# PATTERNS FOR GAINING DIFFERENT PERSPECTIVES

**A part of the Pedagogical Patterns Project pattern language**

**Version 0.6**

Joe Bergin (berginf@pace.edu)
Jutta Eckstein (jeckstein@acm.org)
Mary Lynn Manns (manns@unca.edu)
Eugene Wallingford (wallingf@cs.uni.edu)



## Introduction

Learning technical material can be difficult. In order to learn well, a student must be motivated. At least three factors directly affect motivation:

- a suitable learning environment
- appropriate instructional techniques
- relevant content

A good plan of instruction exhibits patterns that seek to bring each of these factors into play at just the right time to enhance student learning.

The Pedagogical Patterns Project (www.pedagogicalpatterns.org) consists of an international cadre of instructional designers from academia and industry who seek to document the proven patterns of effective instruction and learning. Their work treats pedagogy in its broadest sense, as "systematized learning or instruction concerning principles and methods of teaching." (Webster's New Collegiate Dictionary) Early efforts in this project focused on mining patterns from instructors with diverse experience teaching object-oriented techniques, resulting in a large body of potential patterns. In recent years, those involved in the project have begun to turn their attention to culling the patterns, finding common forces and solutions, and refactoring the patterns into a language of instructional design. The paper "Patterns of Experiential Learning" [JE3], workshopped at EuroPLoP 2001, focused on patterns dealing with abstraction and failure.

This small pattern language presented in this paper consists of patterns that deal with the diversity of instructional techniques. Different learners learn differently, and so the effective instructor must be able to help students encounter material in different ways. We believe that these techniques apply to teaching many topics, but our direct experience with them is in teaching object-oriented techniques, so we have retained that focus throughout the paper.

## The Patterns

This paper consists of the following patterns:

1. Different Approaches
2. Consistent Metaphor
3. Physical Analogy
4. Role Play
5. Reflection
6. Explore for Yourself
7. Spiral
8. Linking Old to New
9. Test Tube

Pattern 1 is the primary entry point into the language, the pattern that recognizes the importance of different learning styles, including visual, auditory, and kinesthetic.

---

### DIFFERENT APPROACHES

This pattern was written by Astrid Fricke and Markus Voelter [AF] and revised by Jutta Eckstein.

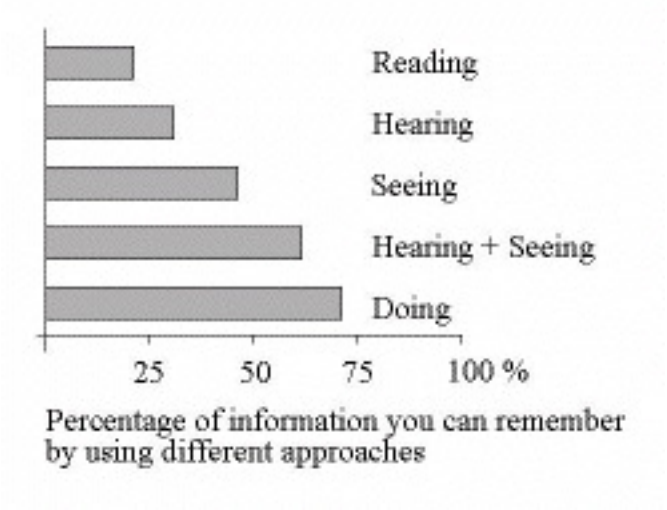
**Communication always takes place between a sender and a receiver, and the effectiveness of communication isn't measured by what the sender says, but by what the receiver understands. Every person obtains information differently, using different sensory modalities. Some people, the visuals, learn most effective by watching; the auditories, by listening; and the kinesthetics, through action. Be aware: Not every student uses the same sensory modality as you!**

φφφ

Therefore, provide different approaches to the same topic. Accept different learning styles by addressing various sensory modalities. It might be difficult to provide different approaches for every single topic, but make sure to at least change the approach when you change the topic, as described in NEW PEDAGOGY FOR NEW PARADIGMS [JB4].

Changing approaches also helps to keep the students' interest, because they don't become too used to a specific style and because you challenge them by addressing other skills. Furthermore, you could use a combination of several modalities, e.g., show transparencies while explaining the topic verbally. The more perception areas in the brain that are activated, the more opportunities the learner has to make associations and to develop a deeper understanding.

At the minimum, be sure to visualize important topics, because an average human being takes in over 83% of her overall knowledge with the eyes.




φφφ

*To better accommodate the auditories, consider the following questions to make a lecture more understandable:*

- *What is this lecture about? Provide an example.*
- *What are the key points? Explain the big picture.*
- *What does this mean exactly? Explain the details.*

*Value the visuals by providing a road map for visualizing the big picture. Emphasize the key points, by writing them on a white board or flip chart.*

*You could acknowledge the needs of the kinesthetics by using ROLE PLAY, simulations, or games as possible actions.*

## CONSISTENT METAPHOR (AKA ANALOGY)

This pattern was written and revised by Joseph Bergin [JB1].

**It is easy to get lost in the details of the current topic, especially when teaching beginning students. Students then may not see how this topic is related to larger goals. It is also difficult for many students to quickly see how things fit together and to make correct predictions about how the technology should behave. We would like to let students draw on what they already know to help them learn new material.**

You may be teaching a complex topic, such as object-oriented thinking. The topic may have many parts, some of which are quite detailed. The students need a way to think of the topic as a whole, but the topic may be highly technical and outside their experience.

φφφ

Therefore, create a metaphor that is consistent with the topic being taught, and with the same basic elements that interact in the same way. Give this to the students as a way to think about the topic. The metaphor must allow students to make valid inferences about the topic by thinking about the metaphor.

If you have good metaphors, you should be able to draw correct inferences by referring to the metaphor. Tell the students about the relationship between the metaphor and the topic of interest.

φφφ

You want to give your students a powerful and consistent shorthand for thinking about some complex topic. The shorthand should relate the topic being taught to things within their experience.

Students may get lost in the detail easily and fail to see the big picture and how the parts relate to each other. This is especially true when the details themselves are unfamiliar and new to them.

It is helpful when learning new topics to relate new ideas to already familiar ideas.

It may take you a long time to teach all of the elements of the topic under review. (See SPIRAL for more on this issue.)

You want students to have an idea about what is happening within a system that permits them to make valid inferences about what should happen.

In teaching object systems, a good metaphor is human beings. People are autonomous, encapsulated actors, who interact with other similar objects.

However, always be aware of the limits of your metaphor. The "people as objects" metaphor breaks down somewhat in message passing as the object that receives a message doesn't necessarily have a reference with which to communicate back to the sender, though people usually do.

φφφ

*A good example of the use of this pattern came out of a workshop that was held at OOPSLA'97 on Non-Software Examples of Design Patterns [MD]. The results can be used to motivate a number of ideas of software design.*

## PHYSICAL ANALOGY

This pattern was written by Phil McLaughlin [PM] and revised by Mary Lynn Manns.

**You are trying to help learners understand the dynamic qualities in a rather abstract concept. You have provided an overview of the concept, and now would like to help students visualize how it works. While it is rather easy for learners to comprehend concepts that are concrete because they are usually easy to visualize, it is not as easy to do this with abstract concepts.**

For example, learners can comprehend code that is executed in a linear top-down procedure because they can follow the code and visualize what is happening as each statement is executed. However, object-oriented code is not executed in a top-down way. In addition, other principles such as the private nature of an object's data can be rather obscure for students to comprehend. But, if the instructor does not take special effort to solidify important principles such as these, this can cause the learner to ignore these principles and write procedure-oriented object programs.

φφφ

Therefore, illustrate the dynamic properties of the abstract concept in a concrete way. Create a physical analogy with the use of visual things such as inanimate objects or people and/or other memorable things such colorful scenarios.

For example, you can illustrate the calling chain in a series of messages as they are executed from one object to the next. To do this, tie a large amount of string to a ball or similar item. Ask each learner to represent an object (give each some kind of a tag to display with the message name). Explain a scenario that could be executed with this collection of objects. Begin to execute this scenario by throwing the ball to the person who has the object that contains the first method which will be executed. This person will then pass the ball to the object with the next method to be executed. Continue the ball passing in this way while, in each case, the "object" should keep part of the string before passing the ball. Continue until the scenario is completed. This physical analogy will illustrate the dynamic nature of message passing while the string that connects everyone along the way will illustrate the path, the calling chain, that was taken through the object-oriented code.

People remember much more of what they see than what they hear or read. Use of an interesting physical analogy will make it more likely that learners will remember a concept because they are able to visualize it in a more lifelike way.

φφφ

*Karen K. Brown, formally of ObjectSpace, reinforces the use of accessors and mutators in this way. The instructor asks a learner (named Greg for purposes of illustration here), "Greg, what did you eat for breakfast?" Greg may answer, "A bagel and coffee." The instructor points out that Greg's public interface has been properly invoked, and Greg is induced to answer in an acceptable format, while hiding his implementation. The instructor then describes that the visceral alternative is to put one's arm down Greg's throat and grab a piece of the bagel. That violates the privacy of Greg's data, bypassing his public interface. Obviously, it is equally hard for Greg to obtain or modify this information without also using accessing methods. Throughout the class, anytime a student attempts to bypass using accessors and mutators, the instructor need only remind them that they are reaching down Greg's throat. (This has been used only in instances when the learners have demonstrated a sense of humor.)*

## ROLE PLAY

This pattern was written by Jutta Eckstein, based on ROLE PLAYING by David Bellin [DB] and INCREMENTAL ROLE PLAY by Jutta Eckstein [JE2].

**The complexity of some concepts makes them hard to understand with only abstract explanations. Furthermore, difficulties in understanding complex concepts may frustrate the students. You not only would like to provide a positive learning environment, so even learning complex topics might be fun, but you also want to**

**take into account that different people learn things best using different sensory modalities. Most teaching styles respect the auditories, a few the visuals, and even fewer the kinesthetics.**

φφφ

Therefore, invite your students to behave as a part of the concept involved in a role play. Every student plays one part of the concept to get a deeper knowledge for its underlying structure. Students see how the different parts of the concepts are all working together to solve a bigger problem.

Role play gives the students the possibility to see what they understand and simultaneously confronts them with their lack of knowledge. If they understand their roles well, then they will know how and with whom to interact. To further encourage this understanding, allow the students to switch roles occasionally, in order to enable different point of views of the system.

Making learning difficulties obvious is also the major drawback of using role play. Not all students enjoy learning in public, which can be an especially difficult political issue when the participants belong to the same organization.

Because role play is almost always implemented by a team of players, it also builds on human interaction and social skills. Role play has a significant beneficial side effect: by using this kind of active learning, the students do not get as bored or tired as quickly. But as with most active learning techniques, role play can require a lot of time. If this is an issue, then you might prefer to use a CONSISTENT METAPHOR instead.

φφφ

*When teaching Enterprise Java Beans, assign a role of the objects involved to the students. For example, one student might act as the Client, one as the Container, and one as the Bean. The script is one scenario, e.g., asking an EJB to fulfill a specific service. Messages can be sent back and forth by throwing a soft ball between the students. Each student can only act according to the interface of "her" object. Also, messages can only be sent to objects one is connected to. This is best supported by handing a CRC card to every student, where each CRC card illustrates both the responsibilities and the limitations.*

*Everyone—students with or without a role as well as the trainer—has the responsibility to watch for awkward communication and missing communication paths.*

*You could take this even further by having competitions between different teams, where the playing team will be observed by the other team. When the observing team detects a miscommunication, it scores a point.*

## REFLECTION

This pattern was written by Jutta Eckstein.

**Sometimes, learners believe that the trainer has to deliver all the knowledge, but the students themselves are knowledgeable. Furthermore, students often anticipate that an instructor will solve each and every problem for them, but the knowledge of the instructor is also limited. You want students to trust their own competency rather than just letting them accept what they have learned by listening passively.**

φφφ

Therefore, provide an environment that allows discovery and not one that is limited to answering questions. Let the students uncover solutions for complex problems by drawing on their own experience. It is the students' debt of delivery or of inquiry. Train students so that they are searching for solutions by exploring the problem.

Redirect students to use their own intellect. Students have to find answers for themselves. Even if a student comes up with a question that you cannot answer, you can challenge the students by explaining to them that if they are clever enough to come up with such an interesting question, they will be also able to come up with a good answer for it.

Make what is learned obvious. Point out the things the participants have learned and ask them how they imagine they can use it in their work environment. Ask the students to discover the differences and commonalities to their experience.

φφφ

When redirecting students to use their own intellect, let them know that you are interested in the answer. This way, you admit that you do not know the answer, but you also underline how difficult and interesting the question is.

Young adults who enter industrial training may not realize that their learning style has matured beyond the pedagogical learning style they became accustomed to in school. They might not know that they can benefit greatly from a learning style that better matches an adult level of cognitive development.

Adults are competency-based learners, meaning that they want to learn a skill or acquire knowledge which they can apply pragmatically in their current circumstances.

φφφ

*EXPLORE FOR YOURSELF shows one way in which students can learn by themselves. ROUND AND DEEP [HS] exploits the students' experiences to complement the trainer's.*

*At the Educators Symposium at OOPSLA '96, Bruce Anderson [BA] used this pattern in the Activity Session. The task was to compare different C++ implementations. Although hardly anybody understood the code in detail, the teams were still able to draw on some other experiences and evaluate the code.*

## EXPLORE FOR YOURSELF

This pattern was written by Jutta Eckstein, based on CLASS CONCEPT MAP by Jeanine Meyer [JM1] , EXPLORE-PRESENT-INTERACT-CRITIQUE by Jorgen Lindskov Knudsen and Ole Lehrmann [KL], and CHALLENGE by Jutta Eckstein [JE1].

**A person's success is based mainly on her ability to learn new concepts efficiently and to act as a team player by sharing knowledge and insights. You want to give your students the ability to learn in the future and to communicate their wisdom, but students are often afraid of taking responsibility for their own learning.**

φφφ

Therefore, assign topics to the students that they have to learn on their own and ask them to present the topic afterwards. It is helpful to provide hints for resources related to the topic.

Presenting a concept to somebody else is an important part of the learning process. A student who is able to explain a concept to someone else understands the better herself.

φφφ

Often students are irritated or uncertain about how to progress. Some people aren't able to handle a situation like this at all. If this is the case, then you have to provide more specific hints, so that students are able to overcome their own uncertainty and handle the situation for themselves. The difficulty for you is to find a balance

between providing too much structure (so that students do not explore for themselves anymore) and *laissez-faire*.

When starting to use this approach, you might be forced to tell the solution at one point, because the students do not yet trust their own knowledge. If this is the case, make sure that the students reflect afterwards on whether they would have been able to uncover the solution on their own [JM2]. Ask students to consider the following questions: Should I have known this? Have I done it already? Is it similar to something else? Is this a combination of two or more things I knew?

Sollmann [SH] reports that the more often students are in this uncertain situation the better they can handle this kind of situation, or rather the longer it takes before they feel uncertain. It is like in a physical training setting; the more you overcome your limits, the higher your limits will become.

φφφ

*Jeanine Meyer used this patterns for teaching technical vocabulary [JM1]. At Aarhus University in Denmark [KL] this pattern was used to introduce various object-oriented concepts, among them garbage collection, CORBA, and OODB.*

## SPIRAL

This pattern was written and revised by Joseph Bergin [JB5].

**Topics in a course are often interrelated. Too often, many different topics are required for students to have enough tools with which to solve interesting problems. If we try to do the topics in any logical order we tend to get bogged down in details and leave the students bored.**

**You want to enable students to solve meaningful problems as early in the course as possible. Students learn best when they are doing things, and meaningful problems motivate them to work harder.**

φφφ

Therefore, organize the course to introduce topics to students without covering them completely at first viewing so that a number of topics can be introduced early and then used. In the first cycle make each topic introduction as simple as possible without leaving out essential details. Cover several topics quickly. This can get students working on interesting problems earlier as they have more tools to use,

though they have not, perhaps, mastered any of the tools. The instructor can then return to each topic in turn, perhaps repeatedly, giving more of the information needed to master them.

On each cycle of the spiral, cover old topics in more depth and include additional topics. The sequencing of the "fragments" is done with an eye to providing students with problem solving skills. Anthony's MIX NEW AND OLD [DA] suggests the importance of mixing new material into what is already known. The course cycles around to a given topic several times during the term, each time to a greater depth. This also provides reinforcement of key ideas.

You need a plan showing the order in which the topics will be introduced and what will be deferred to later cycles. You must also extract subsets of each of the many topics in which the tools introduced can work together in problem solving. Several, increasingly large, subsets must be designed. Problems using most of the features of each subset need to be designed.

One way to design the subsets is to start with the problem and extract a minimal set of tools necessary to solve that problem and similar problems. The next larger subset can often be designed by thinking about how the original problem could be expanded and its solution generalized.

Many fields can only be mastered by individually mastering a large number of different techniques that must interact. Large topics such as design and programming require many parts and much detail to master. Developing these in a sequential manner leaves the students without interesting exercises, as they have not seen enough of the breadth of the topic to do interesting things.

Students like to build things and they like to see how the pieces fit together. They get bored easily if instruction is repetitive and if the instructor spends too much time on one topic or a set of closely related topics. Students can also get bored if exercises are artificially contrived to illustrate arcane details.

Courses do not need to be organized like reference material. Nor should textbooks be. This pattern results in the topics of a course being more fine grained, with just enough of a larger topic introduced at each stage to permit problem solving with other tools/topics which are also, as yet, not completely covered. For example, "iteration" is a big topic. The "while loop" is a small topic, especially if initially introduced with only simple loop tests. Yet the while loop can be used effectively with other constructs to build meaningful programs before iteration is understood in all of its aspects.

To start, the instructor should extract a subset of the material covering several topics that interact. Only simple cases should be introduced at first. The instructor and

class move quickly through the topics until an understanding of how the topics interact can be gained. Students then can work with the tool subset on problems. Then more of each of the original topics, with perhaps simple cases of new topics are introduced to deepen understanding of the topics and of their interactions. Students then work on a richer set of problems. This can be repeated as often as necessary.

Students will get a feel earlier for how the pieces fit together.

However, a potential negative consequence for some students at least is that some of their questions (What if...?) may need to be deferred. You may be able to use TEST TUBE or EXPLORE FOR YOURSELF as a way to overcome this somewhat.

## LINKING OLD TO NEW

This pattern was written by Jutta Eckstein.

**Learning something new is exciting, but it often involves questioning things that the learner already knows. Learning too many new things often leads to a sense of rejection and then to stress.**

φφφ

Therefore, use an old wrapper to introduce new information. This will help the learner to recognize what she already knows and to make associations between the new information and existing knowledge.

Anchor the new information by relating it to what is already known. Connecting and complementing old with new allows the learner's mind to reorganize its knowledge structures. This is sometimes also referred to as the "Advance Organizer", which activates prior learning for the purpose of acquiring new knowledge.

Our brain remembers things through associations with existing memory. This is one reason that human memory is often compared to a map, which reflects its knowledge in the form of a network.

φφφ

*Tim DeClue uses this pattern to introduce object technology and C++ [TDC].*

## TEST TUBE

This pattern was written by Joe Bergin [JB6] and revised by Eugene Wallingford.

**When students encounter holes in their knowledge, we would like for them to seek out an answer. Unfortunately, students often resort immediately to the "easy fix" of asking an authority for the answer. We want students to ask questions, but sometimes they have available to them more effective ways to gain knowledge that they never consider.**

Students are often naive about what resources are available to them. They will ask questions that they could answer for themselves. This is not bad in itself, but what about times when no authority is available to them? What if the instructor is unavailable? What if the documentation is incomplete or missing? Students dependent on asking questions will be stymied unnecessarily in such circumstances. They can also become frustrated when they cannot find sufficiently detailed documentation to answer their questions.

Computer science is unlike many other disciplines. We can write programs and watch their behavior. By observing the behavior, we can infer the answers to many questions about the programming language, about the compiler or interpreter, and about different programming constructs. Often, this sort of experimentation is a quicker and more effective way to learn than by asking a question or reading formal documentation. More importantly, it prepares students to learn independently in a wide variety of situations. Students feel liberated when they learn they can find answers to their own questions quickly.

φφφ

Therefore, give the students exercises in which they are asked to write small programs that us the computer to answer simple questions of the form "What happens if ...?". Make these exercises frequent enough that students develop the habit of probing the machine for what it does, rather than asking a question or seeking out documentation.

Student questions can be a good source for exercises of this kind. Take a student question and turn it into a quick exercise. This may be overnight between classes, but if a laboratory is available, students can immediately test out hypotheses about how things operate.

This sort of informal Test Tube works well even if—especially if—the instructor knows the answer to a question. You want your students to begin to realize that they can learn independently, even when external references such as instructors and manuals, by exploring in their computing laboratory.

This technique works well when students are learning a programming languages. Students in this circumstance can take up a large amount of class time asking questions at a very low level of detail. This time is better spent on larger issues involving the construction of programs. Turn as many "What if ...?" syntax questions into experiments for students to do in the laboratory.

Beware, though, of frustrating students who do not know enough of the language to experiment effectively. Sometimes, you will need to address detailed questions of syntax in class in order to prepare students sufficiently to experiment on their own.

This technique also works well for languages are under-defined. With these languages, the instructor will need to point out situations where there are no rules. For example, the order of evaluation for parameters in C++ is not defined in the standard. The most a program can tell a student here is how the compiler that they are using behaves. It cannot give you a general rule because there is none. Make your students aware of these situations so that they can program defensively and not draw misleading conclusions from their lab work.

φφφ

If the purpose of a TEST TUBE exercise is to generate errors, then it is an instance of MISTAKE [JE3]. However, this pattern is more general than Mistake, because it allows students to learn positive examples, too.

TEST TUBE is also a way for us to ask learners to EXPLORE FOR YOURSELF. But whereas EXPLORE FOR YOURSELF tends to focus on larger-grained topics and on the advantages of presenting what has been learned to others, TEST TUBE works best for smaller-scale questions and when the student is learning in "real time".

This technique can be used in conjunction with FIXER UPPER [JE2]. Students can experiment with different ways to improve the target program and learn by observing the results.

This can be used effectively to make cycles around SPIRAL move quickly without getting bogged down in too much detail. Move the as many details as possible into exercises for students to do at the computer, and use your class time to move forward.

φφφ

*The meaning of for loops in C++ can be explored in a sequence of exercises in which the initialization, test, and "increment" portions of the loop are varied.*

*When a while loop exits (between executions of the body) and when it does not (as soon as the condition becomes true in the middle of the body) can be explored in exercises.*

*The difference between value and reference parameters and the meaning of* `const` *can be explored.*

*Whenever a student asks, "What happens if I do this in my program?", ask the student to do that and report results back to the group.*

---

## Thumbnails

These patterns are not part of our language, but they are referred to by one or more patterns above.

**FIXER UPPER** [JB2]

We want students to work on large artifacts without overwhelming them. Therefore, give them an artifact, such as a program or design, into which the instructor has purposely introduced flaws. Ask students to find and correct the flaws. Ask them to discuss the nature of the flaws found and the reasons for their changes. Finally, ask them to discuss the overall structure of the artifact and draw inferences from it.

**MISTAKE** [JB3]

People make mistakes. Students often don't know how to interpret the error messages provided by their tools or what to do to solve problems that are diagnosed by the tools. Therefore, ask students to produce an artifact with a specific error. Then explore the nature of the error, its effect on the system, and how to find and fix such errors.

**NEW PEDAGOGIES FOR NEW PARADIGMS** [JB4]

Some topics that you need to teach require different modes of thinking on the part of learners. Be sure to use a different pedagogy—not just different examples—to teach the new topic. You need to learn this pedagogy, either by developing it yourself (in the case of state-of-the-art topics) or by borrowing it from those who have been teaching this topic effectively.

**ROUND AND DEEP** [HS]

An experienced student is most likely to gain a deep understanding of a complex concept by relating it to his or her own experience. But the same experience which results in a deep understanding may also limit it because a round and deep understanding of a complex concept can only be achieved by considering different perspectives.

## Acknowledgments

We thank all those who have contributed to the Pedagogical Patterns project, especially the authors of the patterns we have revised and incorporated into this paper. We also thank our PLoP 2001 shepherd, Dick Gabriel, for helping us to improve the paper.

## References

[BA] Bruce Anderson, *Reflective Learning: Empowering the OO Practitioner*, at the Educators Symposium, OOPSLA'96.

[DA] Dana Anthony, MIX NEW AND OLD, in "Patterns for Classroom Education", *Pattern Languages of Program Design 2*, edited by Vlissides, Coplien, and Kerth, Addison Wesley, 1996.

[DB] David Bellin, ROLE PLAYING, http://www-lifia.info.unlp.edu.ar/ppp/pp5.htm

[JB1] Joseph Bergin, CONSISTENT METAPHOR, http://csis.pace.edu/~bergin/PedPat1.3.html#consistentmetaphor

[JB2] Joseph Bergin, FIXER UPPER, http://csis.pace.edu/~bergin/PedPat1.3.html#FixerUpper

[JB3] Joseph Bergin, MISTAKE, http://csis.pace.edu/~bergin/PedPat1.3.html#mistake

[JB4] Joseph Bergin, NEW PEDAGOGY FOR NEW PARADIGMS, http://csis.pace.edu/~bergin/patterns/fewpedpats.html#unp

[JB5] Joseph Bergin, SPIRAL, http://www-lifia.info.unlp.edu.ar/ppp/pp32.htm

[JB6] Joseph Bergin, TEST TUBE,
http://csis.pace.edu/~bergin/PedPat1.3.html#testtube

[TDC] Tim DeClue, THE GAGNE'-AUSUBEL PATTERN,
http://www-lifia.info.unlp.edu.ar/ppp/pp19.htm

[MD] Michael Duell, "Non-Software Examples of Design Patterns",
http://www.agcs.com/supportv2/techpapers/patterns/papers/tutnotes/

[JE1] Jutta Eckstein, CHALLENGE, in *Proceedings of EuroPLoP 2000*, edited by Martine Devos and Andreas Rueping, UKV, Universitaetsverlag Konstanz, 2001.

[JE2] Jutta Eckstein, INCREMENTAL ROLE PLAY, in *Proceedings of EuroPLoP'98*, edited by Jens Coldewey and Paul Dyson, UKV, Universitaetsverlag Konstanz, 1999.

[JE3] Jutta Eckstein, Klaus Marquardt, and Markus Völter, "Patterns of Experiential Learning", in *Proceedings of EuroPLoP'01*, 2001.

[AF] Astrid Fricke and Markus Voelter, "Seminars: A Pedagogical Pattern Language About Teaching Seminars Effectively", in *Proceedings of EuroPLoP 2000*, 2000.

[KL] Jorgen Lindskov Knudsen, and Ole Lehrmann, EXPLORE-PRESENT-INTERACT-CRITIQUE, http://sol.info.unlp.edu.ar/ppp/pp11.htm

[PM] Phil McLaughlin, PHYSICAL ANALOGY,
http://www-lifia.info.unlp.edu.ar/ppp/pp16.htm

[JM1] Jeanine Meyer, CLASS CONCEPT MAP,
http://www-lifia.info.unlp.edu.ar/ppp/pp38.htm

[JM2] Jeanine Meyer, email correspondence, July 12, 2001.

[HS] Helen Sharp, ROUND AND DEEP,
http://www-lifia.info.unlp.edu.ar/ppp/pp59.htm

[SH] Sollman and Heinze, *Visionsmanagement: Erfolg als vorausgedachtes Ergebnis (Vision Management: Success as a pre-thought result)*, Orell Füssli, 1994.

.