# Regulation as an Enabler for Collaborative Software Development

Maryi Arciniegas-Mendez, Alexey Zagalsky, Margaret-Anne Storey, Allyson F. Hadwin
University of Victoria, Victoria, BC, Canada
{maryia, alexeyza, mstorey, hadwin}@uvic.ca

***Abstract*—Collaboration has become an integral aspect of software engineering. The widespread availability and adoption of social channels has led to a culture where today's developers participate and collaborate more frequently with one another. Awareness is widely accepted as an important feature of collaboration, but exactly what this encompasses and how processes and tools should be evaluated in terms of their awareness support remains an open challenge. In this paper, we borrow a theory of regulation from the Learning Science domain and show how this theory can be used to provide more detailed insights into how collaboration tools and processes can be compared and analyzed.**

## I. Introduction

Collaboration is crucial for the modern socio-technical developer. Developers not only write code but also partake in various socially enabled activities (e.g., mentorship and code reviewing), and the tools that developers use play a critical role in promoting and supporting collaboration. A widely accepted framework for evaluating collaborative tools and processes in software engineering is the *3C Model* [1]. This framework examines communication, cooperation, and coordination, with coordination being recognized as a necessity for collaboration.

However, existing models of coordination were proposed before the development community embraced socially enabled channels and tools [2]. This adoption caused a cultural shift and has led to a participatory culture [3] with different coordination and collaboration needs. Developers more readily learn from and work with one another in an *ad-hoc* manner within a broader community of like-minded peers. That is, the means by which developers author code today has drastically changed. This change brings both benefits (e.g., community authored documentation) and challenges (e.g., feeling overwhelmed by the vast choice of information and tools one could use). This participatory culture also brings with it challenges in terms of understanding which tools developers should use. These new collaboration work practices call for a richer framework so that we are able to understand and reason about the myriad of tools developers use today.

Challenges with coordination and collaboration are not unique to software engineering—researchers in other knowledge domains are intimately familiar with these challenges and actively study them. In particular, researchers in the Learning Science domain have achieved a deeper understanding of collaboration by examining *regulation* as the adaptation of strategic responses to new learning and collaboration challenges. This research suggests three types of regulation: self-regulation, co-regulation, and shared-regulation. Self-regulation refers to the individual's vision of the task, and represents a necessary condition for successful collaborative efforts, but one that is insufficient by itself. Co-regulation allows for recognition of the perspectives of others with respect to the tasks to be completed. Shared-regulation represents the last stage of regulation where individual perspectives have been aligned into a common group vision. The accomplishment of co-regulation and shared-regulation are key to successful collaborative efforts.

In this paper, we enrich the 3C Model by augmenting it with the *Model of Regulation* [4]. Using the enhanced 3C Model, we examined GitHub's Issues, Graphs, and News feed features. We find that the Model of Regulation brings new insights about the collaborative support provided by GitHub. Our preliminary analysis demonstrates that the application of this new framework provides better support for evaluating modern collaborative software development tools and practices. Furthermore, this new framework provides a common language for tool builders and practitioners that was previously lacking.

## II. Background

We review the evolution of collaborative software development research and provide an overview of the 3C Model. We then discuss the concepts of community and collaboration from the Learning Science domain as an introduction for the Model of Regulation, which we present in Section III.

### A. Collaborative Software Development

Modern software development is a social and collaborative process [5] involving groups of individuals that contribute towards a common goal with each other's experience and expertise. The recurrent collaborative activity, fueled by the need for different skills and perspectives to problem solving, has led to the formation of *communities of practice* [6] around diverse topics of software engineering.

Communities of practice, empowered by the *"inexpensive and low barriers to publish, as well as the rapidly spreading peer-to-peer, large-scale communications made possible by social media"*, have become an extensive part of software engineering [3]. The social structure created by the community supports the co-construction of knowledge for active participants and learning for all members. Moreover, these communities are a resource for consulting on and distributing software development practices, tools, and resources.

The widespread adoption of socially enabled tools and channels facilitates virtual communities of practice and enables global software teams. Consequently, the availability of additional communication channels (e.g., micro-blogging) and a broad catalog of developer tools have increased the participatory culture among software developers. More importantly, the diversity of these tools and features posits a challenge in understanding which may be the best composition or combination of tools that will enhance the *productivity* of collaborating developers and motivate broader *participation*.

### B. The 3C Model

Understanding collaboration and how it can be improved through processes and tools has been a longstanding challenge in computer science. As far back as 1991, Ellis *et al.* [7] investigated computer-supported group interactions and suggested three key areas that require attention when studying collaborative work: *Communication*, *Collaboration*, and *Coordination*. Their analysis provided the basis for a framework that has been widely used to evaluate collaboration tools. This approach is recognized as the 3C Model.

In this model, the *Communication* aspect of group work refers to the exchange of knowledge and allows for the coordination of group tasks. *Coordination* refers to the awareness of and agreements made regarding tasks to be completed through team interactions, as well as any overhead (e.g., planning) that is necessary for the *Collaboration* effort itself [7].

Later, Gerosa *et al.* [8] suggested that *Awareness* be added to the 3C Model. Awareness, as defined by Dourish and Bellotti [9], is *"an understanding of the activities of others, which provides a context for your own activity"*. Gutwin *et al.* [10] also found that distributed developers need to maintain awareness of one another and the entire team, as well as gather more detailed knowledge of the people they plan to work with. In this paper, we use and enhance a refined version of the 3C Model defined by Fuks *et al.* [11] where they included the concept of awareness, but also replaced the term *collaboration* with *cooperation* due to a difference in terminology (see top part of Fig. 1).

### C. Knowledge Building Communities

Collaboration involves more than just distributing tasks across people. It also involves regulation planning, enacting, monitoring, and evaluating processes, ideally producing something better than any individual could either conceive or produce alone [12]. Furthermore, coordination and communication have been shown to represent an important source for delays in software development projects [13]. The formation and growth of communities rely on the complex interactions of its members [3], suggesting that successful collaboration strongly depends on the social component of the community.

Communities of practice have been the focus of study in many domains. In Learning Science, researchers refer to a *Knowledge Building Community* as a socio-constructivist pedagogical strategy *"that supports discourse and aims to advance the knowledge of the members collectively while still encouraging individual growth that will produce new experts and extend expertise within the community's domain"*[1]. The software developer community is a prime example of a knowledge building community.

## III. Model of Regulation

The Learning Science literature emphasizes that merely providing opportunities to collaborate does not necessarily guarantee successful collaboration. Hadwin *et al.* [14] found that a group's collaborative potential may not be fulfilled by group members due to a lack of **regulatory skills**, such as time management, efficient use of resources, and task distribution. They further found that many learners do not effectively regulate their activities, neither individually nor collectively [15].

The theory of *regulated learning* [16] is an approach to address the above mentioned problems within a collaborative learning context. In particular, it refers to strategic control that allows for the improvement of learning and collaboration processes. Recent studies [4], [15] have led to the formation of a *Model of Regulation* for collaborative tasks which suggests three different types of regulation, with each type of regulation requiring different forms of process tool support.

### A. Three Types of Regulation

The Model of Regulation includes the following objects of regulation: task knowledge, self knowledge, goals and plans, strategy knowledge and use, motivation and emotions, and tools and context. Each regulation type (self, co, shared) is defined in terms of its relationship with the objects of regulation. In particular, *self-regulation* is an activity performed by an individual that assists with self-awareness of their contributions to the objects of regulation. Once the participant in the collaborative activity is aware of their own comprehension of the work, they are able to move onto the second type of regulation. *Co-regulation* is an activity that leads to the recognition of each other's understanding of the objects of regulation, and represents the beginning of the alignment between the individual and the group's perspectives. Finally, when each other's understanding has been recognized and aligned into one group vision, then *shared-regulation* has been reached and represents the *"we perspective"*.

### B. Regulation Process Support

When Jarvela and Hadwin [15] investigated the Model of Regulation, they articulated how computer-based tools can be used to support self-, co-, and shared-regulation within a learning context. Below we describe the different categories of regulation tool support, however, their contribution to self-, co-, and shared-regulation depends on the scope of the information and the way it is presented to the user. Within each category, we provide a description of the type of regulation it can support from a software engineering perspective.

[1] http://edutechwiki.unige.ch/en/Knowledge-building_community_model

*1) Structuring Support:* refers to approaches that aim to guide interactions by designing or scripting a situation before it begins [17]. Structuring support consists of roles, scripts, and prompts. **Roles** are functions intentionally assigned to each member of the team (e.g., designer, developer, or tester). While **scripts** are a list of steps that suggest the correct order of activities required to complete a bigger task. For example, the life cycle of a software development project may require the following order of general activities: requirements gathering, design, implementation, and evaluation. A script could provide the list of activities with a detailed set of steps for each phase. Finally, **prompts** are messages that can be delivered to the team to provide hints and suggestions about correct processes for a particular activity. For instance, when code changes are made, a prompt can remind the developer to register the changes in the team's Wiki or notify the other team members. All three types of structuring support aid in the strategic control of individual performance within the collaborative activity, thus supporting self-regulation.

*2) Mirroring Support:* refers to mechanisms that reflect individual or collective actions by gathering and summarizing data [17]. Depending on the scope of the information presented by the tool, it can promote self-, co- or shared-regulation. An example of mirroring used for self-regulation can consider individual actions as a summary of activities, with completed tasks distinguished by color or some other cue. In addition, mirroring support can be achieved with a visualization, such as a pie chart or a time-line graph that shows aggregated contributions. In this case, the visualization supports a shared understanding of the group perspective and creates context for shared-regulation. Several software development environments provide views to highlight such information (e.g., Trello).

*3) Awareness Tools:* is a similar category to mirroring support, except that it allows for self comparison or comparisons between group members. The scope of the information included within the tool determines whether it promotes co-regulation or shared-regulation. For instance, an awareness tool that invites co-regulation will be restricted to comparisons between a small set of members, while a context for shared-regulation requires simultaneous performance analysis of the whole group. Potential examples of awareness tools include a visualization presenting individual time-line graphs per contributing member, or a visualization showing the lines of code introduced by each member.

*4) Guiding Systems:* are programs that behave like a virtual presence interpreting data and providing instructions when issues in the collaboration process are detected. A guiding system can promote all three types of regulation depending on the context. For instance, a guiding system that sends reminders of an individual's pending work is helping with self-regulatory processes. If the guiding system provides instructions on a subtask designated for a set of members, it supports co-regulation. Whereas a guiding system that shows visualizations of ideas not picked up during previous group discussion creates context for shared-regulation. To the best of our knowledge, guiding system support in software engineering has not been explored much, but we see it as a prominent future research direction.

## IV. Enhancing the 3C Model

We argue that the collaborative nature of software development can only reach its full potential if we can clearly identify which components of group interaction are supported by the myriad of tools and resources that are used today. To address this, we propose to enhance the "awareness" component of the 3C Model (the *Enhanced 3C Model*) with the Model of Regulation (Fig. 1), highlighting that regulation provides the foundation for collaboration. As far as we know, there is no research that considers this approach in software engineering.

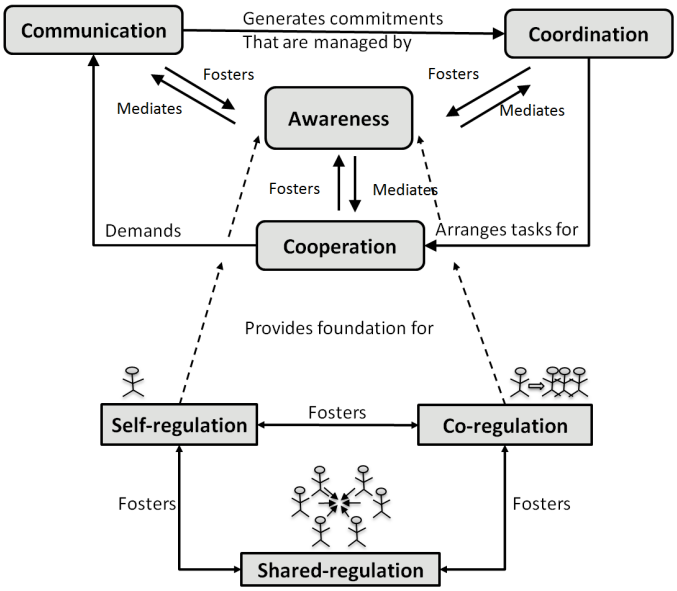


Fig. 1. Enhancing the 3C Model (shown in the top half of this figure) with the Model of Regulation (shown in the bottom half).

The inclusion of the Model of Regulation within the 3C Model leads to a richer framework. It provides terminology that we found was lacking when we attempted to clearly describe and distinguish commonly used collaborative software development tools. To demonstrate how the model can be applied, we examine three features of GitHub, a popular social coding environment: **Issues**, **Graphs**, and **News feed**[2].

**Issues** are defined as work items that *"can be labeled and assigned to a user"*. This definition suggests that the feature offers context primarily for co-regulation, but in practice, users use this feature to contribute to each of the three types of regulation. When an Issue is intended for the same member that created it, it is used as a self-reminder and supports self-regulation. But if the Issue is assigned as a to-do item for someone else, then co-regulation is supported. Also, issues can be created as an open request for everyone in the group (e.g., bug report or feature request). This case supports shared-regulation as any member has the opportunity to comprehend the collective state of the group. As mentioned previously, how

[2]https://help.github.com/articles/github-glossary/

a tool is used and what information it can manage affects the type of regulation it supports. Regarding the classification of Issues within the categories of regulation tools, we place it in mirroring support.

**Graphs** allow team members to visualize individual and collective statistics (e.g., contributor activities, number and time of commits) for a particular repository. For contributor activities, the Graphs feature presents two approaches. The first shows a graph of aggregated contributions to the project whereby individual work is not identified. In this case, the visualization supports the foundation for shared-regulation. Since it is not possible to make comparisons between individual contributions, this feature is classified as mirroring support. The second approach creates the context for co-regulation and provides awareness support as it shows individual graphs per contributing member and permits comparison across all members.

**News feeds** show a list of recent activities related to the people or projects the user is working with or following. This feature provides mirroring support and can be used to promote shared-regulation because it shows an overview of work being completed towards the shared collaborative vision. But it can also be used to provide co-regulation for individual members watching the activity of close collaborators.

## V. Discussion and Future Work

In the previous section, we demonstrated how the enhanced 3C Model provides a more detailed approach and improved terminology for evaluating, classifying, and comparing features within a software development tool or environment. Our use of the enhanced 3C Model helps to reveal and label areas of tool support that may be lacking. For example, more guidance about pending tasks for individuals and groups could be very helpful in building awareness about a project's trajectory towards a milestone. Indeed, in our studies of software developers, we see collaborative software teams using tools such as Trello taskboards to provide additional support for co- and shared-regulation.

For future work, we plan to apply the extended model to additional software development tools, including socially-enabled code hosting tools (e.g., BitBucket), management and collaboration task boards (e.g., Trello, ZenHub), software development-focused communication tools (e.g., Slack, HipChat), and personal & team metric tools (e.g., WakaTime).

Considering that the Model of Regulation was incorporated with the 3C Model in its original form, further research is required to determine if this framework accurately represents collaborative software development processes.

Moreover, when discussions over a broader context of collaboration in software engineering are included, such as virtual teams and global software development, constraints like time zones, culture, and language must be taken into account in order to refine the framework. Future work will include application of the theory of regulation using software teams as the unit of analysis. We expect that the model will evolve as we conduct these studies and learn more about the nuances of regulation and the tools that support it in software engineering.

Finally, van der Hoek *et. al* [18] also propose a Continuous Coordination model that integrates informal and formal communication practices, paying special attention to the need for ongoing coordination in the lifecycle of a project. We emphasize that the approach we proposed can also be applied to other collaboration frameworks and models, such as the Continuous Coordination model.

*Acknowledgment:* We thank Cassandra Petrachenko, Mariel Muller, and Daniel German for providing feedback on our paper.

## References

[1] I. Steinmacher, A. P. Chaves, and M. A. Gerosa, "Awareness support in global software development: a systematic review based on the 3c collaboration model," in *Collaboration and Technology*. Springer, 2010, pp. 185–201.

[2] P. Tell and M. Ali Babar, "A systematic mapping study of tools for distributed software development teams," IT-Universitetet i København, Tech. Rep., 2012.

[3] M.-A. Storey, L. Singer, B. Cleary, F. Figueira Filho, and A. Zagalsky, "The (r) evolution of social media in software engineering," in *Proc. of the 36th Intl. Conf. on Software Engineering, Future of Software Engineering (FOSE '14)*. ACM, 2014, pp. 100–116.

[4] A. F. Hadwin, S. Järvelä, and M. Miller, "Self-regulated, co-regulated, and socially shared regulation of learning," *Handbook of self-regulation of learning and performance*, pp. 65–84, 2011.

[5] P. Layzell, O. P. Brereton, and A. French, "Supporting collaboration in distributed software engineering teams," in *Software Engineering Conf., APSEC 2000. Proc. 7th Asia-Pacific*. IEEE, 2000, pp. 38–45.

[6] M. McLure Wasko and S. Faraj, "it is what one does: Why people participate and help others in electronic communities of practice," *The J. of Strategic Information Systems*, vol. 9, no. 2, pp. 155–173, 2000.

[7] C. A. Ellis, S. J. Gibbs, and G. Rein, "Groupware: some issues and experiences," *Communications of the ACM*, vol. 34, pp. 39–58, 1991.

[8] M. A. Gerosa, "Analysis and design of awareness elements in collaborative digital environments: A case study in the aulanet learning environment," *Journal of Interactive Learning Research*, vol. 14, no. 3, pp. 315–332, 2003.

[9] P. Dourish and V. Bellotti, "Awareness and coordination in shared workspaces," in *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. ACM, 1992, pp. 107–114.

[10] C. Gutwin, R. Penner, and K. Schneider, "Group awareness in distributed software development," in *Proceedings of the 2004 ACM conference on Computer supported cooperative work*. ACM, 2004, pp. 72–81.

[11] H. Fuks, A. B. Raposo, M. A. Gerosa, and C. J. Lucena, "Applying the 3c model to groupware development," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 299–328, 2005.

[12] D. W. Johnson and R. T. Johnson, *Cooperation and competition: Theory and research.* Interaction Book Company, 1989.

[13] J. Herbsleb and D. Moitra, "Global software development," *Software, IEEE*, vol. 18, no. 2, pp. 16–20, Mar 2001.

[14] A. Hadwin, M. Miller, and E. Webster, "Promoting and researching adaptive regulation in cscl: Scripting, visualization, and awareness tools," *Conf. of the European Association for Research on Learning and Instruction*, September 2013.

[15] S. Järvelä and A. F. Hadwin, "New frontiers: Regulating learning in cscl." *Educational Psychologist*, vol. 48, no. 1, pp. 25–39, 2013.

[16] B. J. Zimmerman, "Academic studing and the development of personal skill: A self-regulatory perspective," *Educational psychologist*, vol. 33, no. 2-3, pp. 73–86, 1998.

[17] A. Soller, A. Martínez, P. Jermann, and M. Muehlenbrock, "From mirroring to guiding: A review of state of the art technology for supporting collaborative learning," *Int. J. Artif. Intell. Ed.*, vol. 15, pp. 261–290, Dec. 2005.

[18] A. van der Hoek, D. Redmiles, P. Dourish, A. Sarma, R. Silva Filho, and C. De Souza, "Continuous coordination: A new paradigm for collaborative software engineering tools," in *Workshop on Directions in Software Engineering Environments*. IET, 2004, pp. 29–36.