ECONOMIC OUTPUT UNDER THE CONDITIONS OF SOCIAL FREEDOM IN SOFTWARE DEVELOPMENT

David Kuhlen[1] and Andreas Speck[2]

[1]Datenlotsen Informationssysteme GmbH, Technical Consultant,
Beim Strohhause 27, 20095 Hamburg, Germany.
[2]Christian Albrechts Universität zu Kiel, Head of the Business Information Technology Group, Hermann-Rodewald-Straße 3, 24098 Kiel, Germany.

ABSTRACT

Software developers organize their work autonomously. Agile development approaches give freedom to developers. However, the discussion about economy often leads to the comparison with manu-facturing processes which are tightly organised. If developers spend too much time on inefficient activities, the performance might decrease. The worst example could be breaks. Breaks could be seen as a reason for economic problems by the management.

This paper investigates the impact of breaks on the overall performance in software development. The investigations assume that if developers make pauses in a normal manner, this has not a negative impact on the profitability.

In practice, the human-centric development process brings together a business process and a social process. The interaction of both processes was simulated. Due to the execution of 1.500 simulations, we obtained information on the economic progression of the development process under the influence of breaks. We determined the impact of breaks on the overall profitability.

This investigation contributes to the discussion of freedom in software development. It helps man-agers to assess if employees who make breaks harm the profitability. This could lead to the imple-mentation of further business constraints.

KEYWORDS

Software engineering, Manufacturing process, Self-determination, Operating efficiency

1. ECONOMIC SCHEDULING AND CREATIVE SOCIAL FREEDOM IN SOFTWARE ENGINEERING

In the past decade multiple software producers fail to predict the effort of requirements. This leads to wrong assumptions on the costs and the profitability of software development projects. This fact is surprising, because the development process is well described. Such business process specifications prescribe how development should take place [28, 151f.]. The problem is that employees actions differ from what the process dictates. However, the process is not always followed [11, 27]. As revealed, due to process mining, employees work differs from the model [33, 3].

The way how software is developed is seen as being too slow and too costly [2, 42]. Even if development is a craft-process“, a lack of clear standard methods leads to heterogeneous performance and developers who are less productive than others [7, 4]. Often, the compliance of development approaches is not verified overall [30, 19]. This makes it possible for developers to act differently from the rules, dictated by the process model. Self-determination is considered to be a very important success factor in agile development. Often, developers grew up without classical hierarchy and organize their work autonomously [8, 83].

This freedom is a key success factor which leads to technological innovations. In order to be successful and sustain the cost pressure, development has to enhance the controlling of their process costs [15]. The discussion on economy often leads to the comparison with manufacturing processes. These processes have to pay much attention on constraints such as resource and time [21, 1777]. In order to adjust this manufacturing-perspective to the case of software development, we have to consider individual, self-determined actions of developers in the analysis of the effort.

Individual actions and self-determination often lead to the discussion, whether individuality improves or harms the profitability. If developers spend too much time on inefficient activities, the performance would decrease. The worst example could be breaks: if developers spend too much time on breaks, management could see this as a reason for economic problems of the development. However, there is no evidence which impact pauses have on the profitability.

2. HYPOTHESIS AND CONTRIBUTION

This paper investigates the impact of breaks on the overall performance in software development. The investigations assume that if developers make pauses in a normal manner, this has no negative impact on the profitability. Statistical outliers like developers who are lazy and spending the majority of their time with different activities are not considered in this study. The purpose of this paper is to analyse if pauses (actions which do not fulfil the sprint goal) have a negative impact on the economics of software development.

The investigations focus on a normal course of action the development of software takes. Therefore, the investigations consider that developers who work on requirements which have a normal priority. This paper focuses on the progression of development. Other contract-related influences on the development, (like the delivery of intermediate goods), were excluded. Our objective is to analyse the progression of development itself solely (to answer the above-mentioned question), in order to deliver information to the management. Based on this information, management could enhance the planning of development and define aspects of the contact.

Due to our investigations, it becomes apparent that pauses do not have an impact on the profitability of software development, overall. The illustrated dependencies between pauses and the profitability (compare figure 1) lead to the compensation of negative and positive effects of breaks.
This paper helps to understand the influence of non-value creating actions, like making a break, on the overall performance in human-centric processes. It enables a new perspective in the analysis on the procedure how developers act in a process. The insight that pauses have no impact could allow management to expand the freedom of software developers. This would suit the developers which honour this with increased performance.

3. THEORY OF SOFTWARE ENGINEERING ECONOMY

Our investigations focus on the process of software development. Speck, Franczyk and Kiebusch define a business process as a sequence of events with a definite beginning and a distinct ending [14, 439]. Abramowicz describes business processes as a set of related, ordered activities that contribute to the production of a software [1, 1]. These two mentioned characteristics exclude the transition of states which arise from recurring individual actions from being a process. It is reasonable to concentrate the business inspection on higher-level activities. Management is facilitated by the discussion of elementary activities like making out the production schedule, reading the quality control report, visiting a customer [26, 8].

However, the performance depends on individual actions. Parnas et al. claim that we will never find a process which develops software in a perfect rational way [24, 251]. We refuse this conclusion. Because of the individual actions that could influence the performance, a rational process to describe how software is developed is hard to find. Developers do more than the process description demands or allows.

The basic philosophy of breaks is to protect the working capacity of employees. Breaks are necessary, because employees could not work from the beginning until the end equally efficient and therefore breaks help them to recover [34]. In manufacturing processes for example, breaks are included. This allows manufacturers to enhance their control of (process-) costs.

During the software development, developers could decide independently if they have a break. If developers interrupt their work, this could occur in the personal scheduling. The analysis of personal schedule in respect of its admissibility shows risks of delays in the instances [9, 216]. The interruption of working process due to breaks [34] could have a negative impact on the flow time and the number of completed function points.

Breaks are necessary to protect the health and the working capacity of the employees. At the same time, breaks interrupt the working process and could harm the profitability. To prevent the emergence of bottlenecks, companies often contemplate to the use of a compliance rule. Such compliance rules could ensure that business processes and operations are in accordance with prescribed norms [28, 149]. To implement such compliance, the rules have to be described formally. Business processing rules require a formal notation that defines the change of state, precisely [22, 61]. The construction of business rules corresponds to an investment in the process. As Abramowicz explains, the definition of a fully comprehensive model is a challenging and demanding task which raises effort [1, 2 – 3]. The effort it takes to develop higher level models [13, 1367] has to be worthwhile.

Such an investment in the development process offers the potential to enhance the effort estimation. It facilitates software producers to plan their production capacity. On the opposite, such a compliance rule contradicts to the esteemed freedom of software developers. Empowerment of employees could be the basis for value improvement [23, 12]. In order to judge if freedom could be the basis for performance losses, an analysis of the impact of misspending (time for pauses) could be advantageous.

4. RELATED WORK

We investigated the impact of social effects, like breaks, on the economic profitability in business processes. This research has three dimensions: (1) the social freedom of individuals in software industries, (2) the process of software development, and (3) the profitability of software development.

The dimension of social freedom is investigated e.g. by Eckstein [8]. Especially agile approaches give freedom to developers, like e.g. Scrum [29]. These facts have been analysed during the creation of agile procedure models.

Parnas et al. described problems in the development process [24], regarding to the second dimension. The use of standardized development methodologies was analysed, e.g. by Kuhrmann et. al [19], [20] and [30]. This topic has strong relations to process analysis and business process reengineering. Especially the findings of Davenport [6] and Hammer and Champy [12] are used to analyse approaches which have to improve the profitability of processes. Furthermore, the dimension of processes in software development is investigated e.g. in our previous work [16].

The productivity of software development was topic of the investigations e.g. of Plewan and Poensgen [25]. Fundamental work in this dimension was delivered by Boehm, e.g., [4]. Furthermore, Curtis et al. investigated special economic aspects in software development [5].

5. THEORETICAL FRAMEWORK AND METHOD

In order to investigate the impact of breaks, our method needs to distinguish dependent and independent variables. In our theoretical framework variables which describe the breaks (e.g. expressed by the key performance indicator number of breaks) represent the independent variable. In our simulation, we analysed the impact of different constellations of breaks. We measured the economic output of the progression in each experiment. The variables which describe the economic output (e. g. expressed by the key performance indicator realized function points or the Instance duration time) are the dependent variables in our experiments. By the use of the regression analysis techniques, we analysed the correlation between these variables, on the basis of simulation results. In our experiments, we simulated the progression of the software development process, consisting of multiple activities.

During the development, a number of base activities have to be processed [25, 8]. The details are defined by a standardized development approach. Its process prescribes which activity has to be performed in order to complete a requirement. The requirement needs to be (1) analysed, then it has to be (2) prioritized, (3) a concept needs to be written, (4) the software has to be developed, a (5) quality assurance has to take place, before (6) the solution has to be delivered [25, 8], [16, 160]. Each instance (conform to each requirement) has to pass the activities of development. The employees are responsible for different activities.

Based on different instances in diverse activities, employees could perform different tasks. Each task represents an instance (requirement) in an activity (work order) [17]. The figure 1 shows the progression of employees in the process on a detailed level.

Figure 1 shows the process, how an employee does his work. At first, he picks up a task and then he performs his work (expressed due to the ad hoc activities within the group “work“). Curtis explained that the behaviour in software projects has to be analysed in multiple levels [5]. On the second level, figure 1 shows the effect of different activities on a time frame. The personal level and the base process are incorporated due to the time frame.

The introduction of compliance rules in this process is regarded as highly problematic from a practical standpoint [28, 151]. In order to perform the analysis, mutual dependencies need to be considered. As Speck et al. explained, model checking is an ideal technique especially if the number of states to be considered does not adventure to run into a state explosion [31]. In this case the different states have a strong dependent relationship. In order to model the reality adequately, a vast number of disjoint states could be defined.

The relationship leads to states, changing (1) the probability of other states and (2) the behaviour within other states (intensity and impact). In reality employees often work in different projects (and thereby on different tasks) simultaneously which has an impact on

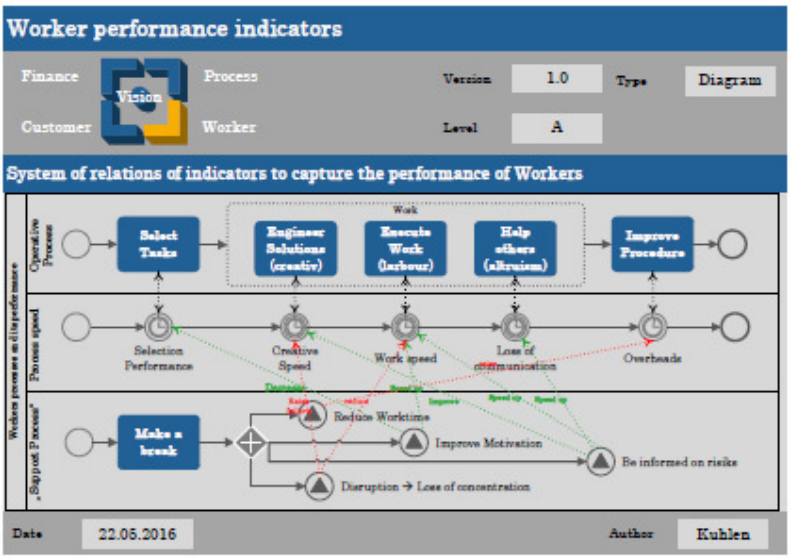

Figure 1: Process model which illustrates dependencies and interactions of the effect of pauses in the progress of software development.

the overall work and performance [25, 127]. Beecham et al. struggled to find a model which considers all the identified factors which have an impact on developers motivation [3, 24]. Endl explained that it is impossible to model all the situations which potentially can occur during the runtime of a process [10, 5]. This challenge needs to be considered due to the method, used to analyse the above mentioned case. A comprehensive analysis of all effects during the development would be impossible, because the cardinality of the set of states is nearly infinite. To reduce the risk of a state explosion we restrict the set of states, we consider in our analysis on a limited number of six different states [17], knowing that in reality a larger quantity of states (and therefore also impacts) exists.

In order to analyse the behaviour of a software development team, a convincing model is needed. Speck et al. require that the model has to be described as finite automata [31, 80]. The illustration 2 shows a brief of our simulation model. It demonstrates the handling of multiple states.

Figure 2 clarifies a state machine which illustrates the behaviour of an employee abstractly.
Each simulation consists of cycles which represent the progression of time. During the whole experiment we repeat the simulations and measure the results. In each cycle, each employee has to pick up one state. Therefore, the finite automate of figure 2 is passed in each cycle repeatedly. The execution of the state machine 2 bases on a list of available actions. An action is an object which belongs to a state (distinguished by the superior class), [17]. Therefore, each employee could have a number of multiple tasks, he has to work on. However, in each time unit (cycle) the developer could just work on one task.

6. CONDITIONS AND INITIAL VALUES OF THE SIMULATION ENVIRONMENT

In order to investigate the impact of breaks on the profitability, we performed a simulation. For each development approach 250 simulations were executed. Each simulation (iteration) consists of 57.600 cycles. One cycle nearly conforms to one minute. Therefore, multiple simulations of 6 months were performed to simulate a team, developing according to Scrum, Kanban, RUP, TDD and V-Modell XT. The execution of these 2.496 simulations took about

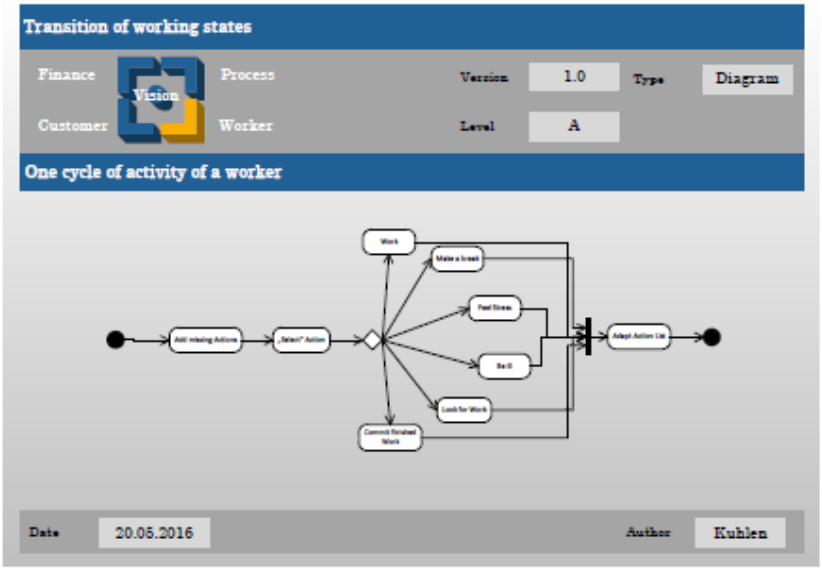

Figure 2: State diagram which illustrates the actions of developers during the simulation.

3 days plus or minus some hours. We used a Windows Server 2008 R2 Enterprise on an industrial machine to perform the simulation efficiently.

During the simulation the state machine (see illustration in 2) would be executed for each cycle multiple times. For each state, we developed an action. Each action belongs to exactly one type of person (employee or customer) [17]. Each action has a probability and overrides the method act which is executed during the cycle. Furthermore, an action has a minimum duration. At last, actions have a boolean attribute finished "which indicates if it has to be executed any longer.

ChooseTaskAction : In this action, the employee inspects every node in the process, he is responsible for. For every related activity, the employee checks whether open instances have arrived or not. If multiple instances are available alternatively, the employee selects that one which is the closest to the end (push for finishing). If the employee has selected an instance, he takes it and derives a task. Then, the employee creates a Work Action finishes his ChooseTaskAction. The ChooseTaskAction was defined to have a probability of 35 %.

WorkAction: An object of WorkAction expresses the normal work (like development, quality assurance, and so on). During an execution of the act method in this action, the employee works on a task. The task was derived from an instance (requirement) which is located at any activity in the process, the employee is responsible to. The work action was defined to have a probability of 75 % and a minimum duration of 15 cycles. It is finished if the task has no more remaining workload. If the activity finishes, an object of the action CommitResultAction has to be created.

CommitResultAction: The creation of a CommitResultAction requires a task. The aim of this action is, to move the related instance to the next step within the process. The progression of the act ()-method leads to the selection of the next flow that connects the activity (the instance is located on) to the next node(s). The employee tries to move the instance forward. The recipient of the instance could be an internal activity. If the product is finished (guided by the process), the employee sends the finished instance to the customer. In normal cases, it is possible to finish the movement during one cycle. Depending on the process definition it could be impossible to move the instance to the target node, maybe because its stock is full. In this case, the CommitResultAction could not be finished until the instance was moved. The probability of a CommitResultAction was defined to be 35 %.

StressAction: If employees work on multiple tasks at the same time, it stresses them. The probability of this action vary, depending on the number of tasks the employee handles parallelly. For each WorkAction, the probability was defined to be increased by 2 %. Each unfinished CommitResultAction increases the probability by 1 %. Each WorkerFailAction increases its probability by 3 %, by definition. If this action is executed, the employee's health would decrease by 2 %. Thus, this action expresses the negative impact of stress that increases the risk of failures (WorkerFailAction) and sickness (AbsenceBecauseSicknessAction). In this way, we modelled feelings as actions. This simplification could be true, however, it could be that feelings happen parallel to other actions or even other feelings.

WorkerFailAction: The probability of the WorkerFailAction is defined by $pFailure := \{x \in R \mid if\ (1\text{- }knowledge) + (1\text{- }health) \leq 1,\ 1 - knowledge \leq x \leq (1\text{- }knowledge) + (1 - health),\ else\ 1\}$
If this action is created, it selects a task, the employee currently performs in a WorkAction. If the action is executed, it simulates a failure. The failure might concern the quality or the remaining workload of the task. The action calculates a fail rate (based on the knowledge of the employee) which expresses the intensity of the failure. With this intensity, the action hurts the task. The WorkerFailAction reduces the speed of instances. If the quality decreases, the instance has to iterate again through activities which already have been passed (depending on the compliance rules of the process). In reality, different solutions like the application of a code generator [18, 92–93] are available to avoid such failures.

AbsenceBecauseSicknessAction: If an employee is sick, we express this case by the AbsenceBecauseSicknessAction. The probability of this action is defined on the interval of $p_{Sickness} := \{ x \in R \mid 0 \leq x \leq 1\text{- health}\}$. The duration of an absence varies between 0,5 PT to 5 PT. After the AbsenceBecauseSicknessAction is finished, the health of the employee is completely recovered (100 %).

BreakAction: The BreakAction represents a period of cycles, during which the employee would do nothing. Its probability is defined by the interval $p_{pause} := \{ x \in R \mid 0 \leq x \leq (1\text{- motivation})\}$. The duration of a break varies between 5 to 40 minutes. If the employee finishes his break, his health would increase by 5 %. This conforms to the aim of a break [34].

Due to the usage of actions, the attributes of an employee have an impact on the performance and on the progression of the process. An employee has three percentage attributes: knowledge, motivation and health [17]. Higher percentages are believed to be better / stronger values. At the moment, when the employee is created, the attributes were initialized by a rational random value. In addition to these attributes which describe the personality of an employee, each worker has a salary. The salary is needed, to determine the costs and financial aspects of the process. Lin et al. explained that costs of activities come from the costs of resources which are associated to the activities [21, 1781]. Conform to this, costs of activities are determined by the salary of employees who are responsible for their progression. To investigate the financial performance,

we performed the above mentioned simulation. During these simulations we let the employees record what they were doing. Based on the summation of information about process instances and environmental conditions, our simulation engine was able to sum up the costs [32, 3]. After this, we calculated key performance indicators which express the economic situation of the development process during the simulation. Therefore, we calculated 2.496 operating numbers on the financial performance of the process. This values would be compared to the progression of breaks.

7. EVIDENCE OF THE RELATIONSHIP BETWEEN BREAKS AND ECONOMIC OUTPUT

In order to investigate the dependency of economic success and breaks, we analysed the above mentioned 2.496 simulations. The whole number of simulations is divided into two groups. We executed 1.314 simulations of a model which empowers the employees to make breaks. For comparison, we executed 1.181 simulations without the permission to make a break.

First, we examine the relation of breaks to accomplished function points. It shows a broad variance of economic output. Apparently, the average number of breaks in each experiment is nearly between 480 to 750 breaks. The average number of finished function points varies nearly between 30 to 150 function points. Mainly, it seems to confirm that breaks have no impact on the economic output. However, some discordant values exist. They are part of experiments which have higher number of breaks in combination with fewer completed function points.

To investigate the significance of the above mentioned observation, we calculated a hypothesis test. Our first hypothesis claims that breaks have no impact on the overall profitability. We compare the progression of simulations, depending on the permission of breaks. The calculation in equation starts with the definition of the hypothesis h_0. Contrary to normal, we defined our target hypothesis to be h_0. We tried to reject our hypothesis with a (very high) α - risk of 84.5%. Our calculation revealed that our hypothesis could not be rejected.

$$H_0 : \mu = 88,16FPs \tag{1}$$

$$H_1 : \mu \neq 88,16FPs \tag{2}$$

$$\sigma^2_{\overline{X}} = \frac{1.314}{1.314 - 1} \cdot 990,22 = 990,97FP^2 \tag{3}$$

$$\sigma_{\overline{X}} = \sqrt{990,97FP^2} = 31,48FP \tag{4}$$

$$\frac{|\overline{X} - \mu_0|}{\sigma_{\overline{X}}} = \frac{|93,86FPs - 88,16FPs|}{31,48FP} = 0,1810 \tag{5}$$

$$\frac{|\overline{X} - \mu_0|}{\sigma_{\overline{X}}} > z[1 - \frac{\alpha}{2}] \Rightarrow discard H_0 \quad (6)$$

$$z[1 - \frac{\alpha}{2}] = z[1 - \frac{0,845}{2}] = 0,19 \quad (7)$$

$$0,1810 < 0,19 \Rightarrow discard H_0 NOT! \quad (8)$$

Conditional probability in spot tests

The significance is (despite of the high α - risk of 84,5%) unknown, because the possible β uncertainty is unvalued. To analyse if breaks have an impact, we changed our hypothesis h_0. Based on the unity of all experiments that allow employees to make breaks, we calculated the 95% confidence interval of the average finished function points. In the equation the whole test is calculated. We compared the arithmetic average of produced function points in the experiments with breaks to the upper and lower bound of the confidence interval. The hypothesis h_0 claims that the real arithmetic average of the produced function points is outside the above mentioned interval in experiments with breaks. As calculated in, the hypothesis h_0 could be rejected. Therefore, we prove (with an α - risk of 1 %) that the mean number of accomplished function points is in the same confidence interval, independent from breaks.

$$P(29,95 \leq X \leq 146,36) = 95\% \quad (9)$$

$$H_{0_1} : \mu_{0_1} \leq 29,95 FPs \quad (10)$$

$$H_{0_2} : \mu_{0_1} \geq 146,36 FPs \quad (11)$$

$$H_{1_1} : \mu > 29,95 FPs \quad (12)$$

$$H_{1_2} : \mu < 146,36 FPs \quad (13)$$

$$\sigma^2_{\overline{X}} = \frac{1.314}{1.314 - 1} \cdot 990,22 = 990,97 FP^2 \quad (14)$$

$$\sigma_{\overline{X}} = \sqrt{990,97 FP^2} = 31,48 FP \quad (15)$$

$$\frac{|\overline{X} - \mu_{0_1}|}{\sigma_{\overline{X}}} = \frac{|93,86FPs - 29,95FPs|}{31,48FP} = 2,03019511 \tag{16}$$

$$\frac{|\overline{X} - \mu_{0_2}|}{\sigma_{\overline{X}}} = \frac{|93,86FPs - 146,36FPs|}{31,48FP} = -1,667739685 \tag{17}$$

$$\frac{|\overline{X} - \mu_{0_1}|}{\sigma_{\overline{X}}} > z[1 - \frac{\alpha}{2}] \Rightarrow discard H_{0_1} \tag{18}$$

$$\frac{|\overline{X} - \mu_{0_2}|}{\sigma_{\overline{X}}} > z[1 - \frac{\alpha}{2}] \Rightarrow discard H_{0_2} \tag{19}$$

$$z[1 - \alpha] = z[1 - 0,845] = 1,64 \tag{20}$$

$$z[\alpha] = z[0,845] = -1,65 \tag{21}$$

$$2,03019511 > 1,64 \Rightarrow discard H_{0_1}! \tag{22}$$

$$-1,667739685 < -1,65 \Rightarrow discard H_{0_2}! \tag{23}$$

Statistical hypothesis test to determine if breaks have no impact on accomplished function points To review our result, we calculated a regression analysis. Instead of comparing breaks to finished function points, we compared it to the cycle time of instances. Our regression analyses the influence of breaks on the instance duration time. The result is described in the calculation. Breaks have a minor impact on instance cycle time which is why the correlation coefficient r_{XY} is 0,27. The adjusted coefficient of determination R^2 is 0,07215.

$$n = 1.314 \tag{24}$$

$$Breaks : \overline{X} = 566,87 \tag{25}$$

$$Instanceduration time : \overline{Y} = 3520,11 \tag{26}$$

$$s_X^2 = 35992,63 \tag{27}$$

$$s_Y^2 = 1671579,56 \tag{28}$$

$$s_X = 189,72 \tag{29}$$

$$s_Y = 1292,90 \tag{30}$$

$$c_X = 65886,06 \tag{31}$$

$$r_{XY} = 0,27 \tag{32}$$

$$R^2 = 0,07215 \tag{33}$$

Influence of breaks on the duration time of instances

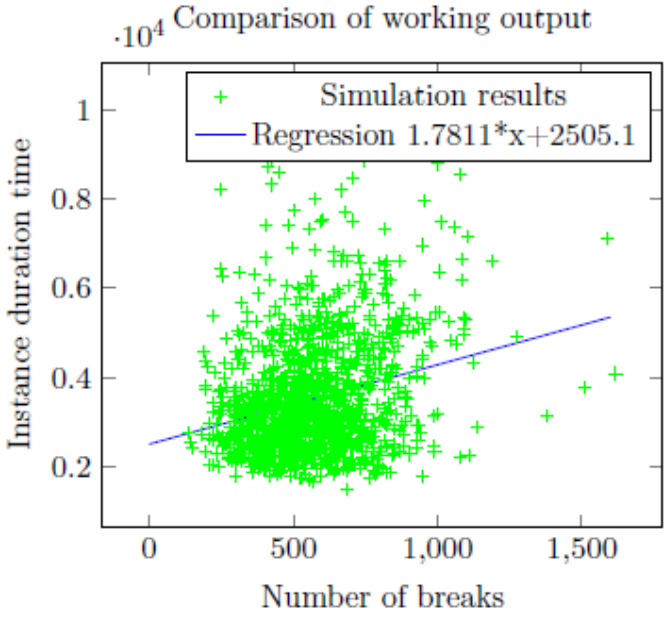

Figure 3: Influence of breaks on the duration time of instances

As illustrated in figure 3, a very low positive correlation exists. Thus, more working hours increase the output. However, more breaks do not decrease the productivity. Diagram 4 shows the spreading of experiments, with regard to the finished function points influenced by breaks. There is nearly no difference between the experiments, regarding to the impact of breaks.

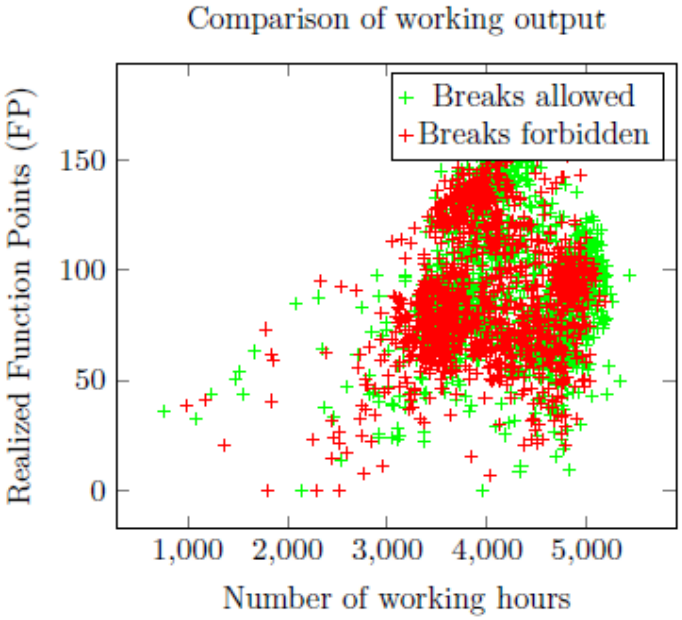

Figure 4: Relation between the number of pauses to accomplished function points

If developers are allowed to make breaks, this does not have any influence on the economic performance. The hypothesis that breaks have no impact on the profitability could not be rejected. Rather, it is possible to prove that software development leads to the same number of produced functions points on average, compared to the 95% confidence interval even if developers are free to interrupt their work to pause.

Considering other social effects, like the absence of sickness, enable the execution of a simulation, in order to investigate the impact of breaks. Our experiment proves that breaks have no impact on the overall profitability. Although breaks reduce working hours, they could reduce the sickness absence rate and fewer faults (under consideration of conditions in). This could prevent a loss of working time. Their positive effects might be able to economize the costs of breaks. Strategies of big companies which empower their employees to work creatively and organize breaks independently are no faults, economically.

During our investigations we revealed that the independent variables which express the breaks are not connected to the dependent, economic variables. Precise, we reveal that (under the prerequisites mentioned above (see)) there is no significant relation between the number of breaks and the amount of finished function points.

8. CONCLUSION

Current trends suggest organizing software production as a free, creative and self-determined process. Standardized approaches (agile and conventional) support this freedom. Such trends contradict to the appropriation, to organize development to be a production process. The idea of industrial production of software is economically attractive, because it facilitates the scheduling of output. Management seeks to reduce confounding factors which hamper forecasts. Employees who pause autonomously could be seen as reason for performance losses.

In fact, the human-centric-characteristic of software development leads to two process models which encounter each other. First, all employees interact in a business process. These business processes determine a sequence of activities. This business process is prescribed by the development approach. It becomes performed by a social process. During this social process, each employee could follow the process, but he is also able to act independently. The implementation of actions gives freedom the employees to interact.

The whole complexity of the opportunities for social actions exceeds the possibility of a simulation model. Therefore, the simulation model is a simplification of possible actions, an employee or a customer could perform. In general, progression happens if the employee works. In our analysis we investigated if progression is harmed if developers make normal breaks. We determined the success of a development team within 2.496 simulations and compared it to the number of breaks. As a result, we observed that breaks have no impact on the profitability of the development.

Our results indicate that freedom would not harm the profitability in software development. It does not quit the discussion about the design of a production process. However, it indicates that developers who are empowered to act independently could not harm the profitability if their action happens in a normal bound. The results of our investigation are usable in the economic. In practice, employees could do more than just making a break. It is possible for employees to do a vast research on a technical subject. As explained above, all potential actions could not be modelled. We ignore that these other actions have a (maybe small) impact on the success and concentrate on breaks.

We defined a normal bound of a break, to take between 5 to 40 minutes. In our analysis we allowed developers to make breaks of this duration, autonomously. The next step has to be a review of the parameters and calculations that limit and control the progression.

As Recker explains, the constitution of a valid business process model is a field of research that has just emerged [27, 44f.]. In the context of software development, this leads to the need to consider more details in the simulation model, in order to improve the match of reality. In further investigations it might be sensible to append the framework of PAS with functionality to consider values of parameters, observed in reality.

9. REFERENCES

[1] Witold Abramowicz, Agata Filipowska, Monika Kaczmarek, Carlos Pedrinaci, Monika Starzecka, and Adam Walczak. Organization structure description for the needs of semantic business process management. In 3rd international Workshop on Semantic Business Process Management colocated with 5th European Semantic Web Conference, Poznań, Poland, January 2008. ResearchGate.

[2]P. G. Armour. The business of software estimation is not evil. COMMUNICATIONS OF THE ACM, 57(1):42–43, 2014.

[3] Sarah Beecham, Nathan Baddoo, Tracy Hall, Hugh Robinson, and Helen Sharp. Mo-tivation in software engineering: A systematic literature review. Information and software technology, 50(9):860–878, September 2008.

[4] Barry Boehm. Value-based software engineering: Overview and agenda. 2005.

[5] Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. Communications of the ACM, 31(11):1268–1287, November 1988.

[6] Thomas H Davenport. Process Innovation - Reengineering Work through Information Technology. Havard Business School Press, Boston, Massachusetts, 1993. Ernst & Young Center for Information Technology and Strategy.

[7] Thomas H Davenport. The coming commoditization of processes. Harvard business review, 83(6):100–108, 2005.

[8] Dipl-Ing Jutta Eckstein. Agilität – ein baustein der dritten industriellen revolution. HMD Praxis der Wirtschaftsinformatik, 50(2):77–83, 2013.
[9] Johann Eder, Horst Pichler, Wolfgang Gruber, and Michael Ninaus. Personal sched-ules for workflow systems. In Wil M.P. van der Aalst, Arthur ter Hofstede, Mathias Weske, Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen, editors, Business Pro-cess Managemment, number LNCS 2678 in Lecture Notes in Computer Science, pages pp. 216–231, Einhoven, The Netherlands, June 26-27 2003. Springer. International Conference on Business Process Management (BPM 2003).
[10] Rainer Endl and Martin Meyer. Potential of business process modelling with regard to available workflow management systems. Number 20 in SWORDIES Report. Springer, February 1999. Published in Scholz-Reiter, B.; Stahlmann, H.-D.; Nethe, A. (Eds) PProcess Modelling"Berlin, Heidelberg, New-York, 1999.
[11] Abbie Griffin. The effect of project and process characteristics on product development cycle time. Journal of Marketing Research, pages 24–35, 1997.
[12] Michael Hammer and James Champy. Reengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness A Division of HarperCollinsPublishers, New York, NY, USA, 1993.
[13] Vlatka Hlupic and Stewart Robinson. Business process modelling and analysis using discrete-event simulation. In Proceedings of the 30th conference on Winter simulation, pages 1363–1370. IEEE Computer Society Press, 1998.
[14] Sebastian Kiebusch, Bogdan Franczyk, and Andreas Speck. An unadjusted size mea-surement of embedded software system families and its validation. Software Process: Improvement and Practice, 11(4):435–446, 2006.
[15] David Kuhlen and Andreas Speck. Wertanalyseverfahren für kundenanforderungen. In Erhard Plödereder, Lars Grunske, Eric Schneider, and Dominik Ull, editors, IN-FORMATIK 2014 Big Data - Komplexität meistern, volume P-232 of Lecture Notes in Informatics (LNI) - Proceedings, pages 2317 – 2322, Stuttgart, September 2014. Gesellschaft für Informatik e.V. (GI). Thanks to Prof. Dr. Andreas Speck and Prof. Dr. Hinrich Schröder.
[16] David Kuhlen and Andreas Speck. Business process analysis by model checking. In 5th International Symposium on Data-Driven Process Discovery and Analysis SIMPDA 2015, pages 154–170, Vienna, Austria, December 2015. Ceravolo, Paolo and Rinderle-Ma, Stefanie.
[17] David Kuhlen and Andreas Speck. How to design a simulation software, to evaluate the economic performance in software development. 2016.
[18] David Kuhlen and Andreas Speck. The potentials of a code generator which faces the stress ratio of requirements engineering processes in agile development projects. In Modellierung 2016 Workshopband, volume P-255 of Lecture Notes in Informatics, pages 87–96, Karlsruhe, March 2016. Gesellschaft fÃ¼r Informatik e.V. (GI), Betz, Stefanie and Reimer, Ulrich.
[19] Marco Kuhrmann and Oliver Linssen. Welche vorgehensmodelle nutzt deutschland. PMV 2014, pages 17–32, 2014.
[20] Marco Kuhrmann and Oliver Linssen. Vorgehensmodelle in deutschland: Nutzung von 2006-2013 im überblick. Projektmanagement+ Vorgehensmodelle 2014, pages 32–47, 2015.
[21] Huiping Lin, Yushun Fan, and Stephen T Newman. Manufacturing process analysis with support of workflow modelling and simulation. International Journal of Produc-tion Research, 47(7):1773–1790, 2009.
[22] D. C. McDermid. Integrated business process management: Using state-based busi-ness rules to communicate between disparate stakeholders. In Wil M.P. van der Aalst, Arthur ter Hofstede, Mathias Weske, Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen, editors, Business Process Managemment, Lecture Notes in Computer Sci-ence, Einhoven, The Netherlands, 2003. Interational Conference BPM, Springer.
[23] John G Mooney, Vijay Gurbaxani, and Kenneth L Kraemer. A process oriented framework for assessing the business value of information technology. ACM SIGMIS Database, 27(2):68–81, 1996.
[24] David Lorge Parnas and Paul C Clements. A rational design process: How and why to fake it. IEEE Transactions on Software Engineering, SE-12(2):251–257, February 1986.
[25] Hans-Jürgen Plewan and Benjamin Poensgen. Produktive Softwareentwicklung: Be-wertung und Verbesserung von Produktivität und Qualität in der Praxis, volume 1. Auflage. dpunkt. verlag, Heidelberg, Germany, Juli 2011.
[26] William F Pounds. The process of problem finding. Library of the Massachusetts Institute of Technology, 1965.

[27] Jan Recker. A socio-pragmatic constructionist framework for understanding quality in process modelling. In Australasian Journal of Information Systems, volume 14, pages 43–63. Citeseer, June 2007.
[28] Shazia Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In Business process management, volume 4714 of Lecture Notes in Computer Science, chapter Business Process Management, pages 149–164. Springer, Brisbane, Australia, 5th international conference on business pro-cess management edition, 2007.
[29] J. Schwaber, K.; Sutherland. Der scrum guide der gültige leitfaden für scrum: Die spielregeln, July 2013. Zuletzt Abgerufen am 06.09.2014.
[30] F Simon, A Kossmann, M Kuhrmann, and D Mendéz Fernández. Wunsch oder wirk-lichkeit? professionelle softwareentwicklung âmade in germany â. OBJEKTspektrum (in German), (1):16–23, 2014.
[31] Andreas Speck, Elke Pulvermüller, and Dirk Heuzeroth. Validation of business pro- cess models. In Proceedings of ECOOP 2003 Workshop Correctness of Model-based Software Composition (CMC), pages 75–83, 2003.
[32] Patrik Spieß, Dinh Khoa Nguyen, Ingo Weber, Ivan Markovic, and Michael Beigl. Modelling, simulation, and performance analysis of business processes involving ubiq-uitous systems. In Advanced Information Systems Engineering, pages 579–582. Springer, 2008.
33] Wil MP van der Aalst. Challenges in business process analysis. LECTURE NOTES IN BUSINESS INFORMATION PROCESSING, January November 2007.
[34] Günter Wöhe and Ulrich Döring. Einführung in die Allgemeine Betriebswirtschaft-slehre. Verlag Franz Vahlen München, 23. auflage edition, 2008.
[35] Thomas Zink. Class for aircc journal submissions, 2012. This is an unofficial Latex class for Authors of AIRCC Papers. Access timestamp: 2016-10-19.