Akademie věd České republiky
Ústav teorie informace a automatizace

Academy of Science of the Czech Republic
Institute of Information Theory and Automation

Research Report

Jindřich Bůcha

ON EFFICIENCY OF LEARNING: A FRAMEWORK AND JUSTIFICATION

Version 1.0

No. 1983 May 2000

GA CR No 102/99/1564
Research and Education in Adaptive Systems

ÚTIA AV ČR, P. O. Box 18, 182 08 Prague, Czech Republic
Telex: 122018 atom c, Fax: (+420)(2)6884 903
E-mail: utia@utia.cas.cz

This report constitutes an unreferred manuscript, which is intended to be submitted for publication. Any opinions and conclusions expressed in this report are those of the author(s) and do not necessarily represent the views of the Institute.

On Efficiency of Learning: A Framework and Justification

Jindřich Bůcha[1]

Abstract. A conceptual framework whose goal is the efficiency of machine learning is proposed and justified. The framework is designed in a broader context, that of problem solver (PS). The design is solved as an integration of all basic cognitive functions and as a software-engineering problem. Many (one hundred) requirements imposed on PS are considered. The most important are the object-oriented nature of the PS environment, reflexivity of PS, and central role of tool and shifted border.

1 INTRODUCTION

In the paper, a framework for machine learning is proposed. It is a new framework[2], even if it is "99%-based" on known concepts, based on their integration. There are a lot of them. They are unified in one super-principle (see later).

The sophisticated robots on RoboCup-99 did not demonstrate a very intelligent game. This was not because they were not able to form teams of autonomous robots, demonstrate multi-source fusion of perception, and use multi-agent game theory. I contrast these important topics covered in [1], with my explanation: The robots were inadequate mainly because they **could not identify or learn** some characteristics of simple and obvious *objects* from their environment. For example, sometimes they could not tell a ball from a teammate. At other times they could not recover from deadlock situations, e.g. when the movement of one robot's leg was blocked by its opponent's leg. The robots could not identify the field edge and repeatedly fruitlessly tried to overcome it. More exactly, even if some robot teams claimed learning[3], it was obviously not a fully adequate learning. Robots demonstrated only a very simple kind of learning. This is not limited to RoboCup only; it is typical for most of AI. It can be seen from the survey [2]. I also surveyed the machine learning group[4] and concluded that more complex approaches to learning are probably not covered there.

I would roughly characterize **simple learning** as a one-shot creation of knowledge, describing one dependency or one function of small part of an environment, represented, for example, by one neural network, one set of rules, or one decision tree, created using one technique, and applied in a manually selected part of some cognitive function. On the contrary, I would characterize **complex learning** as a cumulative creation of knowledge with many parts, which describes many related dependencies and functions of whole environment, and is structured, preferably in object-oriented way, represented e.g. by many ontologies, neural networks, or sets of rules, created by different techniques, applied and integrated in different parts of different cognitive functions, applied in the whole real world. These are two extremes on a scale of kinds of learning. The state-of-the-art is, of course, somewhere in-between, significantly closer to simple learning. It contrasts with some other AI areas, like the use of ontologies or knowledge engineering, where we do use complex knowledge structures [3], [4].

To apply **learning to ontologies** it is necessary to integrate both learning and ontology areas, i.e. modify each area. For example, learning could accept complex knowledge structures; ontologies could accept the approach to approximate knowledge. This would modify e.g. some design "principles" that "have been proved useful in the development of ontologies", like clarity, completeness, and coherence [3], [5]. On the other hand, this may solve e.g. one often-mentioned problem, that "concrete elements are in many cases practically more usable then abstract ones" [3], [6], [7].

The more complex learning is rare, but not absent [8], [9]. Existing examples may provide a basis for the design of such learning [10], [11]. The most advanced contemporary systems seem to be PRODIGY [12] and Soar [13]. These systems have shown that the integration of several cognitive functions is possible, useful, and efficient. From machine learning research, we have gained insight into the suitability of using previous knowledge. Hence, my approach to learning can be characterized concisely as **"learn from learning"**. This means that it is suitable to use knowledge, which we have gathered during the exploration of learning (phenomenon), and apply it as meta-knowledge in the design of improved learning (algorithm). I will refer to this as the **super-principle**. Its more exact meaning greatly depends on one's definition of learning. I understand learning in a very broad sense. This should be clearer after reading this paper (I hope).

The body of previous knowledge is vast. What parts should be used? The simplest choice is to try to use everything. This was already proposed by Minsky[5] [14]. From that time, our understanding and knowledge of learning has grown significantly. My approach is to use **all meta-knowledge that can be integrated**

---

[1] Institute of Information Theory and Automation, Pod vodárenskou věží 4, 182 08 Prague 8, Czech Republic, email: bucha@utia.cas.cz

[2] I'll stress this issue as it's in contradiction with the standpoint of reviewer of previous version of this paper.

[3] I counted 40% of best teams (teams up to the third place) from all RoboCup-99 leagues claiming learning.

[4] machine-learning@eGroups.com

[5] It has not been implemented, as far as I know.

into an efficient learning system. This paper proposes and justifies a framework that aims at such integration. To design and build a corresponding system is a **formidable task**. However, realize four things: First, there is a great redundancy in AI knowledge. Second, there is a difference between the building of a learning system and fully developed problem solving system. I am concentrating on **problem solving (PS)**, which has not yet gathered a massive amount of knowledge, but which can gather it. I am using the frame of PS to stress the importance of the context of learning. Third, there are approaches to the design of complex systems; I am using the software-engineering approach and the object-oriented methodology. Fourth, there is no evidence that this task is unsolvable.[6]

In the design, I primarily use the viewpoint of efficiency of learning [2]. There are many other possibilities, e.g. the viewpoint of design of problem solver, integration of cognitive functions, design of agent [15], essences of intelligence [16]. These viewpoints will manifest in a requirement analysis. This is usually the first design step of a software-engineering approach. Doing this, I have analyzed many projects, approaches and surveys, and I have extracted and classified key requirements either implemented or gained as conclusions from project experiences. I have gathered more then one hundred **requirements**. Most can be found in [17][7]. For some projects and approaches, I estimated the following fulfillment of these requirements[8]: Minsky 11%, CYC [18] 7%, PRODIGY 18%, Soar 23%, Brooks methodology [16] 33%, all these together 65%; my framework 100%. This is one reason why I consider the framework new.

I prefer the Brooks methodology, as it is the closest to my framework. I share with Brooks many principal assumptions, particularly, his emphasis on the role of environment. I do not share his criticism of traditional AI, as I believe that his objections can be solved in range of this AI. (I'll show several cases, at least).

The body of the **paper** is organized in two main sections. In the first, overall PS principles are presented, PS, its knowledge base, cognitive functions, and relations among them, i.e. the object-oriented PS model, are described. It is done partially by the graphical means of Unified Modeling Language (UML) [19]. To model in UML, I am using a tool, Rational Rose 98i [20]. In the second section, the design of PS features is partially justified. Similarities to Brooks' methodology will not be justified; only differences will be covered.

There is no performance system **implementing** the framework (yet). It is in contradiction with AI rules. How do I dare? My justification is: AI has had problems with building very general systems. Trying to overcome them, I want to follow software-engineering methodology and review the current design before continuing it.[9]

[6] Realize, please, also fifth: It cannot be described fully here. The description must be restricted according to the firm space limits.

[7] The list is steadily growing.

[8] I do not want to claim any higher precision of my estimates. They are precisely calculated from subjectively estimated coverings of particular requirements.

[9] It's true that implementation will make clear a number of constraints. However, after abstract design, some abstract, i.e. more important constraints will be revealed too.

2 FRAMEWORK

In this part, I briefly describe PS framework. It is a proposal how to implement all identified requirements. It is made on the level of basic framework ***entities***. The proposal relates basic knowledge base (KB) structures to PS cognitive functions. A more detailed description accompanied by examples can be found in [21].

2.1 Principles

"A good software product is based on a small number of powerful ideas; even if it has many specialized features; they should be explainable as consequences of these basic concepts." (Meyer [22]). Here, I am presenting concrete, basic assumptions about the PS **context**. Just, as Meyer recommends. Brooks and Jennings do it similarly. The assumptions have the form of five **principles** and their consequences. These assumptions partially cover the requirements and form a basic frame for them.

The first principle: ***Environment***[10] is a ***network*** of many heterogeneous, dynamic and even uncertain ***objects***.[11,12]

The second principle: ***PS*** is an *object* of *Environment*. *PS* has a pre-specified ***goal***. If *PS* is not in *goal state*, it has ***problem***, and that should be solved by *PS*. *PS* should control *Environment* to reach *goal*. *PS* control is accomplished by means of its ***cognitive functions*** (see Table 1). The *cognitive functions* have also their (*sub*) *goals*.

Table 1. *Cognitive function goals*

Cognitive function	***Goal***
Implementation	*Implement plan*[13] by means of *inputs/outputs* identified/applied in *Environment*
Identification	*Identify model* (and possibly *problem*) using *inputs/outputs* from/to *Environment*
Reasoning	Produce *plan* to reach *PS goal*
Learning	Create knowledge, using *history,* to aid the solution of *PS goal*
Knowledge Base	Provide means to store and access *knowledge*
Self-control	Control *PS cognitive functions* to reach these *function goals*, to help *PS* to reach its *goal*
Initialization	Initiate *KB* with built-in *knowledge*

The first two *principles* have important *consequences*: The whole object-oriented approach (e.g. [22]) can be applied to *PS* and its parts, e.g. to *KB*. *PS* can recursively control itself (self-reflexivity). This means that *PS cognitive functions*, e.g. *reasoning*, *learning* etc. can be of a recursive kind.

The third principle: ***Knowledge*** is an **approximate description** (of the behavior) of *object*; *knowledge* usage increases the average probability of success of reaching *PS goal*.

These three *principles* again have *consequences*: The more accurate *goal* of *learning* is to improve a precision of *knowledge*. *Learning* is an iterative process. It should be possible to describe all kinds of *objects* of *Environment*, as the *objects* of *Environment* are interconnected and the impossibility of describing one part may affect the impossibility of describing the whole *Environment*. Thus

[10] Take words in italics as names of framework *entities*.

[11] This is very similar to [23].

[12] Concepts like *object*, *network*, heterogeneous, dynamic, uncertain etc. are primitive concepts of the framework. I am not going to define them here; see e.g. [22], [21], [8].

[13] *plan*, *model*, and *history* are specified later.

knowledge should be **general** one, e.g., able to describe deterministic or stochastic, static or dynamic, simple or structured *objects*. *Learned object*, about which *PS* has already *knowledge*, can be used as *tool* for *PS*, e.g., *tool* for the next *learning* (see later). *Knowledge* can play the role of both *knowledge* and data. *Knowledge* has a role of data, when it is *object* for recursive *PS*. *PS* can have *meta-knowledge*. *Knowledge* and *PS* are more **homogenous**; i.e. *PS* does not need different kinds of *knowledge* for (external) *Environment* and for itself, e.g., for ordinary *PS objects* and *plans*.

The fourth principle: *Environment* is relatively very stable. During an interval of *PS* work, very little part of *Environment* is changing.

The fifth principle: *PS* (*learning*) should cope with its complexity.

2.2 *Knowledge base*

PS is a knowledge-based system. Let us start with the description of its *KB*, i.e. with the description of model of *KB entities* and semantics of these *entities*. These *entities* are e.g. *classes*, *relations* between *classes*, *class instances*, *notes*, and *class diagrams*. The UML *class diagrams*, which I am using in the paper, describe relationships among *classes*. The semantics of these *entities* are relationships between these *entities* and *objects* of *Environment*. The model of *KB* (Figure 1[14]) consists of *Description* and other related *entities*. *Description* is *class*. ***description***[15] is an *instantiation* of *Description*. *description* is a core of *KB*. Each *description* (indirectly) describes either *class* of *objects* (i.e. characteristics of a group of *objects*) or a partially instantiated *class* of *objects*, or a specific *object* (i.e. fully instantiated singleton *class* of *objects*). There can be a lattice of *instantiated description*s. The *instantiation* can range from not *instantiated* to fully *instantiated*. The *instantiation* is ISA (is-a kind of) *relation*. "Indirectly" means that the characteristics of *objects* can be derived via *description relations*. *Description* is *meta-class* of *class* of *objects*, i.e. *class* of *descriptions*.

By characteristics of *objects*, I mean their relationships, interfaces, and functionality. *description* can be related to other *descriptions* via various kinds of ***relations***, e.g. *instantiation*, *simplification*, *aggregation*, *association*, *equivalence*, *anchor*, and *is-goal-of relations*. *relation* describes relationship between *descriptions*.

description can have *interface*(s) with *+input*, *+output*, and *+state role*s. ***interface*** is an *instantiation* of *Interface*. *interface* provides for a description of *input*, *output*, and *state* of *object*. It enables that *interface* part of *object* characteristics is partially independent of *description*. For example, a particular *interface* can be *input* for one *description* and *output* for another one. The sharing of *interface* enables to describe a connection of two or more *objects*. Like *description*, *interface* can be *instantiated*. If

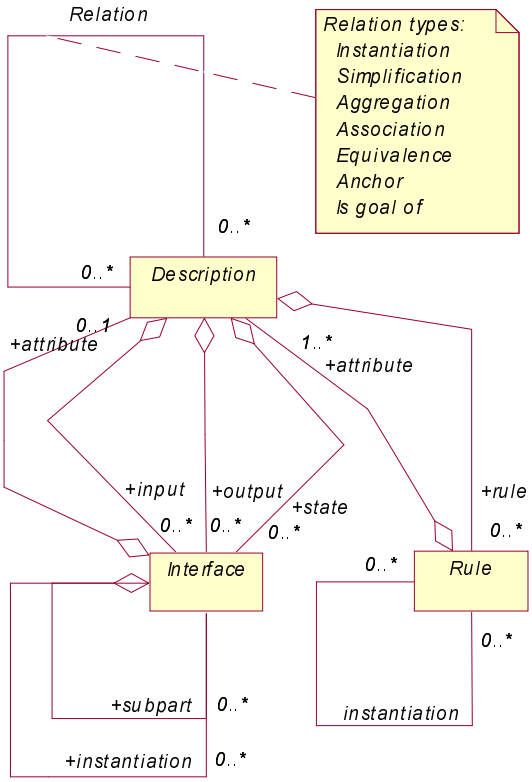

Figure 1. Model of knowledge base

there is a corresponding more general *description* (with *interface*), *interface* can be inherited. *interface* can be structured. It is accomplished by the possibility for *interface* to play the *+subpart role* for another *interface*. Each *interface* has *attribute*(s). ***attribute*** describes *input*, *output*, or *state* component of *object*. *attribute* is *description*, which plays a *role* of *+attribute* for *interface*. If there is *description* with its *attribute* and its *instantiated description* with its fully *instantiated attribute*, then this *instantiated attribute* can be a ***value*** of the original *attribute*. If there is *description* with its not fully *instantiated attribute*, then this *attribute* is a ***variable*** of *description*.

description can have also *rules*. ***rule*** is an *instantiation* of *Rule*. *rule* describes the functionality of corresponding *objects*, i.e. describes either the *state* transition or *output* production. *rule* can be generalized, i.e. related to another more general *rule* by *instantiation relation*. *rule* can be inherited if there is a corresponding more general *description* (with *rule*). *attribute* can be a part of *rule*. It is accomplished by *Description* being part of *Rule* and playing *+attribute role* in this association. I am not specifying any details of *rule* implementation. This is postponed to the later design stages. Now, you can envisage any implementation, like classical if-then rule, fuzzy rule, rule with associated probability, neural network.

Knowledge consists of *description(s)*.

2.3 *Problem Solver*

PS (Figure 2) is connected with *Environment* via its *input* and *output* (*I/O*). *I/O* are *instantiated interface*s of *PS* and *Environment*, their *attributes* have *value*s. *input values* are produced by *Environment* and a part of it is produced as *modeled inputs* (see

---

[14] The short introduction into UML: A named rectangle means *class*, e.g. *Description.* A (named) line connecting two *class*es means an association *relation* between these *classes,* e.g. *instantiation.* It can have an arrow, denoting a client – server *relation.* A (named) arrow with a diamond head means that *class* is a part of another *class*, e.g. *Interface* is a part of *Description*. Association *relation* can have roles, e.g. *+input*. *Relation* can have multiplicity, e.g. "0..*".

[15] Notice that I distinguish e.g. description in its common sense and *description* or *Description* as framework *entities*. It's important.

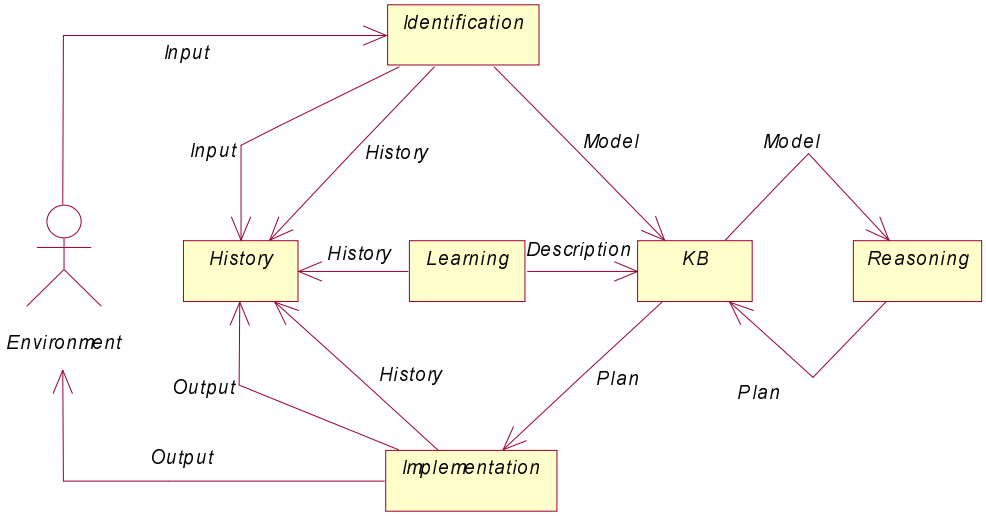

Figure 1. Problem solver

later) by ***identification*** *function*. *PS* perceives *input*. *output values* are produced by ***implementation*** *function* to influence *Environment*. On the lowest level, *output* is a specification of a program (or motor or actuator), that *implements* action. On higher levels, *output* is a specification of an action of some external *object*. Preceding *values* of *I/O* (behavior of *PS* and *Environment*) are stored as ***history***. *history* is a basis for *identification*, *implementation*, and *learning functions*. *implementation* is also based on *plan* (including *plan state*).

Later, it will be seen that *implementation* produces, in some sense, an *intelligent* part of *PS output*. ***plan*** is *description*. ***reasoning*** *function* produces it. It represents a possible future behavior of *PS*. *reasoning* is based on *model*. *reasoning* tries to find or construct in *KB plan* that would transmit *Environment* from current *problem* (*state*) to *goal* (*state*). It can create *sub-problems*. ***model*** is *instantiated description* of *Environment*. *I/O* are (also) *interfaces* of *model*. *identification* tries to find *model* of *Environment*, i.e. it tries to find or construct *description* and it instantiates this *description* in such a way that it corresponds to *history*. Later, it will be seen that *identification* produces, in some sense, an *intelligent* part of *PS input*. As a part of *model*, *problem* can be identified, too. ***goal*** is a specification of set of *Environment states* that should be maintained or reached. It is described as *description*. ***learning*** produces, i.e. creates or modifies, *descriptions*. *learning* is using *history* as a training set. When *learning plan*, it uses an identified *value* of *problem* for evaluation of this training set.

PS and its *cognitive functions* are *instantiations* of *Description*. All *cognitive functions* use *KB* and are controlled by *self-control*. ***knowledge base*** *function* saves or retrieves *descriptions*. ***self-control*** *function* is to be implemented as recursive *PS*. It works like *PS*. Its environment is *Environment* together with the internals of *PS*. *I/O's* of *self-control* are *I/O's* of *PS* and *PS cognitive functions*' *inputs*, *outputs* and *states*, e.g. allocated resources, successes and resource consumptions. The level of recursion is restricted by the extent of *KB* on the deepest level of recursion, e.g. by necessity to initialize the work of *self-control* on the next recursion level. If this *KB* extent is low, then the behavior of corresponding *cognitive function* is simple and it is not efficient to use recursive control, i.e. to initialize new recursive level. *problem* of *self-control* is an unsuccessful work of some *cognitive function*. The success can be either immediate or long-term. For example, it can be immediately revealed that *learning* cannot create *description* from a given training set. Or, it can take some time to recognize that some *description* is wrong. ***initialization*** *function* initiates *KB* with built-in *knowledge*. It initializes connection between *KB* and *PS I/O attributes* and, in a similar way, it initializes connection between *KB* and *self-control I/O attributes*. It initializes the work of *self-control*, too.

plan, *model*, and *goal* are *instantiations* of *Plan*, *Model*, and *Goal*, which are *sub-classes* of *Description*, *history* is a kind of *rule*, and *History* is *sub-class* of *Rule*.

2.4 Shifted Border

In a simple *Environment*, to *identify* (the only one and simple) *object* and its *state* need not be a problem. To *learn description* of such *object, plan* a solution in this *Environment* described as such *object, implement plan* need not be a problem. *self-control* would not be necessary and *KB* would be trivial. I call the corresponding *cognitive functions* simple *cognitive functions*. To implement them the existing cognitive functions or ontologies from AI can be used [2], [3]. However, the real *Environment* is not simple, usually.

To implement (full) *cognitive functions* it is necessary to extend the simple *cognitive functions* in such a way to cover the work with object-oriented *Environment*, and if necessary, to cover the work with both deterministic and stochastic *objects*, with both static and dynamic *objects* etc. Some aspects of object-oriented nature of *Environment* are solved by existing techniques, e.g. *instantiation*, *simplification*, and *anchor relations* in *learning*. To solve other aspects, e.g. *association relations*, the concept of *tool* is used: *PS* uses a known *object* as ***tool*** that helps *PS* to perceive or influence *Environment*. In the range of this concept, *PS* may consider *tool* to be its part. In this way, *PS* virtually shifts its *border* with *Environment*. As it is an important aspect of the whole framework, I describe it a bit more (see Figure 3).

Let us consider *PS*, its *Environment*, some *tool*, as an *object* of *Environment*, and *rest of Environment*. Let *PS* have *inputs* to perceive *Environment*; the part of them connected with *tool* is x. Let *PS* have *outputs* to influence *Environment*; the part of them connected with *tool* is y. x and y together form the part of the real *border* between *PS* and *Environment*, i.e. *border* between *PS* and *tool*. x' and y' form the part of *border* between *tool* and *rest of Environment*. x' and y' are accessible to *PS* during *learning* but not fully accessible to *PS* later. Let us assume that *PS* can *learn* characteristics of *tool* in such a way that it can later 1) *identify tool*, 2) *model* (specify) the sequence of *values* of x' and y', if it knows the sequence of *values* of x and y. x' and y' are called *modeled inputs* and *outputs*. The equality of *modeled inputs* and *outputs*, both in *Environment* and in *PS*, can be interpreted as a shift of

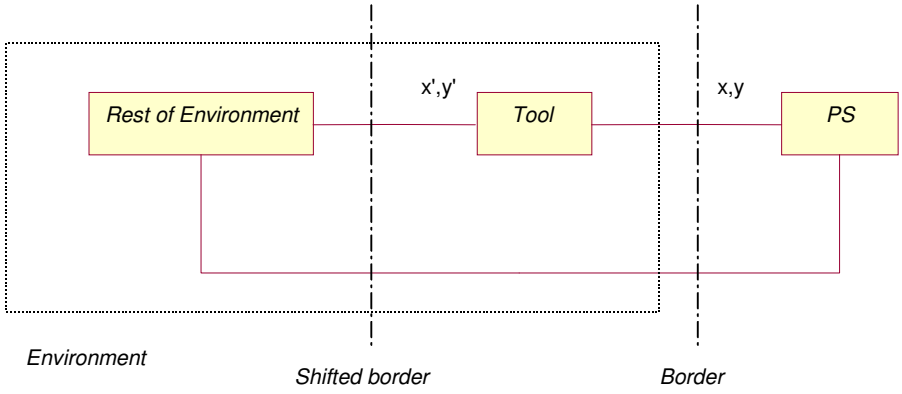

Figure 2. *Shifted border*

border between *Environment* and *PS*. The ***shifted border*** is partially composed from x' and y', instead of x and y. Thus, *PS* can control *Environment* in such a way as if it has a *shifted border* with *Environment*. From the assumption that *PS* can *learn* and use tools directly connected to itself follows, that it can *learn* tools indirectly connected also. In addition, it means that *input* information for *learning*, i.e. *outputs* and *inputs*, can be iteratively extended by information about *modeled inputs* and *outputs*.

How are *PS cognitive functions* implemented in a *shifted border* context? To ***identify*** *object* from the *rest of Environment* it is necessary to *identify tool*, T, and then to use T to get x'. It means to use *implementation* to apply y to T, and *simulation* to derive x' from T and y', which was applied by T. ***simulation*** can be implemented as *self-control implementation*; its *meta-output* is not applied in the external *Environment*, but only internally, in part of *PS model*. To ***learn*** *object* from *rest of Environment*, it is necessary to *identify tool*, T, to use T as a *tool* to get x' and y', i.e. use *simulation* to derive x', *implementation* to apply y, and indirectly via *tool* y', and *learn description* on the level of x' and y'. In accordance with the *tool* concept, newly created *description* extends the set of *I/O attributes*. This, together with the use of *KB* and recursive control of *identification* and *implementation*, forms the basis of ***intelligent I/O***. To ***implement*** *object-plan* from *rest of Environment* it is necessary to *identify tool*, T, use T to apply y', i.e. use *identification* to *identify* x, *simulation* to derive x', *simulation* to derive y from wanted y', and *implementation* to apply y and indirectly via *tool* y'. To ***reason***, i.e. to produce *plan* for *rest of Environment* it is necessary to *identify tool*, T, produce *plan* on level of x' and y'; there can be some constraints implied by T; these can be solved either by theirs *learning* or by using the *association relation* of *descriptions*.

One of tools can be ***teacher***. One of *teachers* can be a ***society***. *PS* can use *knowledge* created by *society* of similar *PS*'s; i.e. *learning* can be of a distributed kind. Some of *tools* can be *tools* for ***communication*** with similar *PS*'s. One of *tools* can be an approach to creation or *learning*. That *tool* can be used to design *learning* itself.

3 JUSTIFICATION

In this section, I justify several *PS* design decisions. In this way, I also illustrate the framework. More justification and new interpretation of old concepts can be found in [21]. Here, justifications primarily cover differences between my approach and that of Brooks [16].

3.1 Principles

The *super-principle* is justified by justification of our *principles*. Our *principles* are justified both by the corresponding analogous and more specific AI assumptions. The AI assumptions are considered justified (by AI). Thus, the justification consists from showing the correspondences.

Brooks has four basis assumptions: "developmental organization, social interaction, embodiment and physical coupling, and multi-modal interaction"[16]. My approach is more general, i.e. as general as possible. I follow an old AI heuristic that the solution of complex problems should be tried on abstract level first.[17] The Brooks assumptions are covered by my more general *principles*: "**Development** forms the framework by which humans successfully acquire increasingly more complex skills and competencies." It is covered by my assumption about approximate *knowledge* and corresponding *consequences*. I do not stress "increasingly more complex skills", I stress *learning* with all above described features. "**Social interaction** allows humans to exploit other humans for assistance, teaching, and knowledge." Brooks concentrates on communication. I prefer the more general concept of *tool*, subsuming concepts of *teacher*, *society*, distributed intelligence, *communication*. *Communication* is also the part of our assumption about *learning* dealing with complexity. "**Embodiment** and physical coupling allow humans to use the world itself as a tool for organizing and manipulating knowledge." The assumption of embodiment is covered by our assumptions about *Environment* and its *relation* to *PS*. In addition to "embodiment", I have an important assumption that *PS* is a part of *Environment* and that *Environment* has an "object-oriented" structure. Jennings assumes object-oriented environment too. "**Integration** allows humans to maximize the efficacy and accuracy of complementary sensory and motor systems." Integration (of sensors and motors) is again covered by my assumption about *Environment* and its *relation* to *PS*. Brooks considers stability of environment and efficiency of learning too, but not on the level of basic assumptions. I think that it is mainly this difference in emphasis, which causes my framework to be, on a conceptual level, a more elaborated one.

3.2 Homogeneity

There are approaches using e.g. genetic algorithms for self-control [24], [25]. They offer evidence of improved efficiency. They use simple learning, in my terminology. Why not generalize this and use complex learning?[12] It should be even more efficient. This advantage is enabled by my *principle*: *PS* is part of *Environment*. Consequently, *PS* should control itself. It can do it with all its power. For ***self-control* to be practicable**, *PS* must be homogeneous. Therefore, it must not be necessary to design and use special control mechanism for each *PS* part. *KB* should be homogeneous; this is most important. However, some homogeneity is beneficial and possible for *cognitive functions* also. On the

[16] In the paper, in quotes, I am using his terminology and quotations.
[17] This is an application of the *super-principle*.

abstract level of design, homogeneity is directly supported by object-oriented methodology.

Let us consider the possibility to implement, in the framework, two types of *descriptions*, i.e. ***description*** of *object* of *Environment* and *description* of *PS* ***plan***. These two *descriptions* should have had many similar properties, e.g. they should have been general, dynamic, complex, hierarchically organized, object-oriented, might have been approximate, uncertain, can utilize *tools* etc. How might they have differed? Only one difference is that (*sub-*)*plan* should be directly related to some (*sub-*)*goal*. This *relation* can be permanent or temporary, depending on the kind of *goal*. There is also another viewpoint: *PS* and its parts have their behaviors. They are described - generated by *plans*. *PS* and its parts are parts of *Environment*. For both, it is suitable to have one common *description*. As *description* of external *object* and *plan* share most features, it is suitable to describe them in some common way.

In the first approximation, it was appropriate to understand *learning* as creation of *knowledge* (Table 1). After making the concept of *knowledge* more precise and after projecting the requirements into the *PS* design the need for discrimination between general and specific *descriptions*, between *variable* and its *value*, between *description* and *plan* nearly disappears. Then it is suitable to characterize some ***cognitive functions*** as in the following table (Table 2). In all cases of listed *cognitive functions*, *function* is specified as the creation of *knowledge*:

Table 2. *Cognitive function* characteristics

Cognitive function	Characteristics: it is the creation of *knowledge* related ...
Identification	... directly to the *state* of *Environment*
Implementation	... to the direct action in *Environment*
Deduction	... to the mid-range influence of *Environment*
Learning	... to the long-range influence of *Environment*

It is not good to have information about *description* being a part of that *description*. For example, to have *attributes* describing the precision or confidence of some *rule* being the parts of this *rule*. It is not a very general approach; it is in contradiction with the requirement of generality. Why use just some (pre-)specified *attributes* and not others? Is it flexible? What if, after some time, it shows that some other attribute will be more suitable? Could not *PS* create this information itself? The answer to these questions is the following solution: **Characteristics of *description*** should be represented with another (*meta-*) *description*.

Notice, that there is only one **common concept**, *description*, for description of various *PS entities*, i.e. *classes* of *Environment objects*, specific *objects*, *model*, *plan*, *value*, *variable*, *state*, *goal*, *meta-model*, *meta-plan*, etc.

3.3 Relation to Classical AI

Brooks considers representations of classical AI to be monolithic, explicit, internal, and uniform. He believes these features do not correspond to human features: "Humans have no full monolithic internal models. ... Humans have no monolithic control. ... Humans are not general purpose." Of course, it depends on the exact meanings of the concepts used. However, structured representations can also be found in classical AI. On ECML 97 [25], I counted 16% of (learning) papers dealing with the object-orientation issue. Implicit representation can be found also, e.g., when some knowledge can be deduced. In my framework, "external representation" can be used in the situation when the *state* of *Environment* can be directly identified from its *outputs* (and it is not necessary to remember it).

The framework's *knowledge* representation is structured, it is both explicit and implicit, both internal and external, and in some sense uniform. *PS* works with *model*, which is only a part of *KB*. It is not *model* of all details of *Environment*, and in that, it corresponds to humans. Because the external *object* and *plan* are described similarly, a similar situation holds for the control of *PS* too.

"Humans are not general purpose" in the sense that they are usually not top mathematicians, top soccer players, or popular novelists, at the same time. Statistics works. However, they can be good mathematicians, good soccer players and not-so-bad singers, at the same time. In this sense, they are general purpose. They can *learn* many different domains. In the same sense, *PS* is designed to be also **general purpose**.

Brooks [27] underscores the requirement of **reactivity**: "The key idea from intelligence is: Intelligence is determined by the dynamics of interaction with the world." Brooks uses highly independent, finite state machines. Reactivity can be reached in my framework by similar means, but without the need to "radically change the model of though". In my framework the reactivity is reached by using structured *plan* (*description*) consisting from *sub-plans*, and by the possibility to *implement* these *sub-plans* in parallel, to *implement* particular *sub-plan* when its *inputs* match *PS inputs*. "Direct physical coupling without intermediary representation" corresponds to *implemented sub-plan*. When there is pre-prepared *plan* (or *sub-plan*), then it is not necessary to *plan* (or *plan* so much).

3.4 Ontologies

I have two brief remarks concerning ontologies: The framework is not specialized on ontologies, as it is, on its level of detail, directly applicable in this area.

In Introduction, I open issues that can be made more clear now: In the ontology area, the reuse of knowledge is stressed. This leads to “principles” like clarity, completeness, and coherence. They are covered by my requirements. However, in the context of the other requirements, especially of my third *principle* about approximation, they should not be taken absolutely, especially if we understand learning as a process, during which *description* does develop. The approximation can also explain the problem – paradox with “the lesser utility of abstract concepts”. They seem less usable as they are less accurate. This is, in turn, caused by generalization; it is known that it is a source of (knowledge, as well as) errors. As soon as you manage to have accurate abstract concepts, they should become also usable.

4 CONCLUSION

This paper proposes a framework to improve the current level of efficiency of learning. The approach shows feasibility in a sense that, on the conceptual level of design, the *PS* design could be done in such a way, that it satisfies all challenging requirements. No significant obstacles impeding the subsequent more detailed design of the framework have been found. This is the evidence for my claim that learning that is more efficient can be designed.

The idea to integrate all AI concepts into one system is not new. However, integration is often preached, but seldom practiced. My

contribution is in doing the "first (small) step" toward such integration. The framework uses many concepts, but not many new ones. The exceptions are shifted border, that learning can create all parts of KB (knowledge base), and knowledge homogenization. The concepts of self-reflection and object-orientation play also very important roles in the proposed integration.

In higher cognitive processes, self-reflection plays an important role. AI is a cognitive process of incremental understanding the phenomenon of intelligence. What is the state of self-reflection in AI? According to my assessment, the field of AI is just at the beginning of its conscious self-reflection. What do I mean by this? I mean that parts of AI community - yet still small - are just starting to use specific benefits of AI work, i.e. they are starting to interpret the results of the AI process - the knowledge produced by AI - to control the AI process itself. I used this knowledge in this paper as a unifying super-principle.

The efficiency of learning has been analyzed and designed on a conceptual level. This is the only possible (and necessary) beginning of a long-term project. My intention is to continue with a detailed analysis and design of cognitive functions and KB, and their implementation using a contemporary software-engineering approach, i.e. utilizing object-oriented, computer-aided software-engineering approaches and tools and using contemporary knowledge about cognitive functions.

ACKNOWLEDGEMENTS

This work has been partially supported by the grant GA ČR No. 102/99/1564. I also acknowledge the help of the reviewers, my colleagues Jiří Grim and Alena Halousková, my daughters, and niece, who helped to improve earlier versions of the paper.

REFERENCES

[1] RoboCup-99 Workshop, http://www.ida.liu.se/ext/RoboCup-99/, 1999.

[2] S. Thrun, C. Faloutsos, T. Mitchell, L. Wasserman, *Automated Learning and Discovery: State-Of-The-Art and Research Topics in a Rapidly Growing Field*, CMU, Pittsburgh, 1998.

[3] A. G. Perez, V. R. Benjamins, Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In [28], 1-1 – 1-15, 1999.

[4] L. Console, and O. Dressler, Model-based Diagnosis in the Real World: Lessons Learned and Challenges Remaining. In [29] 1393-1400, 1999.

[5] T. R. Gruber, Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, **43**, 907–928, (1995).

[6] A. Barnaras, I. Laresgoiti and J. Corera, Building and Reusing Ontologies for Electrical Network Applications, In [30], 288-302, 1996.

[7] H.S. Pinto, A. G. Pérez and J.P. Martins, Some Issues on Ontology Integration. In [28], 7-1 - 7-12, (1999).

[8] J. Bůcha, *Problem Solving with Knowledge Represented by Probabilistic Automata Network*, Ph.D. Thesis, Czechoslovakian Academy of Science, Prague, p. 110, 1978 (in Czech).

[9] J. Bůcha, Incremental Learning, *Computers and Artificial Intelligence*, **4**, 325-334, (1985).

[10] Special Issue on Integrated Cognitive Architectures, *SIGART Bulletin*, **2**, no. 4, (1991).

[11] Special Issue on Knowledge Representation and Reasoning Systems, *SIGART Bulletin*, **2**, no. 3, (1991).

[12] M. Veloso et al., Integrating Planning and Learning: The PRODIGY Architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, **7**, no. 1, (1995).

[13] The Soar Project at Carnegie Mellon University, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/soar/public/www/home-page.html, 1998.

[14] M. Minsky, Steps toward Artificial Intelligence, *Proc. of IRE*, **49**, 8-30 (1961).

[15] N. R. Jennings, Agent-Based Computing: Promise and Perils. In [29], 1429-1436, 1999.

[16] R. A. Brooks, C. Breazeal (Ferrell), R. Irie, C. Kemp, M. Marjanovic, B. Scassellat and M. Williamson, Alternate Essences of Intelligence, http://www.ai.mit.edu/people/brooks/papers/group-AAAI-98.pdf. Also in AAAI-98, 1998.

[17] J. Bůcha, Integration of all Cognitive Functions: Requirements, http://www.utia.cas.cz/AS_dept/bucha/ recent.html, 1999.

[18] D. B. Lenat, The Dimensions of Context-Space, CYCORP, Austin, http://www.cyc.com/, 1998.

[19] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Reading: Addison-Wesley, 1999.

[20] *Rational Rose 98i, Using Rose*, Rational Software Corporation, Cupertino, CA, 1998.

[21] J. Bůcha, On Efficiency of Learning, A Framework, Research Report No. 1969, UTIA, Prague, http://www.utia.cas.cz/AS_dept/bucha/recent.html, 1999.

[22] B. Meyer, *Object-Oriented Software Construction*, Upper Saddle River, Prentice Hall, 1997.

[23] J. Pearl, Reasoning with Cause and Effect. In [29], 1437-1449, 1999.

[24] F. Lemke, Knowledge Extraction from Data Using Self-Organizing Modeling Technologies, http://www.Knowledgeminer.net/index3.htm, 1997.

[25] P. Collard and A. Gaspard, "Royal-Road" Landscapes for a Dual Genetic Algorithm. In [30], 1996.

[26] M. van Someren and G. Widmer (Eds.), *Machine Learning-97, Proceedings of 9th European Conference on Machine Learning*, Prague, Czech Republic, 1997.

[27] R. A. Brooks, Intelligence without Reason, MIT, A.I. Memo No. 1293, 1991.

[28] V. R. Benjamins (Ed.), *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends*, Stockholm, 1999.

[29] T. Dean (Ed.), *IJCAI-99, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Vol. 1 a 2, Stockholm, Sweden. San Francisco, Morgan Kaufmann Publishers, 1999.

[30] W. Wahlster (Ed.), *ECAI 96, 12th European Conference on Artificial Intelligence*, J. Wiley & Sons, Ltd, 1996.