The Autotelic Principle

Luc Steels
VUB AI laboratory - Brussels
Sony Computer Science Laboratory - Paris
steels@arti.vub.ac.be

July 5, 2004

Abstract

The dominant motivational paradigm in embodied AI so far is based on the classical behaviorist approach of reward and punishment. The paper introduces a new principle based on ’flow theory’. This new, ‘autotelic’, principle proposes that agents can become self-motivated if their target is to balance challenges and skills. The paper presents an operational version of this principle and argues that it enables a developing robot to self-regulate his development.

1 Introduction

The design and implementation of self-developing robots has become a focal point of recent efforts in robotics and AI research [20]. It builds further on the work of developmental psychologists, who have a long history of studying 'epigenetic' or 'ontogenetic' development [7], [9]. A lot of research in developmental robotics focuses on finding powerful learning mechanisms that can run continuously in open-ended environments [**?**]. This paper turns to a more global issue: How can the developmental process as a whole be orchestrated.

The problem of regulating development is very challenging for three reasons. (1) Certain things often cannot be learned before other things are mastered, so the developmental process must be scaffolded somehow, to enable bootstrapping from simple to complex. Thus, it is not possible to learn fine-grained control of grasping if there is no ability to identify and track the objects that need to be grasped. (2) In a complex agent, each component depends on others, either to provide input or to produce appropriate feedback. But if there are many subcomponents, each developing at their own pace, regulating global development becomes a non-trivial issue. (3) An agent may reach a level of performance which is adequate with respect to a given environment but which is nevertheless a local maximum in the sense that a richer interaction can be achieved by further exploration and development. So a big challenge is to avoid that the agent gets stuck in development, even if this means a decreased performance in the short run.

Some researchers have proposed that nothing special needs to be done to orchestrate the developmental process, because the development of one skill naturally creates new opportunities for the development of other skills in a changing ontogenetic landscape [18]. For example, once the arm can be controlled, it is possible to start exploring the uses of the hand. Although it is obviously the case that one opportunity may lead to the next, it is now generally recognised that more needs to be done, particularly to avoid that the agent remains in local maxima which do not exploit the full capacity of what is possible. Three approaches have already been discussed in the literature.

- Scaling of input complexity

  A first group of researchers has proposed that development can be organised by regulating the complexity of the external environment. This way the agent can build up capacity in a simple environment before tackling additional challenges. Usually a small subtask is isolated and the agent is trained for that specific subtask with prepared scaffolded data [2], [6]. In more sophisticated applications, several stages and subcompetences are identified and input data is carefully prepared to pull the agent through each stage. [19].

- Scaffolding of reward function

  Other researchers have proposed to scaffold the reward function, i.e. to give external feedback to the agent which makes sure that simpler and foundational skills are learned before more complex skills are tackled and that the stakes are increased as soon as steady performance has been reached [20]. In the case of language development for example, we could envision first a high reward for producing single word sentences, then a higher reward for multiple-word sentences, then a higher reward for constructing grammatical phrases with increased complexity.

- Staging of resources

  Yet another approach is to stage the resources available to the agent in a kind of 'maturational schedule'. For example, Elman [6] has shown that a recurrent neural network can be trained first with a small 'look back' window, then this window is progressively increased to take more of past input into account. Such an approach has been shown to give better performance compared to one where the full complexity of internal resources is available from the beginning.

All these approaches are valuable and have shown to yield interesting results. Moreover they are not completely devoid of naturalness because in the case of infants, caregivers often scaffold the environment or 'up the ante' to push the infant to higher competence. However these approaches assume a very strong intervention by 'trainers' and/or a careful a priori design of developmental scenarios. The real world always presents itself with the same complexity to the learner and it is therefore artificial to constrain it. It would be much more desirable if the agent could develop independently and autonomously in an open-ended environment by actively self-regulating his own development.

This is precisely the goal of the research reported in this paper: a general principle is proposed by which a complex agent could self-regulate its-build up of skills and knowledge without the need for the intervention of a designer to scaffold the environment, stage the reward functions, or bring resources progressively online in a maturational schedule. The main idea is to introduce a new motivational principle gleaned from recent work in humanistic psychology. This principle is introduced in the next section of the paper. Further sections present an operationalisation of this principle. We have already conducted various experiments to exercise the principle in the context of grounded language development, [**?**]. The results are encouraging and will be reported in more detail in forthcoming papers.

2 Motivation and Flow

Reinforcement Learning

Most models in psychology and neuroscience are still rooted in the behaviorist framework of reward and punishment, originallly coming from the work of Skinner and his associates [16]. Also a lot of autonomous robotics work, particularly under the banner of reinforcement learning, is implicitly based on the same approach. This theory makes four major assumptions.

First, it assumes that the overall goal of the organism is to keep its critical parameters for survival within viable bounds [**?**]. The challenge of a developing organism is to acquire the necessary behaviors so that such a viable state is maintained, or to adapt the behaviors if the environment changes.

Second, it argues that certain behaviors get rewarded, for example with food or other means that give direct pleasure, and others are punished, for example through the inducement of corporal pain. Rewards reinforce specific behaviors because they inform the organism that they are beneficial, in other words that a viable state can be reached and maintained. Punishment signals that the behaviors that were enacted need to be abandoned or new knowledge and skills need to be acquired. In natural circumstances, reward and punishment is generated by the environment.

Third, it proposes that organisms start with a repertoire of reflex behaviors and an innate value system. New behaviors are shaped by reward and punishment. When a trainer or educator hands out the reward or punishment, she can push development in specific directions and the trainer's value system may become progressively internalised by the trainee.

Fourth, classical behaviorism proposes that this reinforcement framework is an adequate theory of motivation, in the sense that the main purpose of the organism is to seek reward and avoid punishment, and so all the rest (acquisition of new behaviors and internalisation of a value system) follows.

Flow theory

More recently, a complementary motivational theory has been proposed in psychology, which points to a richer notion of motivation. This theory was originally developed by the humanistic psychologist Csikszenmihalyi, based on studying the activities of painters, rock climbers, surgeons, and other people who showed to be deeply involved in some very complex activity, often for the sake of doing it, i.e. without direct reward in the form of financial or status compensation [3]. He called these activities autotelic. ”Autotelic” signifies that the motivational driving force (”telos”) comes from the individual herself (”auto”) instead of from an external source, administered by rewards and punishments.

Autotelic activities induce a strong form of enjoyment which has been characterised as ”flow”. The word ”flow” is a common sense word and so there is a risk to interpret it too broadly. Csikszenmihalyi intends a restricted usage, being a state which often occurs as a side effect of autotelic activities:

> People concentrate their attention on a limited stimulus field, forget personal problems, lose their sense of time and of themselves, feel competent and in control, and have a sense of harmony and union with their surroundings. (...) a person enjoys what he or she is doing and ceases to worry about whether the activity will be productive and whether it will be rewarded. o.c. p. 182.

Because the activity is enjoyable, the person who experiences this enjoyment seeks it again, and therefore it becomes self-motivated. Moreover due to the high concentration and the strong self-motivation, learning takes place very fast. The learner is eager to find the necessary sources and tools herself and spends time on the acquisition of skills, even if they are not exciting in themselves, as long as they contribute to the autotelic activity.

Given this description, it is quite obvious that many people will have experienced some form of flow in their life, and that children in particular enter into flow experiences quite often, particularly during play. Flow is sometimes associated with the ultimate high experience of the rock climber that has finally managed to climb Mount Everest, but that is an exceptional situation. Flow - as defined here

- is much more common and can just as well happen in every-day experiences like playing with children or engaging in a long term love relationship.

It is also important to distinguish flow from directly pleasurable activities like going down a roller coaster. A key difference is that the activity must in itself be challenging - otherwise there is no feeling of satisfaction after difficulties have been surmounted. Moreover there must be a steady progression in the nature and particularly the level of the challenge. This is the reason why child rearing can be so enjoyable and fascinating. A child keeps developing all the time - which is what makes the interaction fun - and that creates continuously new challenges for the parent to figure out what she is thinking, what she might want to do or not do, and so on. The rock climber can also scale up the level of difficulty with which rocks are being climbed or the kinds of rocks that are tackled. Similarly, the musician can first play easy pieces and then steadily move up. she can first play with other amateur musicians and then play with better and better musicians. The performance can be first for a few friends, but then for a larger and larger unknown audience.

An obvious key question is: What makes activities autotelic? Here comes Csikszenmihalyi's most important contribution, I believe. He argues that it lies in a balance between high challenge, generated through the activity and perceived as meaningful to the individual, and the skill required to cope with this challenge:

> Common to all these forms of autotelic involvement is a matching of personal skills against a range of physical or symbolic opportunities for action that represent meaningful challenges to the individual. o.c. p. 181

When the challenge is too high for the available skill, in other words the opportunity for action is so bewildering that no clear course can be seen, and when there is at the same time no hope to develop appropriate skills by learning, anxiety sets in and the person gets paralysed and eventually may develop symptoms of withdrawal and depression. When the challenge is too low for the available skill, boredom sets in and the long term reaction may be equally negative. The optimal regime is somewhere between the two, when there is a match of challenge and skill. It follows that it is important for the individual to be able to decrease challenge when it is too high so as to get an opportunity to increase skills, but it is equally important that the individual can increase challenge when the skill has become higher than required to cope with the challenge, or that the environment generates new opportunities for the individual to grow.

Let us now see how these intuitive ideas can be operationalised into a design principle that can be implemented on physical self-developing robots.

3 Operationalising the autotelic principle

A cognitive agent is a physically embodied organism embedded in an environment in which there is a steady stream of sensori-motor inputs and a steady stream of decisions for action which translate into motor commands or internal state changes, such as switch goals or move to another location in the world. The key challenge for the agent is to survive in this environment and hence choose the right action based on an interpretation of the current situation.

We assume that the agent is organised in terms of a number of sub-agencies further called components. Each component establishes an input-output mapping based on knowledge and/or skill. For example, a segmentation component takes a camera bitmap and produces a list of segments using some segmentation algorithm. Each component requires a set of resources (memory, computer time) and makes use of knowledge or skill that is typically adapted or learned. For example, the segmentation algorithm may progressively build up a database of the shapes or movement trajectories of the objects in the environment so that segmentation can be done more quickly or more reliably.

A realistic system needs of course many components. For example, in the case of an embodied agent interacting through language with another agent [17], we need components for grounding world models through vision, speech and gesture recognition, speech and gesture production, selection of a topic, conceptualisation of what to say, lexicon lookup, grammatical parsing and production, interpretation of semantic structures, dialog management, etc.

The autotelic principle suggests that the balancing of skill and challenge should be the fundamental motivational driving force of the agent. This implies (1) that each component must be parameterised so that challenge levels can be self-adjusted based on self-monitoring of performance, (2) that each component must have the ability to increase skill to cope with new challenge, and (3) that there is a global dynamics regulating the adjustement (both increase and decrease) of challenge levels. The reward function of the total agent is the degree of balance between challenge and skill for each of its components. The increase in complexity of the agent's behavior (and hence the kinds of tasks and environmental complexity it can handle) will be an emergent side effect of the system's effort to keep this balance between challenge and skill.

The following subsections provide more detail on each of these aspects.

3.1 Parameterisation of components

Each component of the system must be parameterised to reflect different challenge levels. The nature of the parameterisation obviously depends on the task that the component must achieve and on the nature of the algorithms that are used. For example, suppose that a robot has a subsystem that has the task of moving an object using vision and hand/arm motor control. One parameterisation of such a component could concern the precision with which the object is to be moved: Is pushing it aside in a broad gesture enough, or should the object be picked up and put down carefully in a precise location. Another parameter is the nature of the object: Is it of a simple uniform shape or does it contain handles or other structures that need to be recognised and used to manipulate the object. Another parameter concerns the weight of the object. A heavy weight might require the agent to adopt a specific posture so as not to get out of balance.

Formally, we associate with each component c_i a parameter vector $< p_{i,1}, ...p_{i,n} >$. The set of all parameters for all m components in the agent forms a multi-dimensional parameter space P. At any point in time, the agent adopts a particular configuration of these parameters. $p(s,t) in P$.

The problem of self-regulation in development can now be seen as a search process in a multi-dimensional parameter space to maintain optimal (or acceptable) performance. The performance is determined by a cost function $C : P- > R$ where R is a real number between 0.0 and 1.0. Formally, the goal is to find a configuration $p(s,t)$ such that: $C(p(s,t)) = C_{opt}$ where C_{opt} is the optimum cost.

Given this formulation, many techniques from optimisation theory (such as the Simplex algorithm, combinatorial optimisation, simulated annealing, evolutionary programming, etc.) become relevant. There can be little doubt that we are dealing with an NP-hard problem because the parameter space for any realistic developmental system is typically very large. So we must expect approximations, sub-optimal performance, and the use of heuristics. Moreover, the goal of the developmental system is not to reach a stable state, but to keep exploring the parameter landscape so as to maintain a balance of challenge and skill. In other words, as soon as a stable state is reached there should be a force to pull the system out of equilibrium again (see next section).

3.2 Monitoring Performance (the Cost function)

Next, each component must have a subprocess to monitor the performance of that component. Various types of monitors can normally be formulated easily for a particular component. Performance data is collected over a certain window of time, known as the observation window, and values are typically averaged and then compared to desired performance levels.

Thus there can be various performance measures related to the nature of the task that a component is trying to achieve. For example, a component in a language production system concerned with lexicon lookup can monitor how far the lexicon can cover all the meanings that are required to be expressed and how far the words that were chosen have been understood by the hearer. The optimal levels for these performance measures must be defined, and they are often related to challenge parameters. For example in lexicon lookup, one challenge is to keep the ratio between the number of words used and the number of predicates covered low (pushing the system to create words with complex meanings), another challenge is to increase the certainty with which a certain word has a certain meaning (pushing the system to seek disambiguated words as much as possible). In these cases, the monitored value reflects how far actual performance deviates from the desired performance level.

Without loss of generality, we assume that monitors yield a real value in the range $[0, 1]$ with 1 being optimal performance for a specific dimension. We associate with each component c_i and with the total agent c_Ta monitor vector $< m_{i,1}, ..., m_{i,n} >$. The set of all monitors for all m components in the system (and the total) forms a multi-dimensional space and system performance in response to a given stream of environmental stimuli traces a trajectory in this space. The performance of the agent at a time t, denoted as $M(s, t)$, is the averaged sum of the performance of all monitors for all components actively used by the agent. We can then define the cost function as $C(P(s, t)) = 1.0 - M(s, t)$, so that $C_{opt} = 0.0$.

3.3 Learning and Skill Levels

When the developing system is attempting to establish its global input-output mapping by chaining the mappings of each of its subcomponents, various failures may occur. Moreover, even if a mapping could be established, there may be a negative feedback signal later. Each component of the agent should be equiped with mechanisms to try and repair these failures. It is not important in the present context what kind of mechanisms are used. They could range from methods to increase needed resources (for example increase the memory available to a component), simple learning mechanisms (such as various forms of neural networks), or sophisticated symbolic machine learning techniques.

It is necessary for the agent to internally measure characteristics of the skill level of each component so that the system can track whether there is any significant change. For example, the amount of memory required by a component, the number of rules learned, the number of nodes or links in a network, etc. can all be quantified so that their evolution during development can be followed. Each component c_i has therefore an associated skill vector $< s_{i,1}, ..., s_{i,n} >$ which measures knowledge and skill levels.

It follows that each component c_i (and the agent as a whole) has an associated triple $c_i =< P_{i,j}, M_{i,k}, S_{i,l} >$, where $P_{i,j} =< p_{i,1}, ...p_{i,n} >$ is the challenge parameter vector, $M_{i,k} =< m_{i,1}, ..., m_{i,k} >$ is the monitor vector and $S_{i,l} =< s_{i,1}, ..., s_{i,l} >$ is the skill vector.

4 Self-regulation

Assuming that all components of the developing agent are designed this way, we can now focus on the global behavior of the agent, and particular the strategy to regulate challenge levels for a smooth, progressive self-development. The global system is an instance of combinatorial optimisation and hence has the same structure as well-known optimisation algorithms such as simulated annealing [10], in which a configuration of parameters needs to be found which gives optimal performance. There are two complications compared to traditional optimisation tasks: (1) The cost of a parameter configuration cannot simply be computed by applying a simple function (as in the travelling sales man for example, where cost is basically the length of a path) but must be derived from monitoring actual performance of the system over a particular period of time, including enought time to achieve the acquisition of the necessary skills to reach a certain performance level, (2) this monitoring period must include enough time for the system to acquire the necessary skills to reach a certain performance level. It is to be noted that the objective is not to get optimal performance, but rather to explore the landscape of possibilities in such a way that a higher degree of complexity is reached.

Optimization algorithms typically combine iterated improvement, in which there are small-scale changes to a configuration in order to find optimal parameter settings in a hill-climbing process, and randomisation, in which there is a change in a parameter which may initially cause a decrease in performance but helps the system to get out of a local minimum. Both aspects are present in the algorithms that we propose in the form of two alternating phases: (1) a phase in which challenge parameters are clamped until a steady performance level is reached after increases in skill levels through learning or resource allocation, this is called the operation phase, (2) a phase in which the challenge parameters are changed either because a steady performance could be reached, and so the skill level is getting too high for the challenge posed, or because performance could not be reached, and so the challenge is too high for the skill level. This is called the shake-up phase.

In our experiments to date we have found that the system should start with the lowest challenge levels possible for all components (instead of starting with a random configuration) so as to build up steadily in a bottom-up fashion.

4.1 Operation phase

The operation phase assumes that the challenge parameters are set at certain levels. The agent exercises its components and monitors the performance of each. A component becomes active when its various inputs are available. In case of failures, each component is assumed to have a set of processes (called ’repairs’) that can be used to fix the failure. For example, if a grasp action failed, the categorisation component receives a negative feedback signal and must extend its categorial repertoire to distinguish a new situation. Some of the repairs just involve the addition of additional resources, such as more memory or more processing cycles, others may require more sophisticated forms of learning. Because there are many possible failures in a given run and many possible repairs, some choice must be made about which repairs will be tried and how many.

The operation phase can be algorithmically described as follows:

Procedure Operation Phase

1. Select all executable components, i.e. components for which inputs are available, and activate them.

2. Monitor performance of these components (Could the input-output mapping be established? In how far does it satisfy the criteria set by current challenge parameters?)

3. If a component fails, extract a list of possible repairs and add them to the ’possible repair list’.

4. Consider the next series of components (go to step 1). If no more components can be executed, go to step 5.

5. Given a set of repair on the ’possible repair list’. First filter out those repairs that were executed on the same input stimuli, but failed. If there are repairs left, select the one(s) with the highest estimated effectiveness and execute it. If there are no more repairs restart from 0. A possible variation which considerable speeds up development is to restart the execution of components with the same input stimuli in order to attempt a solution.

4.2 Shake Up Phase

The goal of the shake up phase is to adjust the challenge parameters. In most combinatorial optimisation algorithms (such as simulated annealing) this is done in a random fashion by selecting arbitrarily a parameter and changing it. However, given the size of the parameter space for developmental systems of realistic complexity, such a weak search method does not give adequate results. Instead it is necessary to adapt parameters in a more structured way and maximally exploit available heuristics.

The shake up phase takes place after the system has sufficient experience with a given parameter setting. Sufficient experience means that the average performance level of each of the components and of the total agent does no longer change significantly during specific observation windows, and that there is no longer any significant increase in skill.

Two situations can now occur:

1. Performance does not reach anywhere near the desired levels. This means that the challenge levels are too high and that learning is no longer improving performance. We call this the A-state (where A comes from Anxiety).

2. Performance is consistently at a very high level. This means that operation of all components becomes routine and there is a potential for increased challenge. We call this the B-state (where B comes from Boredom).

Depending on the specific state, specific actions can be performed. Moreover a fine-grained analysis of these states is possible because performance for one component can be very high whereas that of another one can be very low. So changes to parameters should be heuristically guided by taking into account which components are in the A-state and which ones are in the B-state.

Another source of heuristic information is the dependency of components on each other. If a component is in the A-state, then this can be due to the complexity of the output coming from components that feed into it.

The final source of heuristic information is performance on the previous parameter configuration. Because optimisation algorithms are known to require an iterated approach towards optimal configurations, it is necessary to locally explore the parameter space hill climbing towards an adequate solution.

Procedure for the A-state

The goal of this procedure is to decide which challenge parameters to decrease.

1. If the previous parameter configuration had a better performance than the current one, then first switch back to the earlier configuration before making any change.

2. Select all components which are in the A-state. Either one of the challenge parameters must be decreased or else, one of the components feeding into it must be signalled to decrease the complexity of its output, by a recursive application of step 2. This step generates a set of possible choices for parameter adaptation. These choices can be heuristically ordered based on the performance of the components involved.

3. Choose one or more parameters, enact the change, and go back to the Operation Phase.

Procedure for the B-state

The goal of this procedure is to decide which challenge parameters to increase. There is a steady performance with the given parameters but there is perhaps an opportunity for further increase in skill. Note that this phase correspondence to the 'randomisation' phase of many combinatorial optimisation algorithms, although the parameter change is not completely random.

1. Collect all components which are in the B-state. The possibilities are: to increase one of the challenge parameters of this component, or else to recursively collect one of the challenge parameters that give input to this component. The possibilities can again be heuristically ordered based on the performance of the components involved.

2. Choose one or more of these parameters, enact the change, and go back to the Operation Phase.

Note that a record must be kept of parameter configurations and their associated performance in order to backtrack if needed.

We also know from our experiments that a conservative strategy (where only one repair is executed in the Operation Phase, and one parameter is changed in the Shake Up Phase) is much more desirable than drastic and rapid change.

From the viewpoint of optimization theory, the need for this shake-up process is not surprising. Optimization algorithms like simulated annealing typically combine iterated improvement, in which there are small-scale changes to a configuration in order to find optimal parameter settings in a hill-climbing process, and randomization, in which there is a change in a parameter which may initially cause a decrease in performance but helps the system to get out of a local minimum [14]. In fact it is only because of randomization that these local minima, i.e. situations where a stable but suboptimal solution is reached, can be avoided. Given that a complex cognitive agent is exploring a vast parameter space, the problem is an NP-hard (i.e. nondeterministic polynomial time-hard) problem as defined according

5 Conclusions

The paper has focused on the problem how an agent can self-regulate his own developmental process. It has proposed the autotelic principle, as a way to go beyond the classical reinforcement learning framework initiated by behaviorist psychology. There must be (i) ways to monitor performance and change in knowledge, skill, or resource use, (ii) ways to control the challenge level for the different components of the agent, and (iii) a general mechanism that self-adjusts challenge levels or shakes the system up to push the agent towards new heights. The motivational structure of the agent continuously tries to strike a balance between the highest possible level of challenge and skill.

6 Acknowledgement

I am indebted to Frederic Kaplan for many discussions on the architecture of agents.

References

[1] Churchland, P. and Sejnowski, T. (1995) The Computational Brain. The MIT Press, Cambridge Ma.

[2] Cohen, L. et.al. (2002) A constructivist model of infant cognition. Cognitive Development, 17(2002) 1323-1343.

[3] Csikszentmihalyi, M. (1978) Beyond Boredom and Anxiety: Experiencing Flow in Work and Play. Cambridge University Press, Cambridge.

[4] Csikszentmihalyi, M. (1990) Flow. The Psychology of Optimal Experience. Harper and Row, New York.

[5] Csikszentmihalyi, M. and I. Selega (Editors) (2001) Optimal Experience : Psychological Studies of Flow in Consciousness. Cambridge University Press, Cambridge.

[6] Elman, J. (1993). Learning and development in neural networks: The importance of starting small. Cognition, v. 48. p. 71-89

[7] Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D., Plunkett, K. (1996). Rethinking innateness: A connectionist perspective on development. Cambridge, MA: MIT Press.

[8] Hopfield, J. (1982) Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings of the National Academy of Sciences, USA, volume 79, pages 2554 to 2558, April 1982

[9] Johnson, M. (2003) The Infant Brain. In: Tokoro, M. and L. Steels (eds.) The Future of Learning Vol 1. IOS Press, Amsterdam. p. 101-116.

[10] Kirkpatrick, S., Gerlatt, C. D. Jr., and Vecchi, M.P., Optimization by Simulated Annealing, Science 220, 671-680, 1983.

[11] Matthews, J. (1993) Art Education as a form of child abuse. Lecture for the National Institute of Education Singapore.

[12] Kaplan, F. and PY Oudeyer (2004) A generic engine for open-ended sensory-motor development. Submitted to Robotics and Autonomous Systems.

[13] Newell, A. (1990) Unified Theories of Cognition. Harvard University Press, Cambridge, MA.

[14] Papadimitrious, C. and K. Steiglitz (1998) Combinatorial Optimization: Algorithms and Complexity. Dover, New York.

[15] Steels, L. and R. Brooks (eds.) (1995) The Artificial life Route to Artificial Intelligence. Building Situated Embodied Agents. Lawrence Erlbaum, New Haven.

[16] Skinner, B.F. (1953). Science and Human Behavior. New York: Macmillan.

[17] Steels, L. (2001) Language games for Autonomous Agents. IEEE Intelligent Systems. Sept/Oct Issues 2001.

[18] Thelen, B. and L. Smith (1994) A dynamic systems approach to cognition and development. MIT press, Cambridge Ma.

[19] Uchibe, E, M. Asada and K. Hosoda (1998) Environmental complexity control for vision-based learning mobile robot. In: Proc. of IEEE Int. Conf on Robotics and Automation. p. 1865-1870.

[20] Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2001). Autonomous mental development by robots and animals. Science, 291, 599-600.