Chaordic Learning: A Case Study

Stephan Krusche
Technische Universität München
Munich, Germany
krusche@in.tum.de

Bernd Bruegge
Technische Universität München
Munich, Germany
bruegge@in.tum.de

Irina Camilleri
Technische Universität München
Munich, Germany
irina.camilleri@tum.de

Kirill Krinkin
Saint Petersburg Electrotechnical University
St. Petersburg, Russia
kirill.krinkin@fruct.org

Andreas Seitz
Technische Universität München
Munich, Germany
seitz@in.tum.de

Cecil Wöbker
Technische Universität München
Munich, Germany
woebker@in.tum.de

***Abstract*—Software engineering is an interactive, collaborative and creative activity that cannot be entirely planned. Inspection and adaption are required to cope with changes during the development process. Software engineering education requires practical application of knowledge, but it is challenging and time consuming for instructors to evaluate the creation of innovative solutions to problems. Current higher education practices lead to a multitude of rules, guidelines and order. Instructors see deviations of students as failures and limit the creative thinking processes of students.**

In this paper we describe chaordic learning, a self-organizing, adaptive and nonlinear learning approach, to stimulate the creative thinking of students. Instructors provide structure and guidance, but also integrate freedom for self-organization and self-guided learning and embrace innovation and creativity. Deviations are seen as opportunities and failures as possibilities for students to learn and improve. We introduced chaordic learning into a games development course and a joint advanced student school and describe the chaordic process of these courses as case studies. Students in these courses report about an increased intrinsic motivation, a higher level of self-organization and more room for creativity leading to an improved learning experience and more fun.

***Keywords*-Experiential Learning, Agile Methods, Creativity, Chaos, Order, Self-guided Learning, Self-organization**

I. Introduction

Software engineering (SE) has undergone several fundamental shifts since the term was first proposed in the 1960s when software systems became larger and more complex to develop [1]. Initially, the focus was on defined process control and ordered processes to mimic established manufacturing processes [1]. In recent years, the emphasis changed towards empirical process control that embraces frequent inspection and adaption to react to changing environments [2]. Software development consists of experimental knowledge work where creativity is important [3]. SE is the process of creating software solutions that have - in their entirety - not been developed before.

SE education requires practical application of knowledge [4], [5], in particular interaction and collaboration [6]. Current higher education practices lead to a multitude of rules, guidelines and order, which in turn "makes creative teachers and students feel less and less in place" [7]. University courses focus too much on analytical and logical thinking, because this is easier to teach and to grade.

Instructors follow a defined teaching approach and see deviations of students as failures. They prematurely set up structures (e.g. assignment requirements, syllabus, instructions) without taking time to determine the desired learning outcomes and how the process of learning should be conducted. When all educational activities are well defined, students only follow instructions, but do not use their creative and intuitive thinking processes and do not learn to act themselves.

Some instructors even sanction creativity if students' solutions to exercises do not follow correctness criteria, even if they include innovative aspects. In such cases, there is no room for self-organization, as instructors fear this might lead to chaos, in particular if students are unexperienced. This leads to a gap between skills that current SE education provides and skills that are required in industry and research.

Chaordic learning is an approach to overcome these problems by balancing education between chaos and order. Instructors define the learning environment and high-level learning goals to provide structure and order. Students have the freedom to choose the concrete learning activities and the solution approaches to given problems to allow creativity, innovation and self-guided learning. Instead of defining all learning steps, instructors provide guidance in this approach and give feedback to the students' learning progress. Examples of courses following this idea are capstone courses as described in [8], [9], and [10]. Capstone courses provide students with a real-life experience to prepare them for their future career [11].

The paper is structured as follows. Section II provides background information, defines the term chaordic, describes the learning organization that focuses on individuals, and presents agile methods as concrete practice for a chaordic development process. In Section III, we define the chaordic learning approach, which is based on a design process, and we present its properties and its benefits. Section IV shows two case studies, in which we applied chaordic learning, a games development course and a joint advanced student school. In Section V, we discuss other chaordic learning courses, constructivism and design thinking as related work. Section VI concludes the paper.

II. Foundations

In the 1960s, software systems became larger and more complex to develop [1]. Projects failed because development concepts and methods were missing and teams worked together in rather chaotic and unstructured ways. The term software engineering (SE) "was deliberately chosen as being provocative, in implying the need for software manufacture to be [based] on the types of theoretical foundations and practical disciplines[,] that are traditional in the established branches of engineering" [12], [1]. Detailed process models emerged that describe the process of engineering software, resulting in a structured software process. The aim was to bring **order and control** into the development approach following strict rules and avoiding deviations, which were seen as errors that need to be corrected.

In the following years, software developers increasingly recognized that too much order and control is counterproductive and that the essence of SE is to deal with changes: defined process models are not capable of addressing this need [13]. SE consists of experimental knowledge work where creativity is important, including unexpected events, incidents and uncertainty [3]. Software development is a complex process with random variables, that cannot be defined completely deterministic: "It is typical to adopt the defined (theoretical) modeling approach when the underlying mechanisms by which a process operates are reasonably well understood. When the process is too complicated for the defined approach, the empirical approach is the appropriate choice" [2].

As response, agile methods emerged with the philosophy that software development should follow an empirical approach: still structured, but not entirely planned, and thus providing more freedom to adapt to changes. Deviations, errors and failures are seen as opportunities to inspect, adapt and improve the methodology. If random variables such as uncertainty and change are allowed, the process control is stochastic (nondeterministic) and allows **chaos**. Stochastic control is not solved analytically and deals with the existence of uncertainty and chaos [14]. Empirical process control balances between **chaos** (stochastic control) and **order** (defined control).

A. Software Engineering Education

Several pedagogic theories have been developed and integrated into education. Educators recognized that SE education requires practical application of knowledge [4], [5], in particular interaction and collaboration [6]. Experiential learning is a methodology in which educators engage with learners in direct experience to increase knowledge, develop skills, and clarify values [15].

Problem-based orientation allows students to identify what they know, what they need to know, and how and where to access new information that leads to the resolution of a particular problem [16]. It is one important aspect in capstone courses [17].

The introduction of computers in education enables students to learn through the delivery of content and instructions via computer mediated activities, digital and online media [18]. It promotes simultaneous, independent and collaborative learning experiences.

In cooperative learning approaches, students work in groups to complete tasks collectively towards a common goal and the educator's role changes from giving information to facilitating learning [19].

Active learning is an educational approach to increase involvement and excitement with the subject being taught. Instead of learners acting as receivers of knowledge by passively listening to lectures, active learning puts the emphasis on developing learner skills and engaging them in activities.

B. Chaordic

In 1995, Hock first coined the term *chaord*[1] [20]. This neologism is the combination of the words **cha**os and **ord**er, meaning a state in between that adapts the principles and properties of both. Hock shares his experiences in managing organizations and concludes that a **chaordic** approach is required in complex situations in order to facilitate innovation and creativity.

> "By chaord, I mean any self-organizing, adaptive, non-linear complex system, wether physical, biological, or social, the behavior of which exhibits characteristics of both order and chaos or loosely translated to business terminology, cooperation and competition." [20]

Complex systems arise and thrive on the edge of chaos with just enough order to give them pattern [21]. Chaord is a universal concept that can be applied to different systems and environments. The core principles of chaos and order are essential for a chaordic system. There is an aversion to disorder in today's world [22], because disorder means a loss of control. Since chaos can lead to disorder, it is important to overcome the fear against disorder to make a positive use of chaos, which is central to the idea of chaord. Chaos is used to increase responsiveness and adaptivity to the environment and order is used to keep the system stable and to make sure that its boundaries are not violated.

Chaordic means organizing and shaping a system in a way to be able to adapt to a changing environment. It can be applied to different types of contexts [21]:

- *Organization:* working environment that is governed by both structure and chaos leading to more innovation
- *Business:* organization that harmoniously blends characteristics of competition and cooperation
- *Leadership:* combining induced and compelled behavior in the relationship between a leader and a follower
- *Education:* approach that seamlessly blends theoretical and experiential learning

[1]"Chaord" is a substantive, "chaordic" is the corresponding adjective.

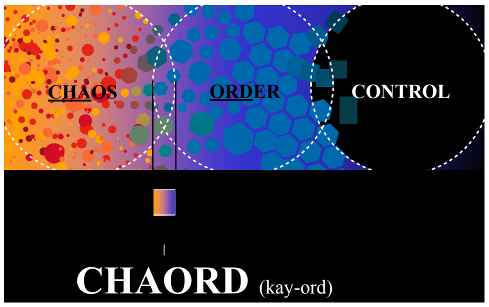

Fig. 1. Chaord is a state between chaos and order (adapted from http://www.chaordic.org)

Organizational forms are still following industrial principles defined more than a century ago [23]. Structure and order are essential in these traditional organizational forms and are often seen as more important than individuals which are treated like gearwheels that either function or get replaced [20]. A chaordic organization however includes characteristics of chaos and order at the same time. It focuses on the individual people in the organization and allows them to innovate and to think out of the box [24]. How decisions are managed plays an important role in governing the organization effectively [21]. Governance in chaordic organizations is distributed to many different people and stakeholders. This is contrary to the traditional, more hierarchical view of organizational structure.

A chaordic system, or a chaordic organization in particular, is defined by the following characteristics [20]: power and function of the system are distributed. There is no single person controlling the whole system or governing all organizational functions. The system is self-organizing: there is no external system or individual required to make the original system last and function correctly. Collaboration and competition are both integrated into the system. The system is malleable and durable: it can adapt while staying successful in a changing environment. The system is owned cooperatively and equitably: everyone is involved in the system's continuing success.

C. Learning Organizations

The learning organization is "a form of organization that enables the learning of its members in such a way, that it creates positively valued outcomes, such as innovation, efficiency, better alignment with the environment and competitive advantage" [25]. While there used to be a divide between individuals and their associated organizations, there is now a consensus that being successful requires collaboration with one another [26]. Individuals play a more important role in today's organizations and especially in how those organizations learn and adapt. This has become important since workers increasingly look for more flexible working arrangement which in turn can increase their satisfaction and commitment [27].

By adapting to the needs of individuals and helping them produce knowledge, organizations can profit themselves. They need to continuously listen to the wishes of their people and transform accordingly. The individual's knowledge can be used to improve organizations or the products and services they provide. The success of the individual is directly linked to the success of the organization as a whole, because knowledge is created when people interact with each other [26].

Learning is an activity of interdependent people working collaboratively with each other [28]. A learning organization is a place where people are "continually discovering how they create their reality. And how they can change it" [29]. Companies applying the principles of learning organizations act between chaos and order, so they are in a chaordic state [20]. Enabling people to learn in these organizations depends on two key factors. Organizations have to provide the possibility and the options for their people to advance themselves, while also providing the culture to learn in the first place [29]. When organizations follow these principles, both the organization itself and its individuals benefit from the chaordic approach. By ensuring that the individual can learn in a chaordic environment, the organization itself can become a learning organization and can improve its efficiency. This is especially important to be successful in the long-run.

D. Agile Methods

In the 1990's, agile development methods emerged with the philosophy that software developement should follow an empirical process model. The empirical model for Scrum described by Schwaber [30] is based on Ogunnaike's definition of a stochastic model [2]. It handles changes and failures as opportunities: a quick reaction to these can lead to advantages compared to competitors. The agile manifesto (visualized between chaos and order in Figure 2) identified new ways of developing software by moving the focus from complete order in traditional development processes towards a more lightweight methodology in between chaos and order [31].

Agile methods play an increasing role in SE [32] and in its education [33]. They allow developers to stay creative while working on complex problems and projects. A team that applies agile methods is self-organizing: the team decides about the concrete procedures, and learns from problems, risks, and failures allowing a more chaotic way of thinking. There are also members in the team, such as the scrum master, who have control over the process by moderating and negotiating with the rest of the team, leading to an ordered course of action. The acceptance of failure is an important aspect of agile methods. Realizing that a mistake has been made, learning from it and iterating on the solution to fix the mistake is of importance to the success in a complex software project.

Due to the application of empirical process control which could also be called chaordic process control, agile methods shifted the focus in SE from order (defined process control) towards chaos (stochastic process control). They include a mix of order and chaos and balance between both. On the one hand, they provide a structured framework with principles such as daily meetings with specific questions or procedures how to organize iterations. On the other hand, they are open to concrete methods, tools and workflows suggested by devel-

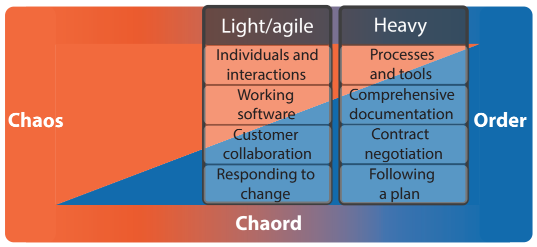

Fig. 2. Agile methods are an instance of a chaordic setup in between order and chaos.

opers, allow changes and facilitate creativity. Therefore, agile methods are a concrete instance of a chaordic setup that has a direct application and impact on the software development process. The goal is to give freedom to the development team while setting the necessary boundaries to control it and to lead it into the right direction.

III. Chaordic Learning

Chaordic learning is an educational approach that "seamlessly blends theoretical and experimental learning" [20] and includes aspects of order and chaos. In this context, structured courses with detailed instructions represent order, while experimental learning and educational innovation represent chaos.

While chaordic learning moves control to students, instructors still provide guidance[2] to ensure that students understand theory in the right way and to avoid misconceptions. Chaordic learning puts the instructor and the students on the same level to remove hierarchies and to improve collaboration. It includes the idea of cognitive apprenticeship [35]: an apprentice (the learner) observes the skills of a master (the educator) who shows how a concept works in practice. Clarifying the thinking process behind the application of the concept makes it easier for the apprentice to imitate the behavior.

Chaordic learning does not require the abandonment of structured learning, but a shift from traditional educational approaches would require creating room for chaos. This can be achieved by promoting creative problem solving experiments, where students apply principles learnt in theory or propose own new ideas. Experimentation and collaboration should be rewarded, while instructors focus on personalization, cooperation and informal learning.

Software development is the process of creating software solutions that have never been created before in the same concrete configuration. Within the context of SE education, development of creative problem solving skills is not addressed properly. We believe chaordic learning can bridge the gap between the skills current education provides and those the SE discipline requires. The encouragement of experimental learning (chaos) teaches the creative skills required in SE. A chaordic balance can be introduced in the form of project courses, where instructors provide a rough structure for the learning environment and the learning goals and students work in a self-organizing, adapting way to achieve them.

[2]Chaordic learning is not to be confused with unguided or minimal guided learning approaches that fail to improve the learning experience [34].

A. *The Chaordic Design Process*

We propose the following chaordic design process including six steps to help instructors facilitate the creation of chaordic learning environments. These steps were adapted, to an educational context, based on the work of Hock on creating a chaordic organization. It is important to point out that such environments cannot be created by instructors alone, but rather through collaboration between students and instructors, where students take charge of their own learning.

1) Purpose: Define the learning objectives of the course and a common understanding of what students aim to gain upon completion. Questions, such as "Why are you here?" often help illuminate purpose. This component is the key to motivating students - educational efforts must feel meaningful. Flexibility in course structure should allow for student input. Depending on the content, students can be asked to narrow down a set of requirements or choose technologies they feel are suitable or wish to explore.

2) Principles: Discuss the fundamental beliefs of how students and instructors shall conduct themselves in pursuit of new knowledge. An example would be emphasis on the positive outcome of learning from failure. Thus, experimentation can be seen as more valuable than a perfect solution. The grading scheme should also reflect these principles whereby students are not penalized for failure.

3) People: Identify people and institutions necessary to achieve the defined purpose. Chaordic principles encourage collaboration across program and institutional boundaries, with government and industry, both national and international. Students should consider engaging the broader community. This mindset helps students appreciate the full context of their learning experience, e.g. the context their software is being developed in. A concrete example taken from agile practices is the inclusion of the customer as participant of the team.

4) Concept: Create key guidance for interaction among the participants in a generally flat chaordic hierarchy. Being the chaordic equivalent of an organizational chart, it includes forming teams and introducing student support mechanisms (e.g. teaching assistants). Students can conceive a new organizational structure that is effective with respect to all team members and that defines how work is distributed internally. Teaching staff should be seen as facilitators and should explain who, how and when students can ask for help. Individual roles - such as those borrowed from agile practices, (e.g. scrum master) could rotate.

5) Structure: Embed the learning goals, roles and responsibilities in the course's official structure. In a higher-education context, it is the course description, assignment requirements, due dates and credits awarded for completion. Assignment requirements should be flexible to allow room for incorporating new ideas or specifying detailed requirements. The grading scheme should reflect and encourage experimentation.

6) Practices: Decide on the activities participants are required to undertake. There should be an emphasis to promote activities inspiring innovation. Mechanisms whereby instructors negotiate with students about their work should be established (e.g. by discussing status, impediments, promises). Work itself e.g. could be organized by using a software configuration management component, such as version control, with changes being reviewed through pull requests [36]. A series of meeting practices helps disseminate information. A course-wide meeting could be used for teams to share their progress, while regular team meetings allow internal discussions of problems or ideas. Teaching staff can guide students to employ best practices, but individual students or teams are allowed to choose own internal team practices.

The chaordic design process is iterative and adaptive, so a decision in each step should be reflected in all other steps. Modifications or refinements of elements may shed light on changes required in other elements. Over time, the environment is established when all elements are defined. Figure 3 shows the six steps of the chaordic design process combined in a creative figure that exemplifies its ideation aspect.

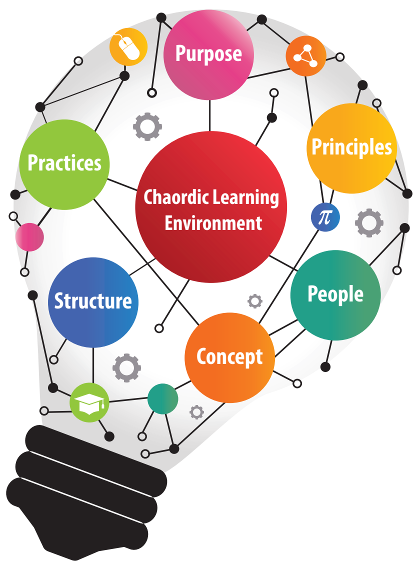

Fig. 3. The chaordic design process includes six steps to facilitate the creation of the chaordic learning environment.

The key to applying chaordic principles in an educational setting is negotiating each of these steps with the students. Students can choose to not use provided infrastructure, adopt own practices or team roles. That way each of the choices being made follows a deeper understanding of the options and the rationale behind them. Instructors act as enablers and moderators, who help students explore, rather than showing them a well-trodden path. A quote by Hock also reflects this well: "In the chaordic age success will depend less on rote and more on reason; less on authority of the few and more on the judgement of many; less on compulsion and more on motivation; less on external control of people and more on internal discipline" [21].

B. Properties

While these six steps in the chaordic design process help to create chaordic courses, they do not necessarily include chaotic properties and instructors might still focus too much on structure and order. It is important to understand the chaotic nature and to include it into the course. There are five principles of chaos [37] that are prevalent in chaordic learning:

1) Consciousness: The essential ground state of learning is mind, more than matter. Ideas are primary and drive the learning experience of students.

2) Connectivity: The learning experience is related to its environment and its context and cannot be seen as independent element. Educators and learners interact and collaborate with each other towards a common goal.

3) Indeterminacy: Learning cause and effect are intertwined. There is no single entity that can completely plan or control what exactly is learned. While the learning process is framed, no one is able to predict the exact next action in the learning process. Guidance and feedback by instructors determine the right direction.

4) Emergence: The learning experience is constantly growing more complex, more coherent and more differentiated, but at the same time not descending into chaos. Students create coherent networks and relationships between knowledge through self-organization.

5) Dissipation: Small learning peer groups are being formed and dissolved based on purpose creating dialogues between students and instructors and generating new knowledge.

These chaos properties provide a mindset that should be included in the design of chaordic courses. They make sure that students evolve their creative skills. However, instructors can not only focus on chaos properties, they also need to incorporate structure into the courses, e.g. with milestones, intermediate deadlines and control points where students obtain feedback and where they can evaluate their learning progress. Examples would be daily or weekly meetings where students report about their progress, their challenges and their promises, while instructors listen and provide feedback and guidance.

C. Benefits

In our experience, chaordic learning leads to the following benefits when it is applied properly in SE education:

1) Increased motivation: Students who can influence the learning subject, methods and process, have a higher intrinsic motivation because they have control over their learning experience and over the learning outcome.

2) Increased self-organization: As the instructor steps aside, students have to act, organize themselves and are responsible for the progress. They have to find the right learning activities and actively ask for feedback.

3) Less hierarchies: Students are treated on the same level as instructors, leading to an improved discussion between students and instructors. Students then come up with new ideas and suggestions to form the learning process.

4) Improved learning from failures: Instructors create a culture of successful failures: they emphasize that students are allowed to make failures and to learn from these failures. This lowers the pressure and allows students to experiment new approaches and to think out of the box.

5) More room for creativity: Decreased pressure to succeed in every step facilitates out of the box thinking and the application of new, innovative approaches.

It is important that instructors facilitate the chaordic learning approach and provide guidance. In particular, unexperienced students will not be able to immediately self-organize themselves. Instructors need to clearly communicate the chaordic learning environment and the focus towards self-organization, innovation and creativity. They ask students in the beginning of the course to actively organize themselves and make the self-organization part of the assessment. In the first weeks, they help unexperienced students to organize themselves without taking over the organization. Frequent inspection and adaption, as in empirical process control, is required to control the learning outcome and to prevent misconceptions.

IV. Case Study

In this section, we present two courses in which we applied chaordic learning to improve the learning experience of students. The first course is on mobile games development, the second course is an international student school on software development for mobile platforms and the internet of things.

A. Games Development Course

The games development course is a two-week block course with up to 40 students. We described the structure of this course in [38]. In this paper, we summarize the course and focus on its chaordic learning characteristics. We describe course design and learning objectives using the six steps of the chaordic design process and the outcome of the course.

1) Chaordic Design Process: While the course is structured in the first week where students learn programming and games development, it provides a lot freedom and facilitates self-organization in the second week.

Purpose: The course is focused on learning games design and iOS app development using the Swift programming language. It includes various topics students get to explore: programming, use of platform specific frameworks, distributed version control, game design, application of design and architectural patterns, and user interface design for games.

Depending on previous experience and interest, students participate in the beginner or advanced modes. The beginner mode targets students who have no experience with Swift and iOS development. However, the course assumes that beginners have basic knowledge of an object-oriented programming language and UML. During the first week, the focus for these students is on learning Swift, the use of iOS frameworks and the Xcode IDE. In the second week, students apply their knowledge in small team projects.

In advanced mode, students participate as teaching assistants (TAs) who have already acquired knowledge of the development environment, including programming language and tool chain. The objectives for TAs include specializing in a technical topic, such as sprite animation within games, and helping beginners during the course. To achieve this, TAs prepare an interactive tutorial on their chosen topic under the supervision of an instructor and present this during the first week of the course. They also guide teams and individuals to apply games development practices in their projects to learn from known methods that achieve best results.

All students practice social and non-technical skills such as working in a team, presenting, communicating and being proactive. Learning objectives and assessment structure of the course are made clear on the first day. During the introduction round, students share “Why are they here?” and what they wish to gain from attending the course. This helps choose topics of interest, set personal goals and illuminate the purpose for the students.

Principles: Students can choose their own game idea and select the programming concepts, frameworks and technologies they want to apply for their game. This increases the motivation of the students because they can try their own ideas without limiting their creativity. Ambitious students might go for more complex 3D technologies, while others prefer the development of simple jump and run games in 2D.

In addition, they have the ability to adapt the game idea during the development. Instructors and TAs provide guidance: students should implement a simple game with easy to apply technologies that is extensible with new features later on. This guidance should prevent them from starting with overly complex games they would not be able to implement in a week. It facilitates the prototyping idea that is important in today’s SE practices. In the intermediate milestone in the second week, the teams present the game idea and justify the chosen technologies. They obtain feedback and decide on their own whether and how to include it.

While the first week provides basic knowledge about game development techniques, students need to obtain further knowledge on their own for details of a specific technique by reading references or by using further tutorials. Students also need to organize their work in teams on their own.

People: Students have to interact with instructors, TAs and their peer students. Two instructors organize the course, one is responsible for introducing Swift, the other one is familiar with games development. Instructors select students, prepare presentations and set up infrastructure. Four to six TAs help in the organization. Game experts from industry provide help for the design of graphics and animations.

Concept: The course promotes a flat hierarchy between instructors, TAs and students and facilitates feedback in both directions, from students to teaching staff and from teaching staff to students. This increases the commitment and the motivation of students and decreases communication barriers.

Structure: The structure of the course is defined by its schedule in the first week and its milestones in the second week. In the first week, instructors and TAs give three tutorials per day based on interactive learning [39] that are required for games development. These tutorials are 90 min long and include a mix of theory, examples, exercises, sample solutions and reflections about the taught concept and its usefulness in different programming situations. During a tutorial, TAs walk around and help students if they face problems. After a tutorial, students get 30 min to solve an exercise on their own that is based on the tutorial content.

In the second week, students build teams to develop their own game idea. They face several milestones: start of development, intermediate feedback and final presentation. These milestones are made clear on the first day. Otherwise, the students can freely choose when, how and where to develop their game providing freedom and a necessity to self-organization. The intermediate milestone is two days after the start of the second week, where the teams informally present their ideas and current progress.

The course phase ends with the final presentations of all teams, which are recorded. Students receive the videos and obtain feedback on content and delivery from the TAs and the instructors. Students are assessed on the following criteria in a descending emphasis: originality and complexity of the game, how well it is implemented, how it was designed and on an additional documentation where the students show UML diagrams and explain those. Students can receive the best grade with both - an unfinished game using complex technologies and a simple game with completed functionality. This assessment scheme is made clear to the students in the beginning of the course, so students are encouraged to experiment knowing that the difficulty of their choices is taken into account during assessment.

Practices: The course does not formally present project management techniques such as the agile methodology Scrum [30] to avoid an increase of its complexity. Instead, only some of the practices, such as stand-up meetings, are introduced to students. These are used to stay organized by presenting the minimum viable products throughout the day. Students work in pairs for one week which encourages the cooperative and self-guided learning process: they work towards a common goal and have the freedom to organize themselves however they want. This gives them the confidence in their programming abilities, required to build larger applications, while at the same time not limiting them to mere implementation of small programming assignments.

2) Outcome: At the end of the course, each team presents their game in a short Pecha Kucha presentation [40] showing 20 slides, each in 20 seconds with automatic transitions. During the presentation, they focus on the game idea, the proposed game features, the software architecture, the object design, the used technologies and frameworks, status and outlook. In addition, they include a demo of the game where they show the most interesting features.

After the course, instructors encourage and further accompany students to finalize and submit their game to the iOS AppStore. Until now, 18 teams published their games.

Our evaluations show that students appreciate a great learning experience with practical aspects that they can further use in their career, as well as the possibility to work self-organized and to integrate their own ideas. They improve their SE abilities, in particular object-oriented programming, and their soft skills with the help of games development and they have a lot of fun.

B. International Student School

The second course is a one week course with 20 students and four instructors. The course on SE in the *Joint Advanced Student School* (JASS) has been conducted for the third time in March 2016, in St. Petersburg, Russia in the office of JetBrains[3], following two previous courses in 2008 and 2012. The idea is to bring together students with different cultural background (Germany and Russia), to work together on innovative topics in SE in a chaordic manner. 10 German students from Munich and 10 Russian students from St. Petersburg participated in 2016. We describe course design and learning objectives using the six steps of the chaordic design process and the outcome of the course.

1) Chaordic Design Process: The topic of JASS 2016 was "software development for mobile platforms and the internet of things (IoT)". Students are organized into balanced teams, each with four students (two Russian and two German students), to work on small projects relevant to this topic for one week while also spending time on sightseeing and team formation. While the project topics are roughly defined by the instructors, the teams have the freedom to adapt the topics and to include their own ideas.

Purpose: Students learn to develop innovative apps for mobile and IoT platforms and get in contact with technologies, such as iOS, Android, micro-controllers, drones, and sensors. This combination enables new scenarios and requires the ability to connect sensors of IoT devices with visualization components and sensors on mobile devices. Students acquire soft skills, gain experiences with international teams and become familiar with latest innovations and technologies. Participants collect experiences in global SE by working in international teams.

Principles: At the core of the educational offering is a more engaging teaching method that comes with less control and moderation and facilitates creativity. The freedom of choice with regards to tools, processes and technologies forces students to organize themselves as a team and to overcome the problems on their own. Students can choose the project topic and influence the problem statement and the requirements. This allows them to include their own ideas and to try out new, innovative approaches.

People: Four instructors organize the course. They arrange the travel, housing, working environment and catering. Although the project environment is established, no strict rules, nor processes of how the projects should evolve are defined. Students apply for the school by providing personal details and a short motivational letter. The instructors review these applications and select a heterogenous group of students with different levels of experience.

[3]JetBrains Research: https://research.jetbrains.org

Concept: The course promotes a flat hierarchy between instructors and students and facilitates feedback in both directions. This increases commitment and motivation and decreases communication barriers: No hierarchies within the team and flat hierarchies between instructors and students.

Structure: The course is scheduled for six days. Students spend about four days for development and two days for sightseeing and self-organized team formation events. The course starts with a short introduction round of the participants, followed by the presentations of the project ideas. The projects are framed upfront by the instructors in a short problem statement that is handed out to the students.

Instructors emphasize that the given problem statements are only starting points and that the students can integrate their own ideas. Relevant hardware and software needed to work on the different projects (e.g. drones, sensors, beacons and mobile devices) are provided to the students. After the team assignment, instructors conduct an icebreaker to lower barriers between team members. Teams organize themselves, choosing which tools and processes to use, and distributing responsibilities. Teams receive regular feedback in a daily standup meeting where they report on progress and impediments, and promise what they will achieve until the next standup meeting. These meetings are similar to standup meetings in Scrum [30].

In a rotating manner each team member reports on its team's progress. The discussion with instructors and other team members is considered as valuable feedback. On the last day of the school, each team presents their project including the design of the system, problem description, motivation, introduction of team members and a live demo. Assessment is based on the final presentation and demonstration of the app functionality. As with the games development course's assessment structure, difficulty reflected in students' choices with respect to technology and functionality is rewarded. Aspects like motivation, personal progress, team skills and quality of the final presentation are also considered in addition to the project functionality. The course is not part of the curriculum: all students participate voluntarily.

Practices: In line with the agile manifesto, the school focuses more on individuals and interactions, than processes and tools and aims for a working prototype instead of documentation. Daily standup meetings structure the communication between instructors and students. Instructors are available for additional questions and guidance. Social activities are an important aspect of the course. All participants have breakfast, lunch and dinner together and go out in the evenings. This motivates the students and strengthens the international teams. The school does not focus on formal management processes, so it is up to the students to organize themselves and work as team to present a viable product at the end of the school.

2) Outcome: After a week of design, development and integration, all teams presented their projects. Students also produce a short, creative trailer to visualize the project idea. The following five projects were presented in 2016:

Multimodel iNTeraction (MiNT): A modeling tool which facilitates real-time collaboration on models during early stage requirements engineering. MiNT allows to collaborate on the creation and editing of informal models, and enables collaborative modeling across iOS and Android devices.

Quadcopter Autopilot: The goal of this project was the creation of an app that allows to capture dynamic scenes using a quadcopter autopilot in several operation modes.

KneeHapp - Rehabilitation Monitoring on the Wrist: KneeHapp is a smart knee bandage that aims to support patients suffering from ligament ruptures. Patients perform required exercises correctly and provide relevant metrics to a doctor over a smartphone app to enable better treatment decisions [41]. KneeHapp Watch extends the smartphone app with a component for the Apple Watch. The smart watch enables the display of real time exercise data directly on a patient's wrist improving the user experience during exercises.

Octopus - Mobile First Responder for Emergencies: Octopus is a sensor system for optimizing the treatment in emergency situations. Octopus consists of a set of hardware sensors (e.g. heart rate, breathing, brain activity sensor) and mobile devices that are used to collect patient data inside the ambulance while the patient is transported to the hospital.

Geo Quest Travel Game: The smartphone app helps travelers to plan city trips with regard to points of interest and food recommendations. It creates individual quests to make city trips more informative and interesting.

C. Discussion

Chaordic courses benefit from flat hierarchies, but also need trust to function properly. Instructors need to be able to move control over the teaching and learning process to students, while still providing enough guidance to keep the learning outcome on a qualitatively high level. It is essential for instructors to provide guidances and feedback to students to control the learning outcome [34]. Instructors need to identify situations where students are learning incorrect approaches. Then, they need to interrupt the process and guide the students back to the correct path. Students need to be comfortable with openness and self-organization to benefit from the chaordic education approach. If students prefer structured courses where all instructions are provided, the chaordic approach will fail.

To summarize, it is essential for students to actively participate in the process and to contribute their own opinions and ideas to the course. Shy students may have an issue with the chaordic approach and should be especially treated to get out of their comfort zone to make sure that they also succeed. We evaluated the case studies in informal discussions and have received many positive comments about the chaordic learning approach. Students have reported an increased motivation, higher level of self-organization and more room for creativity. They appreciate the openness of the courses and the low hierarchies between instructors and students. In particular, they like that they can try out new approaches and have the freedom to make failures and to learn from successful failures.

V. Related Work

In this section, we relate our chaordic learning approach to another chaordic learning context, describe the relation between chaordic learning and constructivism and show how design thinking relates to chaordic learning.

A. Chaordic Learning Context

Leigh argues that “structured activities have the potential to create unpredictable learning contexts” [22]. Simulations and games can be used in cooperation with chaordic approaches inside of a traditional classroom setting. Like in our case study, instructors shift from an orchestrating role to a supporting one.

Leigh uses open simulations, which come with a high uncertainty regarding the possible outcome. She focuses how the chaos framework can be useful in understanding complex interactions. These complex interactions can occur when giving up control over the learning process. Furthermore, Leigh describes what it means to give control away as an instructor and how trying to reestablish order can actually hinder the process of chaordic learning itself. Depending on how instructors react to chaotic circumstances within the chaordic learning experience, the approach can either fail or succeed. It might be difficult to create a positive experience for instructors since they have to adapt their teaching style to a new approach and since transferring control to students can be intimidating.

Our paper focuses on SE education in particular and the different organizational structures that surround it. Furthermore, we directly apply the chaordic organization principles by Hock to the learning contexts and focus on how to give control to students. One of the key aspects is how instructors can steer this process and which decisions they allow the students to make on their own. The students in our chaordic courses have freedom to choose different aspects in a software project and can also choose their own responsibilities in this process. The distribution of the decisions, between instructors and students, play a major role in a chaordic learning approach.

B. Chaordic Learning and Constructivism

Desouza identified the role of ‘Radical Engineers‘ in [42], who are dealing with chaotic requirements, opportunities and problems; they allow projects to incorporate innovative ideas and tools. In this work, the authors do not use the term chaordic, but incorporate the idea behind it. Their conclusion is that working in a chaordic manner is essential for software development projects to provide flexibility to students and to develop a corresponding set of skills. The learning process itself has a chaordic nature. Vygotskii in his works [43] pointed that chaos (and chaos ordering) is a base for learning and acquiring new knowledge.

The growing trend in SE is that development happens under conditions of notable uncertainty. There are no standard rules or methods for setting up the development process itself because it is hard to define the concrete roles which are distributed in the team. Veli-Pekka [44] suggested the following organizational pattern: “develop the development culture before process” and after that let people be self organized.

Uncertainty in requirements and environment requires to experiment not only before the development process was set up, but even during the development. According to Thomke [45], an innovative process (and we believe, that software development is about innovation) encompasses success and failure; it is an iterative process of understanding what does work and what does not. The obvious consequence is that the learning process should have a similar chaordic structure, i.e. incorporate failures and successes and learn from experimentation to create successful failures.

Learning by experimentation is a philosophical viewpoint covered in the constructive approach that tries to describe the nature of learning [46], [47]. It focuses on hands-on approaches where students experiment with concrete problems, try out methods and techniques and learn from the reflection about their usage. Students then make their own inferences, discoveries and conclusions and adapt the behavior. As such, constructivism promotes the chaos side of the chaordic learning process and influences the idea of the chaordic learning approach described in this paper.

C. Design Thinking

Design thinking was originally explored and developed as a human-centric methodology to solve complex creative problems closely associated with conceptual design [48]. Different versions of the design thinking process exist. All describe iterative phases that are not necessarily ordered and can occur simultaneously. One of the latest views of the process proposed by Meinel and Leifer has five phases: (re)defining the problem, need-finding and benchmarking, ideating, building, testing. In the past decade, design thinking has gained attention as a meta-disciplinary methodology relevant in a wide range of contexts beyond the traditional preoccupation of designers [49]. It has been proposed as a “team-based learning process [which] offers teachers support towards practice-oriented and holistic modes of constructivist learning in projects” [50].

Design thinking is an iterative process that guides the teacher to “realize what is recommended theoretically in constructivist theory” [50]. In this sense, design thinking is similar to chaordic learning and can also be applied to SE courses to include creativity and innovation. In order to reconcile the relation between chaordic learning and design thinking, as applied in education, we consider chaordic learning as an overarching educational approach. Design thinking is a process that could be employed as one of the practices (described in Section III-A) when the chaordic learning environment is being established. In our courses, we use agile methods within the practices of the chaordic design process.

VI. Conclusion

Chaordic learning is an approach to balance education between order and chaos to stimulate analytical and creative thinking processes. Instructors provide structure and guidance, but also integrate freedom for self-organization and self-guided learning and embrace innovation and creativity, so that students learn important management and communication skills. The chaordic approach is similar to agile methods following empirical process control. It allows instructors to react to specific situations, while providing an overall plan in their teaching approach. When integrating chaordic learning into their courses, instructors still control the learning outcome and avoid misconceptions.

Chaordic learning is particularly helpful in SE education, where practical application of knowledge, interaction and collaboration is required. Instructors view deviations as opportunities and failures as possibilities for students to learn and improve. We introduced chaordic learning into two courses, a games development course and a joint advanced student school. Students report an increased intrinsic motivation, a higher level of self-organization and more room for creativity which led to an improved learning experience and more fun.

References

[1] M. Mahoney, "The roots of software engineering," *CWI Quarterly*, vol. 3, no. 4, pp. 325–334, 1990.

[2] B. A. Ogunnaike and W. H. Ray, *Process Dynamics, Modeling, and Control*. Oxford University Press, 1994, vol. 1.

[3] V. Basili, "The role of experimentation in software engineering: past, current, and future," in *Proceedings of the 18th international conference on Software engineering*. IEEE, 1996, pp. 442–449.

[4] T. Connolly, M. Stansfield, and T. Hainey, "An application of games-based learning within software engineering," *British Journal of Educational Technology*, vol. 38, no. 3, pp. 416–428, 2007.

[5] D. Shaffer, "Pedagogical praxis: The professions as models for postindustrial education," *Teachers College Record*, vol. 106, no. 7, pp. 1401–1421, 2004.

[6] J. Whitehead, "Collaboration in software engineering: A roadmap," *Future of Software Engineering*, vol. 7, pp. 214–225, 2007.

[7] I. Mulder, "A pedagogical framework and a transdisciplinary design approach to innovate hci education," *Interaction Design and Architecture(s) Journal*, no. 27, pp. 115–128, 2015.

[8] J. Tomayko, "Teaching a project-intensive introduction to software engineering," DTIC Document, Tech. Rep. CMU/SEI-87-TR-20, 1987.

[9] B. Bruegge, J. Cheng, and M. Shaw, "A software engineering project course with a real client," Carnegie Mellon University, Software Engineering Institute, Tech. Rep. CMU/SEI-91-EM-4, 1991.

[10] B. Bruegge, S. Krusche, and L. Alperowitz, "Software engineering project courses with industrial clients," *ACM Transactions on Computing Education*, vol. 15, no. 4, pp. 17:1–17:31, 2015.

[11] A. Dutson, R. Todd, S. Magleby, and C. Sorensen, "A review of literature on teaching engineering design through project-oriented capstone courses," *Journal of Engineering Education*, vol. 86, no. 1, pp. 17–28, 1997.

[12] P. Naur, B. Randell, and J. Buxton, *Software engineering: concepts and techniques: proceedings of the NATO conferences*. Petrocelli/Charter, 1976.

[13] M. Lehman and L. Belady, *Program evolution: processes of software change*. Academic Press, 1985.

[14] M. Kellner, "Software process modeling support for management planning and control," in *Proceedings of the 1st International Conference on the Software Process*. IEEE, 1991, pp. 8–28.

[15] D. Kolb, *Experiential learning: Experience as the source of learning and development*. Prentice Hall, 1984, vol. 1.

[16] D. Boud and G. Feletti, *The challenge of problem-based learning*. Psychology Press, 1998.

[17] J. Dunlap, "Problem-based learning and self-efficacy: How a capstone course prepares students for a profession," *Educational Technology Research and Development*, vol. 53, no. 1, pp. 65–83, 2005.

[18] R. Garrison and H. Kanuka, "Blended learning: Uncovering its transformative potential in higher education," *The internet and higher education*, 2004.

[19] D. Johnson *et al.*, *Cooperative Learning: Increasing College Faculty Instructional Productivity. Higher Education Report*. ERIC, 1991.

[20] D. Hock, "The chaordic organization: Out of control and into order," *World Business Academy Perspectives*, vol. 9, no. 1, pp. 5–18, 1995.

[21] D. Hock, "The art of chaordic leadership," *Leader to leader*, vol. 15, no. Winter, pp. 20–6, 2000.

[22] E. Leigh and L. Spindler, "Simulations and games as chaordic learning contexts," *Simulation & Gaming*, vol. 35, no. 1, pp. 53–69, 2004.

[23] F. W. Taylor, *The principles of scientific management*. Harper, 1914.

[24] D. Hock, *Birth of the chaordic age*. Berrett-Koehler Publishers, 1999.

[25] M. Huysman, "Balancing biases: A critical review of the literature on organizational learning," *SAGE Publications*, 1999.

[26] J. Pfeffer and J. Veiga, "Putting people first for organizational success," *The Academy of Management Executive*, vol. 13, no. 2, pp. 37–48, 1999.

[27] C. Kelliher and D. Anderson, "Doing more with less? flexible working practices and the intensification of work," *Human Relations*, vol. 63, no. 1, pp. 83–106, 2010.

[28] R. Stacey, "Learning as an activity of interdependent people," *The Learning Organization*, vol. 10, no. 6, pp. 325–331, 2003.

[29] F. van Eijnatten and G. Putnik, "Chaos, complexity, learning, and the learning organization: towards a chaordic enterprise," *The Learning Organization*, vol. 11, no. 6, pp. 418–429, 2004.

[30] K. Schwaber, "Scrum development process," in *Proceedings of the OOPSLA Workshop on Business Object Design and Information*, 1995.

[31] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," *The Agile Alliance*, 2001.

[32] VersionOne, "9th annual state of agile development survey," 2015, retrieved January 08, 2016 from https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf.

[33] D. Rico and H. Sayani, "Use of agile methods in software engineering education," in *Agile Conference*, 2009, pp. 174–179.

[34] P. Kirschner, J. Sweller, and R. Clark, "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," *Educational Psychologist*, vol. 41, no. 2, pp. 75–86, 2006.

[35] A. Collins, J. S. Brown, and A. Holum, "Cognitive apprenticeship: Making thinking visible," *American educator*, 1991.

[36] S. Krusche, M. Berisha, and B. Bruegge, "Teaching Code Review Management using Branch Based Workflows," in *Proceedings of the 38th International Conference on Software Engineering*. IEEE, 2016.

[37] L. Fitzgerald, "Chaos: The lens that transcends," *Journal of Organizational Change Management*, vol. 15, no. 4, pp. 339–358, 08 2002.

[38] S. Krusche, B. Reichart, P. Tolstoi, and B. Bruegge, "Experiences from an experiential learning course on games development," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE)*. ACM, 2016, pp. 582–587.

[39] S. Krusche, A. Seitz, J. Börstler, and B. Bruegge, "Interactive learning: Increasing student participation through shorter exercise cycles," in *Proceedings of the 19th Australasian Computing Education Conference*. ACM, 2017, pp. 17–26.

[40] A. Beyer, "Improving student presentations pecha kucha and just plain powerpoint," *Teaching of Psychology*, 2011.

[41] J. Haladjian, Z. Hodaie, H. Xu, M. Yigin, B. Bruegge, M. Fink, and J. Hoeher, "Kneehapp: A bandage for rehabilitation of knee injuries," in *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 181–184.

[42] K. Desouza and Y. Awazu, "Managing radical software engineers: Between order and chaos," in *Human and Social Factors of Software Engineering*. ACM, 2005, pp. 1 – 5.

[43] L. S. Vygotsky and R. W. Rieber, *The collected works of LS Vygotsky: Volume 1: Problems of general psychology, including the volume Thinking and Speech*. Springer Science & Business Media, 1988, vol. 1.

[44] V.-P. Eloranta, "Patterns for controlling chaos in a startup," in *Proceedings of the 8th Nordic Conference on Pattern Languages of Programs*. ACM, 2014, pp. 1:1–1:8.

[45] S. H. Thomke, *Experimentation Matters: Unlocking the Potential of New Technologies for Innovation*. Boston, MA, USA: Harvard Business School Press, 2003.

[46] D. Jonassen, K. Peck, and B. Wilson, *Learning with technology: A constructivist perspective*. Prentice Hall, 1999.

[47] T. Duffy and D. Jonassen, *Constructivism and the Technology of Instruction: A Conversation*. Psychology Press, 1992.

[48] P. Rowe, "Design thinking," 1987.

[49] L. Kimbell, "Rethinking design thinking: Part I," *Design and Culture*, vol. 3, no. 3, pp. 285–306, 2011.

[50] A. Scheer, C. Noweski, and C. Meinel, "Transforming constructivist learning into action: Design thinking in education," *Design and Technology Education: An International Journal*, vol. 17, no. 3, 2012.