26

Ten Steps to Complex Learning

A New Approach to Instruction and Instructional Design

Paul Kirschner
Utrecht University

Jeroen J. G. van Merriënboer
Open University of the Netherlands

The subject of this chapter, ten steps to complex learning (van Merriënboer & Kirschner, 2007), was recently published as a practical and modified version of the four-component instructional design (4C-ID) model originally posited by van Merriënboer in 1997. These ten steps are mainly prescriptive and aim to provide a practicable version of the 4C-ID model for teachers, domain experts involved in educational or training design, and less experienced instructional designers. The model described here will typically be used to develop educational or training programs, which can have a duration ranging from several weeks to several years, aimed at the acquisition of complex cognitive skills (in this chapter referred to as complex learning).

Complex Learning

Complex learning is the integration of knowledge, skills and attitudes; coordinating qualitatively different constituent skills; and often transferring what was learned in school or training to daily life and work. There are many examples of theoretical design models that have been developed to promote complex learning: cognitive apprenticeship (Collins, Brown, & Newman, 1989), 4-Mat (McCarthy, 1996), instructional episodes (Andre, 1997), collaborative problem solving (Nelson, 1999), constructivism and constructivist learning environments (Jonassen, 1999), learning by doing (Schank, Berman, & MacPerson, 1999), multiple approaches to understanding (Gardner, 1999), star legacy Schwartz, Lin, Brophy, & Bransford, 1999), as well as the subject of this contribution, the Four-Component Instructional Design model (van Merriënboer, 1997; van Merriënboer, Clark, & de Croock, 2002). These approaches all focus on authentic learning tasks as the driving force for teaching and learning because such tasks are instrumental in helping learners to integrate knowledge, skills, and attitudes (often referred to as competences), stimulate the coordination of skills constituent to solving problems or carrying out tasks, and facilitate the transfer of what has been learned to new and often unique tasks and problem situations (Merrill, 2002b; van Merriënboer, 2007; van Merriënboer & Kirschner, 2001).

Though the first two goals are essential for education and training and should not be underestimated, the fundamental problem facing instructional designers is education and training's apparent inability to achieve the third goal, the transfer of learning. Instructional design (ID) theory needs to support the design and development of programs that will help students acquire and transfer professional competencies or complex cognitive skills to an increasingly varied set of real-world contexts and settings. *The Ten Steps to Complex Learning* approach to ID (van Merriënboer & Kirschner, 2007) claims that a new ID approach is needed to reach this goal. In the next section, this holistic design approach is presented.

This is VERY relevant for the work we are doing with the CourseFlow pilot.

Holistic Design

Holistic design is the opposite of atomistic design where complex contents and tasks are usually reduced to their simplest or smallest elements. This reduction is such that contents and tasks are continually reduced to a level where they can easily be transferred to learners through a combination of presentation (i.e., expository teaching) and practice. This approach works very well if there are few interactions between those elements, but often fails when the elements are closely interrelated because here the whole is much more than the sum of its separate parts. Holistic design approaches to learning deal with complexity without losing sight of the separate elements and the interconnections between them. Using such an approach solves three common problems in education, namely, compartmentalization, fragmentation, and the transfer paradox.

Compartmentalization

ID models usually focus on one particular domain of learning (i.e., cognitive, affective, psychomotor) and within that domain between models for declarative learning that emphasize instructional methods for constructing conceptual knowledge and models for procedural learning that emphasize methods for acquiring procedural skills. This *compartmentalization*—the separation of a whole into distinct parts or categories—has had negative effects in vocational and professional education.

Any good practitioner has highly developed cognitive and technical skills, a deep knowledge of the work domain, a good attitude toward that work, and keeps all of this up-to-date. In other words, these different aspects of professional competencies cannot be compartmentalized into atomistic domains of learning. To counter this compartmentalization, holistic design integrates declarative, procedural, and affective learning to facilitate the development of an integrated knowledge base that increases the chance of transfer.

Fragmentation

Most, if not all, ID models are guilty of *fragmentation*—the act or process of breaking something down into small, incomplete, or isolated parts—as their basis (see Ragan & Smith, 1996; van Merriënboer & van Dijk, 1998). Typically they begin by analyzing a chosen learning domain. They then divide it into distinct learning or performance objectives (e.g., recalling a fact, applying a procedure, understanding a concept), and then they select different instructional methods for reaching each of the separate objectives (e.g., rote learning, skills labs, problem solving). For complex skills, each objective corresponds with one subskill or constituent skill, and their sequencing results in part-task sequences. The learner is taught only one or a very limited number of constituent skills at the same time, and new constituent skills are gradually added until—at the end of the instruction—the learner practices the whole complex skill.

The problem here is that most complex skills are characterized by numerous interactions between the different aspects of task performance with very high demands on their coordination. Learning and instruction that is based upon such fragmentation of complex tasks into sets of distinct elements without taking their interactions and required coordination into account fails because learners ultimately cannot integrate and coordinate the separate elements in transfer situations (Clark & Estes, 1999; Perkins & Grotzer, 1997; Spector & Anderson, 2000; Wightman & Lintern, 1985). To remedy this, holistic design focuses on highly integrated sets of objectives and their coordinated attainment in real-life performance.

The Transfer Paradox

Instructional designers often either strive for or are required to achieve efficiency. To this end they usually select methods that will minimize the (1) number of practice items required, (2) time spent on task, and (3) learners' investment of effort to achieve the learning objectives. Typical here is the situation in which students must learn to diagnose different types of technical errors (e.g., e1, e2, e3). If a minimum of three practice items is needed to learn to diagnose each error, the designer will often choose to first train students to diagnose e1, then e2, and finally e3, leading to the following learning sequence: e1, e1, e1, e2, e2, e2, e3, e3, e3.

Although this sequencing will probably be very efficient, it yields *low transfer* of learning because it encourages learners to construct highly specific knowledge for diagnosing each distinct error, only allowing them to perform in the way specified in the objectives. If a designer aims at transfer, and with the objective to train students to diagnose as many errors as possible, then it would be better to train students to diagnose the three errors in a random order leading, for example, to a different sequence such as e3, e2, e2, e1, e3, e3, e1, e2, e1.

This sequence will probably be less efficient for reaching the isolated objectives, because it will probably increase the needed time-on-task or investment of learner effort and might even require more than three practice items to reach the same level of performance for each separate objective as the first sequence. In the long run, however, it will help learners achieve a *higher transfer of learning* because it encourages them to construct general and abstract knowledge rather than knowledge only related to each concrete, specific error and will thus allow learners to better diagnose new, not yet encountered, errors. This is the transfer paradox (van Merriënboer & de Croock, 1997), where methods that work best for reaching isolated, specific objectives are not best for reaching integrated objectives and transfer of learning. Holistic design takes this into account, ensuring that

students confronted with new problems not only have acquired specific knowledge to perform the familiar aspects of a task, but also have acquired the necessary general or abstract knowledge to deal with the unfamiliar aspects of those tasks.

Four Components and Ten Steps

The Ten Steps (van Merriënboer & Kirschner, 2007) is a prescriptive approach to the *Four-Component Instructional Design model* (4C-ID; van Merriënboer, 1997) that is practicable for teachers, domain experts involved in ID, and instructional designers. It will typically be used for developing substantial learning or training programs ranging in length from several weeks to several years or that entail a substantial part of a curriculum for the development of competencies or complex skills. Its basic assumption is that *blueprints for complex learning* can always be described by *four basic components*: learning tasks, supportive information, procedural information, and part-task practice (see Table 26.1).

REal life tasks is the tricky issue for most instructional designers.

The term *learning task* is used here generically to include case studies, projects, problems, and so forth. They are *authentic whole-task experiences* based on real-life tasks that aim at the integration of skills, knowledge, and attitudes. The whole set of learning tasks exhibits a high variability, is organized in easy-to-difficult task classes, and has diminishing learner support throughout each task class.

reasoning can also be called "Decision making"

Supportive information helps students learn to perform nonroutine aspects of learning tasks, which often involve problem solving and reasoning. It explains how a domain is organized and how problems in that domain are (or should be) approached. It is specified per task class and is always available to learners. It provides a bridge between what learners already know and what they need to know to work on the learning tasks.

these are the SCAFFOLDS!

Procedural information allows students to learn to perform *routine aspects* of learning tasks that are always performed in the same way. It specifies exactly how to perform the routine aspects of the task and is best presented just in time—precisely when learners need it. It quickly fades as learners gain more expertise.

Finally, *part-task practice* pertains to additional practice of routine aspects so that learners can develop a very high level of *automaticity*. Part-task practice typically provides huge amounts of repetition and only starts after the routine aspect has been introduced in the context of a whole, meaningful learning task.

Each of the four components corresponds with a specific design step (see Table 26.1). In this way, the design of learning tasks corresponds with step 1, the design of supportive information with step 4, the design of procedural information with step 7, and the design of part-task practice with step 10. The other six steps are supplementary and are performed when necessary. Step 2, for example, organizes the learning tasks in easy-to-difficult categories to ensure that students work on tasks that begin simple and smoothly increase in difficulty, and step 3 specifies the standards for acceptable performance of the task which is necessary to assess performance and provide feedback. Steps 5 and 6 may be necessary for in-depth analysis of the supportive information needed for learning to carry out nonroutine aspects of learning tasks. Finally, steps 8 and 9 may be necessary for in-depth analysis of the procedural information needed for performing routine aspects of learning tasks.

Table 26.1 The Four Blueprint Components of 4C-ID and the Ten Steps to Complex Learning

Blueprint Components of 4C-ID	*Ten Steps to Complex Learning*
Learning Tasks	1. Design Learning Tasks
	2. Sequence Task Classes
	3. Set Performance Objectives
Supportive Information	4. Design Supportive Information
	5. Analyze Cognitive Strategies
	6. Analyze Mental Models
Procedural Information	7. Design Procedural Information
	8. Analyze Cognitive Rules
	9. Analyze Prerequisite Knowledge
Part-Task Practice	10. Design Part-Task Practice

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

Designing With the Four Blueprint Components

Figure 26.1 shows how the four blueprint components (also see the left hand column of Table 26.1) are interrelated to each other.

Learning Tasks

Learners work on tasks that help them develop an integrated knowledge base through a process of *inductive learning*, inducing knowledge from concrete experiences. As a result, each learning task should offer whole-task practice, confronting the learner with all or almost all of the constituent skills important for performing the task, including their associated knowledge and attitudes. In this whole-task approach, learners develop a holistic vision of the task that is gradually embellished during training. A sequence of learning tasks provides the

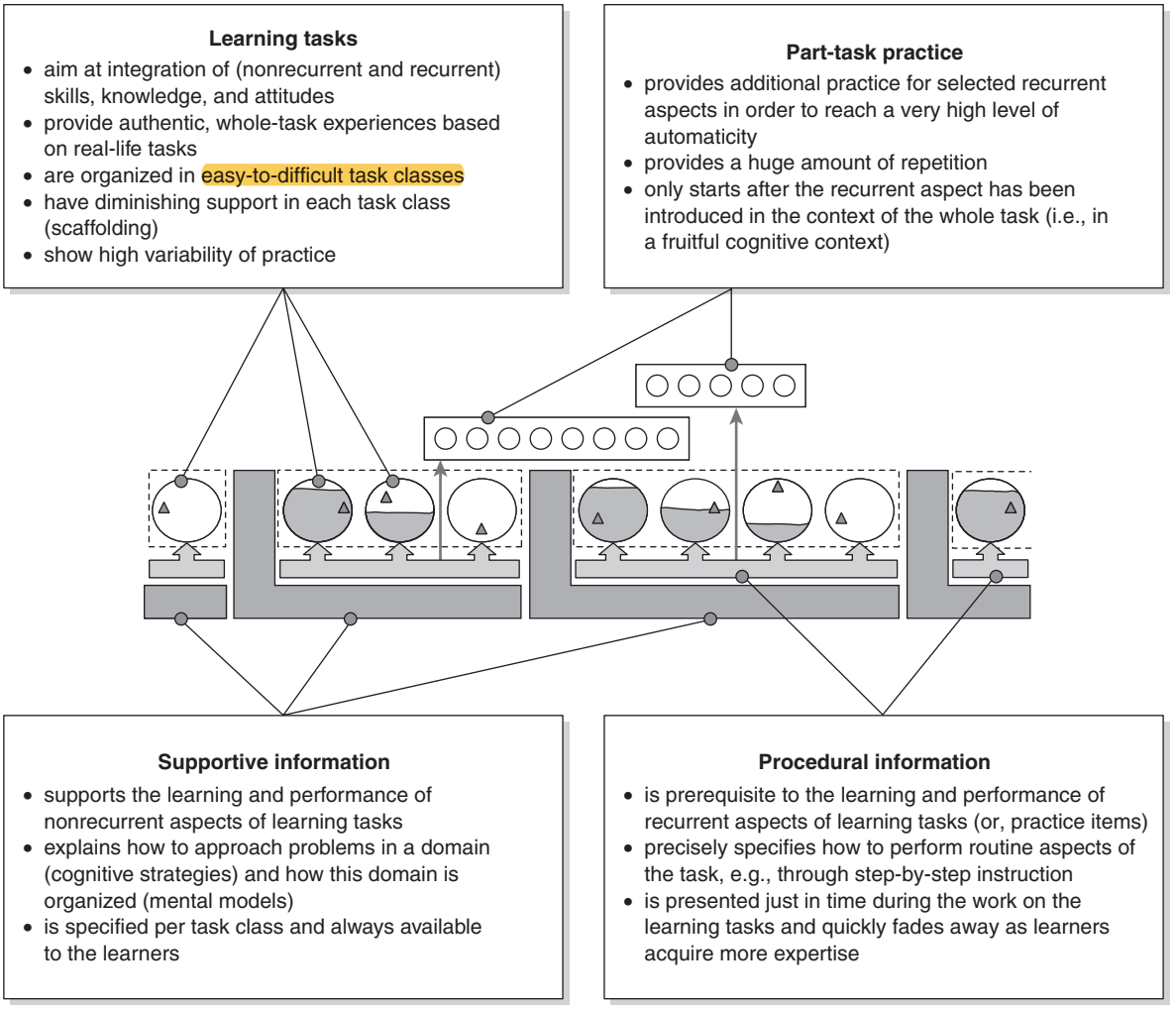


Figure 26.1 A Schematic Training Blueprint for Complex Learning

backbone of a training program for complex learning. Schematically:

Variability

In line with the earlier discussed transfer paradox, it is important that the chosen learning tasks differ from each other on all dimensions that also differ in the real world, so that learners can abstract more general information from the details of each single task. There is strong evidence that such variability of practice is important for achieving *transfer of learning*—both for relatively simple tasks (e.g., Paas & van Merriënboer, 1994; Quilici & Mayer, 1996) and highly complex real-life tasks (e.g., Schilling, Vidal, Ployhart, & Marangoni, 2003; van Merriënboer, Kester, & Paas, 2006). A sequence of different learning tasks thus always provides the backbone of a training program for complex learning. Schematically, it looks like this:

this is interesting and could be examined in the types of alignments that CathyR is doing in mapping out her courses, across time frames - i.e., using CourseFlow feature of looking at LO>assignment alignment from courses in different semesters.

Task Classes

It is not possible to use very difficult learning tasks with high demands on coordination right from the start of a training program, so learners start work on relatively easy whole-learning tasks and progress toward more difficult ones (van Merriënboer, Kirschner, & Kester, 2003). Categories of learning tasks, each representing a version of the task with the same particular difficulty, are called *task classes*. All tasks within a particular task class are equivalent in that the tasks can be performed based on the same body of general knowledge. A more difficult task class requires more knowledge or more embellished knowledge

for effective performance than the preceding, easier task classes. In the training blueprint, the tasks are organized in an ordered sequence of task classes (i.e., the dotted boxes) representing easy-to-difficult versions of the whole task:

IF task support can be viewed as SCAFFOLDS, then we now have a better definition of what scaffolds are (or can be) - i.e., SCAFFOLDS are tangible artifacts that assist learners with their performance of specific tasks that link processes to goals.

Support and Guidance

When learners start work on a new, more difficult task class, it is essential that they receive support and guidance for coordinating the different aspects of their performance. Support—actually *task support*—focuses on providing learners with assistance with the products involved in the training, namely the givens, the goals, and the solutions that get them from the givens to the goals (i.e., it is product oriented). Guidance—actually *solution-process guidance*—focuses on providing learners with assistance with the processes inherent to successfully solving the learning tasks (i.e., it is process oriented).

This support and guidance diminishes in a process of *scaffolding* as learners acquire more expertise. The continuum of learning tasks with high support to learning tasks without support is exemplified by the continuum of support techniques ranging from fully-reasoned case studies through partially worked out examples using the completion strategy (van Merriënboer, 1990; van Merriënboer & de Croock, 2002) to conventional tasks (for a complete description see van Merriënboer & Kirschner, 2007). In a training blueprint, each task class starts with one or more learning tasks with a high level of support and guidance (indicated by the grey in the circles), continues with learning tasks with a lower level of support and guidance, and ends with conventional tasks without any support and guidance as indicated by the filling of the circles:

Recurrent and Nonrecurrent Constituent Skills

Not all *constituent skills* are the same. Some are controlled, schema-based processes performed in a variable way from problem situation to problem situation. Others, lower in the skill hierarchy, may be rule-based processes performed in a highly consistent way from problem situation to problem situation. These constituent skills involve the same use of the same knowledge in a new problem situation. It might even be argued that these skills do not rely on knowledge at all, because this knowledge is fully embedded in the rules and conscious control is not required because the rules have become fully automated.

Constituent skills are classified as *nonrecurrent* if they are performed as schema-based processes after the training; nonrecurrent skills apply to the problem solving and reasoning aspects of behavior. Constituent skills are classified as *recurrent* if they are performed as rule-based processes after the training; recurrent skills apply to the routine aspects of behavior. The classification of skills as nonrecurrent or recurrent is important in the Ten Steps (van Merriënboer & Kirschner, 2007) because instructional methods for the effective and efficient acquisition of them are very different.

Supportive Versus Procedural Information

Supportive information is important for nonrecurrent constituent skills and explains to the learners how a learning domain is organized and how to approach problems in that domain. Its function is to facilitate *schema construction* such that learners can deeply process the new information, in particular by connecting it to already existing schemas in memory via *elaboration*. Because supportive information is relevant to all learning tasks within the same task class, it is typically presented before learners start to work on a new task class and kept available for them during their work on this task class. This is indicated in the L-shaped shaded areas in the schematic training blueprint:

schema-based seems to me like development of HEURISTICS vs. rule-based

This gives us a better definition of SCAFFOLDS - what is and what is NOT faded.

Procedural information is important for constituent skills that are recurrent; procedural information specifies for learners how to perform the routine aspects of learning tasks, preferably in the form of direct, step-by-step instruction. This facilitates rule automation, making the information available during task performance so that it can be easily embedded in cognitive rules via *knowledge compilation*. Because procedural information is relevant to the routine aspects of learning tasks, it is best presented to learners exactly when they first need it to perform a task (i.e., just in time), after which it quickly *fades* for subsequent learning tasks. In the schematic training blueprint, the procedural information (black beam) is linked to the separate learning tasks:

Part-Task Practice

Learning tasks provide whole-task practice to prevent compartmentalization and fragmentation. There are, however, situations where it may be necessary to include part-task practice in the training, usually when a very high level of automaticity is required for particular

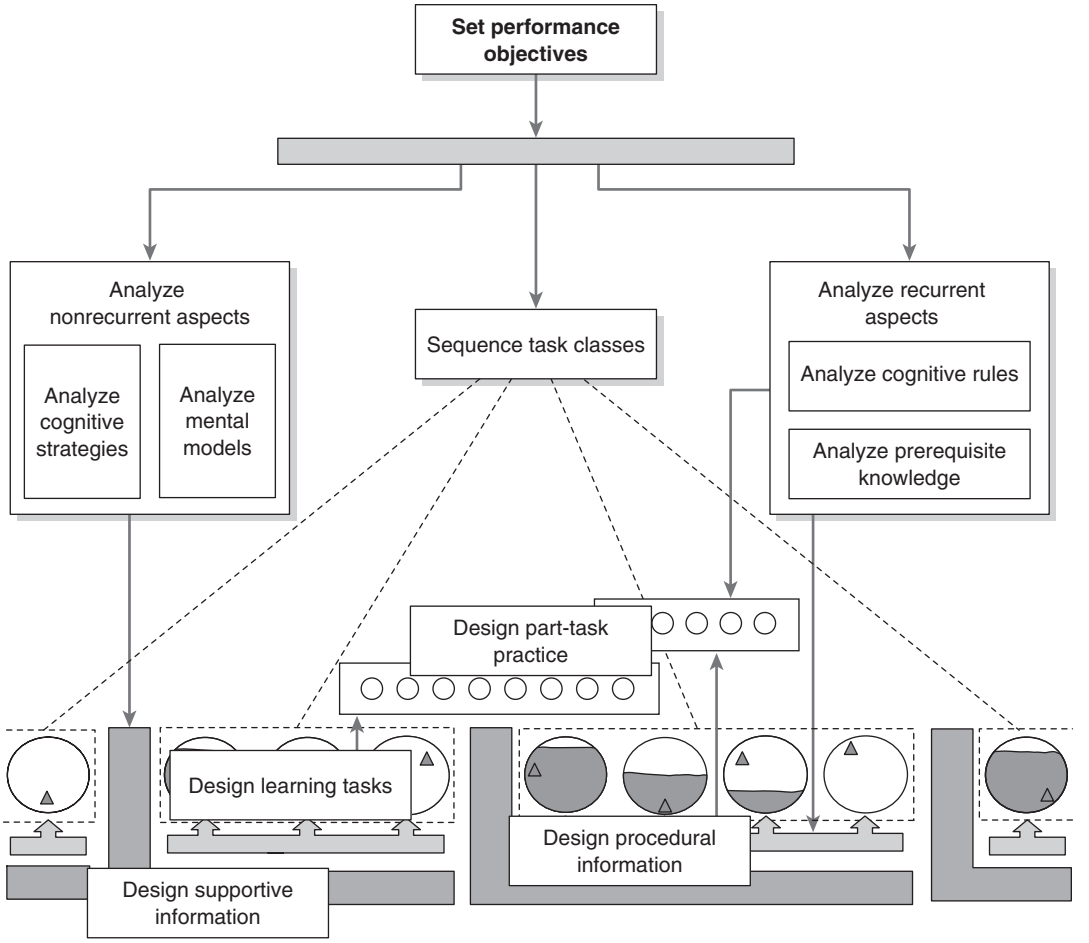
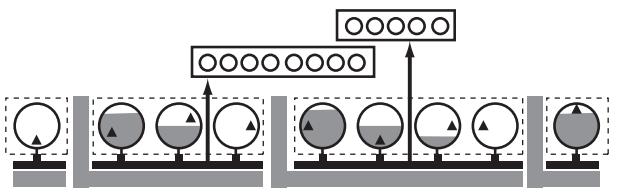


Figure 26.2 The Ten Activities (grey boxes) in Designing for Complex Learning

recurrent aspects of a task. In this case, the series of learning tasks may not provide enough repetition to reach that level. For those aspects classified as to-be-automated recurrent constituent skills, additional part-task practice may be provided—such as when children drill the multiplication tables or when musicians practice specific musical scales.

This part-task practice facilitates rule automation via a process called *strengthening*, in which cognitive rules accumulate strength each time they are successfully applied. Part-task practice for a particular recurrent aspect of a task can begin only after it has been introduced in a meaningful whole-learning task. In this way, learners start their practice in a fruitful cognitive context. In the schematic training blueprint, part-task practice is indicated by series of small circles (i.e., practice items):



Ten Steps

Figure 26.2 presents the whole design process for complex learning. The grey boxes show the ten activities that are carried out when properly designing training blueprints for complex learning. These activities are typically employed by a designer to produce effective, efficient, and appealing educational programs. This section explains the different elements in the figure from the bottom up.

The lower part of the figure is identical to what was just discussed. For each task class, learning tasks are designed to provide learners with variable whole-task practice at a particular difficulty level until they reach the *prespecified standards* for this level, whereupon they continue to the next, more complex or difficult task class. The design of supportive information pertains to all information that may help learners carry out the nonrecurrent problem solving and reasoning aspects of the learning tasks within a particular task class. The design of procedural information pertains to all information that exactly specifies how to carry out the recurrent, routine aspects of the learning tasks. And finally, the design of

Tasks performance objectives are equivalent to CRITERIA, which have to be performed to a specified STANDARD to be considered as demonstrating ability - i.e., level of competency.

part-task practice may be necessary for selected recurrent aspects that need to be developed to a very high level of automaticity.

The middle part of the figure contains five activities. The central activity—*sequence task classes*—describes an easy-to-difficult progression of categories of tasks that learners may work on. It organizes the tasks in such a way that learning is optimized. The least difficult task class is at the entry level of the learners and the final, most complex or difficult task class is at the final attainment level defined by the performance objectives for the whole training program.

The analyses of *cognitive strategies* and *mental models* are necessary for learners to achieve the nonrecurrent aspects of carrying out the task. The analysis of cognitive strategies answers the question, How do proficient task performers systematically approach problems in the task domain? The analysis of mental models answers the question, How is the domain organized? The resulting systematic approaches to problem solving and domain models are used as a basis for the design of supportive information for a particular task class.

The analyses of *cognitive rules* and *prerequisite knowledge* are necessary for learners to achieve the recurrent aspects of carrying out the task. The analysis of cognitive rules identifies the condition-action pairs that enable experts to perform routine aspects of tasks without effort (IF condition, THEN action). The analysis of prerequisite knowledge identifies what learners need to know to correctly apply those condition-action pairs. Together, the results of these analyses provide the basis for the design of procedural information. In addition, identified condition-action pairs help to specify practice items for part-task practice.

The upper part of the figure contains only one activity, *setting performance objectives*. Because complex learning deals with highly integrated sets of learning objectives, the focus is on the decomposition of a complex skill into a hierarchy describing all aspects or constituent skills relevant to performing real-life tasks. In other words, the specification of performance objectives and standards for acceptable performance for each of the constituent skills, and a classification of the skills within these objectives is either nonrecurrent or recurrent.

As indicated by the arrows, some activities provide preliminary input for other activities. This suggests that the best order for performing the activities would be to start with setting performance objectives, then to continue with sequencing task classes and analyzing nonrecurrent and recurrent aspects, and to end with designing the four blueprint components. Indeed, the ten activities have previously been described in this analytical order (e.g., van Merriënboer & de Croock, 2002). But in real-life design projects, each activity affects and is affected by all other activities. This leaves it an open question as to which order for using the ten activities is most fruitful.

A Dynamic Model

The model presented takes a *system dynamics* view of instruction, emphasizing the *interdependence* of the elements constituting an instructional system and recognizing the dynamic nature of this interdependence, which makes the system an irreducible whole. Such a systems approach is both systematic and systemic. It is *systematic* because the input-process-output paradigm where the outputs of particular elements of the system serve as inputs to other elements, and the outputs of particular design activities serve as inputs for other activities is inherent to it. For example, the output of an analysis is the input for the design of supportive information in the blueprint. At the same time, it is actually also *systemic* because the performance or function of each element directly or indirectly affects or is affected by one or more of the other elements—thereby making the design process highly dynamic and nonlinear. For example, this same analysis of nonrecurrent aspects of a skill can also affect the choice and sequencing of task classes.

The Pebble-in-the-Pond: From Activities to Steps

M. David Merrill (2002a) proposed a *pebble-in-the-pond* approach for instructional design that is fully consistent with the Ten Steps. It is a content-centered modification of traditional instructional design in which the contents-to-be-learned, and not the abstract learning objectives, are specified first. The approach consists of a series of expanding activities initiated by first casting a pebble in the pond; that is, designing one or more learning tasks of the type that learners will be taught to accomplish by the instruction. This simple little pebble initiates further ripples in the design pond. This prescriptive model is workable and useful for teachers and other practitioners in the field of instructional design.

this is SUPER interesting and goes against all the ideas of backward design, at least initially. BUT, it seems super intuitive for teachers.

A Backbone of Learning Tasks: Steps 1, 2, and 3

The first three steps aim at the development of a series of learning tasks that serve as the backbone for the educational blueprint:

Step 1: Design Learning Tasks
Step 2: Sequence Task Classes
Step 3: Set Performance Objectives

The first step, the pebble so to speak, is to specify one or more typical learning tasks that represent the whole complex skill that the learner will be able to perform following the instruction. Such a task has in the past been referred to as an *epitome*, the most overarching, fundamental task that represents the skill (Reigeluth, 1987; Reigeluth & Rodgers, 1980; Reigeluth & Stein, 1983). In this way, it becomes clear from the beginning, and at a very concrete

level, what the training program aims to achieve. Normally, providing only a few learning tasks to learners will not be enough to help them develop the complex skills necessary to perform the whole task. Therefore, another unique characteristic of the pebble-in-the-pond approach is—after casting the first whole learning task pebble into the pond—to specify a progression of such tasks of increasing difficulty such that if learners were able to do all of the tasks identified, they would have mastered the knowledge, skills, and attitudes that are to be taught. This ripple in the design pond, or step 2, involves the assignment and sequencing of learning tasks to task classes with different levels of difficulty. Tasks in the easiest class are at the learners' entry level, whereas tasks in the most difficult task class are at the training program's exit level. To give learners the necessary feedback on the quality of their performance and to decide when learners may proceed from one task class to the next, it is necessary to state the standards that need to be achieved for acceptable performance. This next ripple in the design pond, or step 3, consists of the specification of performance objectives that, among other things, articulate the standards that learners must reach to carry out the tasks in an acceptable fashion. In this way, the pebble-in-the-pond approach avoids the common design problem that the objectives that are determined early in the process are abandoned or revised later in the process to correspond more closely to the content that has finally been developed.

Component Knowledge, Skills, and Attitudes: Steps 4 to 10

Further ripples identify the knowledge, skills, and attitudes necessary to perform each learning task in the progression of tasks. This results in the remaining blueprint components, which are subsequently connected to the backbone of learning tasks. A distinction is made here between supportive information, procedural information, and part-task practice. The steps followed for designing and developing supportive information are as follows:

Step 4: Design Supportive Information
Step 5: Analyze Cognitive Strategies
Step 6: Analyze Mental Models

Units of supportive information that help learners perform the nonrecurrent aspects of the learning tasks related to problem solving and reasoning are connected to task classes, and more complex task classes typically require more detailed or more embellished supportive information than easier task classes. If useful instructional materials are already available, step 4 may be limited to reorganizing existing instructional materials and assigning them to task classes. Steps 5 and 6 may then be neglected. But if instructional materials need to be designed and developed from scratch, it may be helpful to perform step 5, where the cognitive strategies that proficient task-performers use to solve problems in the domain are analyzed, or step 6, where the mental models that describe how the domain is organized are analyzed. The results of the analyses in steps 5 and 6 provide the basis for designing supportive information. Analogous to the design and development of supportive information, steps 7, 8, and 9 are for designing and developing procedural information:

Step 7: Design Procedural Information
Step 8: Analyze Cognitive Rules
Step 9: Analyze Prerequisite Knowledge

Procedural information for performing recurrent aspects of learning tasks specifies exactly how to perform these aspects (and is thus procedural) and is preferably presented precisely when learners need it during their work on the learning tasks (i.e., just in time). For subsequent learning tasks, this procedural information quickly fades, often replaced by new specific information for carrying out new procedures. If useful instructional materials such as job aids, quick reference guides, or even Electronic Performance Support Systems (EPSSs; van Merriënboer & Kester, 2005) are available, step 7 may be limited to updating those materials and linking them to the appropriate learning tasks. Steps 8 and 9 may then be neglected. But if the procedural information needs to be designed from scratch, it may be helpful to perform step 8, where the cognitive rules specifying the condition-action pairs that drive routine behaviors are analyzed, and step 9, where the knowledge that is prerequisite to a correct use of cognitive rules is analyzed. The results of the analyses in steps 8 and 9 then provide the basis for the design of procedural information. Finally, depending on the nature of the task and the knowledge and skills needed to carry it out, it may be necessary to perform the tenth and final step:

Step 10: Design Part-Task Practice

Under particular circumstances, additional practice is necessary for selected recurrent aspects of a complex skill in order to develop a very high level of automaticity. This, for example, may be the case for recurrent constituent skills that cause danger to life and limb, loss of expensive or hard to replace materials, or damage to equipment if not carried out properly and quickly. If part-task practice needs to be designed, the analysis results of step 8 (i.e., the condition-action pairs) provide useful input. For a detailed description of the Ten Steps see van Merriënboer and Kirschner (2007).

Ten Steps Within an Instructional Systems Design Context

The Ten Steps will often be applied in the context of *Instructional Systems Design* (ISD). ISD models have a broad scope and typically divide the instructional design

process into five phases: (a) analysis, (b) design, (c) development, (d) implementation, and (e) summative evaluation. In this so-called ADDIE model, formative evaluation is conducted during all of the phases. The Ten Steps is narrower in scope and focus on the first two phases of the instructional design process, namely, task and content analysis and design. In particular, the Ten Steps concentrates on the analysis of a to-be-trained complex skill or professional competency in an integrated process of task and content analysis and the conversion of the results of this analysis into a training blueprint that is ready for development and implementation. The Ten Steps is best applied in combination with an ISD model to support activities not treated in the Ten Steps, such as needs assessment and needs analysis, development of instructional materials, implementation and delivery of materials, and summative evaluation of the implemented training program.

References and Further Readings

Andre, T. (1997). Selected micro-instructional methods to facilitate knowledge construction: Implications for instructional design. In R. D. Tennyson, F. Schott, N. Seel, & S. Dijkstra (Eds.), *Instructional design—International perspectives: Theory, research, and models* (Vol. 1, pp. 243–267). Mahwah, NJ: Lawrence Erlbaum Associates.

Clark, R. E., & Estes, F. (1999). The development of authentic educational technologies. *Educational Technology, 39*(2), 5–16.

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 Associates.

Gardner, H. (1999). Multiple approaches to understanding. In C. M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II, pp. 69–89). Mahwah, NJ: Lawrence Erlbaum Associates.

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. 215–239). Mahwah, NJ: Lawrence Erlbaum Associates.

Kirschner, P. A., Carr, C. S., van Merriënboer, J., & Sloep, P. (2002). How expert designers design. *Performance Improvement Quarterly, 15*(4), 86–104.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 46*(2), 75–86.

McCarthy, B. (1996). *About learning.* Barrington, IL: Excell Inc.

Merrill, M. D. (2002a). A pebble-in-the-pond model for instructional design. *Performance Improvement, 41*(7), 39–44.

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

Merrill, P. (1980). Analysis of a procedural task. *NSPI Journal, 17*(2), 11–26.

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 Associates.

Paas, F., & van Merriënboer, J. J. G. (1994). Variability of worked examples and transfer of geometrical problem solving skills: A cognitive-load approach. *Journal of Educational Psychology, 86,* 122–133.

Perkins, D. N., & Grotzer, T. A. (1997). Teaching intelligence. *American Psychologist, 52,* 1125–1133.

Quilici, J. L., & Mayer, R. E. (1996). The role of examples in how students learn to categorize statistics word problems. *Journal of Educational Psychology, 88,* 144–161.

Ragan, T. J., & Smith, P. L. (1996). Conditions theory and models for designing instruction. In D. Jonassen (Ed.), *Handbook of research on educational communications and technology* (2nd ed., pp. 623–650). Mahwah, NJ: Lawrence Erlbaum Associates.

Reigeluth, C. M. (Ed.). (1987). *Instructional theories in action: Lessons illustrating selected theories and models.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Reigeluth, C. M., & Rodgers, C. A. (1980). The elaboration theory of instruction: A model for structuring instruction. *Instructional Science, 9,* 125–219.

Reigeluth, C. M., & Stein, F. S. (1983). The elaboration theory of instruction. In C. M. Reigeluth (Ed.), *Instructional design theories and models: An overview of their current status* (pp. 335–381). Hillsdale, NJ: Lawrence Erlbaum Associates.

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 Associates.

Schilling, M. A., Vidal, P., Ployhart, R. E., & Marangoni, A. (2003). Learning by doing something else: Variation, relatedness, and the learning curve. *Management Science, 49,* 39–56.

Schwartz, D., Lin, X., Brophy, S., & Bransford, J. D. (1999). Toward the development of flexible adaptive instructional designs. In C. M. Reigeluth (Ed.), *Instructional design theories and models: A new paradigm of instructional theory* (Vol. II, pp. 183–213). Mahwah, NJ: Lawrence Erlbaum Associates.

Spector, J. M., & Anderson, T. M. (Eds.). (2000). *Holistic and integrated perspectives on learning, technology, and instruction: Understanding complexity.* Mahwah, NJ: Lawrence Erlbaum Associates.

Sweller, J., Kirschner, P. A., & Clark, R. E. (2007). Why minimal guidance during instruction does not work: A reply to commentaries. *Educational Psychologist, 47*(1), 115–121.

van Merriënboer, J. J. G. (1990). Strategies for programming instruction in high school: Program completion vs. program generation. *Journal of Educational Computing Research, 6*, 265–285.

van Merriënboer, J. J. G. (1997). *Training complex cognitive skills: A four-component instructional design model for technical training.* Englewood Cliffs, NJ: Educational Technology Publications.

van Merriënboer, J. J. G. (2000). The end of software training? *Journal of Computer Assisted Learning, 16,* 366–375.

van Merriënboer, J. J. G. (2007). Alternate models of instructional design: Holistic design approaches and complex

learning. In R. A. Reiser & J. Dempsey (Eds.), *Trends and issues in instructional design and technology* (2nd ed., pp. 72–81). Upper Saddle River, NJ: Merrill/Prentice Hall.

van Merriënboer, J. J. G., Clark, R. E., & de Croock, M. B. M. (2002). Blueprints for complex learning: The 4C/ID-model. *Educational Technology Research and Development, 50*(2), 39–64.

van Merriënboer, J. J. G., & de Croock, M. B. M. (1997). Strategies for computer-based programming instruction: Program completion vs. program generation. *Journal of Educational Computing Research, 8,* 365–394.

van Merriënboer, J. J. G., & de Croock, M. B. M. (2002). Performance-based ISD: 10 steps to complex learning. *Performance Improvement, 41*(7), 33–38.

van Merriënboer, J. J. G., & Kester, L. (2005). The four-component instructional design model: Multimedia principles in environments for complex learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 71–93). New York: Cambridge University Press.

van Merriënboer, J. J. G., Kester, L., & Paas, F. (2006). Teaching complex rather than simple tasks: Balancing intrinsic and germane load to enhance transfer of learning. *Applied Cognitive Psychology, 20*, 343–352.

van Merriënboer, J. J. G., & Kirschner, P. A. (2001). Three worlds of instructional design: State of the art and future directions. *Instructional Science, 29,* 429–441.

van Merriënboer, J. J. G., & Kirschner, P. A. (2007). *Ten steps to complex learning.* New York: Taylor & Francis.

van Merriënboer, J. J. G., Kirschner, P. A., & Kester, L. (2003). Taking the load of a learner's mind: Instructional design for complex learning. *Educational Psychologist, 38*(1), 5–13.

van Merriënboer, J. J. G., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17,* 147–177.

van Merriënboer, J. J. G., & van Dijk, E. M. A. G. (1998). Use and misuse of taxonomies of learning: Dealing with integrated educational goals in the design of computer science curricula. In F. Mulder & T. van Weert (Eds.), *Informatics in Higher Education* (pp. 179–189). London: Chapman and Hall.

Wightman, D. C., & Lintern, G. (1985). Part-task training for tracking and manual control. *Human Factors, 27,* 267–284.