboration

Bots are frequently used in group or collaborative settings to automate tasks that normally require human interaction. But removing collaboration opportunities can hamper creativity and productivity. Rather than replacing collaboration, bots can be used to reduce friction in communication or task coordination. For example, bots could provide transparency on task progress, make team goals more visible, link experts with novices, and build team trust and cooperation.[6] However, the poor design or overuse of bots might lead to information and interruption overload—issues bot designers must watch out for.

Users Should Always Know What to Expect

In contrast to the Turing bots proposed in the 1950s, bot developers should make it clear to users that they're interacting with a bot, not another human. Similarly, if a bot passes control to a human (for example, when the bot can't understand a command or answer a question), the user should be aware of this, and the handover should be graceful and transparent. This ensures that users don't lose trust in the systems supported by bots and that users understand why control is being passed to them. Also, the bot's purpose—what it can and can't do—must be evident and match user expectations.

User–Bot Interaction Should Be Optimized

Ideally, users' interactions with bots should be smooth and frictionless. This is achievable if designers carefully plan conversational flows, especially for conversational bots. For example, bots might need to repeat commands so that users know that the bots are listening. Bots must also be able to detect dead ends in conversations and prompt users by giving hints on how to continue interaction.

Some bots could incorporate UI elements to reduce the number of user clicks required and to make the conversation more efficient. Implementing global input checks for common navigational keywords (for example, help, back, cancel, start over, and exit) can help avoid the creation of "stubborn" or "clueless" bots.[7] Tools such as BotMock can help you prototype the user's "journey through your bot." Finally, many platforms have specific guidelines for bots to follow; for example, Facebook recommends that its bots follow a set of simple conversational guidelines.[8]

Personality Matters More Than Looks

Because bots are predominantly text based, how they use language—and even how they're named—can influence users' perceptions of their personality, role, and capability. This might seem surprising given that users should know they're interacting with a mere program. However, early research has shown that a bot's personality changes how users interact with it. Even if a bot can effectively accomplish a user's tasks, people won't adopt it if they find it boring. The bot's language should be casual, accessible, friendly, and fun. Bots should expect users to test their abilities and respond accordingly.

But too much personality might not be a good thing, either. According to Slack, "a little goes a long way."[9] Done right, bots can accentuate a company's brand or communicate a company culture (such as playfulness).

Bots Should Do No Harm

Isaac Asimov's three laws of robotics state that robots must not harm humans, must obey orders, and must protect themselves.[10] These rules can also apply to software bots. But perhaps, with the complexity and rapid growth of bots in our software ecosystems, these rules are too simple. Simple mistakes could have devastating effects, and protecting a user's privacy might be difficult. We can expect to see a code of ethics for human–bot interactions in the near future. (One organization working on this is the Partnership on AI; www.partnershiponai.org.)

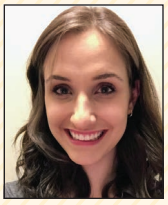
In the meantime, developers should carefully consider how their bots might be misused, intentionally or unintentionally. Many bots are already seen as malicious, so building user trust might also pose a challenge. For now, many bot creation and distribution platforms provide basic principles or best practices for bot design (for example, the platforms from Microsoft,[7] Slack,[9] and Facebook[8]). But developers wishing to use bots or create them for users must be careful which bots they bring to life.

We anticipate a rapid increase in developers using and creating bots. Indeed, Joel Splosky of Stack Overflow claims that "developers are writing the script for tomorrow"[11] by providing insights, through their ability to innovate and lead by example, into how technology can be used in other domains. Developers have the unique opportunity to not only use bots in their knowledge work but also lead the way on how bots can solve both their and others' needs.

References

1. A.M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, 1950, pp. 433–460; www.jstor.org/stable/2251299.
2. M. Murgia, "Can Facebook Messenger Kill Off Apps?," *The Telegraph*, 15 Nov. 2015; www.telegraph.co.uk/technology/facebook/11996896/Can-Facebook-Messenger-kill-off-apps.html.
3. "Conversation as a Platform," video, *Channel 9*, 25 Mar. 2016; channel9.msdn.com/Events/Build/2016/C902.
4. M.-A. Storey and A. Zagalsky, "Disrupting Developer Productivity One Bot at a Time," *Proc. 24th ACM SIGSOFT Int'l Symp. Foundations of Software Eng.* (FSE 16), 2016, pp. 928–931.
5. M. Vakulenko, "Messenger vs. Skype vs. Slack vs. Telegram: How to Spot the Winners," *Mobile Lifestyle*, 6 Apr. 2016; medium.com/mobile-lifestyle/messenger-vs-skype-vs-slack-vs-telegram-how-to-spot-the-winners-adc34b4ca066.
6. C. Lebeuf, M.-A. Storey, and A. Zagalsky, "How Software Developers Mitigate Collaboration Friction with Chatbots," presentation at Talking with Conversational Agents in Collaborative Action Workshop, 2017 Conf. Computer-Supported Cooperative Work and Social Computing (CSCW 17), 2017.
7. "Principles of Bot Design," Microsoft, 4 Aug. 2017; docs.microsoft.com/en-us/bot-framework/bot-design-principles.
8. "Designing for Messenger," Facebook, 2017; developers.facebook.com/docs/messenger-platform/design-resources.
9. "Voice and Tone: Communicating for Clarity," *Building Great User Experiences on Slack*, Slack, 2017; api.slack.com/best-practices/voice-and-tone.
10. I. Asimov, "Runaround," *Astounding Science-Fiction*, Mar. 1942, pp. 94–103.
11. J. Spolsky, "Developers Are Writing the Script for the Future," blog, 9 Dec. 2016; www.joelonsoftware.com/2016/12/09/developers-are-writing-the-script-for-the-future.

ABOUT THE AUTHORS



CARLENE LEBEUF is a master's student in computer science and a researcher at the University of Victoria. Contact her at clebeuf@uvic.ca.

MARGARET-ANNE STOREY is a professor of computer science at the University of Victoria. Contact her at mstorey@uvic.ca.

ALEXEY ZAGALSKY is PhD candidate in computer science at the University of Victoria. Contact him at alexeyza@uvic.ca.