Towards Automated Governance: A DSL for Human-Agent Collaboration in Software Projects

Adem Ait
University of Luxembourg
Esch-sur-Alzette, Luxembourg
adem.ait@uni.lu

Gwendal Jouneaux
Luxembourg Institute of Science and Technology
Esch-sur-Alzette, Luxembourg
gwendal.jouneaux@list.lu

Javier Luis Cánovas Izquierdo
Universitat Oberta de Catalunya
Barcelona, Spain
jcanovasi@uoc.edu

Jordi Cabot
University of Luxembourg
Luxembourg Institute of Science and Technology
Esch-sur-Alzette, Luxembourg
jordi.cabot@list.lu

***Abstract*—The stakeholders involved in software development are becoming increasingly diverse, with both human contributors from varied backgrounds and AI-powered agents collaborating together in the process. This situation presents unique governance challenges, particularly in Open-Source Software (OSS) projects, where explicit policies are often lacking or unclear. This paper presents the vision and foundational concepts for a novel Domain-Specific Language (DSL) designed to define and enforce rich governance policies in systems involving diverse stakeholders, including agents. This DSL offers a pathway towards more robust, adaptable, and ultimately automated governance, paving the way for more effective collaboration in software projects, especially OSS ones.**

I. Introduction

Governance is a key aspect of software development, especially in Open-Source Software (OSS), where the collaborative nature of the process requires clear guidelines and policies to ensure effective decision-making and accountability. Governance policies help to establish roles, responsibilities, and processes for managing contributions, resolving conflicts, and maintaining the integrity of software projects [1]. Such policies make explicit the decision-making process, ensuring that all the involved stakeholders know how decisions are made and who is responsible for them. However, most (OSS) projects do not explicitly define governance policies, and those that do are often poorly described and lack clarity [1]. Furthermore, when described, they are usually scattered across different project resources, making it difficult for contributors to understand the rules and procedures that govern the project [2].

Recently, the landscape of software development is undergoing a profound transformation. The rapid proliferation of AI-powered agents participating in development tasks, coupled with a growing recognition of the critical role of diverse human backgrounds, presents unprecedented challenges to these established governance paradigms [3], [4], [5], [6]. AI agents, powered by advancements in Large Language Models (LLMs), are moving beyond simple automation to become active participants capable of complex communication, collaboration, and even decision-making [7]. On the other hand, human diversity is also beneficial for software development, as it can lead to more innovative solutions and better decision-making [8], while also benefiting end-users [9]. While this human-agent collaboration and broader human diversity promise significant benefits in innovation and productivity, they also expose a critical gap: the lack of governance frameworks designed to explicitly and holistically manage such multifaceted participation.

This paper first replicates a study on governance policies in OSS [2] to demonstrate the persistent need for explicit governance, particularly in today's agentic development context. Then, we address this gap by proposing the conceptualization of a Domain-Specific Language (DSL) designed to define and enforce governance policies in systems involving a diverse set of stakeholders, covering both human and agent collaborators. The proposed DSL provides a structured approach for expressing governance rules, roles, responsibilities, and decision-making processes. It encompasses various dimensions of governance, including decision-making procedures, positive and negative discrimination, and other methods to ensure a fair representation. We envision our DSL to be field-agnostic, extensible, and adaptable to different contexts, allowing for the inclusion of various stakeholders and their specific needs. We also provide tool support to automate and enforce the governance policies defined with our DSL.

The rest of the paper is structured as follows. Section II provides the motivation. Section III and IV presents the design and implementation of the DSL, and the proof of concept, respectively. Section V discusses the roadmap. Finally, Section VI concludes the paper and outlines the future work.

This work has been funded by the European Union under the Grant Agreement No 101189664 (MOSAICO project) and TED2021-130331B-I00 funded by MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them. Jordi Cabot is supported by the Luxembourg National Research Fund (FNR) PEARL program, grant agreement 16544475.

II. State of the Art

A. Diversity and Governance in Software Engineering

Software engineering teams are becoming increasingly diverse across multiple dimensions. Recent research has highlighted the importance of diversity within software development processes [5], [6]. Diversity is being studied in terms of gender, race or ethnicity, but also in terms of cognitive diversity [10], [11]. Studies have shown that diverse teams tend to produce more innovative solutions [12], [13], and impact positively in software productivity [14].

Adding to this human diversity, software projects now include a growing number of non-human participants in the form of bots and AI-powered agents [3], [4]. These automated participants have traditionally performed routine tasks like continuous integration, code quality checks, and dependency management [15]. While these tools have traditionally handled routine tasks and basic governance functions [16], [17], the rise of LLMs is transforming these agents from mere automation tools into sophisticated collaborators capable of reasoning, communication, and participation in complex decision-making [18], [7]. This profound shift introduces an unprecedented dimension of diversity, presenting novel and urgent conceptual challenges for governance that existing frameworks are missing.

B. Governance in practice: the case of OSS development

A previous study illustrated the need for explicit governance policies in OSS [2] but lacked the mechanisms to address the challenges introduced by participant diversity, particularly the inclusion of AI agents in decision-making processes. Our current proposal aims to fill this gap by incorporating diversity-aware governance mechanisms that acknowledge the different characteristics of participants and ensure equitable representation in decision-making processes.

While the landscape of participants is evolving, the explicitness and nature of governance policies in practice, particularly in OSS, remain a concern. To understand this current state, we replicated the study conducted by Cánovas Izquierdo and Cabot [1], which analyzed the top 25 starred software projects and found that most OSS projects do not have explicit governance policies, and those that do are often poorly defined and lack clarity. The governance evidence is analyzed from the explicit indications of four dimensions: (1) whether there is some workflow to contribute, different from the typical pull-based development process; (2) who makes the decisions to accept code, and how; (3) how long it takes to review or accept a contribution; and (4) how to become a contributor.

We found that 68% of repositories reported at least one of the four dimensions, 24% reported none, and 8% reported all four dimensions. The situation is showing slight improvements from the previous study. In comparison, we have seen an increase in the number of projects that partially discussed governance, from 32% to 68%, mainly because of the report of the contribution process. However, the span to review or accept a contribution is only defined in 16% of the projects, how to become a contributor is only defined in 20% of the projects, and who makes the decisions, and how, to accept code is only defined in 24% of the projects. Only 2 projects (8%) provided a full description of the policies governing them. Note that no project included a governance.md file but reported them in the contributing.md file or on their documentation website. None of the analyzed projects adopted a DSL to specify their governance policies, which could help in the definition, enforcement, and automation of the governance policies [1].

III. Defining Governance Policies

A DSL can formalize governance policies, roles, and decision-making procedures, enhancing transparency and reliability. To the best of our knowledge, there is no DSL for defining governance policies in systems that address the diversity of participants.

Several studies analyze the governance of software projects [19], [20], particularly OSS projects [21], [2], [22]. However, to the best of our knowledge, only Cánovas Izquierdo and Cabot [2] proposed a DSL, where they covered the main dimensions to define and enforce governance rules in OSS projects. Note, their proposal did not provide support to assess the diversity of the participants involved in the decision-making process. Our proposal completely revamps that DSL and adapts it to the current reality of software development involving participants with different profiles, the possibility of having uncertain AI agents, and much more complex decision procedures with different participant weighting systems.

To collect the requirements for our DSL, we have carefully considered existing AI governance frameworks and principles [23]. For instance, we incorporate key AI governance indicators identified in the literature, such as autonomy level [24] and explainability [25], thus integrating them directly into our language constructs for agent representation.

Based on these frameworks and our own experience in open-source development and the analysis of open-source communities, we propose, in what follows, our governance DSL. A DSL is composed of three main elements [26]: (1) abstract syntax, which defines the concepts and relationships of the domain where the language is applied; (2) concrete syntax, which defines the notation of the language (e.g., textual, diagram-based, etc.); and (3) semantics, which defines the meaning of the language constructs. In the following, we discuss each component.

A. Abstract Syntax

The abstract syntax of our DSL is defined via the metamodel shown in Figure 1. Metamodels restrict the structure of valid models (i.e., DSL instances, in our case, concrete policies) and define the relationships between the different elements of the language (i.e., the different concepts that can be used to define a policy and how these concepts can be linked to each other) [27]. Governance models conforming to this metamodel represent specific sets of governance policies.

Our proposal is structured around key conceptual areas:

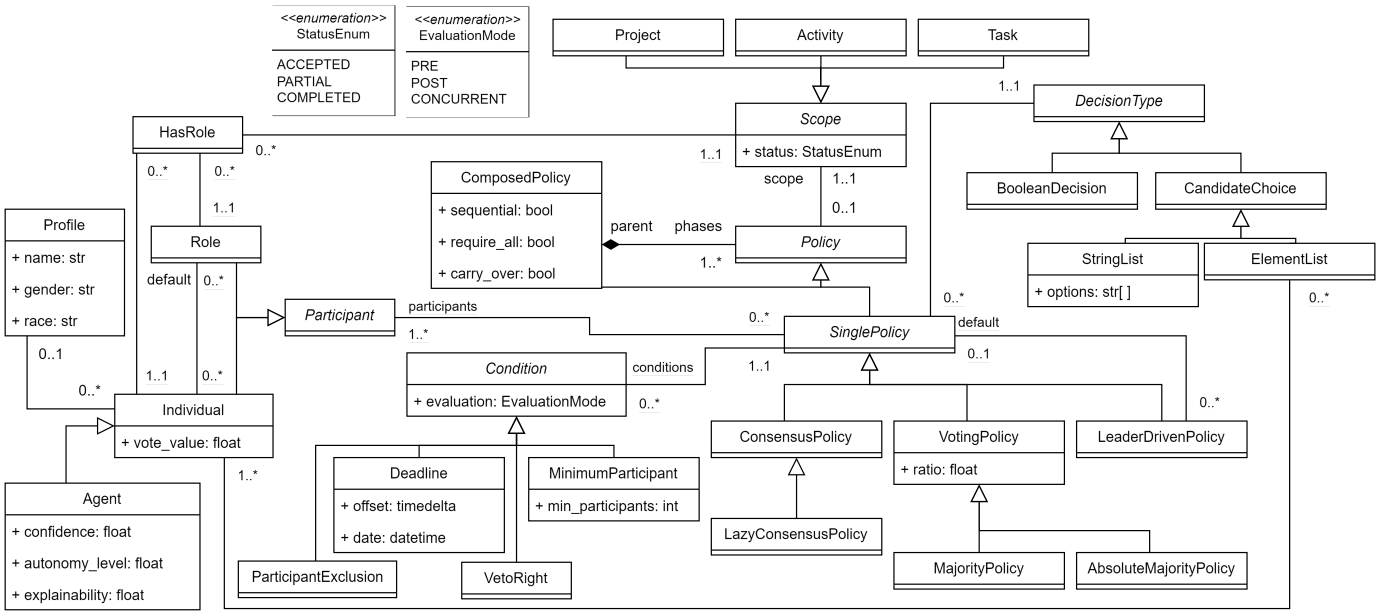

Fig. 1. Abstract syntax metamodel of our DSL.

Participants: Embracing Diversity and Agent Characteristics. A vital aspect of our DSL is the representation of `Participant`. The metamodel distinguishes between human participants (potentially detailed with `Profile` information like gender or role-specific attributes to support fairness or weighted influence) and `Agents`. The influence of each participant in the decision-making process can be adjusted through the `vote_value` attribute, which can be set to favor certain individuals or groups. For `Agents`, we describe further attributes critical for governance, such as `autonomy_level`, `explainability` and `confidence`. This allows expressing complex policies, such as giving a high-autonomy agent more freedom, while decisions from agents with low explainability might be used only as support. Participants can also be grouped by `Role`, while also being flexible to allow certain individuals to act as part of a role in specific policies (see `hasRole`).

Policies: Defining Flexible and Composable Decision-Making. The DSL provides a rich set of `Policy` constructs to capture diverse decision-making processes essential for adaptive governance. It supports various policy resolution strategies, including `VotingPolicy` (e.g., majority and qualified majority based on a `ratio`), `ConsensusPolicy` (including `LazyConsensus` [28] as a practical mechanism for efficient agreement), and `LeaderDrivenPolicy` (with optional fallback mechanisms). Policies specify a `DecisionType`, such as `BooleanDecision` (for accept/reject scenarios like pull request approvals) or `CandidateChoice` (for selecting one out of multiple options, like electing a leader). Simple policies can be combined into `ComposedPolicy` structures (e.g., requiring sequential evaluation or all sub-policies to pass). This enables the modular construction of complex governance scenarios from simpler, reusable components, thus enhancing clarity and maintainability.

Scope and Conditions: Contextualizing Policy Application. Governance policies are usually applied in a particular project or granular component of it. Our DSL allows their application to be precisely contextualized. Policies are applied within a defined `Scope`, such as a `Project`, an `Activity` (e.g., development, testing), or a specific `Task` (e.g., a patch). This hierarchical scoping enables granular control, allowing different governance rules for different project areas or activities. Furthermore, policies can be described with `Conditions` (e.g., setting a deadline or a minimum required participants). These conditions can also act as prerequisites (`pre`) or checks following execution (`post`).

B. Concrete Syntax

As concrete syntax, we use a textual notation following a typical block-based structure (see Listing 1). The syntax is therefore formally defined by a grammar. Each instance of the metaclass is textually represented by its keyword and a block that contains the properties of the instance. Containment references are represented as nested blocks while non-containment references use an identifier to refer to the target element.

C. Operational Semantics: Decision engine

The operational semantics take the form of a decision engine, enforcing the policies on ongoing collaborations. The engine receives platform data to update the decision-making process state and take the necessary actions. For the first implementation of our approach, we rely on GitHub as the collaboration platform.

The decision engine operates through a continuous cycle of state management and policy enforcement. When a collaboration event occurs (e.g., a pull request is created), the engine captures the event data and creates an internal

```
Scopes:
    Project myProject {
        activities : myActivity {
            tasks : myTask
        }
    }
Participants:
    Profiles :
        profile1 {
            gender : male
            race : hispanic
        }
    Roles : Maintainer
    Individuals :
        Joe {
            vote value : 0.7
            profile : profile1
            role: Maintainer
        },
        (Agent) Mike {
            confidence : 0.8
            role: Maintainer
        },
        Paul {
            role: Maintainer
        }
MajorityPolicy TestPolicy {
    Scope: myTask
    DecisionType as BooleanDecision
    Participant list : Maintainer
    Conditions:
        Deadline : 10 days
        ParticipantExclusion : Paul
    Parameters:
        ratio : 0.4
}
```

Listing 1. Simple governance policy evaluated following a majority strategy as a `BooleanDecision` requiring 40% positive votes with a deadline of ten days.

representation. This state management enables the engine to track multiple concurrent decision-making processes.

For new collaborations, the engine identifies applicable policies based on scope. Once found, the engine creates an internal deadline check based on the policy to resolve the process. When voting occurs (e.g., a pull request review), eligible users' votes are registered in the collaboration's ballot box. Each vote is captured with its rationale, timestamp, and association to the voter. At deadline, the engine analyzes the votes and enacts the decision (e.g., merge the pull request). Composed policies resolve phases sequentially or in parallel, combining decisions via conjunction or disjunction. In addition, votes can be carried over between phases.

IV. Proof of Concept

We implemented the metamodel as a set of Python classes, while the concrete syntax was implemented through a grammar definition using ANTLR [29], a parser generator tool. We developed a transformation that converts the Abstract Syntax Tree (AST) generated by ANTLR into instances of our metamodel, which serve as input for the decision engine. The complete implementation, including the grammar and decision engine, are available in the tool repository.[1] We also translated several governance policies identified in our background analysis (see Section II-B) into our DSL syntax to demonstrate its expressiveness and practical applicability.

[1]https://doi.org/10.5281/zenodo.15856633

V. Roadmap

The proposed DSL is just a first step towards a more diverse, transparent and agentic collaboration in software development. Open challenges and future directions are discussed below.

Usability improvement via new syntaxes. To facilitate the adoption of our DSL by less technical users, we envision offering alternative UIs, like chatbots, to enhance usability. This will be especially useful when introducing our DSL to manage governance in other, non-software, domains.

Longitudinal study. To better understand if explicit governance rules help to increase OSS contributions, we plan a longitudinal study where we could compare the evolution of contributions in projects before and after they committed to an explicit governance model. This same study could be useful to see whether projects are sticking to such governance model or they just ignore it even after committing to it.

Improved impact of Multi-agent systems in software development. We plan to test whether our governance layer enables more complex multi-agent systems to collaborate, in a controlled way, to solve more ambitious software engineering tasks. By leveraging SWE-bench [30], we can systematically evaluate how governance policies affect agent collaboration, task completion rates, and adherence to defined constraints.

Participant identification and bias mitigation. A problem some projects face is identifying the right people to play certain roles. Given a governance model, it would be possible to analyze past project data to propose candidates to fill the roles required in the governance model. A real-time monitoring component should also be helpful to proactively change the candidates to continuously improve the diversity of the collaborations. Indeed, a key element of this (semi)automatic participant analysis could be devoted to make sure the participants list is not biased against certain communities and that it respects the profile requirements.

Governance beyond software. We believe other domains could also benefit from an explicit and precise definition of their governance policies based on our DSL. For instance, making explicit the governance systems of organizations (such as NGOs) would enable systematic comparison of their decision-making structures and their assessment of inclusivity.

VI. Conclusion

We have presented a vision of a DSL for the explicit definition and automation of governance policies in software projects, with a particular focus on supporting diversity among participants, including both human contributors and AI-powered agents. Our DSL extends previous work by enabling the specification of participant profiles and agent attributes, thus facilitating more equitable and transparent decision-making processes. This is just a first step in this direction. As future work, we plan to work on the aforementioned roadmap.

References

[1] J. L. Cánovas Izquierdo and J. Cabot, "For a more transparent governance of open source," *Commun. ACM*, vol. 66, no. 8, pp. 32–34, 2023.

[2] J. L. Cánovas Izquierdo and J. Cabot, "Enabling the definition and enforcement of governance rules in open source systems," in *Int. Conf. on Software Engineering*, pp. 505–514, 2015.

[3] M. S. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in OSS projects," *Hum. Comput. Interact.*, vol. 2, no. CSCW, pp. 182:1–182:19, 2018.

[4] Z. Rasheed, M. Waseem, M. A. Sami, K. Kemell, A. Ahmad, A. Nguyen-Duc, K. Systä, and P. Abrahamsson, "Autonomous agents in software development: A vision paper," in *Int. Conf. on Agile Software Development*, pp. 15–23, 2024.

[5] P. Bjørn, M. Menendez-Blanco, and V. Borsotti, *Diversity in computer science: Design artefacts for equity and inclusion*. Springer Nature, 2023.

[6] G. Rodríguez-Pérez, R. Nadri, and M. Nagappan, "Perceived diversity in software engineering: a systematic literature review," *Empir. Softw. Eng.*, vol. 26, no. 5, p. 102, 2021.

[7] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," in *Int. Joint Conf. on Artificial Intelligence*, pp. 8048–8057, 2024.

[8] Y. Yang, T. Y. Tian, T. K. Woodruff, B. F. Jones, and B. Uzzi, "Gender-diverse teams produce more novel and higher-impact scientific ideas," *Proceedings of the National Academy of Sciences*, vol. 119, no. 36, p. e2200841119, 2022.

[9] H. Gunatilake, J. C. Grundy, R. Hoda, and I. Mueller, "The impact of human aspects on the interactions between software developers and end-users in software engineering: A systematic literature review," *Inf. Softw. Technol.*, vol. 173, p. 107489, 2024.

[10] J. Giner-Miguelez, S. Morales, S. Cobos, J. L. C. Izquierdo, R. Clarisó, and J. Cabot, "The software diversity card: A framework for reporting diversity in software projects," *CoRR*, vol. abs/2503.05470, 2025.

[11] R. Dutta, D. E. Costa, E. Shihab, and T. Tajmel, "Diversity awareness in software engineering participant research," in *Int. Conf. on Software Engineering: Software Engineering in Society*, pp. 120–131, 2023.

[12] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci, "Gender diversity and women in software teams: how do they affect community smells?," in *Int. Conf. on Software Engineering: Software Engineering in Society*, pp. 11–20, 2019.

[13] Y. Yang, T. Y. Tian, T. K. Woodruff, B. F. Jones, and B. Uzzi, "Gender-diverse teams produce more novel and higher-impact scientific ideas," *Proceedings of the National Academy of Sciences*, vol. 119, no. 36, p. e2200841119, 2022.

[14] B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. T. Devanbu, and V. Filkov, "Gender and tenure diversity in github teams," in *Conf. on Human Factors in Computing Systems*, pp. 3789–3798, 2015.

[15] T. Kinsman, M. S. Wessel, M. A. Gerosa, and C. Treude, "How do software developers use github actions to automate their workflows?," in *Int. Conf. on Mining Software Repositories*, pp. 420–431, 2021.

[16] M. Wessel, J. Vargovich, M. A. Gerosa, and C. Treude, "Github actions: The impact on the pull request process," *Empir. Softw. Eng.*, vol. 28, no. 6, p. 131, 2023.

[17] A. Decan, T. Mens, P. R. Mazrae, and M. Golzadeh, "On the use of github actions in software development repositories," in *Int. Conf. on Software Maintenance and Evolution*, pp. 235–245, 2022.

[18] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, *et al.*, "The rise and potential of large language model based agents: A survey," *Science China Information Sciences*, vol. 68, no. 2, p. 121101, 2025.

[19] S. Chulani, C. Williams, and A. Yaeli, "Software development governance and its concerns," in *Int. workshop on Software development governance*, pp. 3–6, 2008.

[20] J. D. Herbsleb, "Building a socio-technical theory of coordination: why and how (outstanding research award)," in *Int. Symposium on Foundations of Software Engineering*, pp. 2–10, 2016.

[21] R. van Pelt, S. Jansen, D. Baars, and S. Overbeek, "Defining blockchain governance: A framework for analysis and comparison," *Inf. Syst. Manag.*, vol. 38, no. 1, pp. 21–41, 2021.

[22] S. Keertipati, S. A. Licorish, and B. T. R. Savarimuthu, "Exploring decision-making processes in python," in *Int. Conf. on Evaluation and Assessment in Software Engineering*, pp. 43:1–43:10, 2016.

[23] S. Chesterman, Y. Gao, J. Hahn, and V. Sticher, "The evolution of AI governance," *Computer*, vol. 57, no. 9, pp. 80–92, 2024.

[24] I. Rahwan, M. Cebrián, N. Obradovich, J. C. Bongard, J. Bonnefon, C. Breazeal, J. W. Crandall, N. A. Christakis, I. D. Couzin, M. O. Jackson, N. R. Jennings, E. Kamar, I. M. Kloumann, H. Larochelle, D. Lazer, R. McElreath, A. Mislove, D. C. Parkes, A. S. Pentland, M. E. Roberts, A. Shariff, J. B. Tenenbaum, and M. P. Wellman, "Machine behaviour," *Nat.*, vol. 568, no. 7753, pp. 477–486, 2019.

[25] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, 2020.

[26] A. Wasowski and T. Berger, *Domain-Specific Languages - Effective Modeling, Automation, and Reuse*. Springer, 2023.

[27] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers, 2017.

[28] "Lazy Consensus." https://www.apache.org/foundation/glossary.html#LazyConsensus. Accessed: 2025-06-23.

[29] "ANTLR (ANother Tool for Language Recognition)." https://www.antlr.org/. Accessed: 2025-06-23.

[30] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan, "Swe-bench: Can language models resolve real-world github issues?," in *Int. Conf. on Learning Representations*, 2024.