



## **The Lion, the Ecologist and the Plankton: A Classification of Species in Multi-bot Ecosystems**

Downloaded from: <https://research.chalmers.se>, 2026-03-16 12:07 UTC

Citation for the original published paper (version of record):

Platis, D., Erlenhov, L., Gomes, F. (2024). The Lion, the Ecologist and the Plankton: A Classification of Species in Multi-bot Ecosystems. FSE Companion - Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering: 482-486.  
<http://dx.doi.org/10.1145/3663529.3663782>

N.B. When citing this work, cite the original published paper.



# The Lion, the Ecologist and the Plankton: A Classification of Species in Multi-bot Ecosystems

**Dimitrios Platis**  
Zenseact  
Gothenburg, Sweden  
dimitrios.platis@zenseact.com

**Linda Erlenhov**  
Chalmers University of Technology  
and the University of Gothenburg  
Gothenburg, Sweden  
linda.erlenhov@chalmers.se

**Francisco Gomes de Oliveira Neto**  
Chalmers University of Technology  
and the University of Gothenburg  
Gothenburg, Sweden  
gomesf@chalmers.se

## ABSTRACT

The vast majority of state-of-the-art and practice have, so far, focused on understanding and developing individual bots that support software development (DevBots), while the interactions and collaborations between those DevBots introduce intriguing challenges and synergies that can both disrupt and enhance development cycles. In this vision paper we propose a taxonomy for DevBot roles in an ecosystem, based on how they interact. Much like biology, DevBots ecosystems rely on a balance between the creation, usage and maintenance of DevBots, particularly, on how they depend on one another. Further we contribute with reflections on how these interactions affect multi-bot projects.

## CCS CONCEPTS

• **Software and its engineering** → *Software maintenance tools*; • **Human-centered computing** → *Empirical studies in collaborative and social computing*.

## KEYWORDS

Empirical Software Engineering, Software Development Bots, Bots in Software Engineering, DevBots

### ACM Reference Format:

Dimitrios Platis, Linda Erlenhov, and Francisco Gomes de Oliveira Neto. 2024. The Lion, the Ecologist and the Plankton: A Classification of Species in Multi-bot Ecosystems. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE Companion '24)*, July 15–19, 2024, Porto de Galinhas, Brazil. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3663529.3663782>

## 1 INTRODUCTION

In contemporary software projects a growing number of bots that support software development (DevBots) are added into the software development processes. This affects the number of connections between the DevBots and their surrounding environment, such as project resources and humans. Furthermore, the recent and ongoing increase in number of large language models (LLMs)

utilized by applications and DevBots further contributes to a complicated environment.

When multiple bots are being employed in a project many roles, interdependencies and patterns are formed which can lead to competing or redundant tasks/artefacts operated by those bots. For instance, a previous study reports on an incident where participants lost track of tasks because they had two bots with competing roles; while Bot A created and opened issues, Bot B closed them due to lack of human activity [3]. Based on an investigation of one multi-bot project for a software stack that enables autonomous driving, we propose a taxonomy that reflects these ecosystems.

Our vision is that our taxonomy will help practitioners to identify and prevent bot interactions that are disruptive to software development and, consequently, improve the design and maintenance of DevBots deployed in multi-bot projects.

## 2 RELATED CLASSIFICATION SYSTEMS

Two related research areas to bot ecosystems are multi-agent systems and Software Ecosystems. In the -90s, the precursors to today's bots were investigated under the term agents and research on cooperation between these agents was conducted. In a system of agents, the goal is to have different agents specialized in small tasks and collaborate, also with humans, to aid towards a specific goal. This is similar to the DevBots collaborating with developers and exchanging information with each other to facilitate software engineering processes [5, 12, 16].

In the field of Software Ecosystems (SECO) the goal is to classify the complex relationships formed between the actors (e.g. software developers or vendors) when interacting on a shared software market while sharing a common technical platform [9]. In SECO different roles and responsibilities have been outlined, such as distinguishing between actors that mainly produce information and value; or actors that primarily consume those [1, 6, 18]. Two common challenges in SECO are evolution management and infrastructure tools to foster collaboration between the actors [7]. This resembles the roles that DevBots can take in a bot ecosystem and the struggles that the developers we interviewed described in interaction and maintenance.

Building on this context, previous research in the area of bot classification has primarily examined individual bots in isolation [2, 3, 10, 13, 14, 19]. In contrast, our study explores the roles of



This work is licensed under a Creative Commons Attribution 4.0 International License.

*FSE Companion '24, July 15–19, 2024, Porto de Galinhas, Brazil*

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0658-5/24/07

<https://doi.org/10.1145/3663529.3663782>

multiple bots in collaboration, presenting an orthogonal approach to the existing literature.

### 3 DATA AND ANALYSIS

The initial results presented in this article are based on a case study performed at an automotive supplier in Gothenburg, Sweden, which specializes in the development of a software stack that enables autonomous driving for next-generation vehicles. The stack includes everything from low-level camera-related drivers to software running in the cloud. Consequently, the different components are often heterogeneous in regard to their scientific domain, programming language as well as safety requirements.

Multiple DevBots are being utilized at the company to conduct various activities in different stages of the development and verification process. While there is a diversity of DevBots interacting with software developers, most DevBots collaborating with other DevBots reside in the continuous integration pipeline and typically output information into the communication channel for code reviews. We use existing classification schema [3, 4] to identify and extract 20 DevBots from the company. To develop our taxonomy, we analyzed the source code and functionalities of those 20 DevBots, categorizing their roles (species). We incorporated descriptions from related fields such as Software Ecosystems and Biology.

Due to confidentiality agreements and space limitations, the specifics of these proprietary bots cannot be disclosed. Instead, we illustrate these roles using similar open-source bots available on GitHub. Utilizing a publicly available ecosystem as illustration would have been advantageous however we have no knowledge of such a system. Consequently, the examples provided do not collaborate in a known ecosystem, but the essence of the examples are that of the species we chose to illustrate. However, it is crucial to remember that the species of a DevBot is affected by the relations it has within the environment it inhabits, not solemnly by the intrinsic qualities of the DevBot itself. Finally, we demonstrate the utility of our taxonomy [17] by reporting on the prevalence of each bot species within the industrial project.

### 4 ECOSYSTEMS OF DEVBOTS

A DevBot is a tool which consist of some type of automation in combination with at least one of three essential human-like traits: (i) autonomy representing bots that operate independently, performing tasks without needing explicit instructions from developers (e.g., GitHub bots built on GitHub Actions to automate workflows), (ii) chat/voice interface for intuitive developer interactions using natural-like language (e.g., ChatGPT or DevBots that comment on pull requests or send emails), and (iii) smartness for bots that handle complex and non-trivial tasks (e.g., Copilot, which assists in code generation, and ChatGPT, which can engage in problem-solving activities and explanations). Bots can exhibit one or multiple traits, e.g. ChatGPT [3]. In turn, "A bot ecosystem is a set of bots working on the same or related projects" [15]. We expand on this definition by applying existing terminology in Biology and Software Ecosystems (SECOs) [6].

Using the metaphor of a natural ecosystem, we refer to **species** as the different bot roles, whereas the **environment** is the surrounding context that the different DevBot ecosystems operate

in. The environment comprises a collection of assets that are involved in software development. These assets are often immaterial, for example: source code, test cases, executable files, logs or machine learning models. Assets may also include hardware used for development purposes, such as servers, equipment, or Hardware-In-the-Loop rigs for testing. Further, each species "feeds" on different **resources**. The term resources, here, refers to artifacts or information from the environment. We classify resources as **processed** or **raw** depending on whether the resource has been generated by other bots or not. Resources can also include physical assets. If a resource is raw it is also an asset of the environment, a processed resource cannot be considered an asset unless it persists and outlives the execution of the bot.

We define four different species of DevBots that coexist and function as an ecosystem: (i) Keystone bots, (ii) Niche bots, (iii) Dominator bots, (iv) Observer bots. One bot is defined as belonging to one species within one ecosystem, but can over its lifetime evolve into another species. An overview of the different species can be found in Table 1.

#### 4.1 Keystone Bots (*the Plankton*)

In a SECO, the keystone players provide the technology which constitutes the foundation of the ecosystem and, similarly, keystone bots comprise the foundation of DevBot Ecosystems. They primarily feed on raw resources and while the data they generate may be of use to software developers, the main purpose is to produce or prepare resources for other bots in the ecosystem.

Keystone bots have the potential to majorly disrupt the ecosystem since they can directly affect the operation of other dependent DevBots which rely on the fitness of the generated artifacts or information. Therefore, keystone bots should be characterized by their availability, reliability and correctness. If keystones do not conform to the expectations that the ecosystem has on them, there is a high risk that other DevBots will starve. However, the probability to directly influence software development activities is rather low as humans do not constitute the main stakeholder in the workflow of this type of DevBot.






In our analysis of the company's DevBot ecosystem, from a total of 20 DevBots four were keystone bots. A Github bot that could satisfy the criteria for being a keystone bot is the doxygen-action bot<sup>1</sup> bot found on the popular GitHub Actions marketplace [8]. The bot parses source code and, if both the comments and configuration found in it are correct, produces HTML pages. It is in an ecosystem typically the task of a different bot to fetch the artifacts produced by doxygen-action and either publish this generated content directly or integrate it in something larger.

#### 4.2 Niche Bots (*the Zebra*)

Niche bots are reminiscent of niche players in SECOs. This species provides value for certain niches, i.e., specialized segments in a software development process such as testing, code reviews, design or deployment. They utilize the resources provided by the Keystone bots to do so. Niche bots can feed on both raw and processed resources, indicating that they may depend on other bots but their primary stakeholders are software developers.

<sup>1</sup><https://github.com/doxygen>

**Table 1: Overview of the bot species in a ecosystem of DevBots**

	<b>Class/Species</b>	<b>Unique aspect</b>	<b>Potential to disrupt</b>	<b>Example</b>
	Resources/food	Raw - from source Processed - by a bot	—	Raw - test cases Processed - code coverage
	Keystone bots	Feeds on mainly raw resources. Generate resources to be consumed by other bots.	High for other bots. Low for human users.	Doxygen-action
	Niche bots	Feeds on both raw and processed resources. Generate resources to be consumed by humans.	Low for other bots. High for human users.	DependaBot
	Dominator bots	Primarily consumes raw resources. Produces composite resources to accomplish various goals.	High for other bots. High for human users.	Docker-build-push
	Observer bots	Feeds on both raw and processed resources. Transforms data for human consumption.	Low for other bots. Low for human users.	CodeCov

Niche bots have the potential to disrupt software development activities directly since being unavailable or malfunctioning would impact the human-driven activity that they are designed to facilitate. However, their impact on other members of the ecosystem can be seen as minimal, as their output is typically not consumed by other DevBots or even if they would be, other bots are, by definition, very loosely coupled to Niche bots (otherwise, they would be classified as keystone bots). A majority of the DevBots analyzed at the company were niche bots (45%). Multiple off-the-shelf DevBots can be viewed as members of the Niche species since they are the ones considered as most appealing for the human stakeholders. We chose Dependabot<sup>2</sup> to illustrate this species. Dependabot analyzes the dependencies of a project, determines potentially insecure ones and depending on whether a viable alternative exists, either warns the user about the risk or proposes a solution. Before GitHub acquired Dependabot it could also be given the authority to autonomously change the existing code base and eliminate the risk if it had proven itself trustworthy.

### 4.3 Dominator Bots (*the Lion*)

In a SECO context, dominators seek to assimilate or remove other players from the ecosystem, taking over the entire spectrum of the business needs previously accommodated by those players. Similarly, dominator bots attempt to satisfy multiple development use cases by composing functionality that could have been distributed into separate bots. The data or information they produce is heterogeneous and due to their centralized nature, they exhibit a monolithic nature in their design. Similarly to their SECO progenitors, dominator bots may cause the collapse of the ecosystem if they continue to assimilate functionalities and expand their responsibilities.

As soon as dominators start to incorporate responsibilities that would canonically be attributed to a plethora of keystone or niche bots, an increasing number of bots have to depend on the dominator or even be integrated into its codebase, thus giving the dominators

the power to significantly disrupt the ecosystem. As this happens software development activities also increase their reliance on the dominators, since they accommodate a broad segment of the users' needs. As a result, if problems, such as a decrease in availability or correctness of a dominator are observed, then this has a major impact on both the ecosystem but also the software development that depends on them.

In our observed ecosystem two dominator bots existed. When looking at Github bots it was more difficult to find a match. However, the Docker build-push-action<sup>3</sup> incorporate a set of functionalities that could be split into smaller and more flexible bots. Now it builds the Docker image maintained in a repository and if that succeeds it publishes the image online. The problem with this approach is that one cannot reap, in isolation, the benefits of the two different activities, i.e. building and pushing, as they are coupled together.

### 4.4 Observer Bots (*the Ecologist*)

Contrary to the previously illustrated bots, the observers do not stem from a role present in SECOs but from Biology. Here the observer studies ecosystems in order to understand it, and characterizes the impact of the different species in the environment (and vice-versa). In our context, we define the observer as the role of bots that aim to extract information from the DevBot ecosystem in a non-invasive manner, without obstructing, influencing or somehow meddling with the internal affairs of ecosystem members. The observer strives to merely (i) formulate an accurate view of the system and (ii) for their presence not to affect the behavior of the investigated species.

An observer bot transforms already existing material, for human consumption. It feeds on raw and processed resources to gather different views of the system but not to generate new resources for the ecosystem or its surrounding environment. For example, an observer bot may send notifications about changes in integration pipelines, create reports on the current status and visualize metrics

<sup>2</sup><https://github.com/dependabot>

<sup>3</sup><https://github.com/marketplace/actions/docker-build-push-action>

in a manner that is easily digestible by software developers (e.g., dashboards). The observer bot can be utilized to document and increase the understanding of the different metrics that depict the health of the ecosystem or the environment.

Specimens of this species have a low capacity to disrupt the ecosystem because they do not generate any resources, and other DevBots do not rely on them. While it may become less convenient for humans themselves to interpret metrics on the health of the ecosystem and the environment, the relevant data is still present and can be parsed by different means instead of being summarized by a bot. Observer bots can be considered as a commodity and thus the overall risk for them to destabilize the ecosystem or software development activities is relatively low.

In our company ecosystem, five observers were found in our set of 20 analyzed bots. From Github we classify Codecov<sup>4</sup> as of the observer species, it uses both raw and processed resources to create human-readable reports. Codecov does not generate any new resource to the ecosystem. For instance, DevBots that would require sufficiently high code coverage to operate have no interest in Codecov's reports, as the information they need is already available.

## 5 COLLABORATION PROBLEMS

To gather more information about the ecosystem we interviewed 11 engineers that create or maintain those DevBots at the automotive company. Regarding bot-to-human collaboration, our interviewees stressed the importance of DevBots relaying information in a way that is easy to understand. This significantly affects the design of Observer bots, as they must communicate in a comprehensive manner that appeals to most developers regardless of technical level and prior experience, e.g., via domain-specific or natural languages.

In our interviews many problems on collaboration between the different bots emerged. Managing the bot-to-bot dependencies are essential when creating and maintaining a DevBot ecosystem, particularly for Keystone bots. For instance, not meeting the preconditions to run a DevBot was the main cause for impediment in their development cycles (e.g., availability of resources generated from bots or the environment). The interviewed practitioners mentioned the need to document those dependencies related to each bot since different stakeholders would maintain and run various bots at different times.

Besides dependencies, conflicts between DevBots were reported as a typical source of annoyance. Conflicts occur when DevBots follow contradicting sets of rules or interpret the same rules in opposing manners. This was often the case when applying coding standards using different tools. For example, fixes proposed by a DevBot utilizing a general-purpose open-source static analysis tool may not be considered valid by another DevBot tailored for safety-critical systems in the automotive domain and abides by a stringent set of constraints. Furthermore, a DevBot tasked with linting may contradict a DevBot that formats the code. Those conflicts lead to additional work by manually merging the fixes suggested by each bot.

Practitioners also shared the difficulties of working with or maintaining DevBots charged with a broad spectrum of responsibilities,

i.e., Dominator bots. Generally, there are trade-offs when introducing a single dominator bot versus multiple niche bots based on convenience and modularity. For example, interviewees mention that it is technically easier to develop one single bot, a dominator, that performs multiple development activities instead of having individual niche bots that must collaborate with each other (convenience). However, there is a risk that dominator bot will bear the technical overhead necessary to achieve this synergy and the maintainability of such a monolith is associated with difficulties (modularity).

Since DevBots are not explicitly part of the software product, they rarely receive the necessary attention for repairs and evolution. Most interviewees agreed that DevBot development often gets neglected and the responsibility of their development or maintenance falls on the shoulders of those with the goodwill to welcome it. However, participants showed contrasting views on whether there should be a dedicated team responsible for DevBots or if this task should be distributed among different teams who may be considered the natural DevBot stakeholders. This lack of an explicit 'ownership' of individual DevBots cause many bots to be neglected, consequently jeopardizing the stability of the ecosystem.

## 6 SUMMARY AND OUTLOOK

Our initial work resulted in a taxonomy with four species of bots that define types of interactions between a bot and assets, humans or other bots. Our interview with practitioners confirmed that bringing awareness to different types of interactions helps them decide: (i) which bots to remove/include to the development project, and (ii) how to design bots to minimize conflicts or expedite their maintenance. For instance, changes to a keystone bot directly impact many other bots working on the same project, whereas maintenance of observer bots mainly impact human developers. Future development in the area of LLMs could help improve the human-DevBot experience, for instance Retrieval Augmented Generation (RAG)[11] could be a key to improve the communication of Observer bots.

Further investigation of the attributes and nuances of varied bot-to-bot interactions should uncover more species or refine the existing ones, and doing so might also disclose additional collaboration problems. We encourage researchers and practitioners to advance the field by investigating and classifying species of bots "in the wild" to help us understand, empirically, how bots behave and collaborate in a development environment. Understanding those species helps developers to design and adopt bots that bring harmony and stability to multi-bot projects.

## ACKNOWLEDGMENTS

We extend our gratitude to our industry partner for their invaluable partnership and contributions to this research, and to the interview participants for generously sharing their insights. This research has been funded by Chalmers University of Technology Foundation and the Swedish Research Council (VR) under grant number 2018-04127 (Developer-Targeted Performance Engineering for Immersed Release and Software Engineers).

<sup>4</sup><https://github.com/marketplace/codecov>

## REFERENCES

- [1] P. R. J. Campbell and Faheem Ahmed. 2010. A Three-Dimensional View of Software Ecosystems. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10)*. Association for Computing Machinery, New York, NY, USA, 81–84. <https://doi.org/10.1145/1842752.1842774>
- [2] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and Future Bots in Software Development. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. 7–11. <https://doi.org/10.1109/BotSE.2019.00009>
- [3] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. 2020. An Empirical Study of Bots in Software Development: Characteristics and Challenges from a Practitioner's Perspective. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*. Association for Computing Machinery, New York, NY, USA, 445–455. <https://doi.org/10.1145/3368089.3409680>
- [4] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. 2022. Dependency management bots in open-source systems—prevalence and adoption. *PeerJ Computer Science* (2022). <https://doi.org/10.7717/peerj-cs.849>
- [5] Roberto A. Flores-Mendez. 1999. Towards a standardization of multi-agent system framework. *XRDS* 5, 4 (jun 1999), 18–24. <https://doi.org/10.1145/331648.331659>
- [6] Slinger Jansen, Sjaak Brinkkemper, and Anthony Finkelstein. 2009. Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. *Iwseco@ Icsr 2009* (2009).
- [7] JV Joshua, DO Alao, SO Okolie, and O Awodele. 2013. Software ecosystem: features, benefits and challenges. *International Journal of Advanced Computer Science and Applications* 2 (2013).
- [8] Timothy Kinsman, Mairieli Wessel, Marco A. Gerosa, and Christoph Treude. 2021. How Do Software Developers Use GitHub Actions to Automate Their Workflows?. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 420–431. <https://doi.org/10.1109/MSR52588.2021.00054>
- [9] Eric Knauss and Imed Hammouda. 2014. Eam: Ecosystemability assessment method. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 319–320.
- [10] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. 2019. Defining and Classifying Software Bots: A Faceted Taxonomy. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. 1–6. <https://doi.org/10.1109/BotSE.2019.00008>
- [11] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.
- [12] David L Martin, Adam J Cheyer, and Douglas B Moran. 1999. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence* 13, 1-2 (1999), 91–128.
- [13] Quim Motger, Xavier Franch, and Jordi Marco. 2022. Software-Based Dialogue Systems: Survey, Taxonomy, and Challenges. *ACM Comput. Surv.* 55, 5, Article 91 (dec 2022), 42 pages. <https://doi.org/10.1145/3527450>
- [14] Sivasurya Santhanam, Tobias Hecking, Andreas Schreiber, and Stefan Wagner. 2022. Bots in software engineering: a systematic mapping study. *PeerJ Computer Science* (2022). <https://doi.org/10.7717/peerj-cs.866>
- [15] Margaret-Anne Storey, Alexander Serebrenik, Carolyn Penstein Rosé, Thomas Zimmermann, and James D. Herbsleb. 2020. BOTse: Bots in Software Engineering (Dagstuhl Seminar 19471). *Dagstuhl Reports* 9, 11 (2020), 84–96. <https://doi.org/10.4230/DagRep.9.11.84>
- [16] Katia Sycara and Dajun Zeng. 1996. Coordination of multiple intelligent software agents. *International Journal of Cooperative Information Systems* 5, 02n03 (1996), 181–211.
- [17] Muhammad Usman, Ricardo Britto, Jürgen Börstler, and Emilia Mendes. 2017. Taxonomies in software engineering: A Systematic mapping study and a revised taxonomy development method. *Information and Software Technology* 85 (2017), 43–59. <https://doi.org/10.1016/j.infsof.2017.01.006>
- [18] Ivo Van Den Berk, Slinger Jansen, and Lützen Luinenburg. 2010. Software ecosystems: a software ecosystem strategy assessment model. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. 127–134.
- [19] Mairieli Wessel, Bruno Mendes De Souza, Igor Steinmacher, Igor S Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A Gerosa. 2018. The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19. <https://doi.org/10.1145/3274451>

Received 28-JAN-2024; accepted 2024-04-09