International Journal of Arts & Sciences,
CD-ROM. ISSN: 1944-6934 :: 09(03):277–284 (2016)

TEAM IMPROVEMENT AND LEARNING IN SELF-ORGANIZING CONTEXTS

Stefan Lipowsky and Joel Schmidt

University of Applied Management, Germany

Agile methods for self-organisation of teams have rapidly developed in the IT industry over the last few years and have expanded into other industry areas as well. Understanding of teams and team effectiveness is currently shifting in response to the new challenges in complex working environments and rapid technological developments. This paper makes an analysis of new challenges facing teams and provides a synthesis of new requirements for team effectiveness, concentrating on cycles of continuous improvement and team learning.

Keywords: Self-organisation, Agile groups, Lean management, Team learning, Continuous improvement.

Introduction

Lean and agile methods are becoming increasingly common and are used as an accepted means of organizing work in IT projects. These methods are based on self-organizing teams in order to effectively deal with the two major challenges of digitalization: speed and complexity. This article examines agile methods and consequently agile teams, exploring why it is necessary to change the way of working together, definitions of agile methods and how agile teams function (under which preconditions and with which skillset).

Why Do We Need to be *Agile*?

During the four years leading up to 2015, Apple's revenue *growth* exceeded the total revenue of the Allianz Group in 2015, the largest insurance group worldwide. There is no doubt that both companies are successful, however the rapid growth of Apple is even more evident in a comparison over the last 10 years: while Apple's revenue indicated a growth of 1100%, Allianz Group's revenue grew only by 30% ('statista.de', 2016). The rapid success of Apple is due to the dynamic quality of the digital technology market. The incredible speed of digital technologies and the impact of its products are apparent when compared to other technological advancements in the past: it took 34 years for the radio as a new technology to reach 50 million users; for Instagram, it took only 4 months. Furthermore, IT systems are increasingly becoming connected to each other and to devices and machines – often referred to as the 'internet of things' (Gimpel & Böglinger, 2015).

Given the high connectivity and comprehensive impact and dissemination of products, digitalization changes the life of nearly everybody. Consider, for example, the music industry: during the last 15 years

the revenue from music CDs decreased by more than 85% from 300 Mio CHF to 40 Mio CHF in Switzerland, due to the low cost availability of music through burned CD copies, easy exchange of thousands of songs by mobile hard discs and the upcoming streaming services. While the supply of music to the end user immensely increased accessibility (such as Apple Music providing access to 30 million songs), the average income of musicians significantly decreased, and options to earn money with music are very often digital productions. The example of the music industry, with its massive changes in how music is consumed and produced, illustrates how changes in digitalization can impact and affect even the culture of a society (Chassot, 2016).

The examples above provide evidence of how changes in digitalization can fundamentally impact the market, especially considering the factors of complexity and speed. *Complexity* increases as systems become more involved and as more devices are connected. In terms of *speed*, changes occur so quickly that time to market becomes an increasingly important factor for the survival of a company. Although market changes resulting from complexity and speed are revolutionary for technical companies and for technical products, classical industries are also being impacted: Allianz is trying to achieve faster customer processes by introducing agile methods on a wide base (Maier & Palan, 2016), by reducing maximum project length to 100 days and the share wearing ties to 10%. In response to the changes and demands in the market place relating to complexity and speed, IT projects are increasingly turning to lean and agile methods as an effective solution. These methods are based on dynamic, self-organizing teams. To establish and to support self-organization, agile teams have to be enabled to improve themselves: the team must be able to reflect, to recognize and to revise their collaboration and their effectiveness.

What It Means to be *Agile*?

Historically, paradigms of organizational structure and design have changed approximately every 30 years during the 21st century: during the 30s Henry Ford optimized car manufacturing with the ideas of Frederick W. Taylor's scientific management; during the 60s Taiichi Ohno, with the help of Wiliam E. Deming, developed the Toyota Production System and introduced lean management; during the 90s Ken Schwaber and Jeff Sutherland developed agile management as an advancement of lean management for IT development teams.

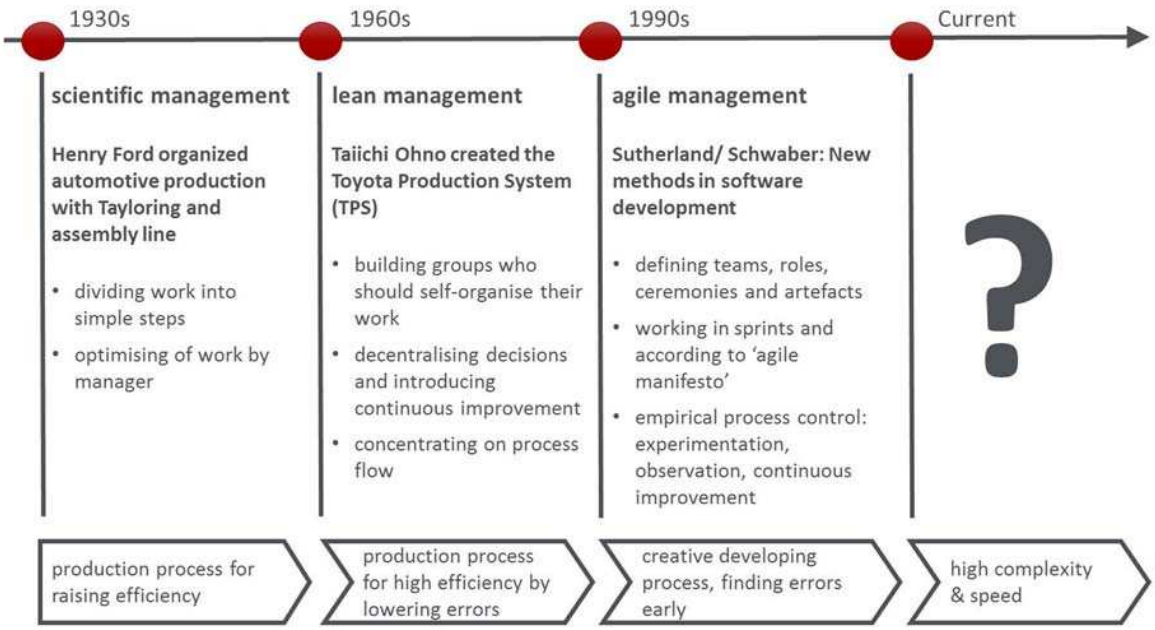

Figure 1. History of lean

Using the approach of "scientific management", Frederick W. Taylor attempted to achieve the optimization of mass production processes with a process-oriented approach – originally referred to as *task management*. Taylor's scientific management is based on four principles: "The development of a true science. [..] The scientific selection of the workman. [..] His scientific education and development. [..] Intimate friendly cooperation between the management and the men." (Taylor, 1911, p. 62). The goal was effectiveness of a company, not the effectiveness of people: "In the past the man has been first; in the future the system must be first" (Taylor, 1911, p. 3). Taylor separated work into intellectual and physical parts, where managers were responsible for the intellectual parts, and divided the physical parts (done by the workers) into very small units and steps, with constant striving for optimization. Consequently, this approach made it possible to hire very poorly educated (and therefore cheap) workers, since the task was to perfom only several, optimally described and optimized physical movements (manual labor). Workers had to follow exactly the manager's standardized work procedures receiving only a very low wage. On the level of organizations, Taylor was successful and increased company effectiveness. Ford integrated Taylor's scientific management in the 30s with his invention of the assembly line for simple, standardized automobiles, and this form of industrial mass production has influenced the world economy for decades.

After the Second World War, as Japan tried to catch up to the American economy, Taiichi Ohno created the Toyota Production System (TPS). Basically this approach uses a customer-centric acting model, which considers the employee as the greatest asset of the company, and takes advantage of the Kaizen-paradigm of continuous improvement. Ohno tried to eliminate waste by implementing just-in-time production and continuous improvement of the production processes. Therefore, he formed working teams who were supposed to organize their work by themselves and initiate regular meetings to reflect on work processes, achievements and errors, and also included the responsibility to improve these processes through learning and small, iterative changes. In this context, managers acted as servant leaders, and focused on enabling their staff and removing impediments (Brunner, 2014, p. 103 ff.). Standardization (as an advantage within all mass production processes), is a central goal of both TPS and Scientific Management, but in the case of TPS standardization is not developed by the manager alone, but is rather in a state of continuous development and improvement by the team. With the knowledge that the maximal throughput of a system is arrived at with a capacity utilization of 80% (Reinertsen, 2009, p. 59 ff.), the TPS is based on a pull principle, meaning that the worker start new tasks only after finishing previous tasks. Another innovation of TPS is the invention of Kanban, as a very strong method of visualization, where actual tasks are represented by a card on a task board, running through the status phases of "to do"; "in work"; and finally "done". Given these concepts, processes and methods, the workers in TPS have much more responsibility within the process than in Ford's assembly line, and consequently workers are much more respected and valued. Still, TPS has been created only to optimize a production process.

Closer to the turn of the century in the 90s, with the upcoming PC and client-server software architecture and new G4 programming languages, market complexity was becoming decreasingly manageable when using a traditional, detailed planning approach. One very important reason for this was the modification of business during long lasting projects, which includes the increasing complexity and uncertainty of planning and managing overhead in projects. In reaction to these changes, Ken Schwaber and Keith Sutherland developed a method to deal with uncertainty and complexity (Schwaber & Sutherland, 2012, p. 3 ff.). They transformed Ohno's concept of lean management applying it to the industry of software development with two major changes: refinement of team methodology, and reorganization of software development processes.

Firstly, Schwaber and Sutherland recognized Kanban as a very powerful tool (for the effective organization and operation of teams) and worked to develop and improve it further into a new method called "Scrum", which is today the most frequently used agile method framework. The method retains and even strengthens the responsibility of the team, dividing the responsibility among three key roles: line manager (disciplinary leader), product owner (responsible for the product that should be developed), and scrum master (responsible for the collaboration process) (Schwaber, 2003, p. 25/53 ff.). As an important nuance, the tasks describe only the desired outcome ("what"), not the way it is to be carried out ("how") – therefore, in Scrum the tasks are called "stories".

Secondly, Schwaber and Sutherland reorganized software development processes into a series of short iterations, called sprints (e.g. lasting two weeks). After each sprint current results in running software should be presented, even with minimal functionality. Further plans are created only for the next sprint, in order to keep flexibility for changes. (Gloger, 2011, p. 19 ff.). Schwaber and Sutherland kept self-organization, responsibility and continuous improvement, and they added the definition of roles, iterative planning/developing/reviewing process. This is a necessary feature, since software development is not a production process which is achieved with high efficiency and quality by lowering errors. Software development is a creative process, where at the start it is mostly undefined; identification of the exact outcome, including experiments and failures, are necessary parts of the process.

A common feature among all these ideas, from scientific management, TPS, to lean and agile methods, is the emphasis on the organization instead of on individuals. Consequently, organizational change becomes a critical factor in the success of an organization. The features of traditional projects in matrix organizations result in two main challenges: One characteristic is that employees are embedded in a hierarchical line organization, and at the same time assigned to one or more projects. This means that a worker can be assigned to more than one superiors resulting in potential conflicts between competing needs. Another characteristic is the switching context, which occurs when working on multiple projects, requiring a lot of time and increases the risk of errors – this produces waste for the company and stress for the employee, along with often lower performance of both employee and project.

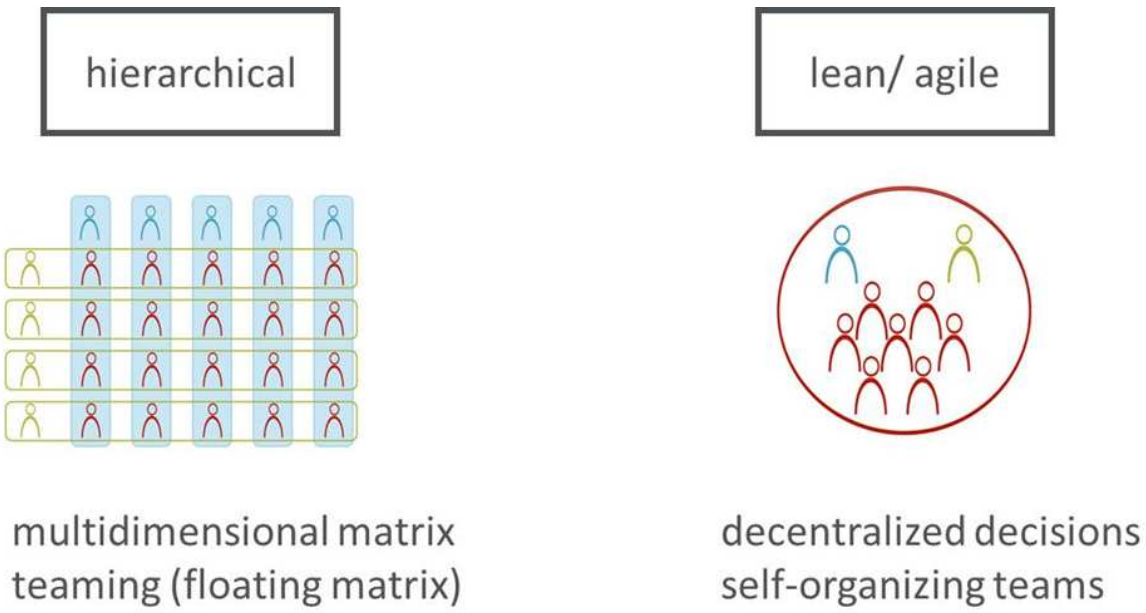

Figure 2. Hierarchical vs. lean structure

Based on these two challenges facing matrix organizations, it is highly recommended to implement agile groups as an alternative, since agility is built on self-organizing groups, using group dynamics. Furthermore, agile teams need to be as independent as possible, forcing them to become multifunctional as most of the tasks are accomplished by themselves (such as engineering, manufacturing, testing and deploying). The best way to do that is to assign co-located people 100% to a project (Larman & Vodde, 2009, p. 193 ff.). Teams need to be co-located, since direct communication is a crucial success factor for good teamwork (Pentland, 2012). If co-location is impossible, teaming (Edmondson, 2012, p. 13 ff.) describes a way to empower teams independently from their organizational embedding, offering an improvement for matrix-based project teams and part-time assigned team members. Teaming may lower the disadvantage of these suboptimal preconditions, but it is not agile.

How Do *Agile* Teams Function?

In Scrum, continuous improvement cycles are part of product development process, as well as in team collaboration as a meta-cognitive process. Product development is organized in iterative sprints. A sprint starts with a common planning session, where the product owner explains the vision of the product and a list of the most important stories which need to be realized, presenting the "product backlog"; the team asks for understanding, discusses how to do and commits the stories it is convinced to finish during the actual sprint, creating the "sprint backlog". In this way sprint planning is divided into two parts: presentation of the stories by the Product Owner ("what") in *planning part 1*, and definition of implementation and commitment, how much can be done ("how") in *planning part 2*. After two weeks of development, the team presents running software during the review meeting to the product owner. The owner then decides if the software fulfils the stories and if the stories can be approved. Some stories might be satisfactory, while others might have to be revised or improved, and some ideas might come up as new – the product owner refines the product backlog and re-prioritizes all stories before the next sprint planning. Therefore in every sprint cycle feedback is used to improve the work (Gloger, 2011, p. 155 ff.). On the meta-level, as a type of metacognition, team collaboration is examined during a retrospective meeting at the end of each sprint. During that meeting, the team strives to find working and non-working processes, using creativity techniques, in order to achieve deeper understanding of the problems with root-cause-analysis to create a list of solutions. At the end of the retropective, the team prioritizes the solutions and commits action points to be done respectively changed during the next sprint (Gloger, 2011, p. 179 ff.).

Self-regulated learning (SRL) is described by Zimmermann as a process of three phases: forethought, performance, and self-reflection (Zimmerman, 2000, p. 16 ff.). A slightly different model is offered by Winnie and Hadwin, who describe a four-step circle of studying: "task definition, goal setting and planning, enactment, and adaptation" (Winnie & Hadwin, 1998, p. 279), and Pintrich with his phases of forethought, planning and activation/ monitoring/ control/ reaction and reflection (Pintrich, 2000, p. 453). Whereas Pintrich emphasizes the monitoring and controlling aspect by dividing these into two separate phases, Winnie and Hadwin focus more on the planning and committing phase. Both perspectives could be helpful for future research, but our current article combines Zimmerman's model of self-regulation to the agile concept of a continuous self-improving sprint team.

According to Zimmerman (2000) self-regulation begins with a *forethought* phase, including sub-processes of task analysis, goal setting and strategic planning. This has its correspondence in planning (I & II) of a sprint cycle. In this initial phase it is often helpful to integrate Winnie and Hadwin's (1998) concept by adding a task definition phase, making it equivalent to the entire content of planning I in a sprint cycle. Zimmerman's second phase is *performance*, which is similar to the development phase of the sprint. The third phase of *self-reflection* is accomplished during a sprint cycle in two ways: review (showing the result to the product owner) and retrospective. Although retrospective can be considered as an independent cycle of self-regulation on a meta-level, in sprint cycle analysis the focus is on the review meeting, including the backlog refinement made by the product owner. This is possible since the product owner is defined as a part of the development team.

According to Zimmerman (2002), self-motivational beliefs, self-observation, and self-reaction, are necessary for optimal self-regulation, and are integral components of the three phases of self-regulated learning. Self-motivational beliefs are connected to the forethought phase and are represented by self-efficacy, outcome expectations, intrinsic interest and goal orientation. Within the performance phase, self-recording and self-experimentation (or self-observation) are needed. For the phase of self-reflection, self-reaction (consisting of self-satisfaction and positive affect and adaptive responses) is an important activity. Additional, prerequisites for effective self-regulation have also been identified, such as error culture (Lipowsky & Schmidt, 2016).

In order to effectively transfer the concepts and advantages of self-regulation to agile teams using Scrum, it is necessary to keep in mind two dimensions of adaptation (see Table 1). First, there are two continuous improvement cycles in Scrum: product development itself, and retrospectives on a meta-level.

Second, it is necessary to distinguish between the individuals working within the group and the group itself. Adaptation of self-regulation on individual level for software development level and retrospective (S-I; M-I) has to be done by mapping the special requirements, skills and preconditions of self-regulation to the activity. In both cases, learning (as mentioned in Zimmerman's work) is not the activity. In software development, the topic would be a task, such as creating a defined functionality or testing a software screen, rather than understanding and remembering or learning content found in educational contexts. In a retrospective session, the topic of examination could be dealing with overload of sudden requests or communication failures rather than a topic of attention focusing as found in learning environments.

Table 1. Two dimensions of adaptation

	sprint level (product development)	**meta level (retrospective)**
group level (team)	S-G	M-G
individual level (person)	S-I	M-I

Consequently, a basic outcome of this current article is the identification of self-regulation as a useful concept for continuous improvement in agile teams, as it offers a framework for the transfer of learning experience to developing agile groups. Eventhough working topics are different, the concepts can be effectively adapted and integrated. Therefore, the preconditions (such as full-time assignment, open error culture, servant leaders and enabled teams) and the skillset (including abilities such as planning, working in multidisciplinary contexts, pairing and retrospectives) to work in agile, self-regulation groups has to be defined in a way that concepts of pedagogical self-regulation improvement can be used in a specific working context. A possible solution is illustrated in Figure 3 and achieved by integrating and connecting similar problems of both perspectives.

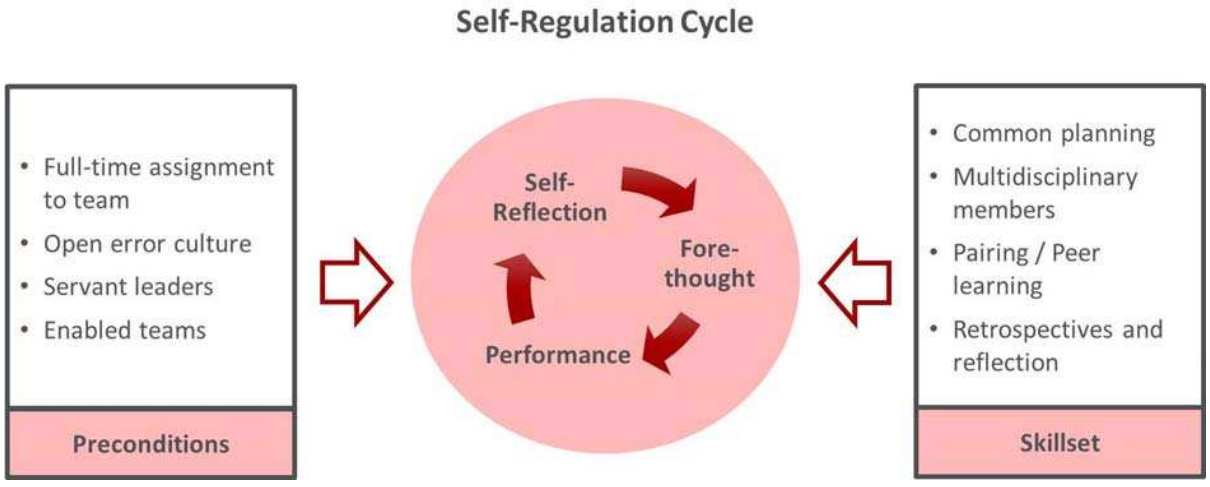

Figure 3. Preconditions and skillset for the self-regulation cycle

In conclusion, central questions for future research and investigations dealing with similar topics are offered as follows: How to develop and to support the self-regulating aspects of agility on individual and especially on group level? What is needed for self-regulating agile teams and how can it be developed?

Literature

1. Brunner, F. J. (2014). *Japanische Erfolgskonzepte: Kaizen, KVP, Lean Production Management, Total Productive Maintenance, Shopfloor Management, Toyota Production System, GD^3 Lean Development* (3rd ed.). München: Hanser.
2. Chassot, S. (2016, July 13). Musikern strömen die Einnahmen davon. *Neue Zürcher Zeitung*. Zürich.
3. Edmondson, A. C. (2012). *Teaming: How Organizations Learn, Innovate, and Compete in the Knowledge Economy*. San Franscisco, CA: Jossey-Bass.
4. Gimpel, H., & Böglinger, M. (2015). Digital Transformation: Changes and Chances – Insights based on an Empirical Study. Project Group Business and Information Systems Engineering (BISE) of the Fraunhofer Institute for Applied Information Technology FIT: Augsburg/Bayreuth.
5. Gloger, B. (2011). *Scrum: Produkte zuverlässig und schnell entwickeln*. München: Hanser.
6. Larman, C., & Vodde, B. (2009). *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale-Scrum*. Boston, MA: Addison-Wesley Pearson Education.
7. Lipowsky, S., & Schmidt, J. (2016). Error-Culture and Self-Regulation. *Journal Association 1901 'SEPIKE'. Social Educational Project of Improving Knowledge in Economics*, (13), 90–95.
8. Maier, A., & Palan, D. (2016). Allianz goes Google. *Manager Magazin*, *20*(6), 50–56.
9. Pentland, A. S. (2012). The New Science of Building Great Teams: The chemistry of high-performing groups is no longer a mystery. *Harvard Business Review*, *90*(4), 60–70.
10. Pintrich, P. R. (2000). The Role of Goal-Orientation in Self-Regulated Learning. In M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of Self-Regulation* (pp. 452–502). New York, NY: Academic Press.
11. Reinertsen, D. G. (2009). *The Principles of Product Development Flow. Second Generation Lean Product Development*. Redondo Beach, CA: Celeritas.
12. Schwaber, K. (2003). *Agile Project Management with Scrum*. Redmond: Microsoft.
13. Schwaber, K., & Sutherland, J. (2012). *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, and Leave Competitors in the Dust*. Hoboken, NJ: John Wiley.
14. statista.de. (2016).
15. Taylor, F. W. (1911). *The Principles of Scientific Management (Transcribed in 2004 by Charles E. Nichols)*. Oxford: Project Gutenberg Literary Archive Foundation.
16. Winnie, P. H., & Hadwin, A. F. (1998). Studying as Self-Regulated Learning. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Metacognition in educational theory and practice* (pp. 277–304). Hillsdale, NJ: Lawrence Erlbaum.
17. Zimmerman, B. J. (1990). Self-Regulated Learning and Academic Achievment: An Overview. *Educational Psychologist*, *25*(1), 3–7.
18. Zimmerman, B. J. (2000). Attaining self-regulation: A social cognitive perspective. In M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of Self-Regulation* (pp. 13–39). New York, NY: Academic Press.
19. Zimmerman, B. J. (2002). Becoming a Self-Regulated Learner: an Overview. *Theory Into Practice*, *41*(2), 64–70.