# A New Generation of Software Bots:

# How Developers Can Create and Use Them

Carlene Lebeuf ([Clebeuf@uvic.ca)](mailto:Clebeuf@uvic.ca))

Margaret-Anne Storey ([mstorey@uvic.ca)](mailto:mstorey@uvic.ca))

Alexey Zagalsky (alexeyza@uvic.ca)

Chistof Eilbert (?)

## The Emergence of Bots

From the earliest days of computer programs, people have dreamed about programs that act, talk, and think like humans. Such programs could not only automate tasks that humans perform, but they could also work with humans to solve intellectual tasks that cannot be entirely automated. Even as far back as 1966, the hope was for these programs to pass the Turing Test (proposed in the 1950 paper Computing Machinery and Intelligence Turing-1950 [1]), where humans are fooled into believing they are interacting with an intelligent human rather than a mere program.

The terms “chatbots”, “chatterbot”, and “bot” were interchangeably used to describe the realization of this vision quite early on, but now they mostly refer to a conversational-style user interface, an anthropomorphized script, or an agent that automates rote and tedious tasks. Bots are not typically intended to fool end users into believing they are interacting with a real person, but many bots do have a pleasant and engaging personality.

Bots typically reside on platforms where users work or play with other users, and they frequently integrate secondary services into communication channels, providing a conduit between users and other tools. They may 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 they may provide feedback and recommendations on individual and collaborative tasks.

Bots are rapidly becoming a *de facto* interface for interacting with software services. This is partly due to the widespread adoption of messaging platforms (e.g., Facebook Messenger for social networking, Slack for developer communication), and partly because of 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 user interface for interacting with these algorithms and data.

Major software companies have started recognizing the value that bots bring in terms of integrating services, users, and communication channels. Facebook aims to ``replace apps'' one bot at a time in their messaging platform [2], while Microsoft claims that ``conversation as a platform'' is the operating system of the future [3].

Alexa, Siri, IBM Watson, and Google Now are all platforms that support this shift towards 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 that are paving the way for bots in other domains.

They are providing the inspiration for developing bots for end users, who are spending an increased amount of time in messaging applications and are openly embracing bots as an alternative to installing and relying on external apps. However, bot developers need to carefully consider where to host bots and how to create them, as well as when not to use bots.

## Botology: Understanding the Bot Landscape

While the development and widespread adoption of bots has occurred in just a few years, what is truly surprising is the diversity of tasks and roles taken on by bots. Rather than attempting to narrowly define bots or chatbots, instead we embrace their diversity and look at ways of characterizing their distinct features.

One way to characterize bots is through the **interaction model** they provide. Some bots support a domain specific language where users interact with the bot using specific commands within a command line interface, while other bots may parse natural language through text or speech.

These bots may also embed rich user interface controls in a platform that allows users to respond quickly.

Bots can support a pull-based approach where users initiate interactions with the bot (e.g., a user invokes the bot using commands such as ``Hey Siri''), or a push-based approach where bots initiate the interaction based on some system or user context.

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

* *Adaptation:* Some bots are ``context aware'' and may use that context to change how it interacts with users.
* *Reasoning:* Some bots follow simple logic rules, whereas other bots use more advanced Artificial Intelligence (AI) to drive their behavior.
* *Autonomy:* Some bots are entirely autonomous, other bots rely on human input before acting, and others use a mixed approach.

Finally, bots can be characterized according to their **purpose**:

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

In the sidebar, we provide examples of how bots are used by developers to support their work.

Joel Splosky of Stack Overflow claims: “developers are writing the script for tomorrow” [4].

## Creating and Hosting Bots

Although simple bots can be built from scratch and self-hosted, many developers choose to leverage third-party frameworks to streamline the creation and distribution process. With the explosion of new tools in the bot development domain, we need to distinguish between the tools used to build bots (creation platforms) and the platforms where the bots dwell (distribution platforms).

Companies such as Microsoft and Facebook offer comprehensive tooling to support both the creation and distribution of bots. Other companies provide customized resources for specific bot creation and distribution processes.



**Figure 1.** An overview of features for popular bot creation and distribution services. The final row of the table contains a collection of additional bot development technologies, sorted by category. This is not a comprehensive list of all possible platforms, but rather a sample of popular technologies for bot builders to leverage.

### Distribution Platforms

Distribution platforms are online ecosystems where bots reside and function. These platforms dictate where and how bots are accessed by end users, and many center around messaging or social networking (e.g., Messenger, Skype, WeChat). Other platforms are domain specific channels (e.g., Slack, Teams, and HipChat) aimed at software developers. 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 provide access to an existing user base [5] --- 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 not only consider the size of the platform’s existing user base, but also the general user demographics and costs for a user to access it. These distribution platforms define and standardize how users can interact with bots, and such platforms 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 as well as the types of tasks that can be performed.

Many distribution platforms offer discovery mechanisms with different ways for users to discover and try out new bots. Like Apple's App-Store, some platforms offer virtual ``bot stores'' where users browse for bots. Third-party sites (e.g., BotList, 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 also have monetization features with mechanisms that allow bots to safely collect payments from users, a particularly useful feature when developing transactional-style bots.

### Creation Platforms

Creation platforms support the design and development of bots, as well as provide a variety of software foundations, frameworks, toolkits, APIs, and other advanced features (e.g., natural language processing, search, image processing). Bot creation platforms may be distribution platform-specific or produce bots that can be deployed across multiple platforms, such as the Microsoft Bot Framework, BotKit, and PandoraBots.

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

## Thoughts on Bots

Bots are rapidly becoming pervasive: we interact with them in cars, at home, in entertainment devices, and at work, and as we discussed 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 what works well, but also what may go wrong. Here are some insights we’ve gained from our research that developers should consider when creating and using bots.

### To Bot or Not: Amplification Doesn't Replace Collaboration

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

### Bot or Human? The User Should Always Know What to Expect

Unlike the Turing bots proposed back in the 1950s, bot developers should make it clear to users that they are interacting with a bot rather than speaking with another human. Similarly, if a bot passes control to a human (e.g., when the bot cannot understand a command or answer a question), the user should be aware of this and the handover should be handled gracefully and transparently. This ensures users do not lose trust with the systems supported by bots and understand why control is being passed to a human. It is also important that the purpose of the bot --- what it can and can't do --- be evident and match user expectations.

### Bot Talk: How to Talk to and Be Heard by Bots

Ideally, users' interactions with bots should be smooth and frictionless. This can be achieved if designers carefully plan potential conversational flows, especially for conversational bots. For example, bots may need to repeat commands so the user knows the bot is listening, and bots need to be able to detect dead ends in a conversation and prompt the user by giving hints on how to continue the interaction. Some bots may 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 (help, back, cancel, start over, exit) can help avoid the creation of “stubborn” or “clueless” [7]. Tools like BotMock can be used to prototype the user's “journey through your bot”. Finally, many platforms have specific guidelines one can follow, e.g., Facebook recommends their bots follow a set of simple conversational guidelines [7].

### Personality Matters More Than Looks

As bots are predominantly text based, how a bot uses language --- even how it is named --- can influence a user’s perception of its personality, role, and capability. This may seem surprising given that users should know they are interacting with a mere program, but early research has shown that a bot’s personality changes how users interact with them. Even if a bot can effectively accomplish a user's tasks, people won't adopt the bot if they find the bot boring. The choice of language should be casual, accessible, friendly, and fun. Bots should expect users to test their abilities and respond accordingly. But too much personality may not be a good thing either. According to Slack [8], ``a little goes a long way''. Done right, bots can accentuate a company's brand and enhance culture.

### Bots Should Do No Harm

Bots, like robots, should do no harm to users. Asimov's laws for robots [9] state that robots shall not harm humans, robots will obey orders, and robots will protect themselves, which can also apply to software bots. But perhaps these rules are too simple with the complexity and rapid growth of bots in our software ecosystems. Simple mistakes could have devastating effects and a user's privacy may be hard to protect. We can expect to see a code of ethics for human-bot interactions in the near future [10], but in the meantime, developers should carefully consider how their bots may be misused, either intentionally or unintentionally. Many bots are already seen as malicious, thus building trust with users may also pose a challenge. For now, many of the platforms that support the creation and distribution of bots also provide developers with basic principles or best practices for bot design (e.g., Microsoft [7], Facebook [11], Slack [8]). But developers wishing to use or create bots for their end users need to be careful which bots they bring to life.

## REFERENCES

[1] A. M. Turing, “Computing machinery and intelligence,” Mind, vol. 59, no. 236, pp. 433–460, 1950. [Online]. Available: http://www.jstor.org/stable/2251299

[2] M. Murgia, "Can Facebook Messenger kill off apps?", *Telegraph.co.uk*, 2017. [Online]. Available: http://www.telegraph.co.uk/technology/facebook/11996896/Can-Facebook-Messenger-kill-off-apps.html.

[3] "Conversation as a Platform", *Channel 9*, 2017. [Online]. Available: https://channel9.msdn.com/Events/Build/2016/C902.

[4] "Developers are Writing the Script for the Future", *Joel on Software*, 2017. [Online]. Available: https://www.joelonsoftware.com/2016/12/09/developers-are-writing-the-script-for-the-future/.

[5] "Messenger vs Skype vs Slack vs Telegram: How to spot the winners", *Medium*, 2017. [Online]. Available: https://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,” Talking with Conversational Agents in Collaborative Action workshop at the Conference on Computer-Supported Cooperative Work and Social Computing (CSCW), 2017.

[7] "Principles of bot design", *Docs.microsoft.com*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/bot-framework/bot-design-principles.

[8] "Building great user experiences on Slack", *Slack Developer Tutorials*, 2017. [Online]. Available: https://api.slack.com/best-practices.

[10] "Home | Partnership on Artificial Intelligence to Benefit People and Society", *Partnership on Artificial Intelligence to Benefit People and Society*, 2017. [Online]. Available: https://www.partnershiponai.org/.

[11] "Design Resources - Messenger Platform - Documentation - Facebook for Developers", *Facebook for Developers*, 2017. [Online]. Available: https://developers.facebook.com/docs/messenger-platform/design-resources.

[9] I. Asimov, “The evitable conflict,” Astounding Science Fiction, vol. 45, no. 4, pp. 48–68, 1950.

[12] M.-A. Storey and A. Zagalsky, “Disrupting developer productivity one bot at a time,” in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016, pp. 928–931.

|  |
| --- |
| Development Bots in Action Software developers have been early adopters and proponents of bots as they recognized their potential for enhancing individual and team productivity as well as significantly improving software quality. Chatbots bring awareness and transparency within the communication channel and enable non-technical team members to engage with operations without needing domain expertise (e.g., using a bot to deploy).  Sendwithus, a company that provides transactional email services, described how they've adopted bots as part of their software development process. With 25 employees and two office locations (Victoria and San Francisco), the company uses Slack as the central hub for their communications and Dev-Ops, and bots fulfill several development tasks. For example, their bots provide support by:   * Setting reminders for themselves or other team members [Slackbot] * Managing and coordinating customer support and helpdesk tickets [Help Scout](https://www.helpscout.net) * Facilitating real-time communication with users visiting their website [Olark piped into Help Scout](https://www.olark.com/) * Managing physical communication (e.g., SMS and Phone) and sending messages to the company's communications hub---Slack [Twilio](https://www.twilio.com/) * Accommodating custom tasks (e.g., ordering a team's lunch via a message on Slack) [Hubot](https://hubot.github.com/) * Automatically sharing resources and documents within the team (e.g., screenshots) [Dropbox & Google drive] * Managing and tracking code migrations and deployments from within Slack, raising team awareness when this happens (as this may cause errors during migrations) [Customized bot with the Heroku integration] * Monitoring and indexing runtime warnings, errors, and exceptions [Papertrail](https://papertrailapp.com/) * Notifying the appropriate team members when errors and exceptions occur [Sentry](https://sentry.io/welcome/) * Tracking and aggregating the service status of other service providers the company depends on [statuspage.io]   Developers can expect to see more and more bots being introduced to support their workflow and development related activities. However, companies like Sendwithus also recognize their dependability on Slack and the many tasks done by bots, which have become crucial to the running of the company. |