Chapter 4 – Constructionism and computers for learning

4.0 Overview of the Chapter

In this chapter, we introduce the theories of constructionism and instructionism and discuss them with reference to the EFL. All approaches to domain learning that involve programming satisfy Seymour Papert's basic definition of constructionism. It is impossible for a learner to construct computer models passively; there has to be a degree of engagement with the task. In this thesis, we take a broad view of constructionism that embraces bricolage and situated learning. We observe that constructionism can be broadly identified with activities at the concrete end of the EFL, and that instructionism can be broadly identified with activities at the formal end of the EFL. We argue that conventional programming is typically concerned with learning activities that are found at the formal end of the EFL, and hence that it is not well suited for supporting domain learning through constructionist model building. In contrast, EM – which gives support to learning activities at the concrete end of the EFL – is better placed than conventional programming to support domain learning through constructionist model building. The construction and use of a digital watch model is used to illustrate the ideas presented in this chapter.

4.1 Constructionism and instructionism

To motivate the theory of constructionism defined by Seymour Papert [Pap93] we shall first briefly review the theories of objectivism and constructivism.

4.1.1 Objectivism, Cognitivism and Constructivism

Psychology is full of debates about how learning occurs [HO96]. Theories of learning and the nature of understanding have a profound influence on the design of instructional materials, teaching philosophies and learning. The epistemological debate between objectivist and constructivist positions can be traced back to issues of ontology debated by the Greeks [Sae67].

Objectivism contends that there is a given reality which learners are expected to reproduce in their minds. There is no personal reconstruction of reality from an individual's viewpoint [Jon91]. The work of Skinner is characterised in the theory of *behaviourism* [Ski74]. Skinner developed a theory from experiments with animals placed in boxes. When an animal discovered the secret to escaping from a box the likelihood of it repeating that behaviour in the future was increased. Skinner proposed a theory of human learning called *operant conditioning*, which claims that learning can be totally characterised in terms of changes in overt behaviour [Dri00]. Learning occurs in response to environmental stimuli where particular stimulus-response patterns are reinforced through reward, and are thereby more likely to occur in the future. Skinner was motivated by his experiments to introduce the behaviourist approach to learning, in which the mind is treated as a black box. Behaviourism contends that the internal processes of the mind are not important in studying learning; it is sufficient to concentrate on the overt behaviour of the learner.

Cognitivism came to prominence in the 1950's [BGA56]. The assumption behind cognitivism is that the brain acts as an information processor. It takes input from the world and processes this to produce overt behaviours. Cognitivism places importance on the internal processes of the mind but is still objective in its approach – it assumes that there is a given reality that the mind processes and a learner's role is to passively acquire knowledge transmitted by an instructor [May99]. Newell and Simon's work on human problem solving was a major research project that took a cognitivist approach; it viewed problem solving as an information processing system [NS72].

Constructivism is founded on Kantian beliefs: it claims that the knower constructs reality based upon their mental activity [Jon91]:

> '… What the mind produces are mental models that explain to the knower what he or she has perceived … We all conceive of the external reality somewhat differently, based on our unique set of experiences with the world and our beliefs about them.' [Jon91]

The origins of constructivism may be found in John Dewey's view of learning as a constant reorganisation or reconstructing of experience [Dew16]. The research of Piaget [Bra78] and Vygotsky [Vyg62] provides the cognitive development theories that underpin the constructivist position. In their view, all meaning is rooted in personal interpretation of the world. The educational researcher Cooper [Coo93] compares constructivism with cognitivism and behaviourism in the following terms:

> 'The constructivist… sees reality as determined by the experiences of the knower. The move from behaviourism through cognitivism to constructivism represents shifts in emphasis away from an external view to an internal view. To the behaviourist, the internal processing is of no interest; to the cognitivist, the internal processing is only of importance to the extent to which it explains how external reality is understood. In contrast, the constructivist view the mind as a builder of symbols – the tools used to represent the knower's reality. External phenomena are meaningless except as the mind perceives them… Constructivists view reality as personally constructed, and state that personal experiences determine reality, not the other way around' [Coo93, p16].

Piaget stated that constructivism requires the learner to actively build their own knowledge structures, based on their own mental activity [Bra78]. Learning builds on existing knowledge, and each learner creates an individual representation of the subject being studied. It is inevitable that our initial constructions are naive and contain misconceptions. Personal constructions become more realistic as our experience grows. This approach is more natural for learners because it directly addresses the process of knowledge construction and is sensitive to mistakes in the learning process [Ben01]. The construction of viable mental models is also an

important part of the learning process, since these are the containers within which the knowledge can be organised [Ben01] (cf. the discussion of construals in section 3.4.1). In 1991, Merrill collated ideas from a variety of sources to identify the assumptions of constructivism [Mer91]. These assumptions are that:

i) knowledge is constructed from experience;
ii) learning is an active process;
iii) learning is collaborative with meaning negotiated from multiple perspectives;
iv) learning should be situated in realistic settings;
v) testing should be integrated with the task, not a separate activity;
vi) interpretation of reality is personal – there is no shared reality.

These assumptions are consistent with Kolb's model of experiential learning, as described in section 3.3.1.

Vygotsky stresses the importance of social and cultural contexts within the learning environment in supporting a discovery-based learning model [Vyg62]. His *Zone of Proximal Development* (ZPD) is an important concept with regard to learning because it defines the potential of a learner, in contrast with traditional tests that give only an accurate measure of current performance. In learning situations, the ZPD is an area within which a learner can interact given suitable assistance from other people and supporting technologies.

In the next section, we consider how the objectivist and constructivist movements have influenced the design of educational technology.

4.1.2 Instructionism and Constructionism

The dominant educational approach in the 20th century assumed an objectivist viewpoint, and was termed *instructionism*. In an instructionist approach, knowledge is transmitted to passive learners, for example through the use of lecturing and whole class teaching. Friere has described this as a 'banking method of education', where a

student's mind is an empty receptacle to be filled up with knowledge, in much the same way that you might top up your bank account [Fri70]. The instructionist approach is epitomised by John Carroll's notion of the Nurnberg funnel [Car90], a mythological device that allows teachers to pour facts directly into a learner's head. Instructionism is characterised by Jonassen in the following terms:

> 'Learning consists of assimilating objective reality. The role of education is to help students learn about the real world. The goal of designers or teachers is to interpret events for them. Learners are told about the world and are expected to replicate its content and structure in their thinking'. [Jon92]

Skinner's operant conditioning, when applied to instructional design, organises material into graded problems to which the student must correctly respond. This model was adopted in early software for computer-based instruction [Dri00]. The software followed the pattern of traditional teaching, where teachers reward students who do well with praise – a form of extrinsic motivation [Cov98]. Students are conditioned to achieve good results by linking the stimuli of good test results with the response of good marks and praise. The influence of the behaviourist approach can be seen in the proliferation of computer-assisted instruction (CAI) and 'drill-and-kill' educational software. This type of software aims to teach students by presenting a topic together with a selection of questions that they have to answer. Feedback is then given on their answers. CAI is simply an extension of the student-teacher transmission model of learning that has come in for strong criticism from many authors [Fri70, Ill71, Pap80, Pos92, Tal95, Opp97].

Constructionism has its basis in the theory of constructivism [Pap80]. In addition to the active building of knowledge structures, Papert claims that construction that takes place in the head often happens especially felicitously when it is supported by construction of a more public sort 'in the world' [PH91]. By 'in the world', Papert means that the public nature of a constructed artefact enables discussion, examination, probing and admiration [Pap93]. Although the theory of constructionism was originally introduced with reference to children, it is relevant

across all age groups, as Papert demonstrated by describing examples from his own learning processes [Pap80]. Knowles's andragogical model of adult learning is related to constructionism and places emphasis on self-direction, positive use of previous experience and internal motivation [Kno70, Kno90]. The important connection between andragogy and constructionism is the emphasis placed on the active role of the learner. The constructionist emphasis in learning is on the process, not on the product [KR96]. Jonassen supports the idea that constructionist educational software emphasises the process of knowledge construction as being more important than the resulting product:

> 'if meaning is determined by the mental processes of the individual, and these processes are grounded in perception and grow out of experience, then those are the things we should evaluate – not the extant behaviour or the product of that behaviour' [Jon92].

Ostwald endorses this emphasis on process over product in learning in his study of knowledge construction in software development [Ost96]. In his view, the building of artefacts enables us to learn through their construction, through experimentation (to see how they work) and through modification (to make them 'better'). He claims that the construction of an artefact and its understanding proceed in tandem:

> 'Experiential artefacts allow us to interact with the world. They provide information that enables us to interpret a situation through our perceptions. The danger is that they don't provide us with the knowledge – they provide us with information that is tacitly interpreted. When what we perceive is different from what we tacitly expect, a breakdown occurs, and the cause of this breakdown is brought to the surface where it can be interpreted and knowledge can be constructed' [Ost96].

The personal construction of artefacts is quite a different kind of activity from the assimilation of material designed by others. A constructionist perspective is well aligned with the learning activities at the concrete, empirical end of the EFL and an instructionist perspective is well aligned with the activities at the formal, theoretical end (cf. Figure 4.1).

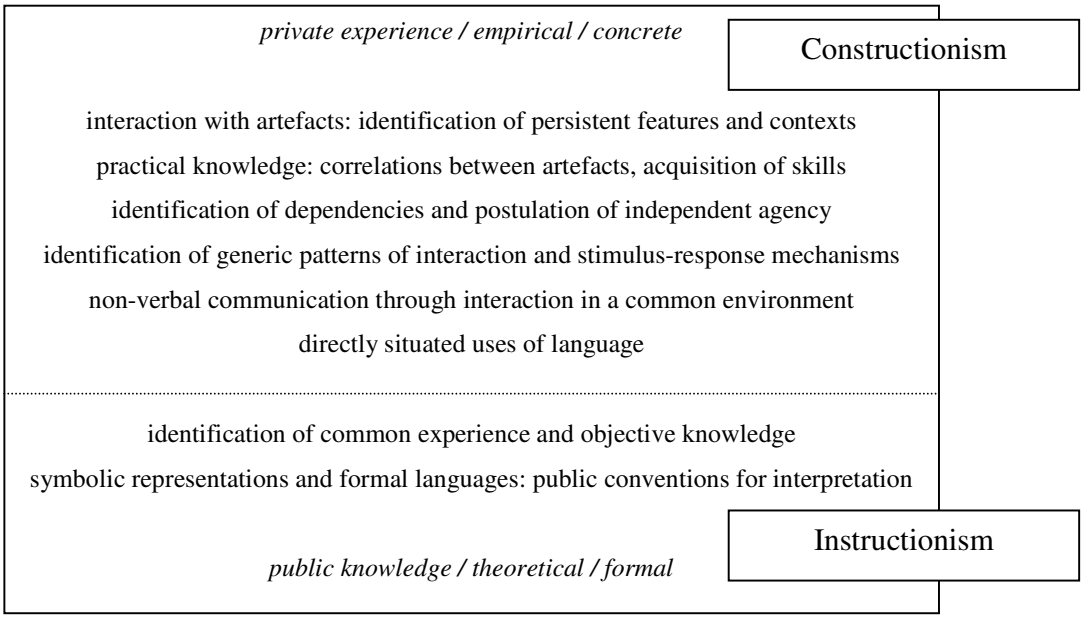


Figure 4.1 – Relating constructionism and instructionism to the EFL

Constructionism is closely linked with the learning activities at the concrete end of the EFL. Active knowledge construction plays a prominent part in interaction with artefacts and practical knowledge. These activities rely on our personal interpretation of the world, based on our private experience. At the formal end of the EFL, an important learning activity is the identification of common experience and objective knowledge. In an instructionist approach this is not an end-goal, but a prerequisite – the goal of education is to transmit objective knowledge. This is reflected in the quote from Jonassen cited above:

> 'Learners are told about the world and are expected to replicate its content and structure in their thinking' [Jon92].

Ever since Papert defined the theory of constructionism, and used the Logo programming language as its vehicle for delivery, constructionism has been closely associated with Logo. There are good reasons to expect computers to play a major role in the future of constructionism in the classroom. On the other hand, it is not obvious that the use of computers is allowing constructionist principles to be fully

expressed in educational practice. For instance, not all educationalists are convinced that programming in Logo promotes domain learning [Sol93, Ste94, Tal95]. To address this concern, we need a better understanding of the relationship between computers and constructionism. To this end, we consider a broader view of constructionism that encompasses model-building approaches such as *bricolage* and *situated learning*, which are not necessarily computer-based.

Bricolage [Lev68] is a style of construction that places emphasis on concrete experimentation and negotiation with artefacts. Bricolage is situated in the realm of primitive knowledge that concerns the acquisition of practical knowledge and the identification of persistent features and contexts. The style and manner of construction is important as well as the finished product. Situated learning [Lav88] advocates learning in the context of interaction in real-world situations, and is a prominent feature of a constructionist approach to learning. This can be seen from Papert's view of the public nature of the artefact constructed in learning and the role it plays in the 'real world' [Pap93]. It is very difficult to create our own model of reality without interacting in the real-world situation we are seeking to understand. We discuss the notions of bricolage and situated learning in more detail in sections 4.2 and 4.3 respectively. This supplies the context for the subsequent discussion of concept mapping (section 4.4), traditional computer programming (section 4.5) and EM (section 4.6) in support of constructionist approaches to learning.

4.2 Bricolage

The concept of bricolage originates in the work of Claude Levi-Strauss, an anthropologist who studied people working in primitive societies [Lev68]. Levi-Strauss was interested in contrasting approaches to task solving in what he characterised as 'primitive' and 'western' societies. In western societies, the most advanced form of thought is generally believed to be abstract and scientific. Jean Piaget's well-known theory of the stages of learning identifies abstract thinking as evidence of maturity [Bra78]. The importance attached to abstract thought can be

seen in the style of instruction and evaluation in the traditional Western schooling system. Levi-Strauss argued that primitive societies view 'concrete' thinking processes as more important than abstract thought [Lev68]. He defined bricolage as a way of performing work that emphasises human involvement and engagement where subjective interaction with the artefact guides solving of a task. A person who is involved in bricolage style activity is called a bricoleur. This French word is best translated as 'a handyman'; it emphasises a working style that takes advantage of whatever tools are at hand to perform tasks for which these tools were not specifically designed.

We can illustrate bricolage by considering an example of a situation in which it is used. Imagine that we are going to build a chair. One approach would be to design the chair and create a plan specifying how the chair should be built before production begins. The end result is a chair that matches our original plans which has been shaped entirely away from the situation in which the construction takes place. There are advantages to this 'planning' approach; we can be confident that the plan will realise a functional chair. However, if requirements for the chair change, or new insights become available during construction then this approach cannot take advantage of them. The planning approach to chair building has parallels in the 'waterfall' stereotype for software development [Boe76] where the knowledge-gathering phase freezes requirements, which are then rigidly implemented according to the specification. In contrast, the bricolage approach emphasises minimal forward planning and continual negotiation with the referent throughout the process of construction. Levi-Strauss characterises this negotiation in the following terms [Lev68, p18]:

> 'Consider the bricoleur at work and excited by his project. His first practical step is retrospective. He has to turn back to an already existent set made up of tools and materials, to consider or reconsider what it contains and engage in a sort of dialogue with it and, before choosing between them, to index the possible answers which the whole set can offer to his problem'.

With reference to chair building, the process of construction is characteristic of a craftsman at work – changes are made to the current prototype chair, and construction is guided by the current state of the chair:

> 'bricolage involves an informal subjective interaction between a craftworker and the artefact he/she is creating that more closely resembles discovery than organised construction. The model-building activity has an experimental and creative quality: if it is successful, the character of the artefact itself changes in the mind of the discoverer as it develops – it is continuously being newly conceived and reinterpreted in stimulating ways' [RB02].

Turkle and Papert have taken up the idea of bricolage and applied it in different fields as a way of validating individual approaches to problem solving [TP91]. Even in mathematics and science, there is problem solving activity that is not centrally concerned with the manipulation of formal symbols. Mathematicians and scientists stumble across discoveries through concrete model-building activities resembling bricolage, and only later do they refine these into scientifically acceptable formal abstractions. In this connection, Turkle and Papert [TP91] stress the importance of the computer as a tool that has 'revalued concrete thought'. They claim that the computer can be used in a way that privileges thinking with concrete artefacts rather than abstract thought and suggest that:

> 'the diversity of approaches to programming suggests that equal access to even the most basic elements of computation requires accepting the validity of multiple ways of knowing and thinking'.

Turkle and Papert refer to this 'validity of multiple ways of knowing and thinking' as an *epistemological pluralism* [TP91].

Although Turkle and Papert recognise that classically computer science promotes a structured planning approach, they argue that the distinction between planners and bricoleurs is manifest in the process of construction, not in the end result [TP91]. Papert identified two styles of programming that he called 'hard-edged' and 'smoky' and observed that to move from a hard-edged to a smoky style requires moving from

an abstract formal approach to a concrete bricolage approach [Pap93]. Turkle and Papert call the users of these two styles 'planners' and 'bricoleurs' respectively, and claim that:

> 'observation[s] of programmers at work calls into question deeply entrenched assumptions about the classification and value of different ways of knowing. It provides examples of the validity and power of concrete thinking in situations that are traditionally assumed to demand the abstract' [TP91].

The importance of concrete thinking in programming has been endorsed over the subsequent decade through the emergence, and growing popularity, of programming paradigms that emphasise non-traditional development cycles (e.g. eXtreme Programming [Bec00] and Rapid Application Development [Mar92]). Planners value hierarchy and abstraction; bricoleurs prefer negotiation and concrete experiments.

Many differences can be identified between the bricoleur's and planner's approaches to programming. Ownership of a program is one important difference. Planners typically want to be able to ignore the detail of individual components in their program and treat each component as a black box. Bricoleurs want the internal workings of their model to be exposed because they are personally involved with their program and want to maintain their engagement with it. If we consider the learning style of bricoleurs and planners, we find another distinction:

> '…the bricoleurs are happy to get to know a new object by interacting with it, learning about it through its behaviour the way you would learn about a person, while the planners usually find this intolerable. Their more analytic approach demands knowing how the program works with a kind of assurance that can only come from transparent understanding, from dissection and demonstration' [TP91, p173].

The above discussion shows that there are fundamental differences between bricolage and planning approaches. Turkle and Papert claim that an epistemological pluralism is a necessary condition for a computer culture that encompasses every individual [TP91]. This view is endorsed in psychology by Gardner's work on multiple

intelligences [Gar93], work on different learning styles [DD93] and approaches to problem solving [Pol57]. It is difficult for current computer programming cultures to accept an epistemological pluralism, because this:

> 'requires calling into question, not simply computational practices, but dominant models of intellectual development and the rarely challenged assumption that rules and logic are the highest form of reason' [TP91, p185].

Turkle and Papert propose to achieve epistemological pluralism through the development of computer programming languages that embrace both planning and bricolage styles. However, their critics claim that what Turkle and Papert identify as bricolage in computer programming is not a case of 'trial-and-error vs planning' but of '*aimless* trial-and-error vs planning' [YB01]. For instance, Ben-Ari is concerned that the development of explicit mental models must take place in tandem with trial-and-error, or learning will be hindered. In respect of programming, Ben-Ari claims that:

> 'premature attempts to write programs lead to bricolage and delay the development of viable models … There is nothing wrong with experimentation and bricolage-style debugging, as long as it supplements, rather than supplants, planning and formal methods' [Ben01].

Although Ben-Ari believes that we all practise some bricolage thinking, he does not see it as a substitute for the conventional programming discipline:

> 'The manifestation of bricolage in computer science is endless debugging: try it and see what happens. While we all practice a certain amount of bricolage and while concrete thinking can be especially helpful – if not essential – for students in introductory courses, bricolage is not an effective methodology for professional programming, nor an effective epistemology for dealing with the massive amount of detailed knowledge (that) must be constructed and organised in levels of abstraction (cf. object-oriented programming). The normative planning style that we call software engineering must eventually be learned and practiced' [Ben01].

In due course, we shall argue for a radically different view of the place of bricolage in computer programming from that represented by Ben-Ari. We attribute the failure of current programming and software engineering practices to support bricolage fully to their lack of maturity and contend that – by adopting different practices – it is possible to integrate the construction of computer-based artefacts and mental models. The possibility of better computer support for bricolage notwithstanding, we endorse Ben-Ari's claim that planning has an essential role in large scale software development. However, in considering the wider agenda of 'learning through creating computer models' – where process is more important than product – bricolage is profitable if it leads to a more valuable learning experience on the part of the student. In this context, what we wish to recognise as bricolage in conventional programming is the province of experts – who shape their programs skilfully in response to emerging requirements, rather than that of novices – who hack their program in an undirected manner. Without better support for bricolage, learners who are not expert programmers are forced to write programs in a style that may not be appropriate for them (cf. [TP91, Gar93]). With reference to Ben-Ari's quotes above, we see bricolage as an important aid in learning through model-construction, and in the prototyping – rather than the production – of a final product.

A further criticism of bricolage in computer programming has been made by Steve Talbott. He claims that – although an approach to programming may be conceived in isolation from the computer as bricolage – at the level of implementation, no matter how the student may think about the problem, it has to be coded in an abstract algorithmic method:

> 'While it may be legitimate to speak of the hard-edged and smoky *effects* the programmer aims at, the programming itself – which is the child's immediate activity – possesses a fundamental character that remains the same regardless of the style of the effects. The programmer may start with an interest in some aspect of the world, but the act of programming *forces* him to begin filtering that interest through a mesh of almost pure abstraction. To draw a figure with Logo, the child must derive a step-by-step procedure (algorithm) by which he can construct the desired result.' [Tal95, p157].

It would be hard to refute Talbott's claim that all computer programming involves some aspects of learning (typically abstract in character) that are not related to domain learning. For instance, in the use of Logo, a child requires some rudimentary knowledge of abstract computational ideas such as parameters and procedures. However, as Nardi has argued, a programming activity such as spreadsheet construction is more closely linked to domain learning than conventional programming [Nar93]. More generally, computer model-building that enables the modeller to focus primarily on the semantic relation β between the model and the real-world (cf. section 2.2.2) can alleviate the emphasis on abstraction. Current computer programming practice arguably obscures the possibility of a bricolage-based approach that does not force the learner to 'filter their interest through a mesh of almost pure abstraction'. As will be discussed in section 4.6, we believe that there is scope for alternative practices that give more access to a concrete, experiential learning style and move away from computer programming as an abstract preconceived activity.

Who is making it?	**Bricoleur**	**Planner**
What is being made?	Concrete artefact	Abstract program
Usage of tools	Uses whatever tools are already available	Uses standard tools with preconceived modes of use
Type of thought	Concrete	Abstract
Level of preplanning	Minimal	Entirely preplanned
Priorities	Negotiation, Engagement	Hierarchy, Abstraction
Knowledge Construction	Through open-ended interaction	Through analysis
Most suitable for	Non-programmers	Software engineers
Most similar software development style	RAD, XP	Conventional software development
Scale of application	Suitable for prototyping and small scale production	Suitable for mass production and large systems
Cognitive implications	Delays the construction of viable mental models	Demands mental models as a prerequisite

Table 4.1 – Differences between bricoleurs and planners, as identified in [TP91, Ben01]

The differences between the approaches to computer programming of the bricoleur and the planner that have been discussed in this section lie in the process and not the product. These are summarised in Table 4.1.

4.3 Situated learning

The proponents of situated learning claim that, for meaningful learning to take place, a learner must be placed within a realistic cultural and situational context. They also argue that the abstraction of problems from their real-world origins does not remove the complexity of the problem – it removes their essence [Lav88, Gog96]. In this section, we outline what situated learning is and how it relates to bricolage and planning approaches to model construction.

Situated learning is linked with John Dewey's claim that learning develops from experience and through social interaction [Dew38]. Learning in a situation is promoted in the idea of *cognitive apprenticeship*:

> 'Cognitive apprenticeship supports learning in a domain by enabling students to acquire, develop and use cognitive tools in authentic domain activity' [BCD89].

Apprentices, in trades such as mechanics and medicine, learn through authentic hands-on activities. Apprenticeship highlights how learning is an active process that is context-dependent, situated and cultural [BCD89]. Apprenticeship skills often arise out of what Lave and Wenger term *legitimate peripheral participation* [LW91]. Learners come to understand a skill by initially watching others perform the activity and gradually they take on some aspects of the role. Learning occurs through hands-on interaction. The full role is learnt legitimately through peripheral participation. This approach is in evidence in job shadowing for new employees.

Problems that are encountered in apprenticeship situations are different in character from problems met in abstract school-like settings. This is illustrated by Table 4.2, which has been extracted from Table 1 in [BCD89]. Table 4.2 can be interpreted as showing how problem solving in a situation (i.e. by practitioners) differs from problem solving in an abstract setting (i.e. by students).

	Students	Practitioners
Reasoning with	Laws	Casual models
Acting on	Symbols	Conceptual situations
Resolving	Well-defined problems	Ill-defined problems
Producing	Fixed meaning and immutable concepts	Negotiable meaning and socially constructed understanding

Table 4.2 – Comparing problem solving in situation and in abstract settings (adapted from Table 1 in [BCD89])

In the spirit of learning through cognitive apprenticeship, many researchers have called for a revolution in the school environment, where problems set for students are often devoid of real-world relevance (see e.g. [Dew38, Fri70, Ill71, Pap80, Pos92, Tal95, Opp97]). In [Res87], Lauren Resnick compares learning in a school environment with what she terms ‘everyday learning’, namely learning that takes place out in the world rather than in a classroom setting. She reasons that school learning often consists of individuals thinking abstractly about ways of solving problems where the emphasis is on the manipulation of formal symbols and the generation of general routines for solving classes of problems. In contrast, everyday learning often involves solving particular concrete problems where the structure of the task or the tools available can guide the solution to a problem, as in bricolage

[Lev68]. The differences between school and everyday learning are summarised in Table 4.3.

	In school	Everyday learning
Type of learning	Primarily mental 'THINKING'	Tool manipulation 'DOING'
Personnel	Individualised	Shared cognition
Applicability	General	Situation-specific
Objects of Thought	Concentrated on manipulating symbols	Contextualised reasoning

Table 4.3 – Comparing school learning and everyday learning [Res87].

CAI provides support for delivering and assessing traditional school problems. In CAI, the computer is used as a rigid device for asking students questions and eliciting responses from them, as in an instructionist approach. Providing support for Resnick's everyday learning requires understanding the part situational elements play in the learning process and their implications on the design of learning environments. Jean Lave promotes the idea of situated learning, claiming that the situation within which a problem is posed is not incidental to its successful solution, but often provides the enabling factor through which learners can solve the problem [Lav88]. Lave gave examples of adult shoppers and tailors who could perform mathematical calculations in a real-world situation, but were unable to solve the same problems posed as abstract mathematical questions.

The psychologist, John Anderson has raised two concerns about situated learning approaches. He argues firstly that abstract knowledge can be transferred between tasks; and secondly that studying parts of an activity in isolation before considering them in combination can be more effective than instruction in complex, social environments [ARS96]. Anderson also observes that learning that is situated in the world is not necessarily to be preferred. In many respects, learning to solve a particular instance of a problem in a specific real-world context is limited in comparison with understanding a general abstract solution to that particular class of

problems. However, the knowledge that leads to comprehension of the general is often gained from the experience of interacting with particular concrete problems.

Anderson's critique of situated learning motivates domain learning techniques that make it possible to combine situational and abstract approaches to learning. With reference to the EFL, such domain learning techniques need to support a fluid migration between concrete and formal learning activities.

Having introduced our broad perspective on constructionism, we now consider the extent to which this is supported by three domain learning techniques: concept mapping (section 4.4), conventional programming (section 4.5) and EM (4.6).

4.4 Concept mapping for domain learning

In this section, we consider concept mapping and discuss how this technique of representing emerging domain knowledge is related to the EFL. We shall argue that concept mapping is particularly closely connected with private and experiential learning activities in the EFL. We shall also argue that concept maps cannot support learning activities across the whole of the EFL, particularly the integration of experiential and formal learning activities.

4.4.1 Reviewing Concept Maps

Concept maps were introduced by Joseph Novak, an educational psyhcologist at Cornell University in the 1960's. Concept maps draw on the psychologist Ausubel's idea that the most important factor in learning is what the learner already knows [Aus68]. Concept maps are an example of a cognitive tool: a mental or computational device that can support, guide or extend the thinking process of its user [KJM92]. Cognitive tools support a constructionist approach because they actively engage learners in organising their knowledge to reflect their comprehension and conception of a domain [GKL+01].

Concept maps are a way of representing knowledge in a visual graph structure, as seen in the concept map of this chapter in Figure 4.2. Nodes in a graph represent concepts. Connections between nodes define relationships between concepts. Novak states that concept mapping comprises three important elements: concepts; propositions; and learning [NG84]. A *concept* in the context of a concept map is 'a perceived regularity, designated by a label'. Concepts can be well understood or only partially understood; the concept map does not require knowledge to be complete or formalised. A *proposition* in the context of a concept map is defined as 'a link between concepts'. Propositions can be labelled with arrows and, if appropriate, annotated with a description of the link that should be concise, as complete as possible, and understandable to another person. A set of concepts linked by propositions constitutes a *concept map*. Novak's third element of concept mapping, namely *learning*, is 'the active construction of new propositions'. Creating a concept map is an active learning process – the actual process of constructing it furthers our knowledge and affects the structure of the map itself.

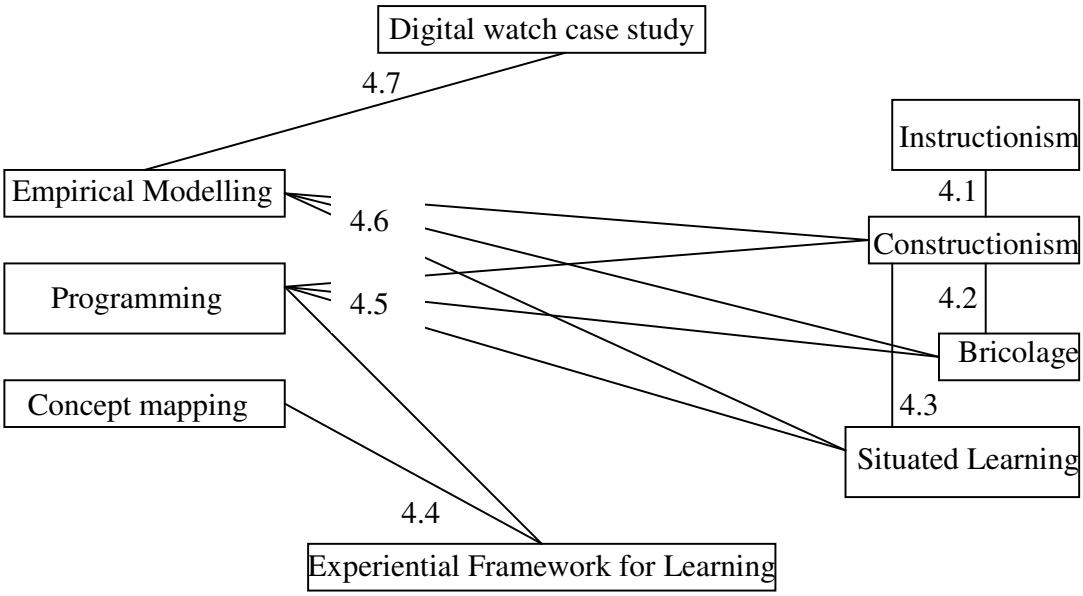


Figure 4.2 – An example concept map of this chapter

Mind maps, developed at the British Broadcasting Corporation (BBC) by Tony Buzan is an idea related to concept maps. They consist of one central word or concept, around which you draw the 5 to 10 main ideas that relate to that word [BB95]. The essential difference between mind maps and concept maps is that a mind map has one central concept but a concept map may contain several.

Learning through linking together concepts is endorsed by Marvin Minsky [Min86]:

> 'The secret of what anything means to us depends on how we've connected it to all the other things we know. That's why it's almost always wrong to seek the 'real meaning' of anything. A thing with just one meaning has scarcely any meaning at all'.

Uri Wilenski has also argued that establishing connections between concepts is a powerful learning objective. He describes a concept as being 'concrete' if it is well connected to other concepts, we have multiple representations of it, and we know how to interact with it in many modalities [Wil93]. In Wilenski's view, a concept becomes concrete through connecting it to other concepts, through many modes of interaction and through engaging in activities followed by reflection (a process he calls 'concretion'). Wilenski's use of the term 'concrete' differs from our use of the term in this thesis where it refers to *existing in a material form; real* [OED97], as opposed to being abstract. His notion of 'concretion' closely resembles the notion of the elaboration of a concept introduced in section 2.2.2.

The general nature of concept mapping means that it is widely applicable. Concept maps can be used, for example:

i) to see connections between current ideas. This is helpful in establishing our current state of knowledge.

ii) to connect new ideas to knowledge that we already possess. This is helpful in organising and understanding the place of new ideas. The assimilation of new knowledge is associated with adding new concepts and connections between concepts.

Concept maps can be constructed using pencil and paper, but there are also many software programs that support their construction (e.g. Inspiration [Ins93]). The major advantage of computer-supported concept mapping over pencil and paper approaches is the forgiving nature of the medium – links and concepts can be edited or removed easily. Anderson-Inman and Zeitz [AZ93] found that the computer-based medium encouraged learners to revise their maps as their understanding changes. In their research, Heeren and Kommers [HK92] concluded that concept mapping software should allow expressive flexibility so that students with different learning styles and techniques can demonstrate and develop their knowledge and understanding.

In summary, concept mapping is a constructionist approach to articulating current knowledge, organising current ideas and establishing links between current and emerging ideas. We now consider how concept maps can be viewed from the perspective of the EFL.

4.4.2 Concept maps and the EFL

Concept mapping, as an activity to explore and classify personal knowledge, has many characteristics in common with learning activities at the private end of the EFL. Concept mapping involves surveying a domain with a view to identifying important features and contexts. The construction of a concept map is not an objectively defined process; it is an iterative process that encourages reflection on construction as active learning to stimulate new knowledge [GKL+01]. The process of constructing a concept map to articulate our current knowledge is related to Gooding's notion of a construal (cf. section 3.4.1) [Goo90]. A concept map can be viewed as a construal because it is a concrete artefact being used to understand a phenomenon. Gooding views construals as concrete and situational interpretations of unfamiliar experience and trial interpretations – this characterisation is consistent with the role that concept maps play in representing the known and connecting the known to our emerging understanding.

In other respects, concept mapping does not necessarily resemble EM at the private end of the EFL. Like the jugs visualisation in Figure 2.19, the concept map in Figure 4.2 serves as an artefact, but its nodes depict concepts that are much more sophisticated than the primitive observables that represent the current contents of a jug. Likewise the relationships between the nodes in the concept map in Figure 4.2 are less precisely prescribed than the dependencies between primitive observables. This loose analogy between observables and nodes, and dependencies and propositions was exploited by Wong in his design of the Dependency Modelling Toolkit (DMT) [Won03]. Figure 4.3 shows the DMT representation of the jugs model shown in Figure 2.19.

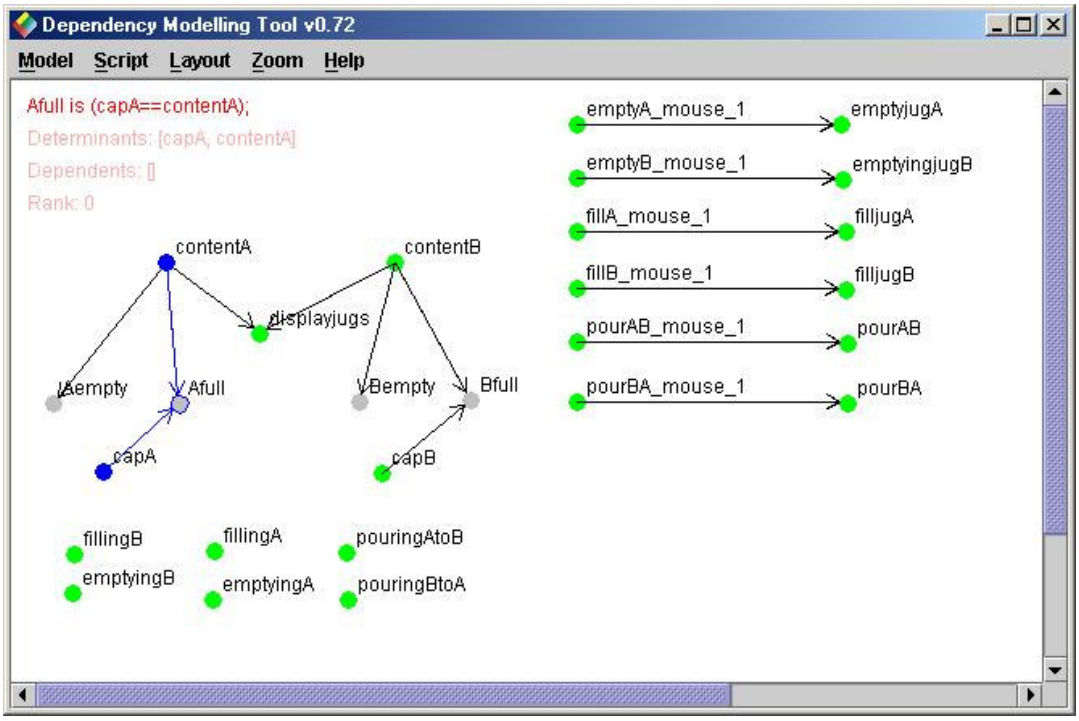


Figure 4.3 – The simple jugs model from Figure 2.19 in the DMT

In [Won03], Wong envisages the DMT as the basis of a visual environment for EM. It is evident that such an environment could be used to construct complex definitive scripts that support EM at the concrete end of the EFL. In effect, Wong has identified a particular kind of concept map that can be used to migrate from primitive learning activities towards objective knowledge (cf. the discussion of heapsort in section 6.3

and in [Run02]). Though concept maps are in general suitable for expressing our current level of understanding and assist as a learning device in assimilating new knowledge neither computer-based nor pencil-and-paper concept maps can integrate experiential and formal learning activities. In effect, the concept map is set aside when constructing a computer model based on the knowledge that it embodies.

4.5 Programming for domain learning

Computer-based modelling has had a crucial role in the development of constructionist approaches to learning. The future prospects for constructionist model-building approaches will depend on the quality of the paradigm being used to build models. For this reason, we need to consider conventional approaches to programming and the implications for their support of constructionism. In this section, we first review the main tenets of the Object-Oriented (OO) approaches to model construction that are so prominent in modern approaches to software development (e.g. [Jac92]). We then describe how programming can be viewed from within the EFL, and conclude that programming as classically conceived has deficiencies with respect to constructionism because of its emphasis on preconception, abstraction and generality over flexibility and experimentation. The work reported in this section draws on material from three previous EM theses, namely Allan Wong's research on software system development in EM [Won03], and comparative studies of OO and EM in Timothy Heron's MSc thesis [Her02] and Ruyuan Wang's MSc thesis [Wan03].

4.5.1 Programming from a learning perspective

If computer-based model building is to support the educational objectives of constructionism, then the programming activity must be well-aligned to domain learning. In general, a program is primarily conceived as a piece of software that fulfils some purpose based on a set of requirements. Programming involves writing computer code to satisfy the intended specification of the program. In general, the

programmer's objective is not to learn from the construction of the program, but to deliver a final product. In the context of the constructionist agenda, the main question is whether the programming process is beneficial from a learning perspective.

If we consider the programming process as having two aspects – the design and analysis of the domain, and the writing of code – then learning is predominantly associated with the design and analysis activities. In the constructionist approach to learning, the evolving artefact is an embodiment of ignorance, and interactions with it serve to shape the artefact and its interpretation. Knowledge of the domain is acquired as the artefact is constructed. The roles that artefacts and domain knowledge play in programming are in contrast quite different. While knowledge of the domain plays a fundamental role in programming, the only essential knowledge is concerned with the intended interaction and interpretation of the program that is prerequisite for conventional programming. There *is* a role for interaction with artefacts (as represented by use cases, UML diagrams [JCJ+92], and prototypes of various kinds) but these artefacts embody the knowledge prerequisite for programming and are typically discarded once programming commences.

In previous EM theses, particular case studies have been used for comparative studies of model building in EM and program construction using OO principles. We will discuss these briefly to highlight their key conclusions. Heron's MSc thesis compares EM and OO development with reference to two case studies: a game of draughts [EMRep, draughtsRawles1997]; and a vehicle cruise control simulator (VCCS) [EMRep, cruisecontrolPavelin2002]. From these comparisons, he concludes that changes in the OO model are as easy to make as in EM as long as the change has been preconceived in advance. In his study of the VCCS model, he observes that the OO programmer has more to do; for example, all the method calls used to propagate changes of state have to be worked out and ordered before coding. Wang's MSc thesis discusses the VCCS models developed in OO and EM in much greater depth. She concludes that the functionality of the object-oriented VCCS is abstractly and precisely prescribed, whereas in EM the VCCS model can take account of issues that

are outside the normal scope of correct operation and that might be useful for learning. Her key observation is that use-cases [Jac92] only give an account of typical interaction between a system and external actors. However, from a learning perspective this requires all interactions to be conceived before any programming takes place – there is no room for evolution of ideas in tandem with the developing program.

Wong's PhD thesis contains a comparative study of EM and OO development of a dishwasher model. Through implementing models using both approaches, he identifies a number of characteristics from which to compare the two development styles [Won03]:

i) Modelling focus – In EM, the focus is initially on the subjective interpretations of the modeller. Observations recorded in the EM model are based on the imagined interactions in the system, but there is no circumscription of the system boundary. In UML, the focus is on modelling the structure and behaviour of the system. Once the system boundary has been established, the system is constructed in isolation from its operating environment.

ii) Interactiveness – In EM, the modeller can always get feedback on any part of the model, because there is always a working model. Experimental interactions to test new insights or consolidate current understanding are part of the modelling approach. In UML, diagrams are abstract representations that are not primarily to be interpreted with reference to a particular state of a system. The main role of the UML is to specify system behaviour; experimental interactions are limited in their extent and changes must be compiled into an executable program for each change.

iii) Comprehension – In EM, comprehension is typically gained from experimental interactions with the model. Exploration in this sense is not part of the UML approach; the diagrams purely represent the predetermined relationships between components rather than allowing the exploration of possible alternatives.

iv) Openness – In EM, there is no particular viewpoint from which a model should be built, and the model emerges from experimental interactions rather than through prior circumscription. There is no system boundary within which construction must take place and the modeller is always in the position of being able to step in and make experimental changes. In UML, a fixed set of diagrams guides the modelling process. Use-cases and class diagrams constrain the interactions between components until implementation is clear. The emphasis is on circumscription as the guiding principle of system construction.

v) Interfaces – In EM, model building and user interface construction occur within the same framework. In UML, there is no support for specifying interfaces of the target system. Interfaces must therefore be constructed and tested separately.

The comparative studies of EM and OO approaches to software development highlight the following distinctions:

- in conventional programming, the artefacts generated to represent the program requirement reflect richer knowledge of the application domain than the program itself. In EM, in contrast, the knowledge prerequisite for realising an abstract behaviour is cultivated throughout the modelling process and is never discarded.
- in conventional programming, domain knowledge and identification of purpose are essential prerequisites for writing a program, and optimisation to purpose is a measure of program quality. In EM, in contrast, the identification of purpose emerges during the process of model construction and this identification does not compromise the character of the model as a construal.

In the next section, we interpret the conclusions from the comparative studies of EM and programming with reference to the EFL and our broad perspective on constructionism.

4.5.2 Conventional programming, constructionism and the EFL

Conventionally, programs are built by defining an external system boundary and circumscribing the possible interactions and interpretations. With reference to the EFL, these principles for programming are targeted at the identification of common experience and objective knowledge together with symbolic representations and formal languages. In other words, programming is associated with activities at the theoretical end of the EFL. What is more – though computer programs may be able to support learning activities at the end of the EFL – conventional programming activity itself is very different in character from the learning activities at the empirical end of the EFL.

The discussion above indicates that conventional programming is only well aligned to a small subset of the learning activities that make up the EFL. It is obvious that model construction using a conventional programming language satisfies Papert's basic definition of constructionism in so far as the act of programming involves actively creating a personally meaningful entity. However, as we shall argue below, conventional programming lacks the characteristics needed to give full support to the broad perspective on constructionism that we adopt in this thesis (cf. Figure 4.1).

The aspiration in bricolage and situated learning is to support concrete learning through interaction and exploration. However, this aspiration is not being well served by the current emphasis on computers in learning. Our experience of EM has led us to believe that the paradigm used for model building has a significant bearing on the quality of support for the constructionist approach that computers can provide. As we have argued in a previous paper:

> 'there are profound conceptual issues to be addressed before such a shift in emphasis [from 'planning' to 'bricolage' in computer model construction] can be achieved: the fundamental preconceptions about computation that inform classical computer science are ill-oriented for this purpose' [RB02].

We agree with Ben-Ari (cf. section 4.2) that computer programming – as practised – contains elements of bricolage [Ben01] and with Brooks's view [Bro95] that programmers see their work as a craft where they wrestle with incompletely understood meaning. Software development identifies two general phases, of knowledge gathering and knowledge deployment. The first phase is one of engagement with the world, and of preliminary knowledge gathering, whilst the second is a complementary phase where knowledge is deployed in program specification and design. Interaction with artefacts is common in gathering knowledge (through the building of prototypes, the creation of use-cases and UML diagrams [JCJ+92]), but this knowledge is targeted at achieving a specific functional goal. The obligation to frame knowledge in this goal-directed fashion reflects the conception of programs within the classical theory of computation. This means that in practice, even though the phases of knowledge gathering and deployment are interleaved, the intimate relationship between the two is obstructed by the way in which knowledge is deployed.

The relationship between knowledge gathering and knowledge deployment in computer programming reflects a commonly accepted view of the relationship between experiment and theory in science [LJ98]. Our concern about the classical separation between the open-ended experience that informs requirements and the circumscribed behaviour of a program mirrors Gooding's concern about the bifurcation of the scientist's world into the empirical and the literary:

> "Scientists' descriptions of nature result from two sorts of encounter: they interact with each other and with nature. Philosophy of science has, by and large, failed to give an account of either sort of interaction. Philosophers typically imagine that scientists observe, theorize and experiment in order to produce general knowledge of natural laws, knowledge which can be applied to generate new theories and technologies. This view bifurcates the scientist's world into an empirical world of pre-articulate experience and know-how and another world of talk, thought and argument. Most received philosophies of science focus so exclusively on the literary world of representations that they cannot begin to address the philosophical problems arising from the interaction of these worlds: empirical access as a source of knowledge, meaning and reference, and of course, realism." [Goo90, p. xi]

The alternation between requirements gathering and program specification testifies to the bifurcation of the computer scientist's world. This is evidenced by the problems encountered in conventional programming approaches when seeking to admit truly experimental interactions and achieve flexible adaptation of the program as it is being developed. In our view, software development to support constructionist use demands a conceptual integration of the pre-articulate exploration and formalisation of knowledge that are respectively associated with the phases of knowledge gathering and knowledge deployment [RB02].

The above discussion suggests that conventional programming is inadequate as a basis for a general constructionist approach to model building (cf. the observations by Soloway [Sol93] and by Steinberger [Ste94] that general purpose programming languages obstruct meaningful domain learning). This may be a factor in accounting for the relative lack of popularity of programming as a learning tool for the non-specialist (cf. [Nar93]), and the emergence and subsequent disappearance of Logo from the National Curriculum in the United Kingdom (cf. [NH96]). In the following section, we consider the merits of EM as an approach to model construction that enables the conceptual integration of concrete and formal learning required to support our broad notion of constructionism.

4.6 Empirical Modelling, constructionism and the EFL

In this section, we discuss the connections between EM, constructionism and the EFL. The strength of these connections determines the extent to which EM can support our broad notion of constructionism. Like conventional programming, EM evidently satisfies Papert's basic definition of constructionism. In section 3.6, we showed that EM can support a wide range of learning activities that are identified in the EFL. It remains to show that EM is well suited to supporting broader aspects of constructionism such as bricolage and situated learning.

4.6.1 Empirical Modelling and bricolage

In this section, we discuss how EM model construction supports bricolage. To do this we review the literature on both EM and bricolage, to identify their common points. Bricolage and EM originate from entirely different contexts. In developing the concept of bricolage, the anthropologist Levi-Strauss was concerned with the construction of physical artefacts within 'primitive' societies. In contrast, EM is concerned with computer-based model construction that has been observed in practice, primarily in the work of computer science students at the University of Warwick over the past fifteen years. This has involved the construction of several hundred models in connection with student projects and academic research (see [EMRep] for a representative sample).

Evidence that the EM modeller is a bricoleur rather than a planner can be seen in key phrases taken from Russ's comparison of EM and programming in [CRB00]:

- 'there is really no counterpart in EM to the 'planning' phase. ... conceptual modelling in EM can conveniently be directly put into a script with a visualisation and experimented with on the computer.'
- 'it is significant that testing occurs in advance of any commitment to a particular form of program.'
- 'in an EM development it is typical that the interface is left until an advanced stage of the development – when the purpose and requirement has been clarified through extensive use of the very open-ended phase of model construction and exploration.'

In EM, the purpose of model building may not be initially clear:

> 'The objective of a (student) project has often been uncertain at the early stages, and a theme has emerged as the model-building activity proceeds incrementally. The profile of work on the project is distinctively different from that practised in other paradigms, such as object-oriented software development. Students typically carry out significant model construction even at the early stages, and are guided by this in their strategic decisions' [Bey01].

Though the modeller may have a general problem in mind, the initial phase of work involves surveying tools and models both to identify useful resources and to shape the provisional direction of development. This preliminary activity typically influences the modeller's original conception of their project. This resonates with Levi-Strauss's account of bricolage in the preliminary stages of a project:

> 'Consider the bricoleur at work and excited by his project. His first practical step is retrospective. He has to turn back to an already existent set made up of tools and materials, to consider or reconsider what it contains and to engage in a sort of dialogue with it and, before choosing between them, to index the possible answers which the whole set can offer to his problem.' [Lev68, p18]

> 'Once it materialises the project will therefore inevitably be at a remove from the initial aim, a phenomenon which the surrealists have felicitously called "objective hazard".' [Lev68, p21]

The influence of the developing artefact over the modeller's conception of his or her project is prominent throughout the development of an EM model, and the final outcome of a project may differ significantly from the initial idea.

As highlighted above, the process of model construction in EM is one of negotiation; the modeller uses the partially constructed artefact (and its real world counterpart if it exists) to further refine their current understanding of it. This emphasis on understanding the artefact under construction is also seen in bricolage. Levi-Strauss says of model building:

> 'Now the model being an artefact, it is possible to understand how it is made and this understanding of the method of construction adds a supplementary dimension.' [Lev68, p24]

The products of EM and bricolage both relate to the embodiment of rich experience in a real-world artefact. In both, the emphasis is on human engagement in the model building and concrete rather than abstract representations of knowledge. Levi-Strauss

refers to the products of bricolage as 'miniatures' and stresses the importance of real-world human engagement:

> '… miniatures have a further feature. They are 'man made' and, what is more, made by hand. They are therefore not just projections or passive homologues of the object: they constitute a real experiment with it.' [Lev68,p24]

The emphasis in EM and bricolage is on the learning that occurs during construction of an artefact rather than on the finished product, as illustrated in the quotes below:

> '... the 'bricoleur' also, and indeed principally, derives his poetry from the fact that he does not confine himself to accomplishment and execution ... The 'bricoleur' may not ever complete his purpose but he always puts something of himself into it.' [Lev68, p21]

> 'Computer models constructed using Empirical Modelling principles are not to be viewed as implementing an abstract mathematical model. Their significance is instead similar to that of the physical model that an experimental scientist might build to account for a phenomena, or that an engineer constructs to prototype or test a design concept.' [BS99]

The qualities of bricolage in relation to the EFL can be inferred from Table 4.1. The defining characteristic of bricolage – of intimate engagement through interaction with the artefact – is found in activities at the empirical end of the EFL. Planners – who preconceive modes of use and functionality of their product before programming – do not engage with the empirical learning activities during construction. This approach is only suitable if they have a good understanding of the situation they are modelling. Model building approaches that embrace bricolage must be capable of supporting learning activities at the experimental end of the EFL.

In summary, the discussion in this section has illustrated that there are close links between EM and bricolage, and that both offer support to the concrete learning activities at the empirical end of the EFL.

4.6.2 Empirical Modelling and situated learning

In situated learning, hands-on interaction with tangible artefacts guides the learning process. The technologist, John Seely Brown claims that educational practice has been dominated by a belief that conceptual representation (typically abstract and symbolic) is of the most importance, and argues that situated cognition, giving activity and perception a prior place over representation, could solve some of the problems in school learning:

> 'For centuries, the epistemology that has guided educational practice has concentrated primarily on conceptual representation and made its relation to objects in the world problematic by assuming that, cognitively, representation is prior to all else. A theory of situated cognition suggests that activity and perception are important and epistemologically prior -- at a non-conceptual level -- to conceptualisation and that it is on them that more attention needs to be focused. An epistemology that begins with activity and perception, which are first and foremost embedded in the world, may simply bypass the classical problem of reference -- of mediating conceptual representations' [BCD89].

In Seely Brown et al [BCD89], situated learning is associated with 'cognitive apprenticeship' in which learning progresses from activity embedded in a situation to general principles of the culture. Apprenticeship goes together with coaching, and students undertake modelling in situ that is scaffolded to get them started in authentic activity.

To give computer support to situated learning as characterised by Seely Brown requires a modelling approach that gives a high priority to activity and perception. As discussed in section 3.3.2, in EM, activity and perception are viewed as 'epistemologically prior to conceptualisation' and the idea of 'one experience knowing another' may be seen as 'bypassing the classical problem of reference' (cf. [Run02, chapter 2]). The fundamental concept of EM is not that 'activity and perception are first and foremost embedded in the world', but rather that activity and perception are first and foremost embedded *in personal experience*, that can then be classified as subjective or objective. In EM, activity and perception that is embedded

in objective experience is nevertheless important. In such a context, EM is a form of situated modelling.

Situated modelling is an essential constituent of computer support for situated learning. The situated nature of EM has been discussed in detail in connection with its potential role in software development [Sun99]. In particular, EM can be seen as meeting Goguen's concern for situatedness in requirements analysis:

> 'EM activities are carried out with reference to an external situation, even though in practice this situation can be imaginary rather than concrete. Practical experience of EM confirms its status as a situated modelling method, and activities in EM exhibit Goguen's "qualities of situatedness": emergence, contingence, locality, openness and vagueness [Gog96]. The main reason why EM exhibits these qualities is that, because of the nature of the modeller's interaction, the process of formulating definitive scripts is never separated from the modelling context.' [BS98]

The qualities of EM as a situated modelling approach are that:

> 'the properties of openness and situatedness reduce the separation of model and world and offer the possibility of a user deriving qualitative knowledge of the world through interactive use of the model' [BRR00].

These qualities are significant for the computer support that EM can give to situated learning. They support the learner in interacting with artefacts and developing practical skills in particular concrete situations and also assist the learner in applying and understanding general abstract solutions.

The organisation of the constituent learning activities within the EFL is consonant with Seely Brown's claim for the prior place of activity and perception over conceptual representation. Modelling an artefact is situated; its construction takes place with and during consultation of a real-world referent. In situated modelling in EM, the important activities have direct counterparts in the real world: namely the identification of salient features (through observation); understanding the nature of indivisible changes in the referent (through dependency); and determining who, or

what, is responsible for those changes (through agency). With reference to the EFL, situated learning is concerned with interaction and experimentation with particular concrete instances of a problem in context and this corresponds to learning activities at the concrete end of the EFL.

In summary, the discussion in this section has illustrated that there are close links between EM and situated learning, and that both offer support to the concrete learning activities at the empirical end of the EFL.

The above discussion has considered situated factors in model construction. The benefits of EM as an approach to situated modelling can also be seen in relation to learning environments where situational factors have an important role to play. This is illustrated by the Clayton Tunnel model (see section 2.4.5). In this model, each participant views the situation from his or her own perspective on a different computer. These perspectives only contain the elements that each participant can see and interact with. For example, train drivers can only interact with controls related to their train and can only see the signalman from particular parts of the track. Despite the limited nature of the visualisation in this model, the open-endedness of the interaction between participants and environmental factors in the situation offers elements of realism that would not necessarily be within the scope of an immersive environment. For instance, it is in principle quite straightforward to simulate failures of communication due to misunderstanding; mechanical breakdowns; and environmental variations that lie within the frame of the construal.

4.7 The digital watch case study

We conclude this chapter with a concrete example to illustrate how EM supports the broad view of constructionism taken in this thesis. The construction of a digital watch model is an example of situated modelling in EM since the referent is an actual digital watch. The digital watch model has been used to illustrate many aspects of EM; for

more detailed information, see [BC95, RBF00, FB00, BRW+01]. Relevant issues illustrated in this section include:

i) bricolage style development involving the re-use of previous models.
ii) knowledge gained through hands-on interaction rather than reference to a user manual.
iii) the combined use of formal and informal artefacts.
iv) multiple perspectives on digital watch design and use.
v) incorporating situated aspects of digital watch use in the model.

An important facet of bricolage is the unprescribed path that the development of the artefact follows. Bricolage involves subjective interaction of the modeller with the artefact, and the character of both the artefact and the modeller's relationship to it are subject to change through exploratory interaction. This can be illustrated with reference to the digital watch through the re-use and extensions made to the model by a group of modellers over an extended period of time (see Figure 4.4).

Four different people were involved in the development of the digital watch model over a period of eight years. The collaboration and communication between the modellers involved was limited; at each stage, the artefact itself embodied much of the knowledge that guided its future development. Beynon constructed the initial digital display and a basic statechart (as presented in [Har87a]) in 1992. Richard Cartwright added the analogue clock and the digital watch buttons in 1994 [BC95]; he also developed a chess clock variant of the model [BC95]. The functionality of the digital watch at this stage was restricted to that of Harel's original statechart, which describes the display functions in detail but omits the functionality of components such as the stopwatch and the mechanisms for setting the time. The full functionality for the watch, together with buttons for updating the date and time, was added by Carlos Fischer in 1999 [FB00]. In 2000, I altered the functionality to match the watch I was using at the time and added an alternative visualisation that aided comprehension of tasks that could be undertaken with the watch [RBF00]. My version of the model is shown in Figure 4.5.

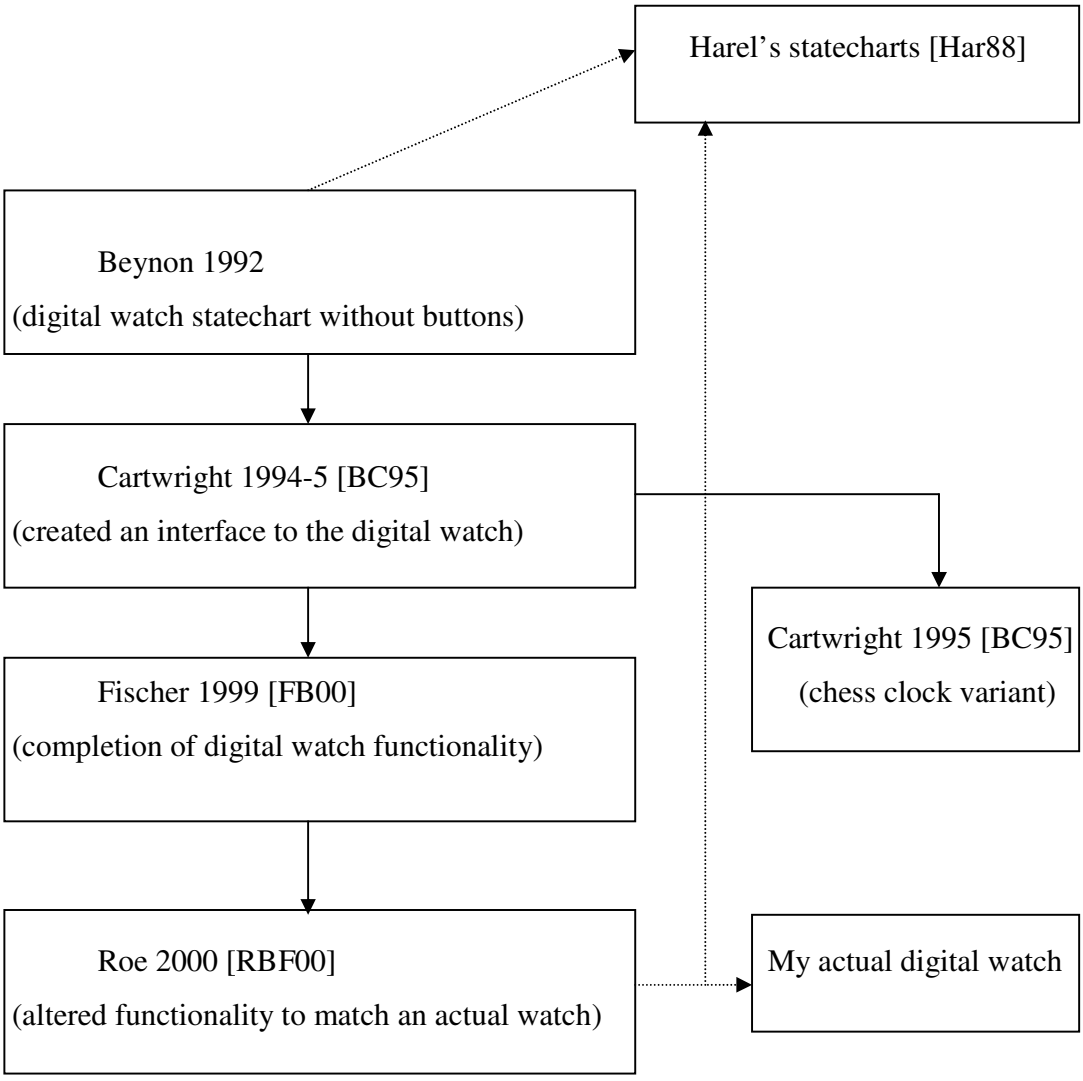


Figure 4.4 – The development history of the digital watch

Conceptually, the model was constructed in one continuous sequence of interactions as in one 'stream of thought' (cf. section 3.3.2). The introduction of new definitions to attach components to the model, and the addition of procedural actions to simulate agents, are examples of typical interactions involved in the development. As in bricolage, the impact of additions to the model guided the modeller in making future changes. There is no sense in which the model in its current state represents a finished product or the development process has reached a point of closure; within EM, the

model always remains open to potential future revision and extension. For instance, the functionality of the digital watch could be extended to include the heart rate monitoring facilities on a sports watch.

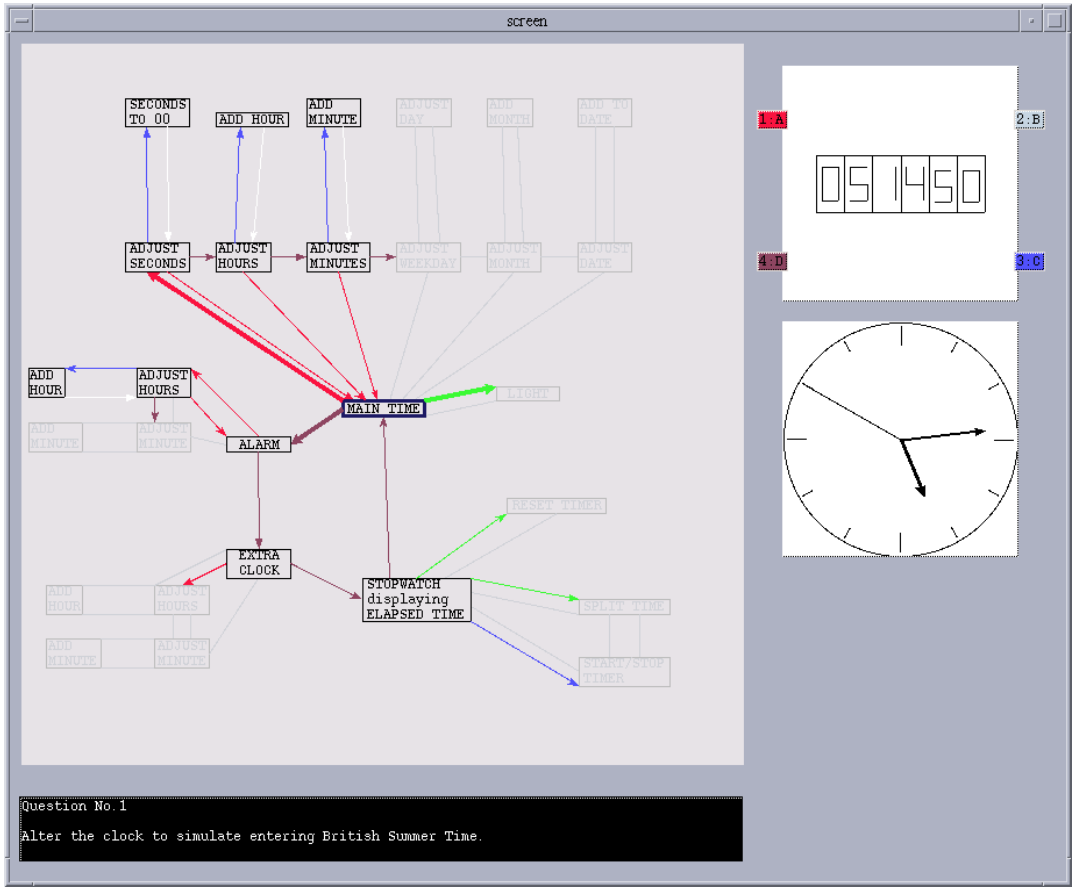


Figure 4.5 – The digital watch artefact (top right), an analogue clock (middle right) and a mental stategraph (left)

The model consists of a digital watch artefact, a corresponding analogue timepiece and a 'mental stategraph' that indicates the user's current level of familiarity with the states of the digital watch. The buttons labelled A,B,C,D on the digital watch interface in Figure 4.5 depict four physical buttons that were similarly located on my personal watch. The functionalities of these buttons correspond to the physical watch operations. There are many dependencies present in the artefact. For instance, the visible elements of the watch are dependent on its internal state. Button pressing

allows certain patterns of state change corresponding to watch operations to be performed through the interface. These patterns reflect the circumscribed functionality of the actual digital watch. They are expressed on the mental stategraph by using coloured arrows to represent the state transitions that will occur if a button is pressed. The current state is indicated with a bold border. For example (see Figure 4.5), if button A is pressed then the watch enters the time setting mode, and if button D is pressed then the watch enters the alarm mode.

Beynon's initial model was originally conceived with Harel's agenda of using visualisation to support complex system development in mind [Har87b, Har88]. It features a statechart (a concept introduced in [Har87a]) that is used to specify state-transitions and events. Statecharts are much richer than traditional state transition diagrams because they exploit the notions of depth and orthogonality [Har88]. A statechart is most suitable for recording reliable and comprehensive system knowledge; it is not necessarily the most appropriate way to represent our emerging understanding of the system that it describes.

The interface to my digital watch model was designed to be as faithful as possible to my construal of the behaviour of the actual watch. In Figure 4.5, the visual organisation of states in the stategraph reflects my conception of the main and subsidiary functions of the watch. The changes of state are precisely correlated to the actions of pressing and releasing buttons. In the stopwatch component, the effect of a button press is dependent on the current state of the stopwatch: specifically the transition made in response to button press B is determined by whether or not the stopwatch is running. In all these respects, the stategraph differs from a statechart. Its primary role is to provide an experiential rather than an abstract representation of state.

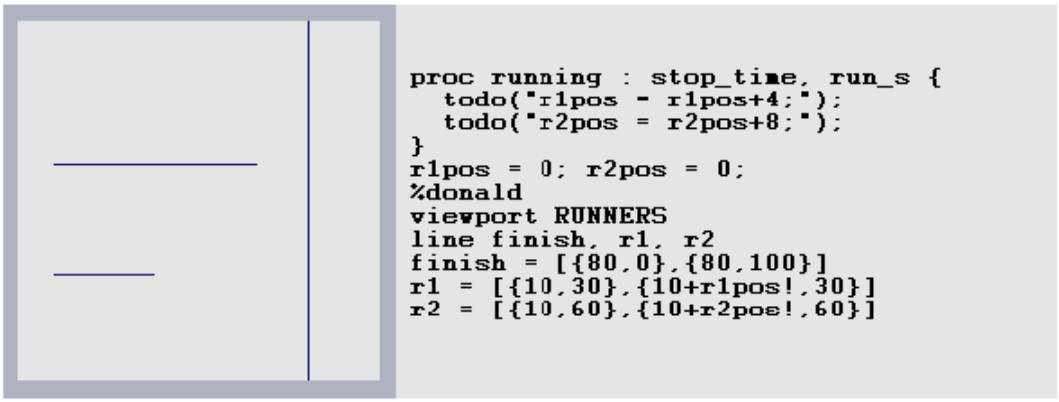
The stategraph is better oriented than the statechart towards studying a learner's interaction with an artefact as it is conceived by Carroll (cf. section 4.1.2 and [Car90]). The learner does not have the comprehensive knowledge of the artefact that

the user manual and the statechart abstractly supply: they develop understanding haphazardly through experiment with the artefact itself. The stategraph in the digital watch model is intended to be helpful in tracing a learner's emerging understanding. With this in mind, the stategraph in Figure 4.5 discloses states to the user as they encounter them. This is a preliminary step towards representing the current knowledge of the learner more accurately. For instance, from Figure 4.5 we can deduce that the user has explored changing the time of the main clock and altering the alarm, but has not yet encountered the other features. Visualisation of this kind is only a first step towards evaluating a learner's understanding; the fact that a learner has encountered a particular function is no guarantee of understanding. More insight can be gained from using the visualisation in conjunction with a worksheet that specifies activities to be undertaken by the learner, such as setting the clock to British Summer Time (see Figure 4.5).

It is evident that the digital watch model supplies a useful environment for situated learning. For instance, it can be used to learn about telling the time on digital and analogue clocks, to learn about the relationship between different time zones and to understand many issues concerning the design and use of clocks, stopwatches and digital watches (cf. Appendix D). Specific applications of the watch model in situated learning are targeted by the worksheet questions that are incorporated into the model.

To illustrate situated use in a broader context, the modeller can introduce extra observables to embellish the current model. These observables – rather than referring to the digital watch itself – will be situational in nature. Situational observables become significant when specific user activities involving the watch are being studied, such as when the stopwatch feature is being used to record the finishing times for two runners in a race. Figure 4.6 is a simple line drawing to represent a race between two runners. The horizontal lines extend to the right towards the vertical finishing line when the stopwatch is started. The watch user can then record the finishing times of both the runners using the 'split time' feature. The TkEden definitions and action required to model the runners in this extension is also displayed

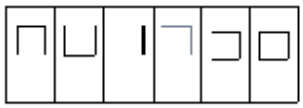
in Figure 4.6. Note that there are two aspects to this extension, of the model: firstly, the actual script has been changed; and secondly, the way that the user interacts with the script changes to reflect the new situational emphases.



```
proc running : stop_time, run_s {
  todo("r1pos = r1pos+4;");
  todo("r2pos = r2pos+8;");
}
r1pos = 0; r2pos = 0;
%donald
viewport RUNNERS
line finish, r1, r2
finish = [{80,0},{80,100}]
r1 = [{10,30},{10+r1pos!,30}]
r2 = [{10,60},{10+r2pos!,60}]
```

Figure 4.6 – Situational observables – timing two runners.

The potential for extension of the digital watch model is such that we can take account of the exceptionally rich aspects of experience and knowledge that can inform everyday interaction. Figure 4.7 depicts the digital watch display as it might appear when observed when lying in bed, where it may be partially obscured.



```
%donald
within led1 {
on3 = false
on5 = false
on7 = false
}
```

Figure 4.7 – A partially obscured digital display

In this situation, careful observation of the clock is required before we can establish the correct time. This will take into account such factors as the parts of the digits we can observe, our knowledge of the patterns that govern the digits changing and contextual knowledge, such as our estimation of the current time [BRW+01].

4.8 Summary of the chapter

In this chapter, we have considered how techniques for domain learning are related to the educational theories of constructionism and instructionism. We have shown that there is a paradoxical aspect to the way that conventional programming offers support for constructionism. It satisfies Papert's basic definition of constructionism, yet on deeper inspection it is not well aligned to the constructionist learning activities in the EFL, or to the ideas of bricolage and situated learning. Further, we have shown that model construction in EM can support the ideas of bricolage and situated learning, and the broad range of learning activities in the EFL. It is for this reason that we regard EM as establishing a more intimate link between domain learning and model construction than other approaches.