Available online at www.sciencedirect.com
ScienceDirect
Computers in Human Behavior xxx (2007) xxx–xxx
Computers in Human Behavior
www.elsevier.com/locate/comphumbeh

Improving the development of instructional software: Three building-block solutions to interrelate design and production

Eddy W. Boot [a,*], Jeroen J.G. van Merriënboer [b], Nicolet C.M. Theunissen [a]

[a] *TNO Defense, Security and Safety, Department of Training and Instruction, Kampweg 5, 3769 ZG Soesterberg, The Netherlands*
[b] *Open University of the Netherlands, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands*

Abstract

Currently, there is a focus on authentic tasks as the driving force for learning in integrated e-learning systems. This sets new criteria for instructional software, which should become much more flexible and allow for domain modeling and pedagogical modeling. A theoretical analysis and a survey ($n = 37$) amongst experienced developers show that current development methods are insufficient to develop such instructional software. New development methods such as "lean production" promise to satisfy the new criteria as they emphasize mass-customization by rigorously applying a pull-principle throughout the whole development process. However, a potential bottleneck is the lack of design languages to transfer the design outcomes to the production phase. Three building-block solutions are proposed to overcome this transition problem: (1) a 3D-model to support designers in stratifying, elaborating, and formalizing design documents, (2) instructional software templates to support designers in producing software themselves, and (3) an integrative approach to support designers in reusing learning objects.

Keywords: Instructional software; Development; Training; Computer-based training; e-Learning; Competency-based learning

* Corresponding author. Tel.: +31 346 356 297; fax: +31 346 353 977.
E-mail address: eddy.boot@tno.nl (E.W. Boot).

0747-5632/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.chb.2007.05.002

1. Introduction

Current educational, technical and organizational innovations are rapidly changing the nature of instructional software, and, thereby, the way it is developed. Recent theories of instruction tend to focus on authentic learning tasks that are based on real-life tasks as the driving force for learning (Merrill, 2002; Reigeluth, 1999). The general assumption is that such authentic tasks help learners to integrate the knowledge, skills, and attitudes necessary for effective task performance; give them the opportunity to learn to coordinate constituent skills that make up complex task performance; and eventually enable them to transfer what is learned to their daily life or work settings. This focus on authentic, whole tasks can be found in practical educational approaches, such as project-based education, the case method, problem-based learning, and competency-based learning; in theoretical models, such as Collins, Brown, and Newman's (1989) theory of Cognitive Apprenticeship Learning, Jonassen's (1999) theory of Constructive Learning Environments, Nelson's (1999) theory of Collaborative Problem Solving, and Schank's theory of Goal Based Scenario's (Schank, Berman, & MacPerson, 1999); and in instructional design models, such as the Four Component Instructional Design model (Van Merriënboer, 1997; Van Merriënboer & Kirschner, 2007).

In addition to educational changes, technical and organizational innovations enable the application of blended learning or integrated e-learning: The combination of face-to-face learning, distance learning, and on-the-job learning. Blended learning is supported by a balanced media-mix of traditional and advanced learning technologies such as books, e-learning, mobile learning, and simulations (Jochems, van Merriënboer, & Koper, 2003). Such integrated e-learning provides both the flexibility to enable the integration of working and learning, in terms of time and place independent learning, and adaptive learning, personalized for individual learners.

The resulting combination of pedagogical considerations (e.g., "How can authentic learning tasks be implemented in the instructional software?"), technical considerations (e.g., "Which media-mix is most optimal?"), and organizational considerations (e.g., "How can working and learning be efficiently integrated by means of instructional software?") makes the development process highly complex, requiring a structural approach towards design, production, and implementation.

In this article, we investigate if current development methods provide for such a structural approach to develop innovative instructional software. First, the new criteria that result from the recent innovations are discussed. Second, a theoretical analysis of current development methods is described. Third, the theoretical analysis is complemented by an empirical analysis, in the form of a survey study. Fourth, lean production is introduced as a new development method that promises to fit the new situation better. Fifth, the problem of lack of design languages, which hampers the implementation of lean production, is discussed. Sixth, three building-block solutions are proposed to enable the implementation of lean production models by supporting designers to (1) improve design documents, (2) use instructional software templates, and (3) reuse learning objects. Finally, conclusions will be drawn about the consequences of criteria and building blocks for future development of instructional software.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

2. New criteria for developing instructional software

The educational, technical, and organizational innovations lead to four new criteria for instructional software development. The first three criteria are related to the flexibility of development processes and products. For example, a blended learning approach requires the efficient and fast creation of multiple configurations of instructional methods and media (''packages''). Atkinson and Wilson (1969) (for more recent discussions see, Gibbons, Nelson, & Richards, 2000; Parrish, 2004) have identified three criteria for developing instructional software related to flexibility. First, *adaptivity*, which is the ability of an instructional software product to adjust itself to learner needs and the history of learner progress, preferences, and choices, provides personalized learning for individual learners. Second, *generativity*, which is the ability to assemble the instructional software product from some combination of parts and sources at the moment of delivery, frees the designer from having to create an infinite variety of products with static designs. Finally, *scalability*, which is the ability to increase the production capacity of instructional software products without a corresponding increase in costs, enables the serving of more and larger target groups.

The fourth, most important criterion is related to the holistic pedagogical view (Van Merriënboer & Boot, 2005), central in current educational innovations. A holistic view on learning assumes that complex knowledge and skills are best learnt through cognitive apprenticeship on the part of the learner in a rich environment (Collins, 1988). Experiences are provided for the learners that mimic the apprenticeship programs of adults in trades, or teachers in internship. It is not possible to immerse the learner to the extent that a traditional internship would imply. However, through the use of simulations and meaningful experiences, the learner would learn the ways of knowing of an expert. As a result, the most important problem of a holistic approach is how to deal with complexity.

Most authors introduce some notion of ''modeling'' to attack this problem. For example, Achtenhagen's (2001) notion of ''modeling the model'' prescribes a two-step approach to modeling, namely modeling reality and then modeling those models of reality from a pedagogical perspective. For developing instructional software, this implies first the *domain modeling* of realistic tasks and systems in such a way that they are simplified (i.e., reduction of complexity) towards the learner's level of ability while at the same time remaining representative for the ''real'' world. Second, it implies *pedagogical modeling* of these domain models to facilitate learning, such as the use of modeling examples, coaching, and scaffolding attuned to the level, progress and interests of the learner. This modeling of the model for instructional purposes allows the designer to determine which elements of the original model can be omitted, and which elements can be increased (not in the original, but introduced for supporting the functions of the model) (Gibbons, Bunderson, Olsen, & Robertson, 1995). For developing instructional materials, a third step should be added to this modeling process, namely *functional modeling*. This works out the two previous models in order to transfer them by means of design documents from the design phase to the production phase. Functional modeling allows the designer to determine how each element should be presented to the learner. The development criterion of modeling, actually consisting of the three sub-criteria, domain, pedagogical, and functional modeling, is conditional for the other three criteria, as adaptivity, generativity, and scalability all depend on an adequate modeling process.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

To which degree do current development methods meet the four criteria discussed in the previous section? This question can be answered from a theoretical perspective and an empirical perspective, discussed in the next sections.

3. Theoretical analysis of current development methods

The vast majority of development models is based upon the standard Instructional Systems Development (ISD) model, an instantiation of the generic Analysis, Design, Development (also called Production; the technical realization of the design), Implementation, and Evaluation model (ADDIE; Dick & Carey, 1996). Every phase in the ISD model identifies specific types of activities and outcomes, for which different specialists (e.g., designers, producers, visual artists and so forth) are responsible. The ISD model is based upon either a craft-production model or a mass-production model (Woll, 2003). Craft production models are directed at providing the highest-quality products, completely adapted towards a specific target group. Development involves highly skilled professionals using flexible, often custom-build tools, in a flexible work process. Products, processes, and tools are not standardized. Developers focus on producing limited quantities: In expanding the volume, costs will rise proportionally. The development of Computer-Based Training (CBT) is a good example of how craft production is applied in the field of instructional software (see, for example, Gibbons & Fairweather, 1998). The first row of Table 1 shows that craft-production models are not able to satisfy any of the new criteria for developing instructional software. Craft production is focused on single solutions for a highly specific target group with particular needs. So design, production, as well as final products, will be focused on that single solution, with very limited use of modeling and adaptivity. There is also no need for products to be modular, so generativity will be difficult to realize. Finally, due to the focus on producing small quantities of unique products, costs will proportionally rise with increase of volume.

Table 1
Production models and criteria for developing instructional software

Type of production model	Criteria			
	Adaptive	Generative	Scalable	Modeling
1. Craft production	No, because it is not necessary for custom-build single-purpose solution	No, because no standards are used and products are monolithic	No, because of the proportional increase in costs of customized processes	No, because modeling is not used as only single-purpose solutions are created
2. Mass production	No, because only single-purpose solutions are created to allow for efficient production	Yes, reached through standardization and modularity of products	Yes, it is an explicit objective, and reached through standardization and modularity of process	No, because modeling is not used as only single-purpose solutions are created
3. Lean production	Yes, it is an explicit objective and reached through the pull-principle	Yes, reached through standardization and modularity of products	Yes, it is an explicit objective, and reached through standardization and modularity of process	No, because the lack of design languages will limit transfer of modeling information

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

The counterparts of craft-production models are mass-production models. These are directed at providing (somewhat) similar products for a broad target group. Development involves narrowly skilled, interchangeable specialists, using expensive, single-purpose tools, in a continuous work process. Products, processes and tools are highly standardized and modularized. Developers focus on producing large quantities: With every increase in volume, costs per unit will decrease. The development of e-learning content is a good example of how mass production is applied in the field of instructional software. For example, recent standardization efforts (see Collis & Strijker, 2004, for examples in the military, commercial and academic fields) explicitly refer to standardized, modularized approaches to increase scalability. The second row of Table 1 shows that mass-production models are not able to satisfy the criteria of modeling and adaptivity for the same reasons as craft production models. However, the criteria of generativity can be satisfied as resulting products are highly standardized and modular. Also, scalability is an explicit objective of mass production.

Summarizing, from a theoretical point of view, it is clear that both craft- and mass-production models fail to satisfy all development criteria. Nevertheless, it is still possible that the practical application of both production models is so flexible that these criteria are still met. In order to determine if the results of the theoretical analysis are in line with the practical application of production models, an empirical analysis will be presented.

4. Empirical analysis of current development methods

To investigate the practical application of current development methods and the degree to which they meet the criteria, a survey study has been conducted. In this study, a questionnaire was used to gather opinions of experienced developers of instructional software on their current practices and experienced problems. A special focus is on the transition of information between the design and production phases, as defined by the third modeling step, namely functional modeling.

4.1. Method

4.1.1. Respondents

Thirty-seven developers of instructional software from the United States of America ($n = 17$) and the Netherlands ($n = 18$), working in large academic, commercial, and military organizations, participated in the study. The number of years of experience in a particular function or job was used to determine the main specialization of the respondents. The respondents' mean experience with developing instructional software was 17.0 years ($SD = 12.7$). Their main specialization was 'instructional designer', with a mean experience of 6.1 years ($SD = 5.4$). Furthermore, they were experienced as 'project leader' ($M = 5.0$, $SD = 4.4$), 'manager or policy-maker' ($M = 2.0$, $SD = 3.8$), 'multi-media specialist' ($M = 3.1$, $SD = 4.6$), or 'programmer' ($M = 0.8$, $SD = 3.1$). All respondents indicated that they used structural, phased approaches based upon the ISD model. All respondents were male and their age varied between 24 and 58 years.

4.1.2. Materials

An on-line questionnaire was used to gather information on the respondents and the problems they experienced in the development process. First, to investigate the

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

respondents' background, they were asked to indicate their overall experience, and the different roles they fulfilled, relevant to developing instructional software. They were also asked whether or not they applied structural, phased approaches based upon the ISD model, and who the responsible persons were for creating the design documents as output of the design phase. Second, to investigate possible development problems, the respondents were asked to rate on a 5-point scale for each of the five ADDIE phases the occurrence of seven typical development problems. These seven problems were drafted based upon interviewing three experienced developers. They are related to the (a) input, (b) core activity, (c) managing, (d) cooperation efforts, (e) labor intensity, (f) return of investment, and (g) changing requirements and conditions of each of the five ADDIE phases, resulting in 35 items. Third, the respondents were asked to rate on a 5-point scale 21 statements that focused on possible causes and consequences of problems in transferring information from the design phase to the production phase (see questions 1–21 in Table 4 further onwards, which present the questions on the transition problem as well as the ratings on these questions). Note that in the USA setting, it was possible to present some extra questions, providing us the opportunity to study the underlying causes of the transition problem more deeply. Therefore, questions 7–21 were only presented to the USA respondents. The 21 statements were established on the basis of suggestions from experienced developers. Fourth, to investigate the need for new solutions for the design to production transition problem, the respondents were asked to indicate on a 5-point scale whether they needed solutions for transition problems for either themselves or for their organization (see questions 22 and 23 of Table 4).

4.1.3. Data analysis

A factor analysis (Principle Component analysis with Varimax rotation) was used to identify the main development problems from the scores of the seven problems for each of the five ADDIE phases (35 items). Using a factor analysis for data reduction, a small number of issues could be identified that explains most of the variance observed in the larger number of item scores. Next, to determine the intra-item reliability of items within each factor, Cronbach's alpha is computed. In general, an alpha larger than .70 is regarded as satisfactory for drawing conclusions about different groups. Scale scores for each factor were obtained by adding item scores within the scale, and transforming crude scale scores linearly to a 0–100 range, with higher scores indicating more problems. Differences between groups (nationality, role, or type of organization) with respect to the issues were tested by MANOVA.

With regard to the possible causes and consequences of the transition problem, and the perceived need for new solutions, one-sample *T*-tests were used to test for differences between the ratings and the neutral score of 3.

4.2. Results

First, the respondents indicated that in their organization the following persons were responsible for creating training blueprints: In 15 cases (43% of the whole group), only the designers were responsible; in 6 cases (17% of the whole group), only producers, and in 14 (40% of the whole group) cases, combinations of designers and producers. So, in most cases, designers were either responsible for or directly involved in the creation of design documents.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

Second, with respect to possible development problems, the factor analysis identified four factors (see Table 2). Combined, they explained 53% of the total variance.

The first factor can be interpreted as *internal design and production difficulties*: respondents experience problems in organizing and executing (i.e., designing, producing) the design and production phases. The second issue can be interpreted as *rolling-out difficulties*: respondents experience difficulties with executing the implementation and evaluation phases. The third factor can be interpreted as *external design and production difficulties*:

Table 2
Factor loadings of the 35 items covering five ADDIE phases in combinations with seven development problems

Item	Scale 1	Scale 2	Scale 3	Scale 4
Internal design and development difficulties				
Problems with managing the design phase	.82			
Problems with production activities in the production phase	.67			
Problems with production phase too time-consuming	.65			
Problems with design phase too time-consuming	.64			
Problems with design activities in the design phase	.61			
Problems with managing the analysis phase	.56			
Problems with implementation phase too time-consuming	.56			
Problems with analysis phase too time-consuming	.53			
Problems with managing the evaluation phase	.51			
Problems with limited return on investment in the design phase	.49			
Problems with analyzing in analysis phase	.46			
Problems with managing the production phase	.41			
Rolling-out difficulties				
Problems with evaluation too labor intensive		.69		
Problems with cooperation with stakeholders in evaluation phase		.66		
Problems with changing requirements and conditions in implementation phase		.65		
Problems with stakeholders in implementation phase		.62		
Problems with limited return on investment in implementation phase		.61		
Problems with changing requirements and conditions in evaluation phase		.61		
Problems in evaluation phase with input from previous phase		.60		
Problems with managing the implementation phase		.58		
Problems in implementation phase with input		.53		
Problems with implementation activities in implementation phase		.48		
Problems with limited return on investment in evaluation phase		.48		
External design and development difficulties				
Problems with changing requirements and conditions in production phase			.73	
Problems in production phase with input			.69	
Problems with cooperation with stakeholders in production phase			.67	
Problems with cooperation with stakeholders in design phase			.64	
Problems with evaluation in evaluation phase			.61	
Problems with changing requirements and conditions in design phase			.59	
Problems in design phase with input			.49	
Problems with limited return on investments in production phase			.40	
Front-end analysis difficulties				
Problems with changing requirements and conditions in analysis phase				.84
Problems in analysis phase with input				.68
Problems with limited return on investments in analysis phase				.59
Problems with cooperation with stakeholders in analysis phase				.52

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

respondents experience problems in dealing with external conditions such as quality of input, cooperation between stakeholders, and changing requirements and conditions of the design and production phases. The fourth factor can be interpreted as *front-end analysis difficulties*: respondents experience problems with executing the analysis phase. Table 3 presents the number of associated items with a particular factor, and Cronbachs' alpha for the items contributing to that factor. MANOVA's showed no significant differences for nationality, role, or type of organization on any of the factors.

Finally, Table 4 shows the ratings on possible causes and consequences of the transition problem, as well as the need for new solutions for that problem.

Table 3
Number of items, Cronbach alpha's, and percentages of explained variance for the four factors

Factors	Number of items	Cronbach's alpha	Explained variance (%)
Internal design and production difficulties	12	.85	16
Rolling-out difficulties	10	.85	15
External design and production difficulties	8	.84	12
Front-end analysis difficulties	5	.67	9

Table 4
Ratings on causes, consequences, and need for new solutions for the transition problem

Questions	*M*	SD
Causes of problems in transferring information from design to production		
1. Lack of instructional design information	2.60	1.22
2. Lack of modularization	2.76	1.25
3. Lack of clear design languages	2.80	1.21
4. Lack of structure in design languages	2.50	1.22
5. Too much information transferred in design documents	2.13*	.86
6. Too little information transferred in design documents	2.70	1.05
7. Producers' lack of design knowledge	2.65	1.22
8. Producers making design decisions	2.70	1.26
9. Designers' lack of production knowledge	3.00	1.00
10. Designers making production decisions	2.82	1.07
11. Communication between designers and producers starts too late	3.00	1.32
12. Communication between designers and producers starts too early	1.94**	0.83
13. Cooperation between designers and producers starts too late	2.82	1.27
14. Cooperation between designers and producers starts too early	2.11**	0.78
15. Instructional Design models incomplete	2.76	1.14
16. Instructional Design models providing too little guidance	2.71	1.10
Consequences of problems in transferring information from design to production		
17. Too long development process	3.47	1.33
18. Planning of development process difficult	3.41	1.12
19. Structuring development teams difficult	3.00	1.22
20. Unpredictability of character of end product	2.76	1.09
21. Unpredictability of pedagogical quality of end product	2.65	1.27
Need for new solutions		
22. For the transition problem. for myself	2.88	1.26
23. For the transition problem. for my organization	3.06	1.34

All questions scored on a 5-point Likert-scale ranging from 1 (''totally disagree'') to 5 (''totally agree'').
* $p < .05$.
** $p < .01$.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

One-sample T-tests were used to test for differences between the ratings and the neutral score of 3. It appeared that three of the possible causes of the transition problem were rejected. First, respondents disagreed significantly ($M = 2.13$, SD = .86; $t = -3.33$, $p < .05$) with "too much information transferred in design documents" as a possible cause. Second, they also disagreed significantly ($M = 1.94$, SD = .83; $t = -5.29$, $p < .01$) with "communication between designers and producers starts too early" as a possible cause. Third, respondents also disagreed significantly ($M = 2.11$, SD = .78; $t = -4.65$, $p < .01$) with "cooperation between designers and producers starts too early" as a possible cause. The respondents were neutral towards all the other possible causes of the transition problem (see Table 4). Also, there were no significant differences between nationality, organization, and role on the ratings for possible causes and consequences and the need for new solutions.

4.3. Discussion

It appears that developers experience problems in organizing, managing and executing each phase, particularly in the design and production phases as indicated by the factors "internal design and production difficulties" and "external design and production difficulties", together explaining 28% of the variance. However, developers were not able to indicate clear causes of these problems. On the contrary, they indicated for three issues that they are not the cause, namely "too much information transferred", "too early communication", and "too early cooperation". Furthermore, they did not indicate possible consequences of the problems. Finally, they did not indicate a need for new solutions for themselves or for their organizations. It seems that designers report problems from a vague feeling rather than from experiencing concrete bottlenecks. A possible explanation is that they are educated in, and experienced with ISD-based development methods, and are not familiar with alternative methods that could help to solve the transition problem. Or, as Womack, Jones, and Roos (1990) stated, workers in a particular manufacturing model will not criticize this model nor move one to another model unless there is a real crisis. The absence of a need for new solutions also implies that developers will probably be rather skeptical in implementing new improvements.

A limitation of this study is that the respondents predominately had design experience and less production experience. This could possibly explain why they were not able to indicate clear causes and consequences of development problems. Producers, for example, could have been better able to indicate what they would need to improve the development process. A second limitation concerns the modest number of respondents, which limits the results of the factor analysis. Except for these limitations, the results of our empirical analysis are fully in line with the theoretical analysis.

5. Lean production as an alternative development method

The theoretical and empirical analyses show that current craft and mass production models do not satisfy the new criteria for developing instructional software. The practical application of these production models is not without problems either. In the manufacturing industry, lean production is introduced as a new model to overcome the problems of craft and mass production (Womack et al., 1990). Lean production aims to provide a high variability of high quality products, which are flexible to adapt to different clients.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

Development according to the lean production model involves autonomous, multi-skilled expert-teams, using *flexible* automated tools in a standardized and modularized work process. Lean production raises efficiency through the continuous, incremental improvement of work processes.

One important implication of lean production is the radical application of the ''pull-principle''. Craft- and mass-production models are supply-oriented: developers ''push'' the product they think is appropriate for the client forward through the work process. Lean production, however, is demand-oriented. In the field of Instructional Systems Development (ISD), this implies that developers have to consider the specific needs of clients and create their products accordingly. The pull-principle applies to both intermediate products, transferred between the phases, and final products. Fig. 1 compares the supply-oriented model with the demand-oriented model, indicating that according to the pull-principle, designers pull information from the analysts, producers pull information from the designers, implementers pull information from the producers, and evaluators pull information from the implementers. This demand-oriented principle ensures a more effective transition process of information and products.

Another important implication of lean production is ''waste-reduction'', which is the continuous process of measuring and analyzing the development process and (intermediate) products to improve quality, increase standardization, and limit waste (Ohno, 1988). Waste is defined as unnecessary delays, defects, and redundancies in the development process. This puts much emphasis on the quality of the transition of information or products, because each transition must be optimal the first time. Otherwise, time-consuming iterations are necessary to provide additional explanations or correct errors.

With respect to the criteria presented in our theoretical analysis of current development methods, the lean production model is the only model that meets the criterion of adaptivity (see Table 1). Therefore, lean production is suitable for the development of innovative instructional software (Woll, 2003). Schellekens (2004) suggests a similar, process-focused

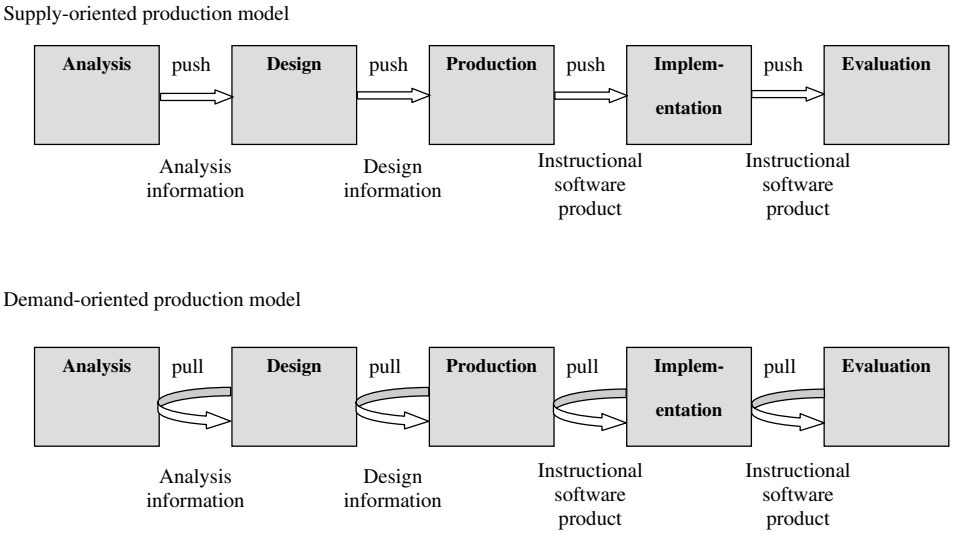

Fig. 1. Supply vs. demand-oriented ISD models.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

strategy, which promotes "mass-customization" with a high degree of adaptation to the needs of clients, high design quality, and volume flexibility.

As can be seen in the third row of Table 1, satisfying the criterion of modeling, particularly the step of functional modeling, remains difficult in the lean production model. The application of the pull-principle between the design and development phases implies that developers have to pull the information they require from designers. However, the designers lack the necessary means to provide the producers with information that ensures an unequivocal interpretation by producers. They have a different background than producers (educational vs. technical) and use different tools (analysis and design tools vs. technical production tools). For applying the pull-principle between other phases, the signaled problem is less relevant because analysts and designers have the same background, and because developers and implementers, as well as implementers and evaluators, exchange concrete products.

6. Lack of common design languages

The lack of standardized design languages, known by both designers and producers, is a fundamental problem in the transition between design and production. Such design languages should be able to allow for functional modeling by capturing and describing the domain and pedagogical models at a level of detail ensuring that producers interpret them unequivocally (Waters & Gibbons, 2004). Design languages require notation systems to convey their message by means of symbolic, graphical, textual or other conventions. An example of a graphical modeling language, not bound to the field of instructional software development, is the Unified Modeling Language (UML; Booch, 1994). The notation system of UML (i.e., diagrams) enables both designers and producers to describe and understand a design. Recent attempts to introduce design languages in the field of instructional software development are IMS Learning Design (IMS LD) (Koper & Tattersall, 2005) and the Educational Environment Modeling Language (E2ML; Botturi, 2006). Both languages are promising but not yet able to provide a complete solution for the transition problem. IMS LD is limited to the configuration of a pedagogical model in an IMS LD compatible e-learning system. E2ML is limited to describing instructional *design* issues such as learning goals, roles, actions, and resources, instead of (relating this to) instructional *software* issues such as its interface design, interaction design, and information flow.

As long as there are no complete design languages available in the field of instructional software development, iteration as proposed by agile methods (e.g., see http://www.agilealliance.com) might possibly provide a solution to overcome the transition problem. Iteration implies that designers and producers model and produce the instructional software incrementally and in direct contact with each other, rather than relying on the once-only transfer of formalized information (i.e., functional modeling) between the design and production phases. Iteration is popular in the fields of software engineering (Fowler, 2003) and instructional development (e.g., Reigeluth & Nelson, 1997; Tennyson, 1995). However, three problems limit the value of iteration: (a) lack of expertise of designers and producers, (b) outsourcing of production, and (c) lower efficiency. The first limitation of iteration is that one of the characteristics of instructional software development is the participation of domain experts such as subject matter experts and instructors, with relatively less instructional design and software production expertise (Hoogveld, Paas, Jochems, & van Merriënboer, 2002; Spector & Muraida, 1997). Iteration, however, requires

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

considerable expertise in order to determine exactly when and how iteration should take place (Verstegen, 2003). This problem may be solved by Verstegen's methodology for the development of a needs statement (Verstegen, 2003). In this method, directed at establishing a thoroughly needs-assessment before the actual development process starts, a series of workshops is organized with stakeholders such as clients, teachers, learners, and instructional designers. Under guidance of an experienced discussion leader, they proceed iteratively through all ISD phases in a structured and standardized manner, and record their assumptions and (provisional) decisions with regard to design, production, implementation, and evaluation issues. The standardized method and experienced discussion leader overcome the problem of lack of expertise. Also, if producers are involved in the workshops, the resulting needs-assessment documents may provide development information that is understood and accepted by both designers and producers, thereby avoiding the need to rely solely on design languages and formal transfers of design documents. However, this will often be impossible due to the second limitation of iteration, namely outsourcing. In large organizations and with professional development projects, there is often a strict juridical and organizational separation between design and production due to outsourcing of production activities to external companies. Note that production-related input in Verstegen's method can be accomplished by involving other producers, to promote at least understanding of the development information by the ultimate producers. The third limitation of iteration is that it may reduce efficiency because it costs extra time, and there is no guarantee that this will be regained in a later phase of the development process. For example, Verstegen's method explicitly emphases iteration within the needs-assessment phase to prevent iteration between later development phases.

Summarizing, lean production promises to satisfy the new criteria for instructional software development, except domain, pedagogic, and particularly functional modeling, due to the lack of common design languages and the limitations of iteration. A possible solution should focus on supporting designers to provide producers with exactly the functional modeling information they need.

7. Three building-block solutions

Currently, production-related information is typically embedded in three types of building blocks for the production process: (a) design documents as input, (b) programming structures as throughput, and (c) learning content as output (see Fig. 2).

In order to better interrelate the design phase into the production phase, it is proposed that designers' attention be focused on these three building blocks, thereby preventing the need to rely solely on design languages and/or iteration. Three building-block solutions are proposed to compensate for the limited production expertise of the typical designer:

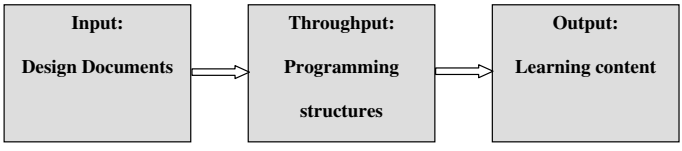

Fig. 2. The three artifacts in the development process that embed production-related information.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

First, the Developing Design Documents (3D) model to support designers to improve design documents; second, instructional software templates to support designers to create programming structures, and third, an integrative approach to support designers to reuse learning objects. According to the first solution, designers work in their traditional role but use functional modeling to interrelate their design more explicitly to production. According to the second and third solutions, designers operate in a producer's role, using programming structures and learning objects to interrelate their designs more explicitly to production. The next sections discuss the three solutions.

7.1. The 3D-model

In functional modeling, design documents such as training blueprints and storyboards serve as input for creating technical specifications by producers. Design documents may be difficult to interpret for three reasons: (a) different instructional and technical structures are often not meaningfully organized, (b) different levels of detail are mixed up, and (c) different expressions are used in a nonstandardized manner. With regard to meaningful organization, Gibbons' model of Design Layers (Gibbons, 2003) may be used for *stratification* of the instructional software design on seven, interrelated layers: content, strategy, control, message, representation, media logic, and data management. Each layer is typified by the designer's selection of design languages pertaining to the solution of different instructional design subproblems. Together, the functional designs at each layer make up the total design. Stratification helps to determine the relations between the functionally different instructional and technical structures, while at the same time staying cognizant of the need for integration of those structures within the complete design.

With regard to mixing up different levels of detail, the three perspectives of Fowler (2003) may be used for the *elaboration* of the instructional software design: (a) a conceptual perspective, with more or less superficial and descriptive information, (b) a specification perspective, with more or less comprehensive and detailed information, and (c) an implementation perspective, with more or less technical and meticulous information. Elaboration helps to determine the required level of detail, depending on the capabilities of the designer and the needs of the producer.

With regard to use of nonstandardized expression, designers may reach *formalization* of their design by making their informal and formal design languages explicit. They should strive for (combinations of) formal languages, but depending on their capabilities and the needs of the producer, they can also select (combinations of) informal languages. Formalization helps to determine the required level of standardization.

The 3D-model uses stratification, elaboration and formalization as its dimensions. Designers, with or without producers, may first analyze their design situation in order to determine the optimal configuration of the 3D-model (e.g., What kind of designers and producers are involved? What kind of training is the design made for? Which support tools are available?), and then use this configuration to stratify, elaborate and formalize their design documents. Fig. 3 presents the 3D-model in its full configuration, in which all dimensions are completely utilized. The 3D-model provides producers with insight into the underlying structure and content of the functional model, even when the design languages used are deficient.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

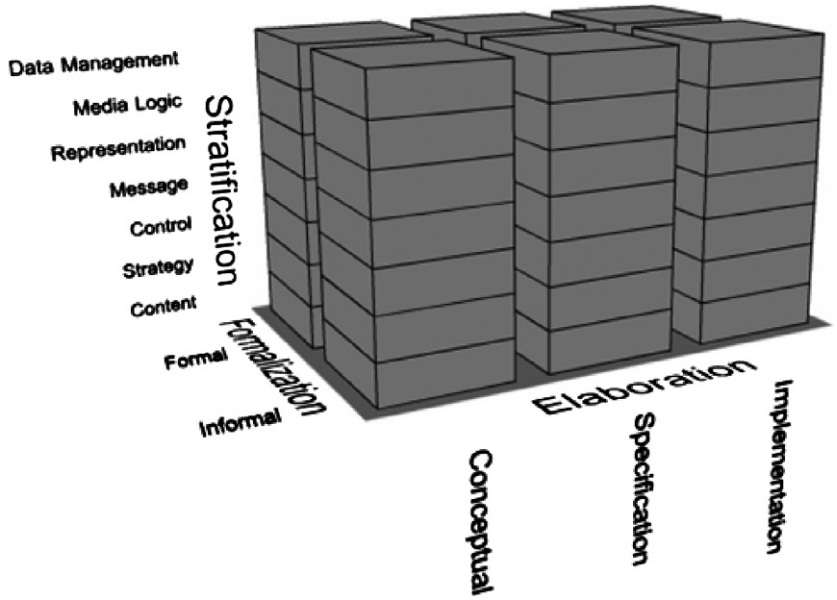

Fig. 3. The 3D-model.

7.2. Instructional software templates

e-Learning systems and authoring tools often provide instructional software templates, which producers can use to easily create or adapt programming structures that make up the instructional software. This ''programmer-less-authoring'' (Hedberg & Sims, 2001) is based on automation of routine tasks and intuitive interfaces such as ''wizards''. They provide support on three levels. First, on the authoring level, the templates offer prefabricated ''moulds'' of programming structures to implement the lessons, practice items, test questions, examples, cases, feedback, learner support, and so forth into the instructional software. Second, on the technical level, the templates automatically produce programming structures that are compliant with current e-learning technologies, learning technology standards, and different operating systems. Third, on the pedagogical level, the templates provide prefabricated structures, for example, drill-and-practice, concept learning, mastery learning, and case-based learning. On the one hand, producers use templates to speed up their authoring and technical activities and to receive pedagogical support for creating programming structures without support from instructional designers. On the other hand, those designers may use templates to speed up their pedagogical activities and receive authoring and technical support.

Instructional software templates explicitly interrelate design and production. First, designers can choose not to rely on the producers at all and select and instantiate appropriate templates themselves. This way, they are able to create and assemble the programming structures they want, thereby making their own instructional software. Second, designers can provide producers with design information by means of implementing their design principles. These templates will then force producers to apply particular pedagogical principles in the instructional software. Third, designers can provide producers with

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

design information by means of example products they have created with templates. Optionally, these examples can be presented to the producers together with the design documents.

7.3. Learning objects

To increase the efficiency of design and production, there is currently much emphasis on the *reuse* of learning content. If learning content is divided into small, modular chunks, often called "learning objects", developers will be able to combine and recombine these objects to create new learning content. Van Merriënboer and Boot (2005) identify six problems with the current reuse of learning objects. The first three problems relate to the nature of learning objects: (1) the *metadata problem* refers to the fact that it is difficult and extremely labor intensive to specify metadata for large sets of learning objects; (2) the *arrangement problem* refers to the fact that combining and sequencing learning objects into larger arrangements is not always easy and self-evident, and (3) the *exchange problem* refers to the fact that it may be difficult from a psychological viewpoint (e.g., due to the "not-invented-here" syndrome) or organizational viewpoint (e.g., due to security issues or intellectual property rights) to exchange learning objects between developers and between e-learning systems. The remaining three problems arise because current approaches of reuse are not consistent with the holistic pedagogical view: (4) the *context problem* refers to the fact that effective learning objects cannot be created in isolation without an implicit or explicit instructional setting, target group, and other contextual descriptors; (5) the *pedagogical function* problem refers to the fact that it is difficult to express pedagogical intentions for a learning object by means of technical properties such as metadata, and (6) the *correspondence problem* refers to the fact that a developer working from a holistic viewpoint will typically not search for one particular learning object but for a set of learning objects that is meaningfully interrelated and aimed at the construction of *one* rich cognitive representation.

Van Merriënboer and Boot (2005) propose an integrative approach, stressing four solutions to improve the reuse of learning objects. The first solution is to reedit instead of reuse learning objects. This increases the chance that the developer will find a useful learning object because it becomes less important to find exactly what is needed. The second solution is to use templates instead of instantiations as learning objects. Templates allow for the easy modification of learning objects (e.g., a change of an American grading system to a European grading system), making them useful for a broader range of situations. The third solution is to automate the creation and reuse of learning objects. The use of automatic analysis of multi-media content and the semantic indexing of this content in metadata fields makes reuse more cost-effective and also yields more objective metadata than indexing by hand. The final solution is to use intermediate products in addition to final products as learning objects. Intermediate products, such as task analysis results and lesson plans, contain rich information that describes the final products for which they were made. This rich information is more suitable than metadata to provide input for searching suitable learning objects.

The integrative approach to the reuse of learning objects explicitly supports designers to interrelate design to production. First, designers can choose not to rely on producers at all and independently select and reuse appropriate learning objects to assemble the

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

instructional software product they want. Second, designers can provide producers with design documents illustrated with example sets of learning objects they have assembled.

8. Conclusions and discussion

New criteria for instructional software development are set by recent pedagogical, technical and organizational innovations: adaptivity, generativity, scalability, and last but least, modeling. From theoretical and empirical analyses, which clearly corroborate each other, it appears that existing instructional software development methods based on a push-principle do not satisfy all these criteria. Lean production, based upon the pull-principle, is suggested as a new development method to enable the required mass-customization of instructional software. However, lean production also suffers from the fundamental problem of a lack of design languages to transfer information from the design to the production phase. In order to overcome this problem, designers may use production building blocks that prevent sole reliance on design languages and/or iteration. We proposed three building-block solutions to support designers in functional modeling: the 3D-model to improve design documents, instructional software templates to create programming structures, and the integrative approach for reusing learning objects.

The suggested building-block methods are predominantly based on practical experiences and theoretical as well as empirical analyses. Further research might go in three directions. First, it should validate the actual value of the building-blocks solutions, separately and in combination, on the success of the transition between design and production. In particular, it is interesting to study the application of the three solutions by domain experts and teachers, because these are often the inexperienced designers that are involved in actual instructional software development projects. Second, as our empirical analysis shows, designers are likely to be rather skeptical towards new solutions. Changing the instructional software development process as drastically as lean production models suggest, and also introducing the proposed building-block solutions that require other design skills than before, will probably meet resistance from designers. So, further research and validation should also be aimed at the development of innovation models that help to promote acceptance of new solutions by designers. Third, further research may pertain to the roles of designers and producers. The pull-principle suggests that designers should be fully responsible for solving the transition problem, as producers are the ''demanding party''. This does not imply that designers should provide any solution producers demand. So, further research should be aimed at clarification of new roles for producers and designers.

This study has some clear practical implications for the use of the three building-block solutions, either alone or in combination. First, they allow designers to improve their design documents through the analysis of instructional software templates and learning objects used by a production team. This yields useful product information and informs designers about the capabilities and preferences of the producers. Second, they allow designers to improve their instructional software templates through the analysis of multiple sets of learning objects and design documents. This yields useful design information to serve as input for creating new templates. Third, they allow designers to improve reuse of learning objects, as the integrative approach incorporates both using design documents (called ''intermediate products'') and using instructional software templates. This way the three proposed solutions may offer a first step toward the implementation of the holis-

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

tic pedagogical view, with a focus on authentic learning tasks, in innovative instructional software.

Acknowledgements

The authors want to thank Craig Woll for his helpful comments on earlier drafts of this article.

References

Achtenhagen, F. (2001). Criteria for the development of complex teaching–learning environments. *Instructional Science, 29*, 361–380.

Atkinson, R. C., & Wilson, H. A. (1969). *Computer assisted instruction: A book of readings*. New York: Academic Press.

Booch, G. (1994). *Object-oriented analysis and design with applications*. Redwood City, CA: Benjamin/Cummings.

Botturi, L. (2006). E2ML: A visual language for the design of instruction. *Educational Technology, Research and Development, 54*, 265–293.

Collins, A. (1988). *Cognitive apprenticeship and instructional technology (Tech. Rep. No. 6899)*. Cambridge, MA: BBN Labs Inc.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453–493). Hillsdale, NJ: Lawrence Erlbaum.

Collis, B., & Strijker, A. (2004). Technology and human issues in reusing learning objects. *Journal of Interactive Media in Education*(4), Special issue on the Educational Semantic Web.

Dick, W., & Carey, L. (1996). *The systematic design of instruction* (4th ed.). New York: Harper Collins.

Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language*. Boston, MA: Addison-Wesley.

Gibbons, A. S. (2003). What and how do designers design? A theory of design structure. *Tech Trends, 47*(5), 22–27.

Gibbons, A. S., Bunderson, C. V., Olsen, J. B., & Robertson, J. (1995). Work models: Still beyond instructional objectives. *Machine-Mediated Learning, 5*(3&4), 221–236.

Gibbons, A. S., & Fairweather, P. G. (1998). *Computer-based instruction: Design and development*. Englewood Cliffs, NJ: Educational Technology Publications.

Gibbons, A. S., Nelson, J., & Richards, R. (2000). The nature and origin of instructional objects. In D. A. Wiley (Ed.), *The instructional use of learning objects* (pp. 25–58). Bloomington, IN: AECT.

Hedberg, J., & Sims, R. (2001). Speculations on design team interactions. *Journal of International Learning Research, 12*(2–3), 193–208.

Hoogveld, A. W. M., Paas, F., Jochems, W. M. G., & van Merriënboer, J. J. G. (2002). Exploring teachers' instructional design practices: Implications for improving teacher training. *Instructional Science, 30*, 291–305.

Jochems, W., van Merriënboer, J. J. G., & Koper, R. (Eds.). (2003). *Integrated e-learning: Implications for pedagogy, technology, and organization*. London, UK: RoutledgeFalmer.

Jonassen, D. H. (1999). Designing constructivist learning environments. In C. M. Reigeluth (Ed.). *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II, pp. 371–396). Mahwah, NJ: Lawrence Erlbaum.

Koper, R., & Tattersall, C. (Eds.). (2005). *Learning design: A handbook on modeling and delivering networked education and training*. Heidelberg, Germany: Springer-Verlag.

Merrill, M. D. (2002). First principles of instruction. *Educational Technology, Research and Development, 50*(3), 43–59.

Nelson, L. M. (1999). Collaborative problem solving. In C. M. Reigeluth (Ed.). *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II, pp. 241–267). Mahwah, NJ: Lawrence Erlbaum.

Ohno, T. (1988). *Toyota production system: Beyond large-scale production*. Portland, OR: Productivity Press.

Parrish, P. E. (2004). The trouble with learning objects. *Educational Technology, Research and Development, 52*(1), 49–67.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002

Reigeluth, C. M. (1999). *Instructional-design theories and models: A new paradigm of instructional theory* (Vol. 2). Mahwah, NJ: Lawrence Erlbaum.

Reigeluth, C. M., & Nelson, L. M. (1997). A new paradigm of ISD? In R. M. Branch, B. B. Minor, & D. P. Ely (Eds.). *Educational media and technology yearbook* (Vol. 22, pp. 24–35). Englewood, CO: Libraries Unlimited.

Schank, R. C., Berman, T. R., & MacPerson, K. A. (1999). Learning by doing. In C. M. Reigeluth (Ed.). *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II, pp. 161–181). Mahwah, NJ: Lawrence Erlbaum.

Schellekens, A. (2004). *Towards flexible programmes in higher professional education.* Unpublished PhD dissertation. Open Universiteit of the Netherlands, Heerlen.

Spector, J., & Muraida, D. (1997). Automating design instruction. In S. Dijkstra, N. Seel, F. Schott, & D. Tennyson (Eds.). *Instructional design: International perspectives* (Vol. 2, pp. 59–81). Mahwah, NJ: Lawrence Erlbaum.

Tennyson, R. D. (1995). Four generations of instructional system development. *Journal of Structural Learning, 12*(3), 149–164.

Van Merriënboer, J. J. G. (1997). *Training complex cognitive skills*. Englewood Cliffs, NJ: Educational Technology Publications.

Van Merriënboer, J. J. G., & Boot, E. W. (2005). A holistic pedagogical view of learning objects. In J. M. Spector, S. Ohrazda, P. van Schaaik, & D. A. Wiley (Eds.), *Innovations in instructional technology: Essays in honor of M. David Merrill* (pp. 43–64). Mahwah, NJ: Lawrence Erlbaum.

Van Merriënboer, J. J. G., & Kirschner, P. A. (2007). *Ten steps to complex learning*. Mahwah, NJ: Lawrence Erlbaum.

Verstegen, D. M. L. (2003). *Iteration in instructional design: An empirical study on the specification of training simulators*. Unpublished PhD dissertation. Utrecht University, Utrecht.

Waters, S. H., & Gibbons, A. S. (2004). Design languages, notation systems, and instructional technology: A case study. *Educational Technology, Research and Development, 52*(2), 57–68.

Woll, C. (2003). *Identifying value in instructional production systems: Mapping the value stream*. Unpublished PhD dissertation. Utah State University, Logan.

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The machine that changed the world*. New York: Harper Collins.

Please cite this article in press as: Boot, E. W. et al., Improving the development of instructional ..., *Computers in Human Behavior* (2007), doi:10.1016/j.chb.2007.05.002