Complexity Engineering
Harnessing Emergent Phenomena as Opportunities for Engineering

Jonas Buchli
Biologically Inspired Robotics Group
School of Computer & Communication Sciences
*Ecole Polytechnique Fédérale de Lausanne, Switzerland**

Cristina Costa Santini
The Intelligent Systems Group
Department of Electronics
University of York, UK†

Despite a lot of knowledge about complex systems the application of this knowledge to the engineering domain remains difficult. Efforts are scattered over many scientific and engineering disciplines. In this contribution we would like to motivate the union of these efforts by establishing complexity engineering as a discipline. We will motivate this initiative and show a few aspects that we consider important to arrive with this goal. After a historically inspired call for unification of the field we will present one of the most fundamental problems engineers are facing when deploying complex systems knowledge, what leads us to the formulation of *the self-organization – specification tradeoff principle*. Then, we dicuss what we consider central ingredients to complexity engineering: i) theory, ii) universal principles, iii) implementation substrates, iv) designing, programming and controlling methodologies v) collecting and sharing of experience in complexity engineering.

I. INTRODUCTION

As engineers we look with admiration, and often some envy at natural systems. It seems that nature has found ways to solve even real hard problems with great ease and elegance. This has lead many engineers and scientist to try to understand and ultimately harness the principles of nature for their own problems - it is not new that natural systems inspire engineers and scientists – but, only recently we started to gain insight and understanding in the mechanisms that give them their unique properties. Remarkably unifying principles have been found to rule very different systems, this research can be subsumed under the term "complexity science". From the knowledge and understanding collected in complexity science we can possibly profit and devise solution hitherto unsolved or partially solved engineering problem. As we will argue, however before we arrive with a methodological approach to such problems there is a need for formalization of the field. Quite some people are proposing ways of how to harness complex systems for engineering, yet we feel that the field is very dispersed and scattered over all different existing disciplines. In this paper we would like to show how the field could profit from a unification and formalization - into a discipline that we propose to call "complexity engineering".

Complexity engineering is in some way the art of exploiting opportunities coming up in the systems dynamics. We can direct and facilitate the occurrences of opportunities rather than just wait for them to pop up. Complex systems understanding can help us with this.

In this paper we would like to discuss some principles and ideas; some guidelines for complexity engineering. As in complexity science one looks for underlying and unifying principles among many systems, in complexity engineering we look into these different systems and their underlying principles from the point of view of applications.

This paper does by no means provide complete answers - its main idea is to motivate the need for complexity engineering and show some of the issues related to it, and especially stimulate people to think about "complexity engineering".

II. MOTIVATION

"The more science becomes divided into specialized disciplines, the more important it becomes to find unifying principles. Since complex systems are ubiquitous, we are confronted with the challenge of finding unifying principles for dealing with such systems. In order to describe a complex system at a microscopic level, we need an enormous amount of data which eventually nobody, even not a society, is able to handle. Therefore, we have to introduce some sort of economy of data collecting or of thinking. In addition, we can hope to obtain deep insights when we find laws which can be applied to a variety of quite different complex systems. When we look at universal laws it is wise to ask at which

*Electronic address: jonas@buchli.org; URL: http://birg.epfl.ch
†Electronic address: ccs500@york.ac.uk

level we wish to formulate them; be it microscopic or macroscopic." [11]

Despite enormous technological progress and abilities, there are still a lot of problems that defy an understanding let alone taking controlling influence over them.

A common property of these systems is that their complexity makes it impossible to treat them with a "divide and conquer" approach, where the problem is split up in smaller problems and in the end the solutions are assembled to give the overall solutions. Complexity science has made an enormous progress in the analysis and understanding of such systems yet the yield for practical problems is scarce.

We argue that the engineer that would like to use this body of knowledge has a different goal then the complexity scientist has in mind and this shapes the way of how to approach complex systems: *"But now one must use this understanding to design systems whose complex behavior can be controlled and directed to particular tasks. From scientific descriptions of the behavior observed in complex systems, one must learn how to engineer complex systems with specified behavior."[37]*

As of now, in many fields engineers deal with complex systems, yet the problems are not approached from a complexity point of view, e.g. the Internet, power grids, traffic planning, airline planning, etc. We see the applications of complexity engineering to 1) problems we do not have (good) solutions to, e.g. economics, robotics (swarms, locomotion, collective robotics), management and deployment of distributed systems, nano-systems, certain aspects of programming eco-engineering) and 2) systems that are complex but were not engineered swith complexity in mind (e.g. the Internet [5, 36], power-grids, traffic systems).

While we will discuss more ingredients for complexity engineering later, we will first focus on what we consider the most fundamental and important aspect of turning complexity engineering into a mature engineering discipline. Namely, the need for a *formalization* of the techniques and concepts in applying complexity science to engineering, thus developing a common language for complexity engineering. We argue that complexity engineering should be empowered by a strong theoretical fundament along with easy access to practical experience, the dissemination of the the gained knowledge, experience and methodologies should be facilitated by establishing curricula and centers of research around complexity engineering. We will highlight the motivation for constituents of complexity engineering in the following.

In order to arrive with an understanding on how engineering disciplines come into existence and mature we can we look back at the history of control engineering. The first industrial exploitation of control concepts has been in the 18th century for the control of steam engines. In order to harness the power of steam engines there was a need of controlling the energy production of these machines, which in the rotary steams engine translated in controlling the speed of revolution. One of the main mechanisms to control the steam engines was the centrifugal fly-ball governor whose design was completed by Watt in 1788 [20]. This was the first use of feedback control which reached the attention of a wide audience.

Many of the problems of feedback control were known from this mechanisms, i.e. loss of stability, oscillations, tradeoffs between speed of control and accuracy.

However, [20]: "The design of feedback control systems up through the Industrial Revolution was by trial and error together with a great deal of engineering intuition. Thus, it was more of an art than a science. In the mid 1800's mathematics was first used to analyze the stability of feedback control systems. [Since] mathematics is the formal language of automatic control theory,[. . .]".

The mathematical formalization and treatment helped to realize that stability problems are not specific to steam engines, to the fly-ball governor or to the way they were implemented, it rather showed that there are *underlying, abstract* principles governing *all* feedback loop systems, disregarding how they are implemented or what they are used for. Mathematics, systems theory in special was the way on how to formulate and understand these principles.

" [. . .] a mathematical understanding enables one to distinguish those model properties which are essentially model-independent or well-understood from properties which add key new elements [. . .]"[8] - while Grossberg talks about modeling here, in the same vein this understanding help us to separate abstract design issues from implementation details and considerations.

Another important aspect of a converging new discipline is the fact that strong theoretical results, often in forms *theorems*, arise. Often they are negative and constraining – saying "you can not do that no matter how hard you will try". This is however quite powerful as it gives boundaries of what is possible and they do not have to be explored by trial and error, the hard way, spending a lot of money effort. Also in control engineering such theorems exist and projects are known that could have saved millions of dollars and a lot crashed equipment by having access to this theorems [23].

It is important to realize that the formalization of the field by no means implies that the field is somehow frozen or does not evolve anymore. Quite the contrary, by formalizing the ideas one can realize

in what aspect these are special cases and thus formulate more general frameworks, leading to a fruitful feedback loop between theory and praxis. This positive feedback loop ultimately gives the field the momentum and the success that can be observed in modern control and communication engineering.

It is only in the second part of this century that the field of control engineering as such has emerged, formalized and put on a firm mathematical basis. By this, many before distinct problems became treatable in a unified way. However, it is in this time that the most powerful theoretical results and concepts have emerged. The formalization in the years before was a necessary prerequisite for the discovery of these principles. Control engineering nowadays is a multi-million dollar business and our modern life relies heavily on it.

Communication engineering has gone a similar way, from intuitive understanding of problems and solutions to a development of a firm mathematical background which helped the field to grow enormously and contributed to the development of new applications in an ever growing pace. It is probably not a coincidence but a common property of emerging and maturing engineering disciplines that they need to go through these stages: problem, intuitive/trial-and-error engineering and solutions, formalization, theoretical results, (commercial) success/adaptation.

In complexity engineering we are in the early stages of the development, in the "trial and error, intuitive engineering" phase. There are no methodologies, no common language and no common body of experience.

III. THE SELF-ORGANIZATION – SPECIFICATION TRADEOFF

Even though emergence and self-organization are beautiful and fascinating phenomena, in technological systems they do not always produce appropriate effects, e.g. no one would dispute that complete, nation-wide failures of power grids are not effects we are striving for when designing and constructing such systems. Examples of systems where self-organization leads to strong undesirable effects are countless: virus spread, cascading failures, buckling in steel structures, earth quakes, avalanches, riots and panics to name only a few.

On the other hand complexity science has clearly demonstrated very powerful properties of self-organization, properties that would clearly be beneficial to harness. Such properties include robustness, adaptivity, the ability to cope with ill-posed problems, optimization under constraints, exploiting synergies, resolving redundancies amongst others.

"But if a system is to be used for engineering, it must be possible to determine at least some aspects of its behaviour. Conventional engineering requires detailed specification of the precise behaviour of each component in a system. To make use of complex systems in engineering, one must relax this constraint, and instead require only some general or approximate specification of the overall behaviour of systems."[37] When trying to exploit self-organization in engineering system we are always faced with the same fundamental problem: On one hand we need certain functionality in the systems, i.e. we have to be able to specify what the system should do (or a part of it). On the other hand, if we specify "every" detail of the system, if we design it by decomposing it, "linearizing" the problem, then no self-organization will happen and no emergent phenomena can be harnessed[40].

Thus, there is a tradeoff between self-organization on one hand and specification or controllability on the other: If you increase the control over your system you will suppress self-organization capabilities. If you do not suppress the self-organization processes by avoiding constraining and controlling many variables it is difficult to specify what the system should do. This a very fundamental tradeoff that we have to consider, we call it the *self-organization – specification tradeoff principle*.

Let us discuss one way to approach this principle in a more formal way by relating it to existing theories.

A. The continuum of systems variables: constrained vs. unconstrained variables

We introduced the self-organization specification tradeoff principle in an intuitive way. Here we capture and discuss this problem in a more concise way.

What we try to achieve by complexity engineering is guided pattern formation. The function that the system should fulfill can be looked at as a partially specified pattern.

Thus, a part of the pattern[41] is specified, the rest of the system completes the pattern on the additional degrees of freedom. It is natural to turn to theories which deal with pattern formation, therefore the background of the following comes from non-equilibrium statistical physics and nonlinear dynamics, where it has been realized, that principles of this field apply to a vast number of other systems and are able to explain many aspects of pattern formation. Thus, we propose to dicuss this in terms of Systems variables, i.e. variables that describe the state of the system on a chosen level of abstraction. This discussion bases of the synergetic viewpoint of systems [10]. It has been realized that in systems with many active interacting

subsystems the system often self-organizes in low dimensional dynamics: the variables that describe this behavior are the order parameters of the system. Variables influencing the behavior of the order parameters in a critical way are called control parameters[42]. This means we can effectively describe the state of the system with the order parameters, however changing the control parameters the system can start to break up the order and more and more variables are needed to describe system (let us assume for the discussion a maximum of N variables describes the systems on the subsystem level, where usually $N \to \infty$).

The functionality of the system can be expressed by the relationship of the state of a subset of the systems variables or aggregates of them (order parameters are clearly a natural choice) and constraints or boundary conditions on another subset of the systems variables (or aggregates, i.e. the control parameters). As an example consider a logic gate implemented in VLSI. If voltages V_1 and V_2 are applied to the inputs of the gate V_o should be $f(V_1, V_2)$[43], where f is the logic function implemented with the gate. $V_{1,2}$ are variable constraints applied to the system, furthermore there are built-in constraints by the choice of the doping, geometry and connection of the different layers, etc. V_o, an aggregate measure, is a statistic measure about the charge carrier distribution in a part of the semiconductor. (Note that the distinction between constrained and unconstrained is somewhat subtle, sometimes it might look like the number of constrained variables is actually small, but in fact it is not. Consider transistors used in digital logic, in both operation modes ("binary 0" and "binary 1") by applying strong fields (external and built-in) the designer ensures that all variables (position and momentum) of the charge carriers follow a very specific statistics which is well known and easy to handle, so the single control variable V in effect strongly constrains an enormous number of state variables). Another example is a hydraulic piston [9]. It can be very well treated with thermodynamic formalism, i.e. it is a system with a huge number of state variables, but for our purposes we engineer the system in such a way that the resulting movement is essentially one dimensional and the position in this dimension can be exactly controlled by the control parameter pressure in the liquid.

Important to realize from this discussion is that we put the system in a mode of operation where to constraints rule the system. The effects of the constraints are designed to be orders of magnitude bigger then any self-organization mechanism, so they can be neglected in the analysis and prediction of the systems behavior.

In the example of the logic gate this corresponds to the fact that the gate is not operated close to the switching region where we would see inherent oscillations, flipping back and forth enhanced fluctuation all hallmarks of self-organization process among the charge carriers. Another example can be found in chemistry, where most chemical reactions are studied in their equilibrium state though being exemplary nonlinear systems [27]. As reported by Epstein in [6], simple chemical systems and reaction mechanisms involving a small number of components may give rise to a remarkable variety of dynamical phenomena if the systems are maintained sufficiently far from equilibrium.

Let N be the number of systems variables needed to describe the system on a chosen level of abstraction. We introduce a new property to characterizes the system: the number of constrained system parameters N_c (along we introduce the number of unconstrained system variables N_u where $N_u + N_c = N$).

By constrained system variables we describe the variables on which as engineers we impose constraints in order to bring the system to exhibit a certain functionality. N_u are unconstrained variables on which only the system takes influence by its intrinsic dynamics.

As we have already seen there seems to be a fundamental conflict between self-organization on one hand and the need for control and specification in engineering on the other hand: if one constrains all the variables (i.e. $N_c \to N$) no self-organization will happen because we completely impose the dynamics by external forcing and no emergent phenomena can possibly be harnessed.

The key to harnessing self-organization for engineering is the let loose of the control over almost all variables but a few, by that the system can find its own way of dealing with the situation on all the dimension that are not controlled and (as supposed by thermodynamics) find efficient solutions.

Thus, through the study of the system's dynamics one will be able to control the system by determining just a few variables.

Sometimes however, there is only a limited way of influence the choice of N_c and N_u, e.g. crowd control, traffic management, ecological engineering, operations research.

We see that in that sense classical engineering is a limit case ($N_c \to N$) of complexity engineering. Further it comes directly clear, that noise in your system can be beneficial as it helps to explore new possibilities in the system as they come up. If the new mode of the system is a strong attractor the system will be stable anyway, if it is not a strong attractor the system becomes instable and branches of to find new stable states. But this can only happen if the system is not constrained too much. On the other hand the system clearly has to be constrained in the variables on which we have to imply

boundaries out of safety or other functionality considerations.

Now, of course it is not sufficient to go to a given system, pull out the control over many of the variables and think it will self-organize into doing something useful after that. Rather we have to careful engineer the right interactions into the system, so that the systems self-organizing capabilities serve our purpose, i.e. they do satisfy and support the constraints on N_c. The methods how we can do that are subject of complexity engineering and complexity science is the method to arrive with this knowledge.

B. Self-Organization

Let us shortly discuss why should we try to use complex systems and self-organization for engineering in the first place? What is interesting about self-organization for engineering?

Robustness – Self-organized processes are often very robust, as they consist of a vast number of low level entities. The behavior is a low dimensional attractor of the high dimensional system dynamics, this compression induces redundancy and robustness.

Adaptivity – Due to the constant operation of the self-organizing mechanisms, when the boundary conditions change, the system rearranges to a new mode of operation.

Emergence– Self-organization offers new opportunities, by showing emergent phenomena that are not a result of the function of the single elements but a consequence of the *interaction* of these elements. The system can produce new unexpected results which can possibly be harnessed. However the emergence of new phenomena can also be difficult to handle, can induce problematic effects, we will discuss this issue in the next section.

Several examples of self-organization can be found in natural and artifical systems. In [15] the authors show how a macroscopic ramified network can minimize the resistance by self-organizing into suitable structures. In [3] the authors show how, if the body properties of a hopper robot is exploited by complementary controllers energy efficient forward locomotion emerges. In chemical systems self-assembly and self-organization can be found and harnessed in various ways: *"In particular, the spontaneous but controlled generation of well-defined, functional supramolecular architectures of nanometric size through self-organization represents a means of performing programmed engineering and processing of nano-materials."*[19]. The field of supramolecular chemistry offers possibilities to build functional molecular and supramolecular devices by the setting of the right state variables, arriving at a system that will self-organize to behave as expected, as designed.

In biological systems one finds striking examples of self-organization, and *"Indeed the phenomenon of self-organization may have been responsible for the emergence of life itself [...]"*[18, 26]. For example, the tobacco mosaic virus (TMV) [17], which is a helical virus particle composed of 2130 identical subunits, each comprising 158 amino acids, can be dissociated into its component parts and then self-reassembles accurately in vitro to give totally functional viral particles.

In a recent review paper [39], Zauner presents new concepts of information processing, stressing that one needs to loose *"narrow prescriptive control over elementary structures and function in order to develop self-organizing architectures"*. Then he points two issues that needs to be addressed: a novel engineering approach that does not depend on the direct control of individual components and the development of computing concepts that exploit rather than suppress the physics of the materials used for their implementation.

IV. COMPLEXITY ENGINEERING

In this section we will describe in more details some of the aspects that we think are needed for a mature field of complexity engineering. Briefly, these are A) theory, B) universal principles, C) implementation substrates, D) designing, programming and controlling methodologies E) collecting and sharing of experience in complexity engineering.

Our working hypothesis is that by (i) identifying the problem through a complexity engineering perspective; (ii) finding the adequate substrate (complex system) and finally (iii) being able to program or design that system (finding its basins of attraction, state variables, initial conditions, boundary conditions and not trying to compute and determine its input-output mapping); it will be possible to solve complex problems and have the artificial systems that would exhibit the desired properties of robustness, adaptivity and emergence.

We focus on a way of how to approach and to think difficult engineering problems and we will try to show how thinking on abstract/meta-level about the problems can help to overcome the traditional, linear and static mind-set and way of thinking.

A. Theory, fundamentals, theorems

The theory will be the unifying language of complexity engineering, complexity science will be at the core of the theory, but will have to be enhanced by engineering specific aspects. Being able to talk about the different aspects of engineering, i.e. systems, specifications, etc in a clear formal way will help to see common principles in seemingly different systems and transfer this principles to different substrates. Our claim is that by having complexity engineering as a field, the theorems that are shared between different areas are going to emerge as *complexity engineering theorems*, much as the waterbed formulas and stability theorems in control engineering (cf. [29]) or the Shannon-Hartley theorem in communication engineering [28].

B. Universal concepts, paradigms and principles

The theory will help us to look at different systems and understand their common properties. To illustrate this, consider yourself looking at a movie of a traffic jam filmed from a helicopter, the movie is sped and all of a sudden you are able to observe wavelike structures that travel through the traffic jam, it reminds you of water waves. Are there common principles ruling both systems? Can the knowledge gained and tools developed in one field (the water waves) possibly be harnessed for the other problem (traffic control)? Recent research in the area of complex systems suggest an affirmative answer, not only for the concrete example [24], but for finding universal laws governing very different systems [10].

Thus, there are principles which can be formulated on an abstract level (i.e. independent of their implementation or specific system supporting it). Pointing out these abstract principles is important because when different systems share common principles in their mode of operations, in their dynamics, it is possible to transfer knowledge and techniques between these systems.

Next we give a few examples of abstract principles that can be applied to solve different problems. This list is by no means conclusive and we hope that it will grow enormously over time.

Reaction Diffusion systems – Create coordinated spatio-temporal patterns [4, 34, 38].

Pattern recognition – Compare input against templates [12].

Optimization – Optimize one or several aspects of a system under constraints [15].

Stochastic Resonance – Improve a system's performance by adding noise [31, 35].

Diffusion – Exploit gradients to distribute information or material.

Traveling waves, Solitons Packets of activity travel through the system, transfer information and interact.

Winner-takes-all – Implements a decision process, one out of many is chosen.

Voronoi diagrams – Find points of equal distance from a number of points [1].

Edge detection – Find and enforce regions in which properties change in an abrupt fashion [4, 22].

Ant algorithms – Solve shortest path problems [33].

Coupled Oscillators – Create coordinated but flexible cyclic patterns and waves, synchronized and coordinated with input. Control and modulate them with low dimensional control signals. Reduce the dimensionality of the control problem [2, 3, 14].

Note that many of the principles in this list Turing complete machines can implement, but this is probably not the interesting fact, since von-Neumann computing is probably not the appropriate concept to approach many real world problems, i.e. people deal efficiently with many NP complete problems on a daily bases while the "theory of computation" has a hard time with them. Thus, "vN-computing" is not the principle you are looking for when approaching this kind of problems. Often one does not look for "common purpose" or "universal" machines, but rather for special purpose, non-universal machines that are then however orders of magnitudes more efficient in dealing with the given problem. It is our goal to strive for "universal principles" i.e. methodologies of how of devise such special purpose machines for a given task.

C. Implementation "substrates"

Now there are a wide range of different systems that support one or several of the above principles, consequently they can be used as an implementation substrate for such. Important to note here is the fact that the physical systems have a characteristic time and length scale (or a range thereof) - thus depending on which scales we would like to implement above one or the other substrate might be more or less suitable, e.g. reaction-diffusion dynamics can be exploited in chemical systems (with

time scales ranging from minutes to fraction of seconds) or in semi-conductors with time scales in the order of $10^{-3} - 10^{-12}$ s. We highlight the connection between implementation substrate and the abstract principle (cf. A.) for a few examples, then we give a list of more systems that can possibly be used. Again this list is by no means complete and its main point here is to stimulate the discovery, documentation and exploitation of many different possibilities and last but not least a discussion on how we implement our solutions.

Reaction-diffusion media – In [1] Adamatzky presents space-time patterns in active nonlinear media and suggest some useful applications such as image processing, computing logic functions and control of robots. He presents some principles, such as the Voronoi diagram that can be constructed in reaction-diffusion processors, a light-sensitive chemical controller, a micro-array of coupled oscillating gel actuators in BZ reaction, etc.

Amorphous computing – Amorphous Computing [7] aim is to invent programming methodologies for multi-agent systems. In [25] Nagpal presents how organizing principles from multi-cellular organisms may apply to multi-agent systems, by presenting examples of biologically-inspired local primitives (such as morphogen gradients and positional information) and global patterns (such as cell differentiation) for engineering robust collective behavior.

Locomotion control in robotics – In locomotion control for dextrous, legged robots often the biological concept of the central pattern generator (CPG) is deployed [14]. The CPGs, from the abstract point of view, are systems which generate complex coordinated, usually cyclic patterns for locomotion, respiration, flying, stomatogastric systems etc. CPGs are often modeled by coupled nonlinear oscillators which are then in turn implemented in microcontroller/computers, but also in analog electronics [21, 30].

Thus, this three examples give an idea of how the following systems could be used to implement functional systems:

- doped materials
- supramolecular systems - liquid crystals
- biomolecules: proteins, enzymes
- analog electronics
- chemical reactions
- reaction-diffusion media
- quantum systems
- Josephson junctions
- crowds
- waves
- living systems: bees, ants, bacteria, virus, etc.

For the successful exploitation of many of the above substrates one of the biggest obstacles is the interfacing problem, i.e. how to give input and read the output of the system. As for now, the interfacing to electronic systems is developed best.

Of course for many implementations, just for their ease of configuration, modification and interfacing, digital computers of some sort will still be the implementation substrate of choice - at least unless their lack of intrinsic dynamics and parallelization outweighs their universality. Furthermore, sometimes a part of the "implementation" substrate is given, e.g. in ecological engineering or traffic management.

D. Designing, programming, controlling

In complexity engineering one works on all (or at least several) scales, from micro to macro, as opposed to traditional engineering,which generally searches for solutions on the same scales. For example, one possible way to approach a problem is first to identify suitable macroscale behavior; then investigate on the microscale the mechanism that causes this behavior and finally find the suitable substrate to implement the microscale mechanism.

Looking at a problem from the complexity perspective is already the way towards the solution. A few points that help to look at a system from the "complexity point of view":

- What happens in non-nominal states?
- Which regions are "problematic" (enhanced fluctuations, oscillations, ...)?
- What are the important assumptions, simplifications and abstractions and bounds of validity of the way of analyzing and designing the system? What happens if they are invalidated?
- Think about nonlinearities and second order effects [32].
- Identify gradients, find the throughput trough your systems (what makes them open, active systems) [13].

¿From a conceptual point of view, in order to tackle complexity engineering problems, one should (i) be able to take an abstract perspective (cf. A. and B.); (ii) take what could be considered disadvantage as advantage (e.g. stochastic resonance, robotics and intrinsic dynamics) and (iii) enable the system to intrinsically harness opportunities.

E. Gained methodology and experience

In the past every field of engineering has started out from practitioners, i.e. people gaining a lot of experience with common problems occurring and developing intuitively approaches on how to solve them. In a later stage theoretical fundaments are laid and the experience is collected in a systematic fashion.

Furthermore, in every field of engineering there is a lot of rules of thumb, lessons that can be learned from former projects, methodologies and so on, essentially connecting theory and practice. There are formal opportunities (journals, meetings, colloquia, etc) and more informal channels where this information is collected and distributed. Engineering students get exposed to it early in their formation and learn "the art of engineering" next to the formal concepts largely by being involved into discussions, projects and being exposed to the "common knowledge".

We should aim at forming such opportunities for complexity engineering as well. This means organizing conferences, research networks, and especially also curricula aimed specifically at complexity engineering.

V. DISCUSSION

Thus far we would like to discuss a few important points related to the presented ideas.

We have motivated the need for establishing complexity engineering as a discipline, however a common fear that opposes the deployment of complex systems knowledge in engineering is the idea that all control over the system will be lost, however the issue about complexity engineering is not that the developed systems will be unpredictable, nondeterministic or uncontrolled. The output (i.e certain aspects) may be predicted and controlled - it is more how the systems arrived to that output that can not be known, unpredictable, complex or not computationally reproducible. Moreover, the guarantees about the functioning of the system will be of statistical nature. This is not really new, in all engineering work, the guarantees that can be made about a system are limited and essentially of statistical nature. Furthermore, it shows a wrong understanding of how the development of technology works if a 100% understanding of the functioning of a system is demanded before it is accepted. Humans have worked throughout history with systems which they did not fully understand but nevertheless brought them good services, (e.g. plants, cattle, electricity), often the understanding is deepened after which can lead to a yet better or more efficient use. Complexity engineering is the way to find the balance between controllability,predictability and a letting loose of some aspects of the system. We hope to show one way on how to think about this balance in this contribution. Further, complexity engineering is not meant to replace "traditional engineering" approaches, but is merely an extension or generalization to it. Complexity engineering has to be applied to the appropriate problems.

Even if the complexity engineer will take inspiration and learn a lot from physical and living system, his goal is not in first line to understand nature, thus if he finds that a few modifications to a model serve him well, even if this modifications are not justified by observations in the real systems, there is no reason why he should not use the modified system and no justification for the change is needed. If on the other hand the goal is to devise a model which should serve for understanding the real world phenomena more care with modification is in place, but this is the primary concern of the scientist.

In order to realize the importance of complexity for engineering one needs to consider that every system is basically a complex one. most of the traditional systems are operated in a regime where the "complexity properties" are neglectable. While traditional engineering was the art of taking the complexity out of the systems, we now have to gently allow complexity to come back in and learn to exploit it for our own good.

VI. CONCLUSION & OUTLOOK

While we are certainly aware of the fact that we can not force a new discipline to emerge we are convinced that we can stimulate its formation. We have tried to motivate the need for complexitiy engineering as a displine and hope that many engineers and scientist will contribute in the future. We argue that complexity science with a complexity engineering perspective in mind could be more goal directed than todays distributed attempts.

While the ideas presented here, even if founded on rigorous theories, are rather conceptual we plan to investigate some of the aspects we presented here in more detail, especially the self-organization – specification tradeoff principle. It should be possible to show the tradeoff on a simple model where one can adjust the number of constrained variables and investigate on the effect it has on the system.

Lastly, it could be that the discussion of constraints and how to organize them goes well beyond sole application to engineering but is of more fundamental importance: In [16] Kauffman writes "I said we have no theory of organization, but I have the deep suspicion that this reciprocal linking of work and constraints on the release of energy that constitutes work is part of that theory. If so, notice

that this is not part of physics at present, nor of chemistry, nor of biology."

Acknowledgments

Thanks to the people that have brought surfing and paragliding to the world. In both of these sports a lot about harnessing opportunities in self-organized system dynamics for our own good can be learned. *Force it and you loose – be prepared, work with the elements and have the ride of your life!*

We are grateful to the Santa Fe Institute for the organization and support of their annual summer school in the frame of which the initial work on this paper has been done. We would like to acknowledge the importance of the discussions with all the lecturers and participants of the CSSS05, especially the tumbleweed gang.

Further, we are grateful to our supervisors, Auke Jan Ijspeert (EPFL) and Prof. Andy Tyrrell (University of York) for discussions and support.

J.B. acknowledges support from the Swiss National Science Foundation through a Young Professorship Grant to Auke Jan Ijspeert.

C.C.S. is supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship no. E04D028324BR.

[1] A. Adamatzky. *Molecular Computing*, chapter Computing in Reaction-Diffusion and Excitable Media: Case Studies of Unconventional Processors, pages 63–90. MIT Press, Cambridge, 2003.

[2] J. Buchli and A.J. Ijspeert. Distributed central pattern generator model for robotics application based on phase sensitivity analysis. In A.J. Ijspeert, M. Murata, and N. Wakamiya, editors, *Biologically Inspired Approaches to Advanced Information Technology: First International Workshop, BioADIT 2004*, volume 3141 of *Lecture Notes in Computer Science*, pages 333–349. Springer Verlag Berlin Heidelberg, 2004.

[3] J. Buchli, L. Righetti, and A.J. Ijspeert. A dynamical systems approach to learning: a frequency-adaptive hopper robot. In *Proceedings of the VIIIth European Conference on Artificial Life ECAL 2005*, Lecture Notes in Artificial Intelligence, pages 210–220. Springer Verlag, 2005.

[4] M. Conrad and K.P. Zauner. *Molecular Computing*, chapter Chemical based computing and problems of high computational complexity: the reaction diffusion paradigm, pages 1–31. MIT Press, Cambridge, 2003.

[5] J.C. Doyle, S.H Low, F. Paganini, G. Vinnicombe, W. Willinger, and P. Parrillo. *Robust Design*, chapter Robustness and the Internet: Design and Evolution, pages 231–271. Santa Fe Institute studies in the science of complexity. Oxford University Press, 2005.

[6] I. R. Epstein. Complex dynamical behavior in "simple" chemical systems. *J. Phys. Chem.*, 88:187–198, 1984.

[7] Abelson et al. Amorphous computing. *Communications of the ACM*, 43(5), 2000.

[8] S. Grossberg. *The Adaptive Brain I. Cognition, Learning, Reinforcement, and Rhythm*, chapter Associative and competitive principles of learning and development: the temporal unfolding and stability of STM and LTM patterns. North-Holland, Amsterdam, 1988.

[9] S. Guerin. Personal communication.

[10] H. Haken. *Synergetics. An introduction.* Springer Verlag Berlin Heidelberg, 3rd edition, 1983.

[11] H. Haken. *Information and Self-Organization - A Macroscopic Approach to Complex Systems.* Springer, 1999.

[12] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd edition, 1998.

[13] A.W. Hübler. Predicting complex systems with a holistic approach: The "throughput" criterion. *Complexity*, 10(3):11–16, 2005.

[14] A.J. Ijspeert. Vertebrate locomotion. In M.A. Arbib, editor, *The handbook of brain theory and neural networks*, pages 649–654. MIT Press, 2003.

[15] J.K. Jun and A. Hübler. Formation and structure of ramified charge transportation networks in an electromechanical system. *PNAS*, 102:536–540, 2005.

[16] S. Kauffman. Molecular autonomous agents. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 361(1807):1089–1099, 2003.

[17] A. Klug. From macromolecules to biological assemblies (nobel lecture). *Angew. Chem.*, 22(8):565–582, 1983.

[18] H. Kuhn and J. Waser. Molecular self-organization and the origin of life. *Angewandte Chemie International Edition in English*, 20(6-7):500–520, 1981.

[19] J.M. Lehn. Supramolecular chemistry: from molecular information towards self-organization and complex matter. *Reports on progress in physics*, 67:249–265, 2004.

[20] F.L. Lewis. *Applied Optimal Control and Estimation*, chapter Introduction to Modern Control Theory. Prentice-Hall, 1992.

[21] M.A. Lewis, R. Etienne-Cummings, M. Hartmann, Z.R. Xu, and A.H. Cohen. An in silico central pattern generator: silicon oscillator, coupling entrainment, and physical computation. *Biological Cybernetics*, 88:137–151, 2003.

[22] M. Mahowald. Analog vlsi chip for stereocorrespondence. In *Proc. IEEE Int. Symposium on Circuits and Systems*, volume 6, pages 347–350. IEEE, 1994.

[23] M. Morari. Personal communication (Lectures on Control Theory, ETHZ).

[24] T. Nagatani. The physics of traffic jams. *Rep. Prog. Phys.*, 65:1331–1386, 2002.

[25] R. Nagpal. A catalog of biologically-inspired primitives for engineering self-organization. In G. Di Marzo Serugendo, A. Karageorgos, and Rana O.F., editors, *Engineering Self-Organising Systems:*

Nature-Inspired Approaches to Software Engineering, number 2977 in Lecture Notes in Computer Science, pages 53–62. Springer, 2004.

[26] D. Philp and J.F. Stoddart. Self-assembly in natural and unnatural systems. *Angewandte Chemie International Edition in English*, 35(11):1154–1196, 1996.

[27] Stephen K. Scott. *Oscillations, Waves and Chaos in Chemical Kinetics*. Oxford chemistry primers. Oxford University Press Inc., 1994.

[28] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656, 1948.

[29] S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control*. John Wiley & Sons, Chichester, 1996.

[30] S. Still and M.W. Tilden. Coupled oscillators and walking control: a hardware implementation of a distributed motor system. In N. Elsner and R. Wehner, editors, *Proceedings of the 26th Goettingen Neurobiology Conference*, volume 2, page 262, 1998.

[31] R. Stoop, J. Buchli, G. Keller, and W.H. Steeb. Stochastic resonance in pattern recognition by a holographic neuron model. *Physical Review E*, 67, 2003.

[32] S.H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001.

[33] H. Van Dyke Parunak. "go to the ant": Engineering principles from natural multi-agent systems. *Annals of Operation Research*, 75:69–101, 1997.

[34] V.K. Vanag, Y. Lingfra, M. Dolink, A.M. Zhabotinsky, and I.R. Epstein. Oscillatory cluster patterns in a homogeneous chemical system with global feedback. *Nature*, 406:389–391, 2000.

[35] K. Wiesenfeld and F. Moss. Stochastic resonance and the benefits of noise: From ice ages to crayfish and squids. *Nature*, 373:33–36, January 1995.

[36] W. Willinger and J. Doyle. *Robust Design*, chapter Robustness and the Internet: Design and Evolution, pages 231–271. Santa Fe Institute studies in the science of complexity. Oxford University Press, 2005.

[37] S. Wolfram. Approaches to complexity engineering. *Physica D*, 22:385–399, 1986.

[38] L. Yang and I.R. Epstein. Oscillatory turing patterns in reaction-diffusion systems. *Physical Review Letters*, 90(17), 2003.

[39] K.P Zauner. Molecular information technology. *Critical Reviews in Solid State and Materials Sciences*, 30:33–69, 2005.

[40] Respectively the self-organization that will happen in such systems was not planned in and accounted for, it will start to be apparent in non-nominal situations only, when the interactions that were not taken in account start to be important. In this cases this effects are usually negative or even catastrophic, e.g. cascading failures in power grids, congestion breakdown in communication networks, burn out of components.

[41] Note that with pattern we do imply spatio-temporal pattern as well as purely spatial or temporal patterns

[42] This effects chain up, i.e. the order parameters of one level can be the subsystems of the next "higher" level.

[43] Instead of simple functions this can be extended to sequences or similar concepts in order to account for dynamics, but this is of no importance for the discussion in this section.