Jul 1st, 12:00 AM

Agilists and the Art of Integrated Assessment Tool Development

Rob Knapen

P. J. F. M. Verweij

Sander Janssen

International Environmental Modelling and Software Society (iEMSs)
2010 International Congress on Environmental Modelling and Software
Modelling for Environment's Sake, Fifth Biennial Meeting, Ottawa, Canada
David A. Swayne, Wanhong Yang, A.A. Voinov, A. Rizzoli, T. Filatova (Eds.)
http://www.iemss.org/iemss2010/index.php?n=Main.Proceedings

Agilists and the Art of Integrated Assessment Tool Development

Rob Knapen[1], Peter Verweij[1], Sander Janssen[1]
[1]Alterra, Wageningen University and Research Centre, The Netherlands
Email: rob.knapen@wur.nl

Abstract: Software engineering for Integrated Assessment needs to address the fact that systems and models developed are increasingly used in participatory settings. Applying old principles of inside-out design and development using waterfall model based processes is no longer sufficient. New software engineering insights based on interaction design and iterative, agile processes for the development help in building systems from a more outside-in perspective. Based on two case studies from large European projects on integrated assessment this switch from applying old to applying new principles will be described, and its effects discussed. Elements were taken from interaction design: personas, story boards, mock-ups, focus groups and focus tasks. These were mixed with an iterative and incremental development process, using agile elements such as: daily stand-ups, user stories, planning games, Test Driven Development and continuous integration. The resulting process was used for the development of the two systems (SIAT and SEAMLESS-IF) for integrated assessment, giving them a more user-oriented focus than what would have resulted from following the old principles.

Keywords: agile software development, interaction design, integrated assessment

1. INTRODUCTION

Policy makers are confronted with complex, socially relevant problems. When handling these problems, there is a general tendency to increase participation of citizens and other stakeholders. In this way policy makers want to improve the quality of their policies and obtain a broader support for, and understanding of the proposed policy measures. In this new governance concept, the policy making process is the product of complex interactions between governmental and non-governmental organizations, each seeking to influence the collectively binding decisions that have consequences for their interest. Policymaking is more and more a process of cooperation and participation in which the policy maker becomes a facilitator of the process (Wien, et al. 2010).

1.1. Integrated Assessment

To account for this new governance, policy measures need to be assessed in an integrated context. Such an Integrated Assessment (IA) can be defined as an interdisciplinary process of combining, interpreting and communicating knowledge from diverse scientific disciplines in such a way that the whole cause-effect chain of a problem can be evaluated from a synoptic perspective with two characteristics: (i) it should have added value compared to a single disciplinary assessment; and (ii) it should provide useful information to policy or decision makers. Thus IA is an iterative participatory process that links knowledge (science) and action (policy) regarding complex global change issues such as acidification and climate change (Van der Sluijs 2002). This puts it very close to the upcoming fields of scenario analysis and sustainability science (Swart, Raskin and Robinson 2004).

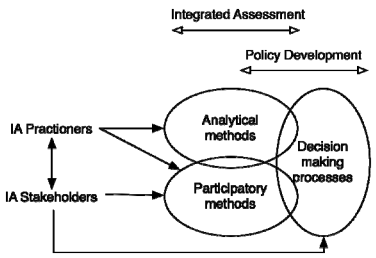

Figure 1: Integrated Assessment and policy development (Van der Sluijs 2002)

It is important to note that Integrated Assessment is understood as a process and not as a method. Where it once used to revolve around scenario development and Integrated Assessment Models (IAM) for the technical analytical part, it is now recognized to be a broader participatory assessment process with IAMs being one of the usable tools. IAMs are computer simulation models in which knowledge from many different disciplines is combined in an analytical computational framework to analyze the problem at hand in an integrated fashion. They are used for several purposes, such as scenario analysis and the (ex-ante) evaluation of the environmental, economic and social consequences of different policy strategies. The role of IAMs in the IA process has been closely linked to the policy development process. In the agenda-setting and pre-negotiation phases of this process they are helpful to bring together experts from a multitude of scientific disciplines to arrive at an integrated view on the problem and possible solutions. In the negotiations and post-agreement phases the build up of know-how and the trust required to develop effective policies becomes more important and implies a growing need for the integration of social science research, and in particular of participatory techniques. This is required to provide knowledge about stakeholders and their ways of opinion forming, and also to provide opportunities for including the knowledge of stakeholders and their judgments about controversial issues in policy making (Van der Sluijs 2002). This is the field of Participatory Integrated Assessment (PIA) were computer models should serve as catalysts supporting the discussion, providing new information and insights, but not dominate the discussion, because the main intention in participatory IA is not to test software systems, but to launch a learning process and to elicit views and values of the participants (Dahinden, et al. 2000).

1.2. Sustainability Software Development

With the growing interest in sustainability science, scenario analysis and the development of participatory integrated assessment, comes the need to look at the software tools available for them and the process used to build them. As a result from the shift from IAM to PIA software tools are being given higher capability for analysis targeted at non-domain experts, which also makes them better suited for use in PIA workshops. However, as multi-disciplinary and crosscutting this science is, the field of Computer Science and the Software Engineering discipline are either not included at all or regarded as utilitarian or operational. The expressed need for participatory approaches to get stakeholders involved is rarely extended to the software development process used to build the tools. To improve this e.g. a Computer Science department should make sure to be included already in the project proposal writing stage, should better promote its work and development processes, and should separate clearly contributing to projects from the general IT business (like office automation and server maintenance).

The software development process of Integrated Assessment (IA) tools will in particular be the focus of this publication. Software systems that support the end-user in combining and interpreting knowledge from diverse scientific disciplines in order to improve understanding of natural and socio-economic systems have over the last decades evolved, starting from Geographical Information Systems (GIS) into Environmental Information Systems (EIS) and Environmental Decision Support Systems (EDSS). These systems however have mostly an *inside-out* focus (GIS Software Engineers built them for GIS Professionals) and are constructed using traditional software development methodologies (e.g. the waterfall approach), if any at all (Argent 2004). In inside-out developed systems typically the scientists or model experts develop her or his models and tools, while, in contrast, in *outside-in* software development the non-domain experts decides on the software requirements.

Considering the four typical building blocks found in these systems: Models, GIS, DSS and Data Management (Denzer 2005), the focus currently is still basically on getting the individual blocks to work properly and to get them integrated. Available valuable project resources have to be spent on achieving sub-optimal in-project integrations, resulting in solutions that are difficult to re-use.

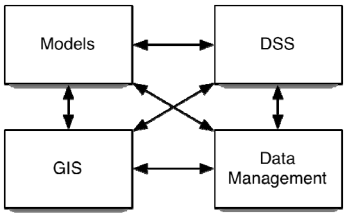

Figure 2: The four EDSS core building blocks (Denzer 2005)

Lately a lot of potentially useful technologies are falling into place that could simplify the construction of the system core (the four major building blocks) and allow more time for a focus shift to outside-in oriented development. Considering the EIDSS building blocks for example technologies and standards like from the Open Geospatial Consortium (OGC) (http://www.opengeospatial.org/) for geographical data exchange, OpenMI (http://www.openmi.org/) for model linking, and Ontology and Web Services for componentized solutions with generic access to semantically rich data, could drive new types of user interfaces to be built. Steps like those outlined by (Jakeman, Letcher and Norton 2006) should help in making models more accessible. When the back-end of the system is easier to construct more resources are available for building the user interface(s) that actually work for the end-user and are more than a technical front-end that can only be operated by scientists and developers.

The next step could be that these information systems turn into real tools for Integrated Assessment and Sustainability Science, which can be operated (to a limited extent) by teams of end-users and help in participatory settings.

2. SOFTWARE DEVELOPMENT PROCESS

Software engineering is the application of a systematic, disciplined and quantifiable approach to the development, operation and maintenance of software, and the study of these approaches. Or in other words: the application of engineering to software. The term *software engineering* first appeared in the 1968 NATO Software Engineering Conference and was meant to provoke thought regarding the "software crisis" (the difficulty of writing correct, understandable and verifiable computer programs) at that time. Since then it has continued as a profession and field of study dedicated to designing, implementing, and improving software that is of higher quality, more affordable, maintainable and quicker to built. Since the field is still relatively young compared to other fields of engineering, there is still much debate around what software engineering actually is, and whether it conforms to the classical definition of engineering or not.

A software development process is a structure imposed on the development of a software product. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. The overall goal is to find repeatable, predictable processes that improve productivity and quality. Project management techniques are typically part of the process. Of the existing models each one has its advantages and disadvantages, and usually it is up to the development team to adopt the most appropriate one for the project. Sometimes a combination of elements from several models may be the best suited. Among the most common models are:

- *Waterfall*: In the strict approach after each development phase (requirements analysis, conceptual modelling, functional design, technical design, implementation, testing) has finished it is signed-off and the work proceeds with the next phase. Reviews may occur before transitioning to the next phase, which allows for the possibility of changes (maybe through a formal change control process). However, going back phases typically is very expensive and therefore discouraged.
- *Spiral*: A risk-oriented development model that breaks a software project up into mini projects. Each mini project addresses one or more major risks, which can refer to poorly understood requirements, poorly understood architecture, potential performance problems, and so on. After the major risks have all been addressed the spiral model terminates as a waterfall lifecycle model would (McConnell 1996).
- *Iterative and incremental*: Types of development models that prescribe the construction of initially small but ever larger portions of a software project to help all those involved to uncover important issues early before problems or faulty assumptions can lead to disaster. A functional version of the software is built at the start of the project but with a very limited feature set. New functionality is then added in an incremental way and in several development iterations. The early versions of the software typically help in communicating with the customer and getting clearer requirements.
- *Agile*: Models based on iterative development but advocating a lighter and more people-centric viewpoint than traditional approaches. Customer feedback driven by regular tests and releases of the evolving software, rather than planning, is used as the primary control mechanism. Examples of agile lifecycle models are: Extreme Programming (XP) (Beck 2000), Rational Unified Process (RUP), and Scrum. The term agile software development was coined in the year 2001 when the *Agile Manifesto* (www.agilemanifesto.org) was formulated.

Agile software development refers to the last type of models. Using iterative development requirements and solutions evolve through collaboration between self-organizing and cross-functional teams. In contrast to the waterfall method, which is very process-oriented, agile methods focus on people (participation, collaboration) and solutions (working software). A set of engineering best practices (e.g. unit testing, test driven development, continuous integration) is used to allow for rapid delivery of high-quality software, and a business approach (planning game, story cards) that aligns development with customer needs and organizational goals.

The work of Patton (2002) and Haikara (2007), for example, shows ways of incorporating elements of Usage-Centered (interaction) Design (Constantine and Lockwood 2002) into the agile software development process. Instead of having to do it as a big upfront design task, which would conflict with the agile approach.

3. CASE STUDIES

In this section our experiences will be presented from applying elements of agile software development and of Usage Centered Design to two large-scale European research projects on IA. The first is the SENSOR project (www.sensor-ip.eu) and the second is the SEAMLESS project (www.seamless-ip.org). While these projects where large (over 30 contributing partner organizations each), ran for four years and produced lots of scientific output we will here only focus on the developed software tools.

3.1. Developed Systems

SENSOR developed a web application called SIAT (Sustainability Impact Assessment Tool, http://siat.cgi-systems.nl/SiatGUI/) for estimating the possible consequences of different policy assumptions on multi-functional land use and its sustainability within different images of the future (Verweij, et al. In Press). A model chain is used to derive from the drivers the impacts expressed as land-use changes. Combined with constant-factor maps (containing parameters which are expected to be constant throughout the modelling) they result in social, economic and environmental indicators. Land use functions are then used to assess regional sustainability limits for these indicators (Pérez-Soba, et al. 2008). As the modelling chain used to calculate possible land-use changes is complex and time consuming, meta-models were derived from its results (Sieber, et al. 2008) for use in SIAT.

SEAMLESS developed an integrated framework (SEAMLESS-IF, http://www.seamless-if.org:8080/gromit/wallace/index.html) for ex-ante integrated assessment of agro-environmental policies and agro-technological innovations in the European Union (EU). This framework provides analytical capacity to assess sustainability of agricultural systems in the EU and contributions of the EU's agricultural systems to sustainable development at large, including some effects on the entire production chain and other land uses (e.g. nature). It is designed to compare policy alternatives, in interaction with agro-technological options, for a defined time-horizon. These time-horizons are defined by the policy questions at hand (Van Ittersum, et al. 2008). The framework provides a software infrastructure for component-based modelling and allows the re-use of model components for scale and disciplinary integration and their inclusion in model chains for the computation of indicators. A web application targeted at two user groups, policy experts and integrative modellers, provides access to the system.

3.2. Development Process

Both projects started using a waterfall model like development approach. This was to be expected since the original Description of Work (DOW) of the projects had to comply with the strict EU guidelines that expected a standard division of the work in clear project milestones (18, 24, 36 months etc.) and steps understood by project managers like "requirements gathering", "conceptual domain modelling", and so on. About the first year typically was used to achieve a shared vision between all organizations about each partner's scientific contribution. The forced development of a first software prototype based on a very limited set of clear requirements was needed to provoke adequate feedback on the conceptual domain model, the expected feature set, and in general to give direction to the software development.

3.3. Outside-in Development

Any software system and its architecture can be viewed from multiple perspectives. This in fact is needed for its design and implementation, where owner, architect and builder all look at the same system from a different point of view (Zachman 1987). Even within each of those points of view, e.g.

that of the architect, there is more then a single perspective to consider. According to Perry and Wolf (1992) the distinction between processing, data and connections views is rather important. Naturally all these views are intertwined. However looking at the design and construction of a software system for Integrated Assessment two broad overall approaches can be distinguished, the inside-out and the outside-in way of thinking (Kessler and Sweitzer 2008). Like in our cases the systems tend to be studied and built as part of (large) research projects where the chosen approach is mostly inside-out, which emphasizes the construction of the building blocks and the linking of them, and perhaps the development or re-use of a framework to support it. In the end a mostly rather technical looking (graphical) user interface is added to it (Knapen, Verweij and Wien 2009). To no surprise models and systems constructed this way are not very suited for use in PIA, e.g. for focus group discussions (Dahinden, et al. 2000). For that it is very important to keep the user in mind when developing, which means taking a more outside-in perspective during the design and implementation of the software. Tell-tailing would be a greater focus on the graphical user interface (its design, content and functionality) of the system, probably being built first at least as mock-ups, and its usefulness studied with the intended users (or stakeholders). Many of them likely only care to a limited level about the intricate internal machinery of connected interdisciplinary models and other components that make up the system, as long as it is easy to operate and delivers useful results at the right moment, timely for the policy decision taking process.

3.4. Personas

Both SIAT and SEAMLESS-IF were designed and developed as a response to a research call from the Directorate General (DG) Research of the European Commission (EC). Frequent interactions with DG Research were needed to shape the requirements. Interactions with various other DGs of the EC were also initiated to engage with foreseen users of the systems, in particular DG Agriculture, DG Environment and DG Economics and Finances. SEAMLESS organized User Forum meetings twice a year and smaller meetings in between upon request. In general it was difficult to get in touch with real users of the system and personas (Haikara 2007) were used instead. Initially SEAMLESS distinguished six classes of users, i.e. coders, linkers, runners, providers, viewers and players, with distinct requirements. While these classes determined requirements for the development of the framework, for the development of the graphical user interfaces they have been merged into three groups of users, i.e. the policy expert, the integrative modeler and the domain specific modeler. These were the focal roles (Patton 2002). The policy experts are engaged in the pre-modeling and post-modeling phases to define scenarios and indicators and are mainly interested in the results of the IA. The integrative modelers set-up integrated assessments with SEAMLESS-IF, implement and run model chains, and work in tight interaction with the policy expert and other integrative modelers. The domain specific modelers are experts of the individual model components and their code or data, and eventually were considered to not need a specific user interface for the system but rather be allowed to work with the tools they already knew. Based on the remaining two focal roles in the end two graphical user interfaces were made available as part of the web application. A system of user accounts, roles and permissions was implemented (Poch, et al. 2003) to decide which functionality to show and make available to the person at the computer, and to some extend also the look and feel of the graphical user interface.

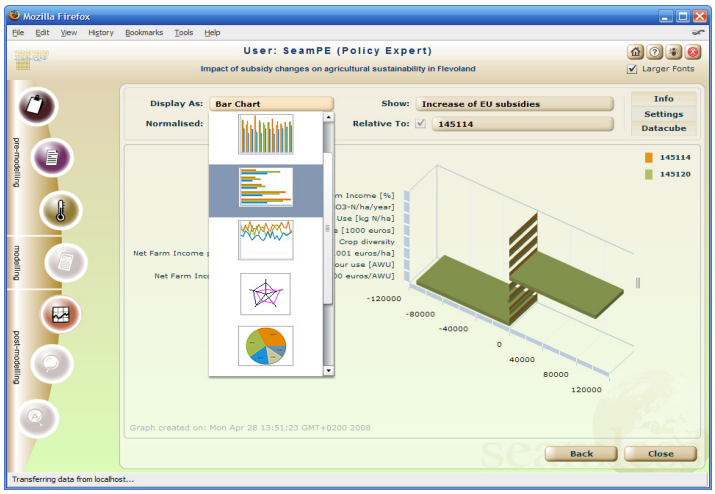

Figure 3: SEAMLESS Policy expert user interface

An important aspect of this system was the ability for integrative modelers to create visualizations of calculated data (scenario outcomes) as maps, graphs, spider diagrams, etc. Policy experts in contrast were only allowed to view these prepared visualizations. This way the integrated modeler could make sure that the policy expert was looking at a well-balanced set of indicators and results, and not (only) focus in on a specific item of interest. An indicator framework (Alkan Olsson, et al. 2009) was implemented into the system to help the user to make a balanced selection.

3.5. Agile Development

Recognizing the vagueness of the requirements and the unpredictable outcome of a large-scale exploratory scientific undertaking with changing and evolving insights both projects abandoned the waterfall model for development and switched to an agile model. Metaphorically this can be seen as switching from a building construction approach to a gardening approach (Hunt and Thomas 1999). When working agile options can be (relatively quickly) tried and cultivated when they look promising, or discarded when they do not work. This more organic approach embraces change (Beck 2000) where in construction change is usually expensive and to be avoided after the initial design phase. Development iterations were chosen to be about 4 weeks, which is a bit longer than typically recommended (2-3 weeks). However this was needed to provide enough time to get feedback from the large group of international research partners and end-user representatives.

Another aspect of agile development that was adopted was the use of user stories (Cohn 2004), comparable to task cases described by Patton (2002). Simplified, a user story is a conversation description by a user (stakeholder) of a task that needs to be fulfilled by (or rather with) the system. Besides documenting the expected feature set, they are used for priority planning, implementation scheduling and as a reminder for developers to talk to the users about the stories topic when it is about to be implemented. User stories are written on cards, or in our case in very simple PowerPoint slides. In a planning game the cards are sorted based on priority by the users and the effort needed from the developers. This is done before the iteration starts. A set of stories is then selected from the top that matches the available resources (time, capacity). Unit testing, Test Driven Development and continuous integration were used to assure a level of software quality during the development. Sadly enough this rarely gets enough attention in a science project.

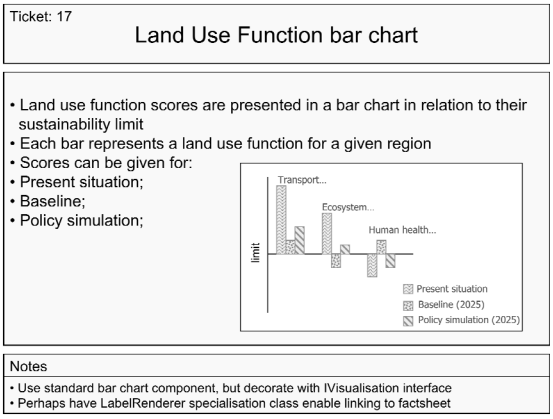

Figure 4: User story slide from the SENSOR project

Online tools like e.g. a WIKI, audio and video conferencing, and messaging were used on a regular (often day-by-day) bases to facilitate software development by multiple, geographically distributed partners. For example a Skype-based daily stand-up was introduced. Besides this development workshops ("hackathlons") were organized when though useful or required. End users were invited to join in at the stand-up meetings, allowed to listen only during the first 15 minutes when the developers discussed what they had done, what they planned to do that day and what major hurdles they faced. After that there was room (each day) for end users to ask questions for clarification, but not to change the order of the user stories. This only could be done after the iteration (of 4 weeks). At or near the end of the iteration a version of the system was deployed as web application and made available to a larger group of end users for providing feedback. Resulting comments were discussed and included in the stack of user stories.

3.6. Interaction Design

SENSOR took the course of actually using an interaction designer upfront (although she was not allowed to spend the half year of work she requested on it), while in SEAMLESS this work was done by properly skilled software developers. A graphical designer was contracted for both projects. Also both projects used storyboards, low-fidelity (PowerPoint slides) and high fidelity (Adobe Flash) mock-ups to sketch and test graphical user interface designs before actually implementing them. These proved to be very helpful. Work on storyboards and low-fidelity mock-ups was done during workgroup sessions at the regular project conference meetings, and open to everyone interested. Typically this meant only a small group (15 or less) of participants, but they were keen to contribute and also actively engaged in follow-ups. From these meetings usually a stack of flip-over posters, whiteboard pictures, post-its and other materials were taken home by the developers to work with. This material was used to create high fidelity mock-ups that were discussed by audio/video conferencing and e-mail. When agreed upon, the design could then be implemented for the next release of the software. These steps were made part of the user stories and the development cycles.

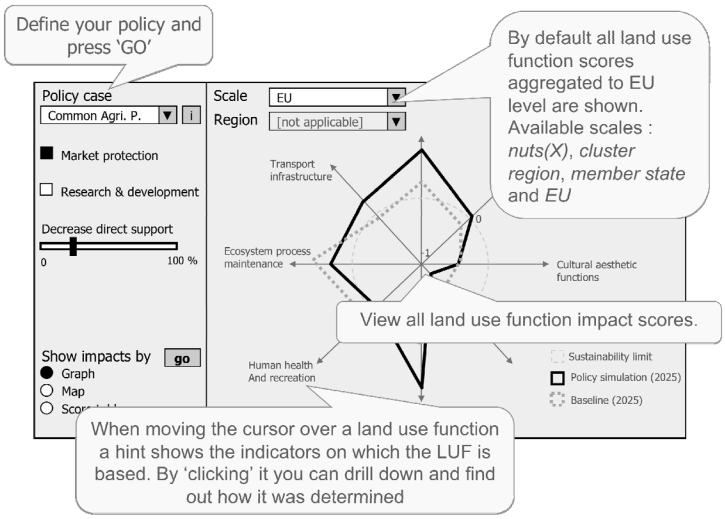

Figure 5: SIAT low-fidelity mock-up

4. DISCUSSION AND CONCLUSION

Driven by the emerging need to make systems for Integrated Assessment better suited for use in participatory settings and a desire to use good software engineering principles and actually deliver systems that can be used by end-users and re-used by developers the work done in the case studies described illustrates how elements of interaction design and agile software development methodologies can be used in IA research projects. Both products, SIAT and SEAMLESS-IF live on after the projects have ended. Developing them in close interaction with end-users and stakeholders definitely helped in creating usable systems with possibilities and limitations that were better understood. The Test Driven Development (TDD) approach resulted in more maintainable software, but unfortunately still with many parts marked as To Do's or shortcuts. Apparently this is more acceptable to scientists.

Agile development and matching project management strategies should get a better place in scientific research proposals. This would avoid the costly and time consuming transition from waterfall to iterative approaches, including communication needed between partners to get a shared view. As observed, implementing a first prototype of the system is usually needed to get relevant feedback from end-users. Advancing in short iterations with visible and testable deliveries of the software, and quickly acting on comments, helps keeping users motivated and involved in the development cycle until the final product. However as long as the real end-users are hard to find and get involved into the agile development process, building a good user interface will remain a challenge.

With the progress of technology and available open standards there is an opportunity to take a more outside-in focus that should help in making IA systems (including EIS and EDSS) more accessible and usable as tools for sustainability science.

REFERENCES

Alkan Olsson, J.A., et al. “A goal oriented indicator framework to support integrated assessment of new policies for agri-environmental systems.” *Environmental Science and Policy* (Elsevier) 12, no. 5 (2009): 562-572.

Argent, Robert M. "An overview of model integration for environmental applications - components, frameworks and semantics." *Environmental Modelling & Software* (Elsevier), no. 19 (2004): 219-234.

Beck, K. *Extreme programming explained: embrace change.* Addison-Wesley Professional, 2000.

Cohn, M. *User Stories Applied: for agile software development.* Addison-Wesley, 2004.

Constantine, L.L., and L.A.D. Lockwood. “Usage-centered engineering for web applications.” *IEEE Software* (IEEE Computer Society Press) 19, no. 2 (2002): 42-50.

Dahinden, U., C. Querol, J. Jäger, and M. Nilsson. “Exploring the use of computer models in participatory integrated assessment - experiences and recommendations for further steps.” *Integrated Assessment* (Kluwer Academic Publishers) 1 (2000): 253-266.

Denzer, R. “Generic integration of environmental decision support systems - state-of-the-art.” *Environmental Modelling & Software* (Elsevier), no. 20 (2005): 1217-1223.

French, S., and J. Geldermann. “The varied contexts of environmental decision problems and their implications for decision support.” *Environmental Science & Policy* (Elsevier), no. 8 (2005): 378-391.

Haikara, J. “Usability in agile software development: extending the interaction design process with personas approach.” Edited by G. et al. Congas. *XP 2007.* Berlin: Springer-Verlag, 2007. 153-156.

Hunt, A., and D. Thomas. *The pragmetic programmer: from journeyman to master.* Addison-Wesley, 1999.

Jakeman, A.J., R.A. Letcher, and J.P. Norton. “Ten iterative steps in development and evaluation of environmental models.” *Environmental Modelling & Software* (Elsevier), no. 21 (2006): 602-614.

Kessler, C., and J. Sweitzer. *Outside-in Software Development; a practical approach to building successful stakeholder-based products.* IBM Press, 2008.

Knapen, M.J.R., P.J.F.M. Verweij, and J.J.F. Wien. “Towards outside-in development of integrated assessment systems.” *Proceedings of the 7th EFITA conference.* Wageningen: Academic Publishers, 2009.

McConnell, S. *Rapid development: taming wild software schedules.* Microsoft Press, 1996.

Pahl-Wostl, C. “Participative and stakeholder-based policy design, evaluation and modeling processes.” *Integrated Assessment* (Swets & Zeitlinger) 3, no. 1 (2002): 3-14.

Patton, J. “Hitting the target: Adding interaction design to agile software development.” *Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA).* 2002.

Pérez-Soba, M., et al. “Land use functions - a multifunctionality approach to assess the impact of land use changes on land use sustainability.” In *Sustainability impact assessment of land use changes*, edited by K. Helming, P. Tabbush and M. Pérez-Soba, 375-404. Springer, 2008.

Perry, D.E., and A.L. Wolf. “Foundations for the study of software architecture.” *ACM SIGSOFT Engineering Notes*, no. 17 (October 1992): 40-52.

Poch, M., J. Comas, I. Rodríguez-Roda, M. Sànchez-Marrè, and U. Cortés. “Designing and building real environmental decision support systems.” *Environmental Modelling & Software* (Elsevier), 2003.

Schwaber, and Beedle. *Agile Software Development with Scrum.* 2002.

Sieber, S., et al. “Transfer into decision support: the sustainability impact assessment tool (SIAT).” In *Sustainability impact assessment of land use changes*, edited by K. Helming, P. Tabbush and M. Pérez-Soba, 107-128. Elsevier, 2008.

Swart, R.J., P. Raskin, and J. Robinson. “The problem of the future: sustainability science and scenario analysis.” *Global Environmental Change* (Elsevier), no. 14 (2004): 137-146.

Van der Sluijs, J.P. “Integrated Assessment.” *Encyclopedia of Global Environmental Change* (John Wiley & Sons, Ltd.) 4 (2002): 250-253.

Van Ittersum, M.K., et al. “Integrated assessment of agricultural systems - A component-based framework for the European Union (SEAMLESS).” *Agricultural Systems* (Elsevier), no. 96 (2008): 150-165.

Verweij, P., et al. “An IT perspective on integrated environmental modelling: the SIAT case.” *Ecological Modelling* (Elsevier), In Press.

Wien, J.J.F, et al. “A web-based software system for model integration in agro-environmental impact assessments.” In *Environmental and Agricultural Modelling: Integrated Approaches for Policy Impact Assessment*, edited by F.M Brouwer and M.K. Van Ittersum, 325. Elsevier, 2010.

Zachman. “A framework for information systems architecture.” *IBM Systems Journal* (IBM) 26, no. 3 (1987).