DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Project Based Learning + Agile Instructional Design = EXtreme Programming based Instructional Design Methodology for Collaborative Teaching

Domenico Lembo
Mario Vacca

Technical Report n. 8, 2012

Project Based Learning + Agile Instructional Design = EXtreme Programming based Instructional Design Methodology for Collaborative Teaching

Domenico Lembo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti"
Sapienza Università di Roma
lembo@dis.uniroma1.it

Mario Vacca

Direzione Generale per gli studi, la statistica e i sistemi informativi
Ministero dell'Istruzione, dell'Università e della Ricerca
mario.vacca1@istruzione.it

Abstract

In the last years, the use of ICT in teaching and learning activities is widespread and "Course design has developed from a craftsmanship-like process to a structured production, which involves interdisciplinary teams and requires more complex communication skills." [Botturi 2006], making methods and modeling languages more and more important. Many instructional design methods have been developed in the last years, but they seem to be inadequate if applied in the context of the 21th century school. In fact, nowadays, new skills for students are requires, like to be able to perform social useful activities or to collaborate to solve real problems [De Vincentis 2007, Pearlman 2009, Pearlman 2010], which, in turn, make it necessary that learning and values in the national instruction programs naturally embody and encompass these new activities. These yields other problems to be faced, like the students and parents' satisfaction, the administrative transparency and the effectiveness of the documentation, the need for cooperation among the teachers of a team. In this paper we map both the principles of Agile methodologies and the features of eXtreme Programming method (XP) into a new agile instructional design methodology which is suitable to solve the previous problems: it redefines the role of the teacher and introduces a new collaborative way to design and manage courses; it allows the realization of the new concept of administrative transparency and access to documents, that we call *active transparency*.

Keywords: agile methods; eXtreme programming; instructional design; software engineering methodology; smart city.

1. Introduction and motivations

Lately, the use of ICT in teaching and learning activities is widespread, fostering the change of educational environments. As a consequence “Course design has developed from a craftsmanship-like process to a structured production, which involves interdisciplinary teams and requires more complex communication skills.” [Botturi 2006]; hence, the instructional design process is always becoming closer to the software production development, making greater the need for methods and modeling languages.

Based on the software engineering methods, many instructional design methodologies have been developed and many others are emerging to face the problem of the course design, each of them carrying the same advantages and disadvantages, limits, conditions of applications and problems such as the originating methods.

Moreover, living in our society, new skills for students are required to perform social useful activities or to collaborate among each other to solve real problems [De Vincentis 2007], which, in turn, make it necessary that learning and values in the national instruction programs naturally embody and encompass these new activities. These ones yields other problems to be faced, like the students and parents’ satisfaction, the administrative transparency and the effectiveness of the documentation, the need for cooperation among the teachers of a team.

Today, the most common methodology in Italian schools is the Dick and Carey one [Dick and Carey 1990], or its several variants; some of the typical problems with the application of this method are: the marginal role of students and their parents in the instructional design process and the related problems of satisfaction; the often unbalanced student workload for each subject and term; the compelling choice of materials and technological tools at the beginning of the instructional process; the scheduling and the revision of the plan in case of failure; the difficulty of collaboration among teachers in the design of the course.

These problems are very similar to those arising from the application of traditional software engineering methods like, for example, the waterfall model [see, for example, Pressman 1997].

In order to solve the problems arising from classical software engineering methods, in 2001 a group of software developers and designers formed the Agile Alliance, which published the Manifesto for Agile Software Development [Manifesto for Agile Software distribution, 2001] based on the assumption that individuals and interactions, working software, customer collaboration and responding to change are, respectively, more important than process and tools, documentation, contract negotiation and following a plan.

Agile methodologies aim to satisfy the customer, to welcome changing requirements, to deliver working software frequently (e.g. every few weeks), to make customers and developers collaborate, to motivate individuals by suitable environment and to support, considering dialogue essential to exchange information about the project (for the complete list of principles see http://agilemanifesto.org/principles.html).

There are many agile methods and the eXtreme Programming is one of these, whose paradigm can be summarized using the word of its inventor Kent Beck: "Driving is not about getting the car going in the right direction. Driving is about constantly paying attention, making a little correction this way, a little correction that way." [Beck 1999] “This is the paradigm for XP. Stay aware. Adapt. Change.” [Beck 1999]

In this paper we show that it is possible to apply the Agile and XP principles to instructional design; therefore, we propose EPIC (EXtreme Programming based Instructional Design Methodology for Collaborative Teaching), a new agile instructional design methodology, based on the XP, with two

concepts at its base: the instructional design as a cooperative/collaborative act; the active participation of students and parents to the instructional design process.

These assumptions lead to the possibility to solve the previous mentioned problems based on the redefinition of the role of the teacher along with the introduction of a new collaborative way to design and manage course design. Moreover, it makes possible the introduction of a new concept of administrative transparency, and access to documents, we call *active transparency* and synthesize by the statement:

transparency + participation = active transparency

This means not only having the possibility to access to documents, or being informed of the individual marks got in a test of in a final report, but also participating to the writing of documents or negotiate the kind of right evaluation.

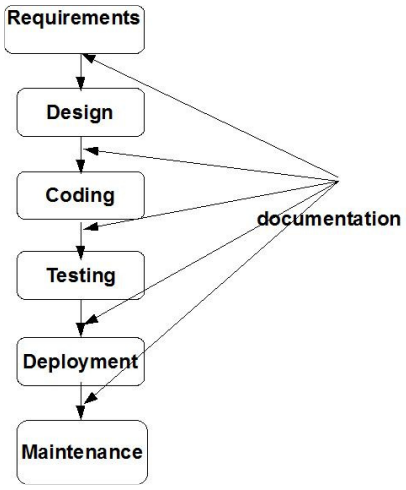
2. Background knowledge

2.1. A short overview of software development methods

A software development method is a way to rationalize the process of software development, that is to structure, plan, and control it, through the definition of the software life cycle (SLC), which is the sequence of the activities to be performed on the software itself from the inception to its dismissal. These activities can be specified through either a language (e.g. a modeling language like UML) or informally.

There are many software development methods, each characterized by a specific cycle of life: the waterfall; the incremental and evolutive method; the spiral method; the unified process; the MDA; agile methods. In the following we only give a short account of some methods (for further readings see [Arlow 2005, Beck 1999, Kleppe 2003, Pressman 1997).

The Waterfall model is the most famous software engineering method; it is a sequential approach, whose SLC is divided in phases, each of them characterized by a kind of deliverable (documentation): requirements analysis, design, coding, testing (validation), deployment and maintenance (see figure below).



The aim of these phases can be roughly summarized as follows: in the phase of *requirement analysis* the features of the software system are identified; the *activities of design* consist of designing the architecture, i.e. of dividing the software to be developed in parts*; the coding* is translating the parts defined during the phase of design into a given programming language; *testing* serves to discover programming bugs, while validation tests must be passed by the software to be accepted by customers; *the deployment and maintenance* are dedicated, respectively, to the delivery of the system and to its improvement.

The waterfall is a "big bang" model because it provides the software all at once, at the end of the process. The main risk, linked with the linear nature of this method, is that problems (for instance software bugs or customer dissatisfaction) arise only at the end making it difficult to fix them. To solve this problem, evolutive and incremental methods allow splitting the project in smaller parts, so that the software is produced incrementally. One or more prototypes (either working or only demonstrative versions of the final software) are often provided during the course of the development process; in this way the users have the opportunity to validate the software while it is produced and the developers can control the process (costs, the time scheduling of the project, etc).

Some of the problems arising in software development are the customer satisfaction, the increase in costs during the process and the time scheduling of the project, the efforts needed to produce the documentation and its usefulness for the process; the management of change and, in particular, of the requirement change which can pervade the whole project.

2.1.1. Agile methodologies and the eXtreme Programming

In order to solve the previously mentioned problems, in the 2000, a group of software developers and designers formed the Agile Alliance, which published the *Manifesto for Agile Software Development* [Manifesto for Agile Software distribution 2001] based on the following assumptions:

"We have come to value
Individuals and interactions over process and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
That is, while there is value in the items on the right, we value
the items on the left more".

A list of principles behind the agile methodologies have also been formulated here, among which there are: satisfying the customer, welcoming changing requirements, delivering working software frequently (e.g every few weeks), making customers and developers collaborate, motivating individuals by suitable environment and support, considering dialogue essential to exchange information about the project (for the complete list of principles see http://agilemanifesto.org/principles.html).

In 2005 the *declaration of interdependence* stated the basic principles for the agile project management, among which there are the following, quoted from http://pmdoi.org/: delivering "reliable results by engaging customers in frequent interactions and shared ownership"; expecting "uncertainty and managing for it through iterations, anticipation, and adaptation"; "unleashing creativity and innovation by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference"; "boosting performance through group accountability for results and shared responsibility for team effectiveness".

There are many agile methods like Crystal, the eXtreme Programming or Scrum; the eXtreme Programming's paradigm can be summarized using the metaphor of Kent Beck, who proposed the

method: "Driving is not about getting the car going in the right direction. Driving is about constantly paying attention, making a little correction this way, a little correction that way. This is the paradigm for XP. Stay aware. Adapt. Change." [Beck 1999]
Therefore, according to this paradigm, managing a project is like driving a car, that is it's necessary to adjust the route continuously. Therefore, it is important to stay aware and taking care of changes. In order to apply this philosophy practically, the method extremes the good practices in software production process, like code revision, testing, design, simplicity, architecture, integration, short iterations and documentation (this is the reason for its name). The extremes practices are reported in the following table:

Good practices in software development	Extremes practices in software development
code revision	pair programmino
testing	continuous testing
simplicity	simple design, simple and standard coding
architecture	shared metaphor
design	Re factoring
integration	continuous integration
short iterations	planning game
documentation	collective ownership of the code

The XP extremes practices aim to realize

a continuous development process with continuous feedback, steadily documented, and realized by an efficient and comfortable organization.

The **continuous process** is realized by small releases, refactoring (code restructuration and improvement) and continuous integration of the new produced code in the already developed one. The **continuous feedback** is reached through a development team including the customer, the planning game (periodical meeting of the team), the pair programming (a programmer write the code while another one check and periodically vice-versa), the continuous testing. To make the whole process **steadily documented**, it is proposed to use the collective ownership of the code (each programmer is responsible for all the code), a system metaphor synthetically describing the project, the use of a standard coding to make it easier to understand the code and the use of simple design. Finally, the organization has to support the workers, trying to avoid stressing them (extraordinary work is forbidden) and let them work in a comfortable environment which promote dialogue and collaboration.

The phases of the XP method are the exploration, the planning, the design, the coding and the testing. During the exploration, the users and the developers write the stories related to the working of the system; the planning phase serves to decide which stories have to be implemented and which others can wait; the design states the right architecture (the classes involved and their responsibilities); during the phases of coding and testing, the stories selected in the planning

phase with the architecture of the design one are, respectively, implemented using some programming language and tested using the tests formulated both by programmers and the users.

2.2. Instructional design methodologies: a short state of the art

2.2.1. Notes on the instruction process and related theories

What does it means teaching or what is an instruction process is a question of point of view. There are, in fact, many views of the instructional process depending on the theory taken in account. In these short notes we consider the behaviorist, the cognitivist, and the constructivist view of instructional process, showing only the main features.

According to the view "The instructional process, or teaching, has traditionally involved instructors, learners, and textbooks. The content to be learned was contained in the text, and it was the instructor's responsibility to "teach" that content to the learners. Teaching could be interpreted as getting content from the text into the heads of learners in such a way that they could retrieve the information for a test." [Dick and Carey, 1990]

The Gagné behavioral psychology theory [Gagné, 1985] is based on the statement that if students have learned, then it is more likely that they will exhibit a desired behavior in a given situation.

The cognitivist model sees instruction process aiming to improve students' mental processes (memorizing new information to be used, for instance, to perform new skills, through the organization of activities and the providing of information.

The constructivist model see instruction process as one where learners construct their own interpretation of the world based on the use of new and old information and experiences.

2.2.1. Instructional design methods: a short overview

All the theories in the previous section, other than other ones, can be applied in a systematic way using instructional design methods. According to merril et al. [Merril et al.1996] "Instructional design is a technology for the development of learning experiences and environments, which promote the acquisition of specific knowledge and skill by students. Instructional design is a technology which incorporates known and verified learning strategies into instructional experiences which make the acquisition of knowledge and skill more efficient, effective, and appealing".
There is a strong link between the software engineering methodologies and the instructional design ones, as many authors stressed [Kennedy 1998, Rawsthorn 2005, Stewart et al. 2009, Tripp and Bichelmeyer 1990].
Rawsthorn claimed that "Since the 1960's computer technologies and related practices and methods have had a significant influence over Instructional Design methods. One of the major trends is the influence of Software Development Life Cycle methodologies over Instructional Design methodologies. This influence is evident in the ADDIE, Dick and Carey, Rapid Prototyping and other Instructional Design methodologies" [Rawsthorn 2005].

Stewart et al observed that "At first glance, the similarities between the software development methodologies and the educational methodologies are easily seen. Both teaching and software development require detailed planning and scheduling. Each requires management and constant assessment and feedback from all involved. Making sure a course is delivered correctly and on time presents similar difficulties to those encountered in software development projects" [Stewart et al. 2009].

Tripp et al. observed that “Engineering and education are both disciplines which fit Simon's definition of artificial sciences. Software design and instructional design are fields that have similar methodologies and purposes. The waterfall model (Maher & Ingram, 1989) of software design and the interservices ISD model (Branson, 1975) represent two well-known models from the respective fields. Both models consist of five steps. The waterfall model includes Analyze, Design, Implement, Test, and Maintain. The interservices ISD model specifies Analyze, Design, Develop, Implement, and Control. The superficial similarities are obvious. At a deeper level, Maher and Ingram (1989) note that in both fields, designers attempt to be systematic in approaching large, complex problems. Designers in both fields attempt to bring orderly and replicable practices to disciplines which are dominated by individual practitioners. Both have typically advocated the use of formative evaluation procedures in the development of systems. Additionally, the two often deal with similar constraints in planning, budgeting, scheduling, and tracking the development of materials. The most fundamental difference between the two fields is the degree of rigor that can be expected in each. Software designers deal with systems that are based on mathematical logic. Instructional designers deal in part with computer software, but primarily with systems based on human cognition, which entail more uncertainty and accept more ambiguity. Based on the large number of similarities and the minor differences that exist, practitioners in the two fields have often used similar models in their efforts to create effective materials” [Tripp and Bichelmeyer 1990].

In this section we overview the most used methods, give a short account about new methods arising in the field of instructional design.

The ADDIE method

ADDIE (Analyze, Design, Develop, Implement, Evaluate) is the simplest and most common instructional design method. It is constituted by five phases as its name suggests:

Phase 1) *Analyze*
identify instructional goals and tasks, analyzing learner characteristics; formative evaluation.

Phase 2) *Design*
develop learning objectives, choose an instructional approach, define performance objectives, develop assessment instruments, develop instructional *strategy*

Phase 3) *Develop*
choose materials; design formative evaluation.

Phase 4) *Implement*
deliver instructional materials; apply instructional activities; formative evaluation.

Phase 5) *Evaluate*
summative evaluation.

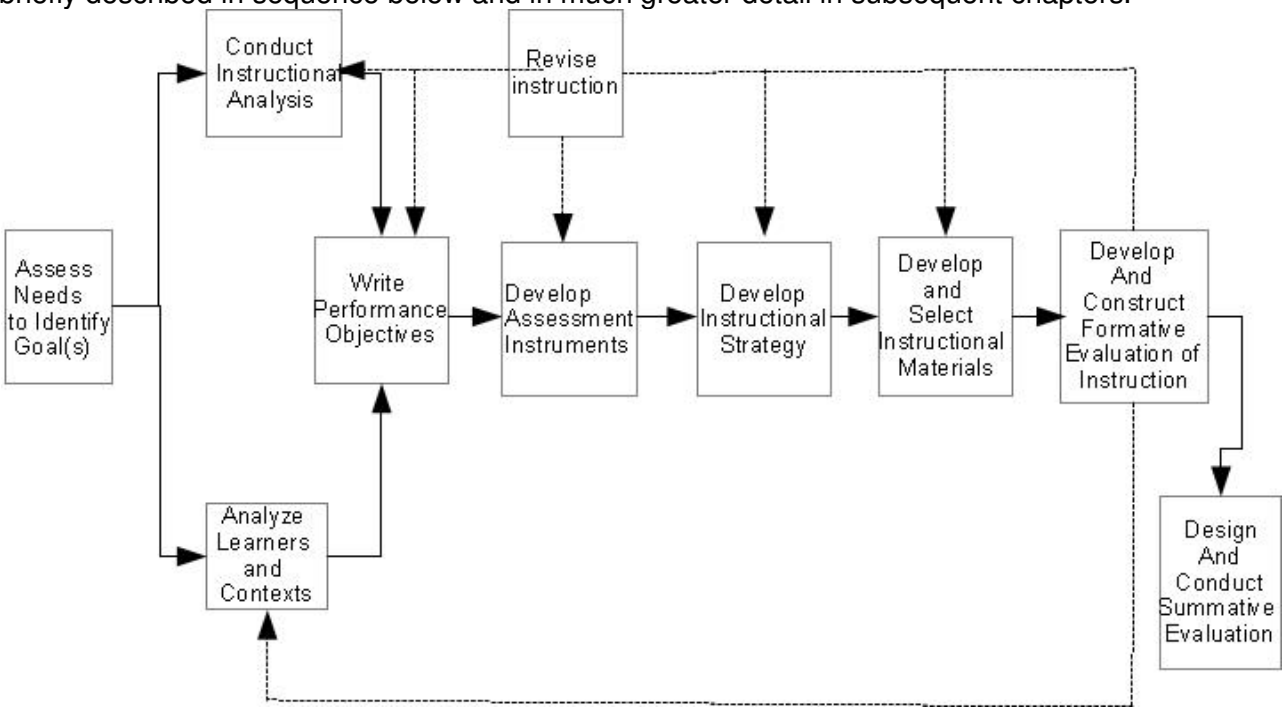
The Dick & Carey method

The Dick and Carey method [Dick and Carey 1990] belongs to the class of Instructional System design (ISD), where the instructional process is viewed as a feedback system, whose interacting components are the learners, the instructor, the instructional materials, and the learning environment and whose goal is to bring about learning; the testing gives the feedback to control the system, making some changes.

The phases of ISD are analysis, design, development, implementation, and evaluation.

The Dick and Carey method is constituted by a series of steps, all of which will receive input from the preceding steps and will provide output for the next steps. All of the components work together in order for the user to produce effective instruction. The model includes an evaluation component that will help determine what, if anything went wrong and how it can be improved.

The model includes ten interconnected boxes and a major line that shows feedback from the next-to-last box to the earlier boxes. The boxes refer to sets of procedures and techniques employed by the instructional designer to design, develop, evaluate, and revise instruction. The steps will be briefly described in sequence below and in much greater detail in subsequent chapters.



Phase 1) *Assess Needs to Identify Goal(s)*
Determine the instructional goals

Phase 2) *Conduct Instructional Analysis*
Determine the required skills, knowledge, and attitudes.

Phase 3) *Analyze Learners and Contexts*
Analyze the context in which the learners will learn the skills and they will use them.

Phase 4) *Write Performance Objectives*
Determine the conditions under which the skills must be performed, and the validation criteria.

Phase 5) *Develop Assessment Instruments*
Develop assessments to measure the learners' ability to perform the skills.

Phase 6) *Develop Instructional Strategy*

Phase 7) *Develop and Select Instructional Materials*

Phase 8) *Develop and Construct Formative Evaluation of Instruction*

Phase 9) *Design and Conduct Summative Evaluation*

Phase 10) *Revise instruction*

The data from formative evaluation are used to revise the whole instructional process.

The Dick and Carey methodology linearity is broken by the revise instruction phase whose effects pervade the whole process.

Rapid prototyping

These classic instructional design models so far seen (ADDIE and Dick and Carey) are linear, in the sense that the instructional design process, even if it include a revision process, advances through the advancement of the Analysis, Design, Development, Implementation and Evaluation phases.

The process proposed by the *Rapid Prototyping* method is an iterative one and it is based on the observation that analysis is rarely complete. For this reason the phases *Set Objectives*, *Construct Prototype*, *Utilize Prototype*, *Install & Maintain* could be overlapping.

Assess needs & Analyze Content	Set Objectives
	Construct Prototype (Design)
	Utilize Prototype (Research)
	Install & Maintain System

The rapid prototyping model (from Tripp and al. 1990)

Recent trends in instructional design

The design of courseware's has further increased the use of software engineering methods in instructional design: in fact, in [Dwolatzky et al. 2002, Luden 2002] the UML and the UP method are proposed to design courseware's; Dodero et al. [Dodero et al. 2006] propose to use the Model Driven Approach (MDA) to generate learning material.

2.3. Agile approaches in education

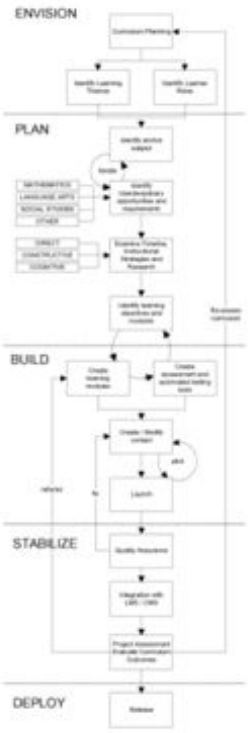
In [Stewart et al. 2009] a survey of agile methods can be found, concerning the use of agile teaching in course about software development or computer science or similar. Common features are: minimized use of lectures in favor of, possibly, group application; short activities; use of feedback. The results include more motivated students and more satisfactory experience both for students and teachers. In the sequel we illustrate some agile education approaches.

Agile Instructional Design **[Peter Rawsthorne 2005]**

(Agile Methods of Software Engineering should Continue to have an Influence over Instructional Design Methodologies.)

The author maintains that the software engineering methodologies have had influence over Instructional Design methodologies and, now agile methodologies are spreading and then they can provide new techniques to Instructional Design methodologies. According the author, from the point of view of learning theories, Agile Instructional Design methods enhance constructivism as they involve the learner in the curriculum development process.

The phases of the proposed method are as in the following figure from [Peter Rawsthorne 2005]:



Rawsthorn P. propose the following change upon the Dick and Carey model:
Stage 1. Curriculum Planning
Stage 2. Identify Learning Themes and Metaphors (Anchors)
Stage 3. Identify Learner Roles
Stage 4. Trawl for Learning Objectives, Modules and Competencies.
Stage 5. Identify Test Cases (Proof of Competencies)
Stage 6 & 7. Pair Programming (Development)
Stage 8. Unit Test (Automated Testing)
Stage 9. Release to Production (Refactor, Refactor, Refactor)
Iterate to Stage 4. within curriculum plan.

Despite their importance within the XP methodology (see Beck 1999), the Rawsthorn paper doesn't consider the roles of actors involved in the instructional process

Agile Education [De Vincentis 2007] spde@deakin.edu.au
In this approach, it is taken in account the problem that "Students need to develop not only excellent numeracy and literacy skills, but problem solving skills, creative solution skills, strategy skills, relationship skills, think-on-your-feet skills" because of the need to create new jobs and be able to change job.
The author observes that State curricula are essentially based on values and the essential learning and that life skills are considered through the blending of discipline based content with values and life skills they. Moreover, equity and standardization are opposite and teaching is often for testing.
She proposed a student-centric approach based on the following *Agile Education Manifesto* based on the Agile Manifesto:

"We are uncovering better ways of developing education by doing it and by helping others do it. Through this effort we have come to value:

individuals and interactions over processes and tools,
working education over comprehensive documentation,
customer collaboration over contract negotiation, and
responding to change over following a plan.

While we value the items on the right in this list, we value the items on the left more."

[De Vincentis 2007, based on The Agile Manifesto, Agile Alliance 2001].

Agile principles, active and cooperative learning [John C. Stewart et al. 2009]
The authors maintain that in order to learn, students have to participate actively to the learning process, that is they have to discuss, to read, to write, but also to solve problem, to analyze, to evaluate and to synthesize. To be active, students have to do things in addition to think about the think they are doing; moreover, to be cooperative students have to participate in tasks as a group. In that paper, according to the authors, *agility is referred to student learning* and it is interpreted as a means to make teaching student-centric and effective.
The authors propose the following *student-centric learning oriented Agile Manifesto*

Students over traditional processes and tools.
Working projects over comprehensive documentation.
Student and instructor collaboration over rigid course syllabi.
Responding to feedback rather than following a plan.

They also mapped the principles of the Agile Manifesto into Corollary to the Pedagogical Environment (see afterward in the paper). The authors show that agile teaching methodologies fall in the categories of Active and Cooperative learning.

3. The change of learning environment and the educational needs in the 21th century society

The technologies are changing our daily life and with it all the social activities, from trade to the processes of government, relations with the government up to the schools and universities, where the change is even more important because it affects not only processes, but comes to teaching, ie the basic tools for citizen education.
To manage the relationship technology-education becomes a crucial issue for society today, as it directly affects the formation of future citizens.

The relationship between the use of technological tools and education is manifested in the creation and testing of new learning environments, or environments in which the actors are always the students and teachers, but they may have, by virtue of the use of technologies, a different connotation space-time, or "pushing the limits of the classroom" and "school time" to allow a more personalized learning and to measure student.

From another point of view is the importance of knowing how to use technology to solve problems, as established and adopted by the Digital European Digital Agenda Italian. So the new learning environments conducive to the spread of technologies, driver of the economy.

We must add, that the flourishing of languages associated with the new medium of communication that young people are common (they are the so-called digital natives) makes the combination of technology-teaching or learning technological environments, of further importance, since in able to attract young people to culture and resolution of problems in language and modes of operation that young people commonly use.

Another observation to make, concerning the importance and role of knowledge in Internet time. Today, knowledge is widely scattered, but many, so it takes two types of skills: building survey, or systematize the knowledge on a comprehensive framework and be able to use knowledge to solve problems. We can say that the lofty goals of Bloom's taxonomy, analysis, synthesis, problem solving, today constitute basic skills that every citizen should have as their culture, much more than individual knowledge.

Therefore more than learning notions it is important solving real problem in a real way, even because this could have a positive influence over the student motivation. On the contrary, school curricula are based on values and learning, neglecting the real life problems which are considered only out of the curricula values and learning (in the Italian educational systems projects can be dedicated to this end).

"Digital technology potentially brings great many advantages to education, including ease of connecting with the world, ease of sharing, ease of getting and giving feedback, and better, faster ways to create and communicate (just to name a few).
But digital technology is not, by itself, the answer to education, or our educational problems. In fact, just adding technology to the old "tell-test" pedagogy can actually *hinder* education and learning, by distracting students from listening, while not taking maximum (or any) advantage of the powerful tools they have. So the pre-requisite for adding technology to change teachers' pedagogy to some form of partnering" [Prensky 2011].

To sum up, as many authors observe, the 21th century societies are characterized by smart cities each of them with a smart specialization. Moreover, modern societies, the information society, ask citizens for new skills like the ability to change job frequently or to create new jobs, the habit to collaborate with other workers or to do research to solve the problems of their community, using the ICT to retrieve the information needed.
In other words, it is required to be creative.

3.2. Inadequacy of the current instructional design methodologies to face the problems of the 21th century way of teaching and learning

If instructional design methodologies have been useful so far, now they provoke some problems, like the following:

- effectiveness of documentation and plan revision

The project plan related to each subject is made at the beginning of the course; each time that some problem happens imposing the plan revision, the new project plan should be rewritten in order to make this document effective. Rewriting documents requires time and efforts to make the document coherent with the others related to the other subjects.

- the time scheduling of the project

It is well known that projects are often delayed; it can happen that, in order to meet deadlines, the realization of some activities could be accelerated, yielding problems in student understanding skills and increasing the workload.

- marginal role of students and their parents in the design of course

Students and their parents have little space in the course design: they participate a few times a year to teachers' team meetings. Their contribution is often limited to the discussion about general problems arising and to the possibility to contribute to choose
From an engineering point of view, the students and parents' role is essentially limited to the validation of the instructional contract.
It is worth to note that other instructional systems based on different methods allow, for example, students to negotiate the activities with teachers in order to follow their own attitude and will, but always within the school general [Garnier 2011].

- unbalanced student workload among the disciplines and the term.

because of each discipline produces its own workplan applying the more appropriate method and without a detailed verification of the relations among the modules or activities of the different disciplines, it could be possible that in some period the student workloads results unbalanced or unbearable, producing negative results on the quality of learning.

- the choice of the more appropriate technological tools

There are cases where the didactical project has to follow a special idea of teaching aiming to modify the learning environment (the so called Idea 2.0) [Barca 2011, iTEC]; to realize this kind of ideas (that is to design the project and to realize it; the teacher team is supported by an expert, a coach. This case shows how the classic project plan used in the Italian school is inadequate.

Moreover, as it is observed in [Rawsthorn 2005] "there is also a growing discontent among many practitioners with the ISD methodology." In addition traditional has been criticized for being too slow and having an outdated world-view.

4. PBL: teaching and learning by projects

As underlined from many authors the didactical formula based on lectures where the teacher teaches (teacher telling or talking or lecturing) and students learn is not more adequate: the new paradigm fostered by the use of technologies is "students teaching themselves with teacher's guidance (a combination of "student-centered learning," "problem-based learning," "case-based learning," and the teacher's being the "Guide on the Side.""[Prensky, 2008].

To this end, Pearlman proposed to consider real life problem to be solved by real methods, so that useful outcomes requires student active participation, that is students that learn by themselves with the teacher guidance. Moreover, parents participation is also important within a newer evaluation system.

The Pearlman proposal is called *PBL* (Project Based Learning) and can be defined as "a system-atic teaching method that engages students in learning knowledge Designing New Learning Environments and skills through an extended inquiry process structured around complex, authentic questions and carefully designed products and tasks" [Pearlman 2010]

The main features of the PBL are:

- the main teaching activity is the project (which is one to three weeks long);

- projects are realistic (real word);

- projects generates a set of knowledge to be known in order to solve them;

- projects are designed to complex problems and require critical thinking, collaborative activities, problem solving capabilities;

- students as Workers and Producers (constructors of knowledge);

- students work and engage in self directed learning;

- students have to be motivated to learn and have personal “need to knows“;

- projects have associated rubrics for content, collaboration, written communication, oral communication, critical thinking, etc., all posted online for students, so that they can decide on their own whether to achieve basic, proficient, or advanced work;

- Self-direction is a learned behavior accomplished by students motivated
to learn, and having information on “how am I doing?” and “what do I want to accomplish?”

- Assessment and feedback are crucial.

In other words the process (project) comes before the content (information) which is a direct consequence. This is not new in philosophy and mathematics: in fact, reasoning by problems and on the problems has been studied and emphasized by many philosophers and mathematicians like Cellucci [Cellucci 1998, 2002], Hintikka [Hintikka 2007], Polya [Polya 1957] or Popper [Popper 1999] and it is considered as opposed to the traditional logic (the Aristotelic one) based on the reasoning on affirmations and using deduction as its main reasoning tool.

In the Italian school is a long time now that the practice of teaching by projects is increasingly shifting from the extra-curricular to the curricular context.

Teaching by projects using a form of horizontal teaching involves, the same as Pearlman notes, a number of difficulties and problems, starting with the management of a class that adopts the PBL.

In fact, students will:

- need access to project materials;
- must always be aware of the times;
- must know the criteria and methods of project evaluation;
- need to know about their assessments in order to improve their performance;
- must be able to choose the level of the tasks to be performed in accordance with their aspirations and abilities.

As Pearlman points out the same, this is where technology can make a difference: technology used to enable the activities mentioned above, as well as collaborative activities (wikis, blogs, video conferencing, etc.).

But not only technologies: a methodology is needed to help develop and manage the project.
A problem relates to the difficulties of managing a PBL classroom.

“Students can’t work effectively as individuals or as members of a team unless they can access all their project materials, calendars, and rubrics for how the project will be assessed. They also need to check their grades constantly to see how they are doing and also see the criteria for how they can do better. In addition, teachers need to design projects, project calendars and benchmarks, and assessments and post them online for student access. This is an area where today’s technology can make a huge difference.”

5. Agile management of learning projects: adapting the Agile and eXtreme Programming principles to the instructional design context and PBL.

“The result of using the systems view of instruction is to see the important role of all the components in the process. They must all interact effectively, just as the parts in a heating or cooling system must interact effectively in order to bring about the desired outcomes. There is not an overemphasis of any one component in the system, but a determination of the exact contribution of each one to the desired outcome. And it is clear that there must be both an assessment of the effectiveness of the system in bringing about learning and a mechanism to make changes if learning fails to occur.”
[Dick and Carey 1990]

Methodology has roles, activities performed by roles and producing artifacts and possibly ordered in phases.

Instructional design as software design paradigm implies the description of the instructional design process in terms of activities performed by roles

The instructional design process is therefore viewed as constituted by roles (students, teachers and headmaster, parents, consultants, etc.) each of them performing some activities (lecturing, checking, solving problems, discussions, exercises, personal study, presentation production, etc.).

In order to give the dynamics of the method, activities have to be arranged in phases or ordered in some way.
In this paper a teaching/learning process is constituted by activities that have to be designed, realized, performed. Activities help in fullfilling a goal
Activities involve students: single, groups or the whole class The acting subjects could be a single learner along with the tutor, a whole class with the instructor, or a tutor alone.

The XP paradigm is well suited also for teaching, because in teaching everything changes as well students are human and their learning response to teaching is not completely predictable, change with the time and from student to student; needs change; the technology and the materials change; the team members change; the team change.
“The whole team drives the development process. XP lets you adapt by making frequent, small corrections; moving towards your goal with deployed software at short intervals. You don't wait a long time to find out if you were going the wrong way.” [Beck 1999]

Students and parents participate to the definition of the content of the system

Our interpretation of the agile manifesto from a didactic point of view is as follows:

The collaboration between students and teachers over processes and tools

The collaboration between students, parents and teachers over educational agreements (patti formativi)

Interesting activities over instructional design documentation

The design, the problem solving and task performing over **notions and knowledge**

Responding to feedback over **following plans**

1. **Principles of the Agile Manifesto [**agilemanifesto.org/principles.html **]**	**Corollary to the Pedagogical Environment [Stewart et al. 2009]**	**Our interpretation**
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Our highest priority is to prepare the student to contribute to an organization through continuous delivery of course components that reflect competence.	Our highest priority is to satisfy the students and their parents through the continuous production of real projects and the achievement of results.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advanta- ge.	The instructor and students welcome and adapt to changes even late in the semester. Agile pedagogical methods use problems and change as an opportunity to facilitate learning and better develop marketable skills in the students.	
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Requiring working deliverables from the students over short time periods allowing for frequent feedback and guided problem solving and experimentation.	
Business people and developers must work together daily throughout the project.	There is iterative interaction between the instructor and students (or student groups) during each iteration of course components.	A collaboration between teachers, students and parents takes places during each iteration of the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Trust that most students are motivated. Give them the environment and support necessary that for them to be successful.	Motivate the students through design and realization of real projects.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	To the extent possible, allow for direct face to face interaction with students or student groups.	The most efficient method to inform is the face to face one within the group students, teachers and parents.
Working software is the primary measure of progress.	Working deliverables (i.e. models, software, project deliverables, presentations, etc.) are the primary measure of student progress (not necessarily midterm & final exams that require rote learning and memorization).	
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	The cooperative learning environment where students actively seek guidance and tools to solve problems is the basis for teaching the skills needed for life-long learning.	
Continuous attention to technical excellence and good design enhances agility.	Continuous attention to technical excellence and good design enhances learning.	
Simplicity--the art of maximizing the amount of work not done--is essential.	While in education there is some value in exploring subjects in depth just because there is student interest, understanding the problem and solving it simply and clearly is essential.	
The best architectures,	Student groups and	

requirements, and designs emerge from self-organizing teams.	teams should self organize, but all should participate equally in the effort.	
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	At regular intervals, the students and instructor reflect and offer feedback on how to be more effective. All stakeholders then adjust accordingly with the goal of being more effective.	

Good practices in teaching	Extremesed practices in teaching
Activity revision	Pair programming
Formative evaluation	Testing all the goals
Summative evaluation	Transversal evaluation
Restructuring of activities	Continuous design
Simplicity	Simple design
Architecture	Shared metaphor
The Council instructional design	Collective ownership of the instructional design
Short iteration	Planning game
Participation of students and parents	Active participation of students and parents

The metaphor plays an important role in learning projects where the use of technologies is relevant, as it happens in the action cla@ssi 2.0. Examples of metaphors can be found in [Barca 2001].

6. The method: activities, strategies and roles.

The feature of projects are:

- the teaching of basic annual educational programming is to define a set of projects;

- a project must be: designed to solve complex real life problems, solvable with real instruments and means, and of short duration;

- a project must generate a set of useful knowledge to the resolution of the same;

- the set of projects in a school year shall exhaust the knowledge, skills and capabilities provided by the ministry guidelines;

- a project must require critical activities, analysis, synthesis, problem solving, to be applied individually or in a cooperative and collaborative way;

- project proposals are written in the form of stories and shared with students and parents (stakeholders).

6.2. The activities and strategies

The Council's work within the class of extreme programming is organized in four tasks (repeated several times during the school year): listening, design, coding, testing.
The *listening* is listening to the desires and needs of users and staff, in setting goals and in consideration of the technological opportunities offered by the market to meet the needs of users and teachers.
The *design* layout of the project concerns the teaching and integration of project activities in the curriculum.
The *coding* consists in designing and implementing activities, while the verification *testing* is to achieve the objectives set out in the business of listening to the proper functioning of the planned activities.

The method therefore provides four strategies: strategy planning, strategy design, development strategy, the strategy of testing.

6.2.1. The planning strategy and game planning

The planning strategy aims to identify activities to be undertaken and the priorities and estimate costs and duration of the project.
As happens in XP, projects must involve only the objectives and activities necessary to be able to pursue the achievement of the curriculum.
The planning strategy is achieved through a game, *planning game,* with two participants: the management (students, parents, teachers, coaches, consultants) and teachers.
The strategy for the Council is to reduce the risk by investing as little as possible to realize the most important activities
The members of the Planning Game are the tabs containing the stories of operating activities.
The *user stories* are the characteristics of the activities are written on index cards of the official school history to each is assigned a value of importance is that a verb that indicates the priority (must, should, could, etc..).
(For the use of teaching stories see Jonassen, David H., Hernandez-Serrano, Julian (2002) Case-based Reasoning and Instructional Design: Stories to Support Problem Solving Educational Technology Research and Development, Vol 50, No. 2, pp. 65-77)
The Planning Game, the two actors are the development team and management. The group development comprises a total of all the people who will be responsible
of implementation. The management consists of total of all
those taking the decisions about what should be done.

The game is developed in three phases: **exploration, commitment** and **management.** The exploration phase aims to identify new goals and activities, the commitment to choose the activities to real ize in the next step, then the management has the task of address on the basis of what happens (adjustments based on reality).
Each phase consists of the moves.
The moves of the exploration phase:
1. Writing a story
management writes a story that describes an activity
2. Estimate the length of a story
teachers estimate the time necessary to produce and implement the story.
3. Subdivision of a story

The moves of the management phase:
1. Iteration (by management)
choice of stories to implement.
2. Recovery (by management)
choice of stories to keep the issue in the course if there are
problem of overestimation of the speed of implementation.
3. New story (by the management and developers)
The management can introduce a new story (and delete them).
4. New estimates (by developers)
if the old estimates are not more realistic

6.2.2. The design strategy

As mentioned before, the XP is based on short release, XP also applied to the teaching activities will be feasible and practicable in 2-4 weeks.
This phase is called the architecture education, whose basic components are the activities, each activity must identify what it does (responsibility) and ATRE activities that must work together to fulfill the responsibility.

The documentation of the design phase is the production of cards CRA (Cooperation, Responsibility, Activity).
The architecture is characterized by "project metaphor" that defines a simple and concise what you are doing at that time.
The simple design means designing teaching a small number of activities (as necessary), preferably interdisciplinary or multidisciplinary collaborations with a few.

6.2.3. The encoding strategy

The coding tasks is to create (e.g. choice of technologies, creation of slides, etc..) And in their implementation.
The practices in support of this activity are the pair programming, collective ownership of assets, The integration continues, the standard coding, refactoring.
The pair programming consists in realizing the activities in pairs (not in the program them), using standard (eg models of slides; shooting LIM with certain characteristics, etc.).
The programming takes place in pairs, both to facilitate the control to favor solutions
mutual control of the property / product code and to stimulate the generation of innovative solutions by comparison between different people. It is advisable to replenish the pairs.

The collectivization of the code should help simplify the structure of work, together with the adoption of coding standards. Refactoring (restructuring) is used to restructure the activities sualla abase of new requirements (eg the next year or for the recovery or because it has identified a more effective).
The ongoing integration of new activities within the existing architectural scheme involves understanding the role of training activities in the context of a class.

6.2.4. The test strategy evaluation

The tests are written before the activities are carried out continuously, and frequently.

Since the tests take time, we must design tests that help track down the important issues, to help people understand where the action exerted through educational activities has failed.

Because the tests take time, we must design tests that help track down the important issues, to help people understand where the action exerted through educational activities has failed.

The different modes of evaluation should be written before the activities are performed continuously, and (often) to monitor the status of implementation and the quality of the project.

He must not play around with: activities must propose some simple things that test is useless, thus remain to be tested only the important things.

6.3 The roles

If the programmer is the heart of XP, the teacher will be the heart of EPIC.
As in XP, even in EPIC, the main value is communication with other people.

6.3.1 Students

The student:

- must be motivated to participate in the project;
- must practice self-learning;
- must have personal knowledge needs and adhering to the project;
- participates in the writing of stories;
- participate actively in the planning and implementation;
- is always aware of the activities and duties;
- choose the activities to be carried out while participating in the project based on their aspirations and abilities;
- are aware of the criteria for evaluation of individual activities and overall value.

6.3.2 Teachers

The teacher:

- plans activities;
- assesses the scope of activities;
- guides and directs the students during the definition and implementation of the project;
- defines the evaluation activities.

A teacher takes on the role of **tracker** who manages the result of tests; takes in account the problems, solutions and tests; collects information on the project realization;

Another teacher or an outside expert's role is the **coach** who is responsible for the entire process. S/he controls the operations of the council.

6.3.3 Parents

Parents participate in the

- writing of stories and functional tests (test type);
- definition of final validation tests

6.3.4 Headmaster

In our proposal the headmaster is interested in the instructional process and assure the fulfillment of the goals.

6.3.5 Consultants

The consultants are teachers who solve the problems for which they was called.

The participation of students, parents, counselors and coaches should also be facilitated by the use of technology and, in particular, those collaborative, parents and students should be seen as analogous to an inherent customer, effective contributors to the definition of the training project class, not just subjective to which the school has only the obligation to fulfill the training agreement.

We must create a working space open for the group, with a central common area for programming and small private spaces around.

7. Conclusions and future work

In this paper the needs of the society are taken on account, according with the concept of smart community. The school becomes a part of the smart community and this foster the participation of the maximum number of stakeholder. To this end a new methodology based on the eXtreme Programming is presented.

New concepts are introduced:

- collective instructional design;
  the collective guarantees a right equilibrium between the load and importance of the disciplines.
- active transparency;
- active role of students and their parents.

In addition to the necessary experimentation , future work include the definition of

- teacher 2.0;
- tools 2.0 for supporting the methodology.

Moreover the relation between the presented methodology and the concept of school 2.0 has to be investigated.

References

2. Manifesto for Agile Software distriubution (2001), http://agilemanifesto.org/ www.agilealleance.org,.
3. J. Arlow, I. Neustadt (2005)``UML 2 andTthe Unified Process'',. Pearson Education, 2005
4. Barca D. (2011) "2.0 in classe Dalla rete: il bisogno, l'idea" (in Italian) . Retrieved from http://www.comprensivocigliano.it/Classe%202.0%20dalla%20rete%20il%20bisogno%20l'idea.pdf on19th November 2011
5. Beck K. (1999). Extreme Programming Explained: Embrace Change, Addison–Wesley.
6. Botturi, L. (2006). E2ML. A visual language for the design of instruction. Educational Technologies Research & Development, 54(3), 265-293.
7. Andy Hon Wai Chun (2004): The Agile Teaching/Learning Methodology and Its e-Learning Platform. ICWL 2004: 11-18
8. Cellucci Carlo (1988) Le ragioni della logica, Laterza (in Italian)
9. Cellucci Carlo (2002) Filosofia e Matematica, Laterza (in Italian)
10. Cla@ssi 2.0 http://www.scuola-digitale.it/classi-2-0/il-progetto/introduzione-2/
11. Dall'Agnol Michela, Sillitti Alberto, and Succi Giancarlo (2004) Jutta Eckstein Hubert Baumeister (Eds.) Project Management and Agile Methodologies: A Survey
12. Dick, W. & Cary, L. (1990), The Systematic Design of Instruction, Third Edition, Harper Collins
13. Declaration of Interdependence http://pmdoi.org/
14. De vincentis Sue (2007), Agile education: Student-driven knowledge production ACEL/ASCD conference, New Imagery for Schools and Schooling Sydney, October 2007 http://www.acel.org.au/conf07/papers/De%20Vincentis%20paper.pdf
15. Dodero Juan Manuel, Díez David (2006), Model-Driven Instructional Engineering to Generate Adaptable Learning Materials, Advanced Learning Technologies.
16. Dubinsky Y., Hazzan O. (2004). Roles in Agile Software Development Teams. XP 2004: 157-165
17. Dwolatzky B., Kennedy I.G. and Owens J.D. Modern software engineering methods for developing courseware Engineering Education 2002
18. El-Abbassy Ahmed, Muawad Ramadan, Gaber Ahmed (2010), Evaluating Agile Principles in CS Education IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.10, October 2010
19. Garnier X. (2011). Personalised teaching and flexible time management in a digital environment in "Implementing 21st century ways of learning and schooling" 12 May 2011, Roma Conferenza internazionale sui nuovi scenari dell'apprendimento organizzata a Roma da ANSAS e EUN http://www.scuola-digitale.it/eventi/scuole-2.0/
20. Hintikka J. (2007), Socratic Epistemology; Explorations of Knowledge
21. iTec – "Design the future classroom" - Project http://itec.eun.org/
22. Gagné, Robert M. (1985). The conditions of learning (4th ed.). New York: Holt, Rinehart and Winston.
23. Jonassen, David H.; Hernandez-Serrano, Julián (2002) Case based Reasoning and Instructional Design: Stories to Support Problem Solving Educational Technology Research and Development, Vol. 50, No. 2, pp. 65–77
24. Kennedy, David (1998). Software Development Teams in Higher Education: An Educators View. Retrieved on Jan. 02, 2011 from http://www.ascilite.org.au/conferences/wollongong98/asc98-pdf/kennedyd.pdf
25. Kleppe Anneke G., Warmer Jos, Bast Wim (2003), "MDA Explained: The Model Driven Architecture: Practice and Promise", Addison-Wesley Professional.
26. Luden Lorna (2002), Courseware Engineering Methodology, JOURNAL OF COMPUTING IN HIGHER EDUCATION Volume 14, Number 1

27. Merrill, M. D., Drake, L., Lacy, M. J., Pratt, J., & ID2_Research_Group. (1996). Reclaiming instructional design. Educational Technology, 36(5), 5-7.
28. Pearlman Bob (2009), Making 21st Century Schools: Creating Learner-Centered Schoolplaces/Workplaces for a New Culture of Students at Work ,EDUCATIONAL TECHNOLOGY September–October 2009
29. Pearlman Bob (2010), Designing New Learning Environments to Support 21st Century Skillsin James Bellanca and Ron Brandt (Eds.) 21st Century Skills Rethinking How Students Learn, 2010
30. Polya G. (1957), How to solve it Garden City, NY.
31. Popper K. (1999), All life is Problem Solving, Routledge
32. Prensky Marc(2008), The Role of Technology in Teaching and the Classroom (in Educational Technology Nov-Dec 2008
33. Prensky Marc (2011), Interview in l'Unita http://www.marcprensky.com/writing/Prensky-l'Unita-Interview-6-11-english.pdf
34. Pressman, Roger (1997). Software Engineering: A practitioner's approach (Forth ed.)
35. Principles behind the Agile Manifesto agilemanifesto.org/principles.html
36. Rawsthorn P. (2005). Agile Methods of Software Engineering should Continue to have an Influence over Instructional Design Methodologies. http://www.rawsthorne.org/bit/docs/RawsthorneAIDFinal.pdf
37. Razmov, V., Anderson, R.J. (2006), "Experiences with Agile Teaching in Project-Based Courses," In ASEE 2006, available from asee.org.
38. Robinson Hugh and Sharp Helen (2004) The Characteristics of XP Teams in Jutta Eckstein Hubert Baumeister (Eds.) Extreme Programming and Agile Processes in Software Engineering
39. Stewart John C., DeCusatis Carolyn Sher, Kidder Kevin, Massi Joseph R., and Anne Kirk M. (2009) Evaluating Agile Principles in Active and Cooperative Learning Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 8th, 2009
40. Traxler, J. Courseware Engineering : A Paradigm for Development [On-line] Available WWW:http:llds~ace.dial.vi~ex.com/mouldindCisedsreid/S IGTOSElmtn 92. htm#John Traxler
41. Tripp,Steven, Bichelmeyer,Barbara (1990), Rapid prototyping: An alternative instructional design strategy, Educational Technology Research and Development, 38, 1, 3/18/1990, Pages 31-44.