Does Continuous Requirements Engineering need Continuous Software Engineering?

Peter Forbrig

University of Rostock, Chair in Software Engineering,
Albert-Einstein-Str. 22, 18055 Rostock
Peter.forbrig@uni-rostock.de

Abstract. Agile development methods allow rapid adaptations of requirements to fast changing needs in businesses and society. Their combination with Continuous Requirements Engineering seems to be very effective. However, agile methods are applied to the development process only. Maintenance is often not organized in the same way. Continuous Delivers might be a solution for that.
Additionally, the focus on users is important. Currently, often customers are mentioned only.
The paper discusses aspect of Continuous business process modeling, Continuous Requirements Engineering, and Human-Centred Design in relation to Continuous Software Engineering. It especially focusses on the problem whether Continuous Requirements Engineering without Continuous Software Engineering makes sense.

Keywords: Continuous Requirements Engineering, Continuous Software Engineering, Continuous Human-Centered Design, Continuous Business-Process Modeling, Agile Software Development.

1 Introduction

Originally, software was developed according to the waterfall life cycle model. The spiral model changed this approach dramatically. However, the two process phases of development and application were still distinguished. From the software engineering point of view application of software goes together with maintenance. Maintenance begins when software development ends. A lot of attention has been paid to development methods and related life cycle models. However, the maintenance activities stayed unstructured at the same time. Because of their advantages for rapid changing application domains, agile software development methods became more and more popular. There have been nearly no discussions about agile methods for maintenance. In [19] the question of never ending projects was discussed. It was stimulated by ideas of Continuous Requirements Engineering [22] and Continuous Software Engineering [11]. In this paper details of both approaches are discussed and it will be discussed whether Continuous Requirements Engineering and Continuous Software Engineering are depended of each other. In other words, Continuous Requirements Engineering does not make much sense without Continuous Delivery. However, Continuous Delivery might

be necessary for improving problems in the implemented software that are not related to changing requirements. Therefore, Continuous Requirements Engineering might not always be necessary. The idea of integrating CRE into Continuous Software Engineering will be presented. Some arguments will be presented to support this ideas and some further ideas.

2 Continuous Software Engineering

2.1 Related Work

Recently, Fitzgerald and Stol [12] updated their model of Continuous Software Engineering [11]. It contains now Continuous Budgeting. Additionally, Continuous Innovation had been extended to Continuous experimentation and innovation. **Fig. 1** provides an overview of the identified activities.
The term Continuous Software Engineering can be traced back to 1998. During that time, it was called "Continuous Engineering for Industrial Scale Software Systems" by Weber and Müller [10]. A collection of different approaches to Continuous Software Engineering are contained in [6] edited by Bosch. Some ideas can be found in the workshop paper by Lichter et al. [21].

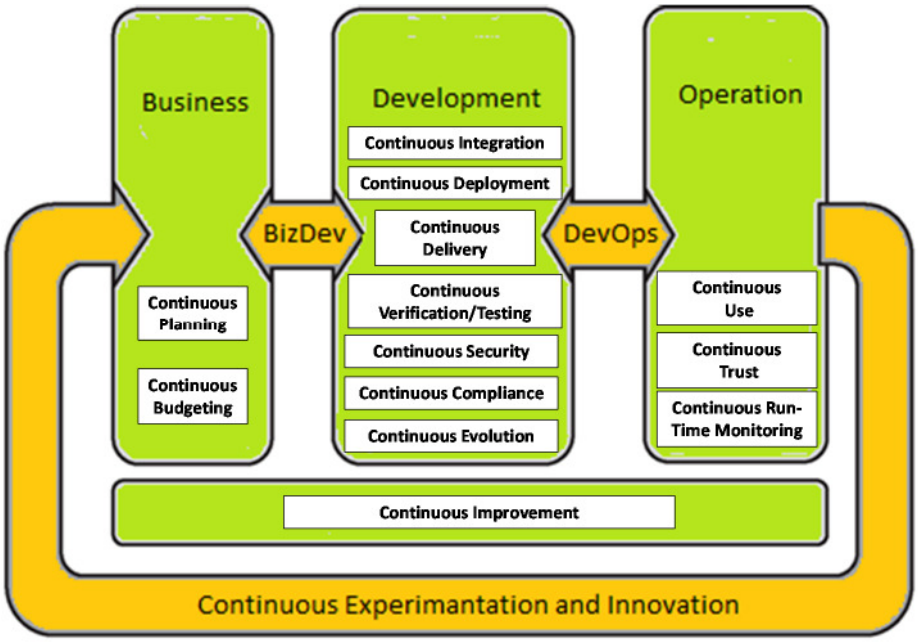

Fig. 1. Continuous∗: a holistic view on activities from business, development, operations, and innovation (from [12]).

Fitzgerald and Stol use the term BizDev (business development) for a continuous influence between business strategy and development. The more often used term DevOps (development and operations) refers to a "set of practices that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes"[1]. It relates development of software to operations in the sense of Continuous Monitoring the way the software interacts with by the users.
For DevOps the quality assurance plays an important role. This is reflected by **Fig. 2**. It represents DevOps as intersection of software development, quality assurance, and operations.

Fig. 2. Venn diagram describing DevOps[1]

It can be considered to start with a plan and create the software afterwards. After verifying the quality of the software it can be packaged and release. After configuring (deploying) the software its usage can be monitored. Based on the monitoring a new plan for creating software can be specified. The interconnection of development and operations of DevOps is visualized by **Fig. 3**.

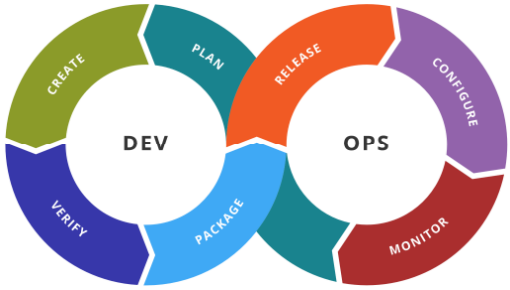

Fig. 3. Stages in a DevOps tool chain[1]

[1] https://en.wikipedia.org/wiki/DevOps

There are several attempts to provide such a pipeline of a tool chain. Such a tool chain can provide an immediate feedback from users to developers. However, there is no need to have always automatic feedback. Observation by humans can help as well. In some way, the monitoring activity can be considered as some kind of requirements elicitation. From this point of view, DevOps can be considered as part of the requirements engineering activities. However, those activities are part of the Business Strategy as well. We will come back to this aspect in the discussions.

2.2 Continuous Human-Centred Design

During the last two decades, agile development methods have been used in a lot of successful projects. They became more and more popular. Nevertheless, most of the time those methods are focused on customers and not on users.
In the first agility principle of [1] it is e.g. mentioned: "Our highest priority is to satisfy the customer through early and Continuous Delivery of valuable software."
From the business perspective, it makes sense to focus on the customer because the customer will pay the bill. However, to produce quality software it is very important to serve the needs of the users as well. In the same way as agile development methods are popular for software engineering experts Human-Centred Design (HCD) is popular for usability and user experience experts. HCD focusses on tasks users have to perform, usability and user experience. Unfortunately, these aspects do not play their important role for software engineering in general. In software engineering the focus is currently often on the technical aspects of an application only.

ISO 9241-210 is the standard for the HCD process that consists of a planning phase and four phases that are performed in an iterative way.
A visual impression of the HCD process model is provided by **Fig. 4**.

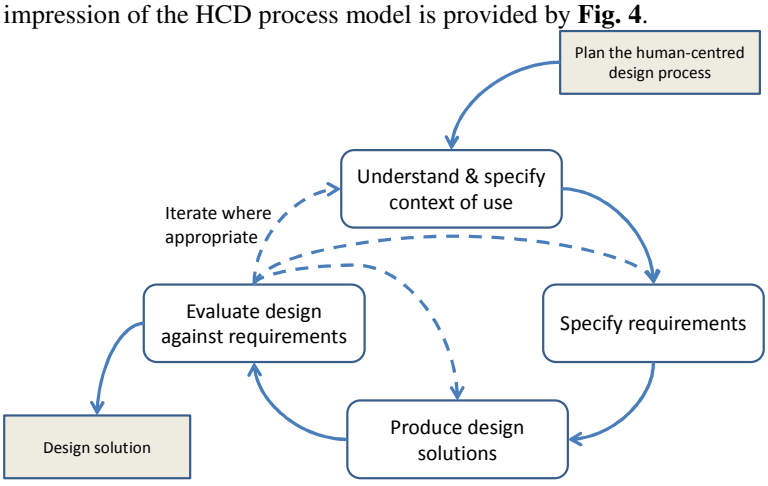

Fig. 4. Human-Centred Design process model [2]

[2] https: //thestandardinteractiondesignprocess.wordpress.com/

The context of use is tried to be understood in the first phase. Stakeholders are identified. Their roles and tasks are analyzed and typical application scenarios are specified. They include artefacts and tools from the domain. Additionally, surrounding people, services, and objects together with their location are identified.

This analysis allows the specification of requirements. Additionally to the goals of users, functional and nonfunctional requirements are identified.

Based on the requirements, first design solutions are produced. They include first ideas of user interfaces.

The design solutions are evaluated against the requirements. If they are met, the development process comes to an end and the implementation of the application core can be performed.

Otherwise, three possible continuations exist. In case of serious problems, one has to analyze the context of use again and has to proceed with the first phase. This can be considered as the worst case. If the context of use was understood correctly but some requirements were specified in the wrong way, one has to rewrite them or identify some new ones. Finally, it can be possible that only new design solutions are necessary.

Unfortunately, the process model of **Fig. 4** does not consider the integration of HCD into a development process. We do not want to focus on this aspect here. The interested reader is referred to the following papers [18], [24], [25], [27], [28], and [29].
From our point of view Human-Centred Design hast to be a prominent part of Continuous Software Engineering.

2.3 Continuous Requirements Engineering

Currently engineering-based approaches for software development are mainly applied to large enterprises that have relatively long change cycles. Much higher flexibility is required if modifications are necessary more frequently. The engineering processes grow in such cases into continuous engineering that requires Continuous Requirements Engineering. This can only be successful if rigid engineering principles are combined with agility, emergence, and spontaneity to support sustainability and viability of the systems under development.

Innovative enterprises need new approaches, methods, and tools to be capable to embrace the growing variety of opportunities and challenges offered by fast changing and hardly predictable environment. For this type of companies, Continuous Requirements Engineering seems to be very supportive. However, it has to be integrated with management and design approaches.

It is common ground that wrong requirements cause a lot of problems. Many projects totally fail because of wrong specified requirements. Others waste a lot of money because the correction of resulted errors in the implementation is very time consuming and labor intensive.

Therefore, new ideas in identifying continuously the correct requirements are very important.

This was the reason for organizing workshops with this topic. The first one was organized in 2015 at REFSQ in Essen [22] and the second one in Gothenburg [4]. While discussing different aspects of requirements participants of the workshop were able to

agree that continuous elicitation and continuous specification of requirements is extremely important for successful software projects.

Leah Goldin et al. [20] discuss the question whether in the development of large scale systems the institutionalized, proactive requirements reuse pays off. In their case study, they found out that at least for the studied project it paid off to meet the moving target of requirements based on existing specifications. This might be one way to reduce the time to market. However, there are still a lot of other aspects to consider.

By Qureshi et al. [26] a framework CARE (Continuous Adaptive Requirements Engineering) was provided. The framework is as well goal- as user-oriented. It is designed for self-adapting systems. The authors distinguish requirements engineering during design-time and run-time.

Indeed, monitoring of running systems might be very useful. However, this is not specific to adaptive systems. It seems to be useful for any kind of application. In this way, Continuous Requirements Engineering has to find its way to general software systems. Its important role should be reflected in the model of CSE.

2.4 Continuous Business Process Modeling

The term Continuous Business Process Modeling is not used so very often. However, the idea of Continuous Business Process Improvement (CPI) has been discussed for several years. The book from Harrington [9] might be an example for that. It is from 1991

The term CPI is perfectly defined by Professional Business Solutions Inc. (PBSI): "To maintain their competitive advantage organizations must streamline their operations and processes. Continuous Process Improvement (CPI) is a strategic approach for developing a culture of Continuous Improvement in the areas of reliability, process cycle times, costs in terms of less total resource consumption, quality, and productivity. Deployed effectively, it increases quality and productivity, while reducing waste and cycle time. Since many business processes rely on information and participation from more than one department and even different organizations, CPI is designed to facilitate these processes by integrating the various components into one streamlined system that runs smoothly and efficiently on a partially or completely automated flow of steps." [3]

Milewski et al. [23] discuss the technological process innovation from a life cycle perspective. They provide a framework based on case studies.

It seems to be common sense that Continuous Process Innovation is an integrated part of companies. Bergener et al. [3] claim: "Business Process Management (BPM) has evolved as an integrated management discipline that aims to enable organizations to continuously innovate and improve their operations."

It is widely accepted that models of organizations in connection with business process models are very important for BPM and the corresponding IT support. According to Fleischmann et al. [13] models play an increasing role in adaptive process environments.

---

[3] http://www.epbsi.com/capabilities/bmp.html

Fitzgerald and Stol [11] argue that “Enterprise Agile and Beyond Budgeting concepts have emerged as recognition that benefits of agile software development will be sub-optimal if not complemented with by an agile approach in related organizational functions such as finance and HR”. They additionally argue: “that the link between business strategy and software development ought to be continuously assessed and improved”.

Additionally to those aspects, the idea of modeling seems to be attractive. From our point of view, Continuous Business Process Modeling seems to be an important activity as well in software engineering as in business administration. It should be integrated into the general approach of Continuous Software Engineering of business applications.

Continuous Business-Process Modeling should be part of the Business Strategy. This idea is supported by Bukša et al. [7]. Those authors presented a method for integrated semi-automated business process and regulations compliance management. They especially referred to the changing business process models: “However, there is a gap between continuously changing business process models that are maintained in a specific set of tools, and continuously changing regulatory requirements that usually are maintained outside organizations. There are tools that provide support for compliance management by means of Business Rules Engine, however, in most cases business rules must be entered manually and there is no live linkage with external legislative and regulative sources”. Additionally, they asked for specific tool support related to Continuous Business Process Modeling.

An approach that allows the rapid execution of business-process specification was provided by Fleischmann et al. [14]. It is called Subject-oriented Business Process Modeling and uses the language S-BPM [13]. Processes are modelled from the perspective of subjects that communicate via messages. These subjects are Most of the time these subjects are humans. This approach perfectly fits to both ideas of Continuous Business-Process Modeling and Human-Centered Design. Users can adapt the models that describe their activities during runtime.

2.5 Discussion

Within the previous paragraphs, the idea of Continuous Software Engineering, Continuous Human-Centred Design, Continuous Requirements Engineering, and Continuous Business Process Modeling were recapitulated. Originally, there was the suggestion in [18] to consider Continuous Requirements Engineering as part of the Business Strategy. However, this seems not to be true. Requirements engineering is part of development and operations as well. Therefore, it is suggested that it plays the same role as Continuous Innovation. Therefore, it is visualized in **Fig. 5** accordingly.

Additionally, in [18] it was also mentioned the importance of Business Process Modeling for software development.

Fleischmann, Schmidt, and Stary [14] even call their business process model as requirements specification and executable software design.

Obviously, Continuous Business Process Modeling has to be combined with Continuous Requirements Engineering to become successful.

The usability of software systems will be very important in the future as well. Humans have to be supported in their daily work as good as possible. Good technical solutions and good business processes are not enough. They have to be supported by good user interfaces. Therefore, following extensions visualized in **Fig. 5** are suggested for the concept of Continuous Software Engineering.

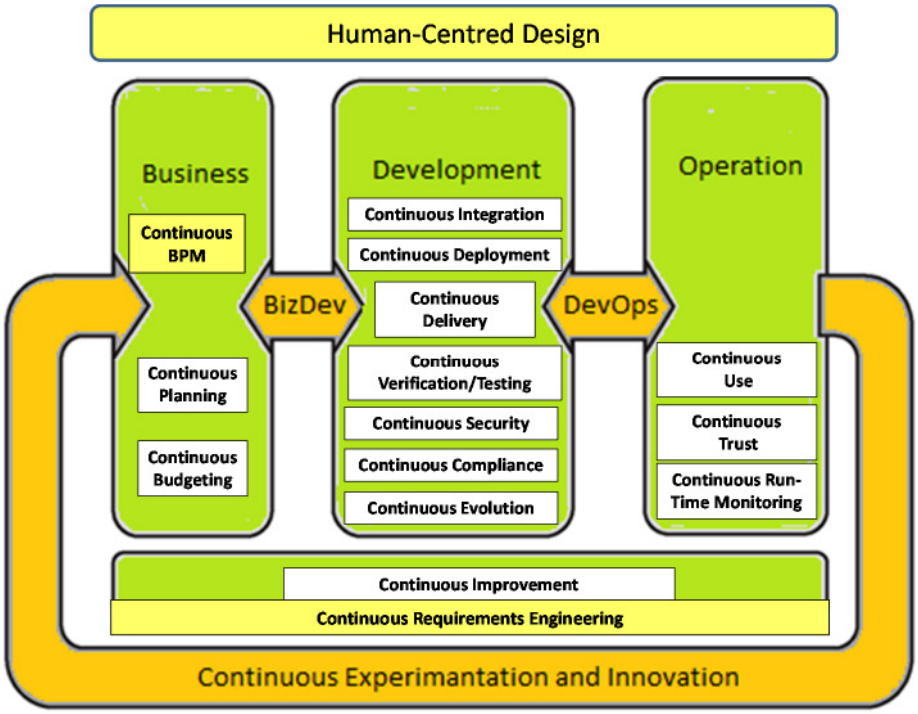

Fig. 5. Extended view on activities of Continuous Software Engineering.

Continuous Software Engineering might make sense without Continuous Requirements Engineering when no further development of functionalities is planned and only bugs have to be eliminated in a software system. However, this seems to be a very rare situation. Changing environments in reality ask for continuous elicitation of new or changing requirements.

On the other hand, Continuous Requirements Engineering does not make sense if no conclusions are reached. The software has to be adapted to the new needs as soon as possible. Therefore, Continuous Delivery is needed. This can be reached best with Continuous Software Engineering.

3 Summary

Based on references and own experiences some ideas regarding shaping the concept of Continuous Software Engineering were presented. It was suggested to integrate Continuous Human-Centred Design, Continuous Business Process Modeling, and Continuous Requirements Engineering into the approach of Fitzgerald and Stol [12]. It was argued, that CRE is not only a part of CSE but that both concepts influence each other. Additionally, none of both concepts makes really sense without the other one. It would be nice, if participants of the CRE'17 workshop could discuss a model for integrating BizDev and DevOps, A model for BizDevOps would be nice to have.

4 References

1. Agile Manifesto, http://agilemanifesto.org/, last visited January 11, 2017.
2. Baresi, L., and Ghezzi, C. C.: The disappearing boundary between development-time and run-time. In Future of Software Engineering Research, 2010.
3. Bergener, K., vom Brocke, J., Hofmann, S., Stein, A., vom Brocke, C.: On the importance of agile communication skills in BPM education: Design principles for international seminars. KM & E-Learning: 4(4), 415-434, 2012.
4. Bjarnason, E., et al.: Joint Proceedings of REFSQ-2016 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2016), Gothenburg, Sweden, March 14, 2016. CEUR Workshop Proceedings 1564, CEUR-WS.org 2016
5. Bogsnes, B.: Implementing Beyond Budgeting: Unlocking the Performance Potential, Wiley 2008.
6. Bosch, J. (Ed.): Continuous Software Engineering, Springer Verlag, 2014.
7. Claes, J. , Vanderfeesten, I. Reijers, H. A., Pinggera, J., Weidlich, M., Zugal, St., Fahland, D., Weber, B., Mendling, J., and Poels, G.: Tying process model quality to the modeling process: the impact of structuring, movement, and speed, in Business Process Management, Springer, Berlin, pp. 33-48, 2012.
8. Bukša, I., Darģis, M., and Penicina, L.: Towards a Method for Integrated Semi -Automated Business Process and Regulations Compliance Management for Continuous Requirements Engineering, in 22 p. 25 – 33.
9. Harrington, H. J.: Business Process Improvement: The Breakthrough Strategy for Total Quality, Productivity, and Competitiveness, McGraw Hill Inc. 1991.
10. Fichtenbauer, Ch., and Fleischmann, A.: Three Dimensions of Process Models Regarding their Execution. In Proceedings of the 8th International Conference on Subject-oriented Business Process Management (S-BPM '16). ACM, New York, NY, USA, Article 7, 8 pages. DOI=http://dx.doi.org/10.1145/2882879.2882892
11. Fitzgerald, B. and Stol, K.-J.: Continuous software engineering and beyond: trends and challenges. In Proc. 1st International Workshop on Rapid Continuous Software Engineering – RcoSE 2014, ACM, New York, NY, USA, pp. 1-9.
12. Fitzgerald, B. and Stol, K.-J.: Continuous software engineering: A roadmap and agenda, Journal of Systems and Software, Volume 25, July 2015, Pages 1–14.
13. Fleischmann, A., Schmidt, W., and Stary, C.: Open S-BPM= open innovation. In S-BPM ONE-Running Processes, pp. 295-320. Springer Berlin Heidelberg, 2013.

14. Fleischmann, A., Schmidt, W., and Stary, C.: Requirements Specification as Executable Software Design – A Behavior Perspective, in 22 p. 9-18, 2015.
15. Forbrig, P.: Generic Components for BPMN Specifications Perspectives in Business Informatics Research - 13th International Conference, BIR 2014, Lund, Sweden, September 22-24, 2014. Proceedings , pp. 202—216, 2014.
16. Forbrig, P.: Reuse of models in S-BPM process specifications, Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, S-BPM ONE 2015, Kiel, Germany, April 23-24, 2015 , pp. 6-16. 2015.
17. Forbrig, P. and Herczeg M.: Managing the Agile Process of Human-Centred Design and Software Development, In: Beckmann Ch. and Gross T. (Eds) INTERACT 2015 Adjunct Proceedings, pp. 223 -232, 2015.
18. Forbrig, P.: Continuous Software Engineering with Special Emphasis on Continuous Business-Process Modeling and Human-Centered Design, In Proc. S-BPM ONE 2016.
19. Forbrig, P.: When Do Projects End? - The Role of Continuous Software Engineering. BIR 2016: 107-121.
20. Goldin, L.,.and Berry, D. M.: Reuse of requirements reduced time to market at one industrial shop: a case study, Requirements Engineering, Springer, vol. 20, Issue 1, pp. 23-44, 2015.
21. Lichter, H.; Brügge, B.; and Riehle, D.: Workshop on Continuous Software Engineering, http://ceur-ws.org/Vol-1559/paper15.pdf
22. Matulevičius, R. et al. (Eds.): REFSQ Workshop proceedings, http://ceur-ws.org/Vol-1342/, 2015.
23. Milewski, S. K., Kiran Jude Fernandes, K.J., and Matthew Paul Mount, M. P.: Exploring technological process innovation from a lifecycle perspective, International Journal of Operations & Production Management, Vol. 35 Iss: 9, pp.1312 - 1331, 2015.
24. Paelke, V. and Nebe, K.: Integrating Agile Methods for Mixed Reality Design Space Exploration. In Proceedings of the 7th ACM conference on Designing interactive systems (DIS '08). ACM, New York, NY, USA, 240-249.
25. Paul, M. Systemgestützte Integration des Usability-Engineerings in den Software-Entwicklungsprozess, PhD Thesis, University of Lübeck, 2015.
26. Qureshi, N. A., Perini, A., Ernst, N.A., and Mylopoulos, J: Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems, In First International Workshop on RE @ Runtime at 18th IEEE International Requirements Engineering Conference (RE '10), pp.9-16 ,Sydney, September 2010.
27. Salah, D., Paige, R. and Cairns, P.: A Practitioner Perspective on Integrating Agile and User Centred Design, Proceedings of the 28th International BCS Human Computer Interaction Conference (HCI 2014), pp. 100–109, 2014.
28. Singh, M.: U-SCRUM: An agile methodology for promoting usability, Integrating usability engineering and agile software development: A literature review. In Proc. AGILE 2009, IEEE Press, pp. 555-560, 2009.
29. Sy, D.: Adapting usability investigations for agile user-centered design. J. Usability Stud. 2 (3), pp. 112–132, 2007.
30. Weber, H., and Mueller, H. (Eds.): Continuous Engineering for Industrial Scale Software Systems, Dagstuhl Seminar 98092, 1998, http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=98092