International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Contribution of Code Generation in Order to Save Process Costs in Software Development

*David Kuhlen**

**IU International University of Applied Sciences, Waterloohain 9, 22769 Hamburg, Germany* david.kuhlen@iu.org

DOI : https://doi.org/10.55248/gengpi.5.0724.1814

A B S T R A C T

The use of code generators is expected to be an improvement of the software development process. However, the introduction of such new techniques means a shift of work. Some activities will be more expensive, some benefits might be realized. Advantages and disadvantages have to be balanced, to appraise the investment decision. In this paper an analysis of a changed software development will be presented, based on a simulation. The results show benefits of the introduction of code generation.

Keywords: Code Generation, Simulation Analysis

Introduction

The success of software development depends on the performance of its stakeholders. Software developers need to perform their tasks quickly and with a high quality. Modern development techniques may assist developers to perform their tasks. Tools like code generators can play an important role to derive software quickly which meets the requirements Kuhlen and Speck (2016c).

In this paper, the benefits offered by an investment in generated requirements engineering on the performance of the software development process will be evaluated. The paper builds up on previous work, published in Kuhlen and Speck (2015a, 2016a, 2016b, 2016c, 2017b, 2017c). In Kuhlen and Speck (2015b) and Kuhlen and Speck (2016c) aspects of code generation are analyzed. The introduction of code generation promises to improve the software development process Kuhlen and Speck (2015b, 2016c). In this paper, an assumption on the process improvement is made to evaluate its benefit. It should be possible to find out which strategy is successful and rate it in terms of costs W. M. van der Aalst (2008).

Theory

Kuhlen and Speck declared that the use of a code generator in the requirements engineering could produce better results and it helps shifting effort from software development to requirements engineering Kuhlen and Speck (2015b, 2016c).

A process analysis might show if an investment is valuable. Before a process becomes changed, the performance of the target process has to be evaluated. The correctness, effectiveness and the efficiency of the business process supported by an information system are vital for an organization W. M. van der Aalst (2008).

The introduction of code generation could disburden the software developers from doing routine job. This brings a shift to more complex work done by developers. However, such more complex jobs may require a further specialization. The exchange of information will be profitable, if developers could specialize themselves. Multiple jobs could be completed simultaneously, in a reengineered process (Hammer &Champy, 1993, 3). Parallelism supports the real-time execution of orders (Davenport, 1993, 250). In software developing projects, it is often remarkable, that the construction of the team doesn’t support parallelism: if project members are just working in one project at the time, they need to be involved in the moment when something to do is available. This is a disadvantage of a high specialization. The involvement of software developers which are high specialized requires a high quality project planning. If experts are involved to early (so they can’t work effective parallel) or to late (so a congestion emerges) this fails! Plewan and Poensgen differentiate between projects which starts with a lack of employees (Plewan&Poensgen, 2011, 228) and projects which are overstaffed (Plewan&Poensgen, 2011, 217).

This circumstance underlines the necessity to perform individual process analysis in practice, before changing a procedure. Business process analysis is part of the business process management W. M. P. van der Aalst, ter Hofstede, and Weske (2003). Projects should operate a bit in an isolated capsule (Plewan&Poensgen, 2011, 219). The process model is the (first) base for an automated checking Speck, Feja, Witt, Pulvermüller, and Schulz (2011).

Designing a process model is far from trivial, because it requires long discussion with the team members and the management about the business practices W. M. P. van der Aalst and van Dongen (2002). A new process model prescribes how business processes should be executed Schimm (2003). Therefore, the model is essential for the future success of the business. To prevent the business from making mistakes during the design, a valid assistance is needed.

Especially if the business process contains multiple rules which control the order of steps, the verification of such systems is quite hard for human beings and requests automated assistance Speck et al. (2011). To solve these difficulties in the design, work flow mining techniques could be used to get an insight in the life-cycle of different cases W. M. P. van der Aalst and van Dongen (2002). However, often it is hard to set up a mining procedure and without a process model the insights may be useless.

As the reality often differs from what is modelled W. M. van der Aalst (2008), the correction of the model has to be possible. It could also be an objective to correct the reality, to be closer to the model. If data about the reality is hard to obtain, it should be possible to make assumptions on the behavior of the process more easily.

Method

On the basis of the state of research and practical experience, a code generator affects the performance of the software development process in multiple areas. These benefits have to be mapped to the process of software production.

- The costs of writing a concept may increase, because the code generator has to be used.
- The costs to accept a concept may increase, because of the formal syntax of the code generator.
- The costs of software development decrease, because of the generation of code.
- The probability of software faults decreases, because generated code has a proofed quality.

Figure 1: Assumed effects of a code generator on the performance of the software development process

In (Kuhlen& Speck, 2015a, p. 160), an example model of the classic software development process is designed. This model could be analyzed by simulation. To assess the impact of a code generator on the performance, a new version of the process model has to be designed at first. Therefore, an alternative version (compared to the original version in (Kuhlen& Speck, 2015a, p. 160)) of the process model is designed. This new version bases on the above-mentioned assumptions displayed in Figure 1 about the benefits obtained by a code generator. In Figure 2 the new version is illustrated. The deviations in the new process version are highlighted in blue color (Cf. (Kuhlen& Speck, 2015a, p. 160)). The changes in the proceeding affect the costs of some activities and the probabilities of several flows. The costs may be influenced, because some activities take more time (e.

g. writing or accepting a concept) and some activities take less time (e. g. development). For example, it can be expected that a generated code contains less bugs which leads to an increased probability to join the activity "roll-out" after the quality assurance. These consequences are conceivable. Maybe there are further effects or some of the assumptions might differ from real effects. This deviations from reality doesn’t matter now. The analysis of different versions of a process has to be possible by making assumptions. In practice, business economists are able to make individual assumptions, based on operating numbers from the controlling to design a decision model which fits into reality.

Figure 2 shows the new version of the development process. This new version has to be compared with the original version, presented prior.

Both process models are analyzed by PAS, see Kuhlen and Speck (2015a, 2016a, 2017c). The analysis algorithm of PAS constructs two lists of computable process paths. Each of these process paths was appraised with its costs and its probability. The expectable costs of the processes were calculated by summarizing the costs of the paths, weighted with their probability Kuhlen and Speck (2015a, 2017a, 2017c).

Especially in the phase of process design, the results of a process analysis could be helpful, to hinder (financial) problems. The costs of changing a process during its design phase are (like always) lower, than after the process is used in operative business. Especially if the process models may involve the core business transactions, its design has to be accurate W. M. P. van der Aalst et al. (2003).

Analysis

During the phase of requirements engineering a procedure is necessary, to check the profitability of the new process design. Without a precise specification of the required process change (requirement) Chaudron, van Hee, and Somers (2003) the evaluation of its profitability will be a problem. By determining flow times, bottlenecks

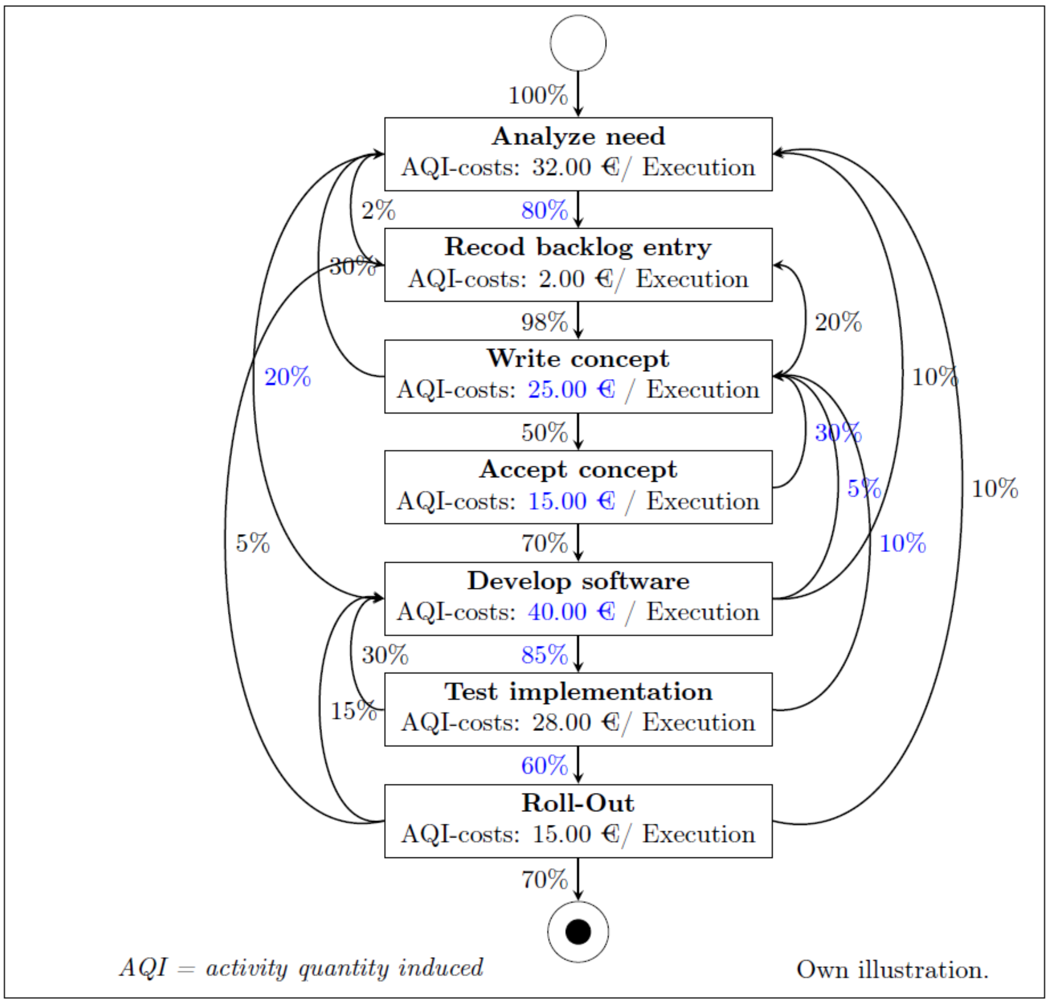
100%
Analyze need
AQI-costs: 32.00 €/ Execution
2%
80%
Recod backlog entry
AQI-costs: 2.00 €/ Execution
30%
98%
20%
Write concept
AQI-costs: 25.00 € / Execution
20%
10%
50%
30%
Accept concept
AQI-costs: 15.00 € / Execution
5%
10%
70%
5%
10%
Develop software
AQI-costs: 40.00 € / Execution
30%
85%
Test implementation
AQI-costs: 28.00 €/ Execution
15%
60%
Roll-Out
AQI-costs: 15.00 €/ Execution
70%
AQI = activity quantity induced
Own illustration.

Figure 2: New possible variant of the process by using a code generator, compare the first version of this illustration in (Kuhlen& Speck, 2015a, p. 160), deviations are highlighted. Own illustration created with TikZ (tikzpicture) being part of TeX Live Version 2024

and the utilization of a process W. M. P. van der Aalst et al. (2003), business improvements could be made. If parts of a process cannot be reached, there is a mistake in the process model W. M. P. van der Aalst et al. (2003). In this analysis, the financial performance before and after the process improvement becomes compared. (Kuhlen& Speck, 2015a, p. 160) illustrates a process model which describes a common approach of software development. The progress of software development often follows a process model which was influenced by different procedure models in practice. Approaches like scrum, waterfall model, kanban, extreme programming (xp), rational unified process (rup) or the v-model have influenced the process of software development (Sommerville, 2016), like many others. (Kuhlen& Speck, 2015a, p. 160) shows one alternative of the huge variety of processes. In general, the process in Figure 2 and the process in (Kuhlen& Speck, 2015a, p. 160) describe the core workflow of software development which does not take aspects of the overall procedure model into account. However important functions for the software development which are described in procedure models have to analyzed too, in order to design a valid simulation model.

A classic process for software development might consist of seven activities. It starts with the analysis of the customer's needs. The need analysis belongs to the phase of requirements engineering. In the v-model this activity may correspond to the concept of operations (Sommerville, 2016). In scrum the product owner receives this information directly from the customers (Sommerville, 2016). In the waterfall model it belongs to the step of "requirements" (Sommerville, 2016). After the analysis of the needs, the requirement is recorded in a list for the development. In scrum this list is called "backlog" (Sommerville, 2016). In the v-model this step matches to the activity "system requirements" (Sommerville, 2016). In the waterfall model, the next activity is called "design" (Sommerville, 2016). Often the design is written in a concept, therefore (Kuhlen& Speck, 2015a, p. 160) calls the next step "Write a concept". After the concept is written, it has to be accepted by the customer. Subsequently the development could start, and the product could be tested. The v-model differentiates between multiple phases of quality insurance (Sommerville, 2016). However, the waterfall model distinguishes between "implementation" and "verification", close to this process in (Kuhlen& Speck, 2015a, p. 160; Sommerville, 2016). The last step of development aims to put the software in production. The step "roll-out" may be adequate for the activity of maintenance in v-model or in the waterfall model (Sommerville, 2016). However, "maintenance" might include changes in the system which are often treated like "normal" requirements. These required changes have to pass the whole process again. In comparison, the installation of the software belongs to the "roll-out". Most of the activities could lead back to a restart of the whole process (cycles). A restart could have different reasons like the emergence of faults or unexpected complexity. This "steps back" lead to an iteration through the process.

The processes given in (Kuhlen& Speck, 2015a, p. 160) and Figure 2 describe state machines. The distinction of seven activities allows the transition to a huge number of different states. Speck et al. describe the problem of state explosion as a problem in the context of model checking Speck, Pulvermüller, and Heuzeroth (2003). Especially, because the defined process contains multiple cycles and short cuts its number of states will be enormous. However, the verification is profitable because multiple errors could be found more often W. M. van der Aalst (2008).

Results

After the simulation was performed the obtained results for the two processes in (Kuhlen& Speck, 2015a, p. 160) and Figure 2 are compared. First the impact of requirements engineering on the costs will be analyzed. Costs are expected to decrease because less iterative cycles are needed to produce software by using a code generator. Costs in software development are mainly fixed, as a result of the salaries. Therefore, monthly costs are stable, more or less. A key performance indicator was calculated to assess the costs of each instance, individually. First these costs base on the time, the developers spend on the realization of a requirement (=instance). This development time was divided by the total number of working hours. The costs of instances are expected to be decreased, because each requirement needs less development time.

The introduction of a code generator improves the requirements engineering and the development within the software production process. In Kuhlen and Speck (2016b) the effect of an improvement to requirements engineering was analyzed. Based on the investigation in Kuhlen and Speck (2016b) figure 3 shows the potential benefit of such improvement in general without connection to code generation. A comparison of efficiency improvement and the average costs of requirements is displayed in 3. The box plot facilitates to compare the mean requirements costs in distinguished classes of efficiency. Comparing the median shows, that the costs are nearly stable, independent from the efficiency of requirements analysis. This result is reasonable, considering a stable workload in combination with increasing costs, due to a more comprehensive requirements engineering. Average instance costs are the wrong key performance indicator to compare the impact of improved requirements engineering.

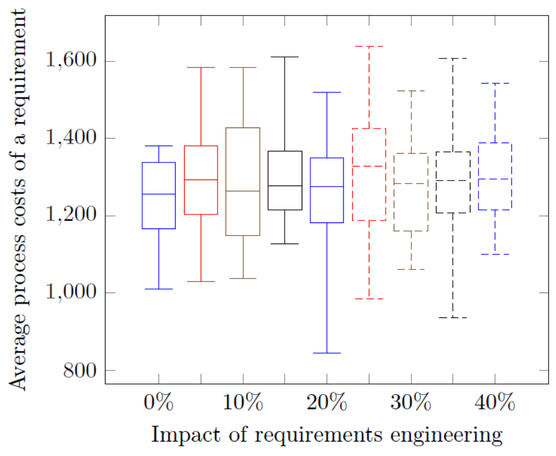

Figure 3: Total costs of implementation, influenced by requirements engineering's impact. Own illustration created with TikZ (tikzpicture) being part of TeX Live Version 2024.

The classic model of the process leads (with a save probability of 76,65%) to costs of 284. 83 € per execution. In comparison, the new process leads (with a save probability of 77,37 %) to costs of 213. 91 € per execution. Therefore, a reduction of operating costs about 70.92 € per execution can be expected by this process improvement. The reduction is the result of a different distribution of the work. In Figure 4 the average numbers of repetitions of each activity are displayed. The bar chart displays the focus of the work in the context of different process versions.

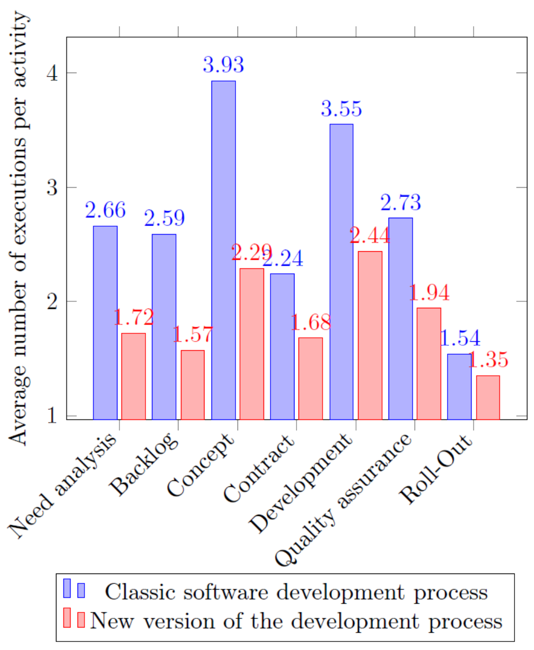

Figure 4: Average number of repetitions per activity. Own illustration created with TikZ (tikzpicture) being part of TeX Live Version 2024

Figure 4 shows how often an activity will be repeated in the iterative process of software development, during the experiment. In the new version of the process, less repetitions, e. g. of the activity "write concept", can be expected. This might happen, because a formal specification of requirements is much more precise and would not be readjusted as often as classic pedestrian specifications. In contrast, the repetitions of the activity "rollout" will not differ significantly. Overall, a faster and tighter development process can be expected, because of this change in the development process.

The positive contribution of requirements engineering to the economic profitability of software development is shown. An improvement of requirements engineering improves the number of produced function points Kuhlen and Speck (2016b). To realize great costs effects, further adjustments (for example on the model of payments) are necessary in practice.

Conclusion

The use of a code generator offers a great potential to decrease the costs. In nearly all activities in the software development process the number of repetitive cycles will be decrease by using the described process improvement. This leads to lower costs.

Further investigations could analyze more potentials to reduce the costs in the software development. To express the financial efficiency of a process, the activity-based costing (ABC) approach could be used for example. In practice this approach would not be used during the design of a process model. The effect of process controlling itself on the process performance might be an interesting extension of simulation analysis with PAS.

References

Chaudron, M., van Hee, K., & Somers, L. (2003). Use Cases as Workflows. In W. M. P. van der Aalst, A. ter Hofstede, & M. Weske (Eds.), Business Process Managemment. International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings (Vol. LNCS2678, pp. 88–103). Berlin, Heidelberg, New York: Springer-Verlag. (DOI: 10.1007/3-540-44895-0_7)

Davenport, T. H. (1993). Process Innovation. Reengineering Work through Information Technology. Boston, Massachusetts: Havard Business School Press. (Ernst & Young. ISBN: 0-87584-366-2)

Hammer, M., & Champy, J. (1993). Reengineering the Corporation. A Manifesto for Business Revolution. New York, USA: HarperBusiness A Division of HarperCollins Publishers. (ISBN: 0-88730-687-X)

Kuhlen, D., & Speck, A. (2015a, December). Business process analysis by model checking. In P. Ceravolo & S. Rinderle-Ma (Eds.), SIMPDA 2015. Data-driven Process Discovery and Analysis. Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015) Vienna, Austria, December 9-11, 2015. (Vol. 1527, pp. 154–170). CEUR Workshop Proceedings. (Available at CEUR: https://ceur-ws.org/Vol-1527/paper15.pdf)

Kuhlen, D., & Speck, A. (2015b). Facing technical debt with the help of code generation and requirements engineering. TechRxiv. (Unpublished, Manuscript submitted for publication)

Kuhlen, D., & Speck, A. (2016a, November). Economic output under the conditions of social freedom in software development. International Journal of Software Engineering & Applications (IJSEA), 7(6), 17 – 31. (DOI: 10.5121/ijsea.2016.7602)

Kuhlen, D., & Speck, A. (2016b, November). Efficiency of Software Development after Improvements in

Requirements Engineering. International Journal of Software Engineering & Applications (IJSEA), 7(6), 65–74. (DOI: 10.5121/ijsea.2016.7605)

Kuhlen, D., & Speck, A. (2016c, March). The potentials of a code generator which faces the stress ratio of requirements engineering processes in agile development projects. In S. Betz & U. Reimer (Eds.), Modellierung 2016. Workshopband (Vol. P-255, pp. 87–96). Bonn: Köllen Druck+Verlag GmbH.

Kuhlen, D., & Speck, A. (2017a, September). Efficient Key Performance Indicator Calculation. International

Journal of Software Engineering & Applications (IJSEA), 8(5), 1–15. (DOI: 10.5121/ijsea.2017.8501)

Kuhlen, D., & Speck, A. (2017b, July). Magnitude economic effects of requirements in the development process - a simulation study of workload behaviour. In Proceedings of the 2017 SAI Computing Conference (SAI) (pp. 953–960). IEEE. (DOI: 10.1109/SAI.2017.8252209)

Kuhlen, D., & Speck, A. (2017c, January). The way of designing a simulation software in order to evaluate the economic performance in software development. In Proceedings of The 8th International Conference on Computer Modeling and Simulation. ICCMS 2017. Canberra, Australia January 20-23, 2017 (pp. 109 – 113). New York, USA: Association for Computing Machinery (ACM). (DOI: 10.1145/3036331.3036347)

Plewan, H.-J., & Poensgen, B. (2011). Produktive Softwareentwicklung. Bewertung und Verbesserung von Produktivität und Qualität in der Praxis (1. Auflage). Heidelberg: dpunkt.verlag GmbH. (ISBN: 978-3-89864-686-4)

Schimm, G. (2003). Mining most specific Workflow Models from Event-based Data. In W. M. P. van der Aalst, A. ter Hofstede, & M. Weske (Eds.), Business Process Management. International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings (Vol. LNCS2678, pp. 24–40). Berlin, Heidelberg, New York: Springer-Verlag. (DOI: 10.1007/3-540-44895-0_3)

Sommerville, I. (2016). Software Engineering (Tenth Edition). Harlow, England: Pearson Education Limited. (ISBN: 978-1-292-09613-1).

Speck, A., Feja, S., Witt, S., Pulvermüller, E., & Schulz, M. (2011, May). Formalizing Business Process Specifications. Computer Science and Information Systems (ComSIS), 8(2), 427–446. (DOI: 10.2298/CSIS110111015S)

Speck, A., Pulvermüller, E., & Heuzeroth, D. (2003, July). Validation of Business Process Models. In R. Van Der Straeten, A. Speck, E. Pulvermüller, M. Clauß, & A. Pleuss (Eds.), Proceedings of Correctness of Model-based Software Composition (CMC). ECOOP 2003 (pp. 75–83). Darmstadt. (Technical Report No. 2003-13)

van der Aalst, W. M. (2008). Challenges in Business Process Analysis. In J. Filipe, J. Cordeiro, & J. Cardoso (Eds.), Enterprise Information Systems. 9th International Conference, ICEIS 2007, Funchal, Madeira, June 12-16, 2007, Revised Selected Papers (Vol. 12, pp. 27 – 42). Berlin, Heidelberg: Springer. (DOI: 10.1007/978-3-540-88710-2_3)

van der Aalst, W. M. P., ter Hofstede, A. H. M., & Weske, M. (2003). Business Process Management: A Survey. In W. M. P. van der Aalst, A. ter Hofstede, & M. Weske (Eds.), Business Process Management. International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings (Vol. LNCS2678, pp. 1–12). Berlin, Heidelberg, New York: Springer-Verlag. (DOI: 10.1007/3-540-44895-0_1)

van der Aalst, W. M. P., & van Dongen, B. F. (2002). Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, & D. Wikarski (Eds.), Engineering and Deployment of Cooperative Information Systems. EDCIS 2002 (Vol. LNCS2480, pp. 45–63). Berlin, Heidelberg: Springer. (DOI: 10.1007/3-54045785-2_4)

The TeX Users Group (TUG). TeX Live, 2024. Version 3.141592653-2.6-1.40.26