Contextualization of regulatory processes in programming learning

Phases, Areas and Processes for Regulated Learning/Contextualization in Programming					
	Type of Regulation	Areas for Regulation			
phases		Cognitive / Socio-cognitive	Motivation/Affection	Behavior	Context
Phase 1. Forecasting, planning and activation **Planning before Coding**	Self-regulation	setting goal Evaluating prior knowledge about content activating metacognition **Detecting important problem items before coding.** **Drawing the solution before coding.** **Using diagrams to explain solution design and its connections to code.** **Practicing and understanding the fundamentals before starting to program.**	Preparing to reach the goal judging self-efficacy Realizing the difficulty of the task Realizing the value of the task activating interest **Viewing Obstacles from a Positive Perspective on Programming**	Planning time and effort Planning self-monitoring of behavior. **Using Kanban for scheduling task planning.** **Using Kanban for self-assessment of theoretical and practical programming content.**	realizing the task perceiving the context **Establishing strategies for executing and monitoring programming tasks.**
	co-regulation	Establishing shared understandings of task demands, negotiating problem meaning, and setting goals. Interacting with team members about actions to be taken. **Establishing a shared understanding of programming concepts.**	Anticipating good relations in the group. Encouraging future participation and interactions. **Use motivating phrases in a good mood in programming.**	Creating workflows to achieve goals, including setting timelines. Negotiating the division of labor **Using Scrum to plan collaborative programming tasks.**	Negotiating and describing roles according to the student's profile. Organizing the team (communication protocol/rules of engagement). **Choosing groupware technologies for programming.** **Planning a collaborative programming script.**
Phase 2. Monitoring **Monitoring during Coding and Testing**	Self-regulation	Monitoring cognition and mete-cognition **Understanding programming patterns.** **Experiencing programming patterns.** **Monitoring problem solving in programming.**	Monitoring of motivation and affect **Monitoring motivation in programming.**	Monitoring effort, time use, need for help Self-observing behavior **Using Kanban for monitoring scheduling tasks.**	Monitoring changing tasks and context conditions **Monitoring individual programming context.**
	co-regulation	Monitoring shared understanding.	Monitoring group motivation for participation and interactions.	Tracking group goals and progress.	Monitoring the change of functions and communication protocols.

		Monitoring the general processes of the group. Accompanying the advancement of knowledge. Detecting errors and checking plausibility. Detecting socio-cognitive conflicts in the group. **Understanding programming patterns together.** **Experiencing programming patterns together.** **Monitoring collaborative problem solving in programming.**	Detecting socio-emotional group conflicts. **Keeping track of the group's commitment to programming**	Using workflows to monitor the progress of activities. **Using Scrum to track collaborative programming tasks**.	Following rules of engagement. **Monitoring the context of collaborative programming**.
Phase 3. Control **Coding and Testing**	Self-regulation	Selecting and adapting cognitive strategies for learning, thinking **Adapting programming patterns.** **Combining programming patterns.**	Selecting and adapting strategies to manage motivation and affect. **Reducing Anxiety in Programming.**	Increased/decreased effort persisting/giving up Help seeking behavior **Using Kanban for Task Management in Programming**	Changing or renegotiating tasks Changing or leaving the context **Acting in the individual context of programming.**
	co-regulation	Communicating with team members about actions being taken. Making collaborative plans to achieve goals, including selecting socio-cognitive strategies. Discovering the type of collaboration. interaction to solve the problem along with the objectives. Moving forward and explaining solutions. Coordination of socio-cognitive conflicts. Tracking the overall progress of group solutions. Facilitating criticism and building the perspectives of others. **Subdividing the computational problem.** **Analyzing and building third-party software artifacts.**	Controlling the quantity and quality of group participation and interactions. Providing feedback on group participations and interactions. Avoiding and controlling socio-emotional conflicts in the group. Promoting respect by criticizing the other's point of view. **Promoting participation in programming.** **Developing trust relationships in programming.**	Seeking teacher help when a conflict of ideas fails to reach consensus Managing workflows. **Using Coding DOJO (Kata) in introductory programming.** **Using Coding DOJO (Randori) in introductory programming.**	Controlling group roles and communication protocols Providing feedback on group roles and communication protocols. **Analyzing pros and cons in programming.** **Working in the context of collaborative programming.**

		Coding together from past experiences.			
Phase 4. Reaction and Reflection **Reflections on Program Coding**	Self-regulation	cognitive judgments Critical thinking and metacognition **Learning from errors and successes in programming.**	affective reactions Intrinsic and extrinsic goals, task value, control beliefs, self-efficacy and test anxiety. **Reflecting on student motivation in programming.**	choice behavior Effort regulation Seeking help Study time/environment **Using Kanban to reflect on scheduling tasks.** **Reflecting on the pros and cons of Kanban for individual progress in scheduling.**	Assessment of tasks Context assessment Peer learning, study time/environment **Reflecting on pros and cons in programming.** **Reflecting on the context of individual programming.**
	co-regulation	Reflecting and repairing shared understanding. Evaluating current joint solutions. Reflecting on different points of view. Monitoring the results of actions and evaluating success in solving the problem. Reflecting on group goals, progress and achievements. Making adaptations to collaborative goals, plans, or strategies. **Reflecting on different computing solutions.**	Evaluating the emotional aspects of group members with regard to mutual respect and engagement in group activities. Group evaluation regarding the number of interactions and how many different people interacted. Preventing lack of participation and interactions. **Reflecting on the motivation of the group in programming.** **Reflecting on trusts in programming.**	Reflecting on the group's goals and progress. Reflecting on workflows to check productivity. Adapting workflows. **Reflecting on the pros and cons of Scrum for collaborative programming**.	Reflecting on group roles and communication protocols. Adapting group functions and communication protocols. **Reflecting on the context of collaborative programming.**