Chapter 5
Systems Sciences and Cognitive Systems

Abstract. The evolvable multi-scale engineering design is presented in correlation with general design theory. The role of meta-models for evolvable and creative conceptual design is emphasized.

The potential of active cases base reasoning systems and their interaction with designs of experiments is evaluated.

Evolvable diagnosis strategies for failure analysis and security purposes are proposed.

Manufacturing systems developments from fixed to flexible, reconfigurable and lastly evolvable with reference to assembly operations are presented. Multiple-scale agent architectures based on cognitive science studies allows integrative closure and autonomy.

5.1 Evolvability for Engineering Design

5.1.1 Modeling Design Processes

As products become increasingly complicated and technology becomes increasingly advanced, the amount of engineering knowledge and of operations required from engineering designers is extensive (Bar-Yam 2003).

Several major problems in developing the computer aided design, CAD systems are related to the complexity advent in industry. Because designs are subject to a dynamical industrial context they must be able to change with the context.

The evolvable designs, and the evolvable CAD, ECAD, represent the target methodology to confront complexity advent in design. Evolvable designs are those that are more easily modified in accordance to shifting consumer demands, safety constraints and dynamic environment. Those designs that have ability to evolve can change more quickly in concert with the dynamic market and then have a better chance for continued survival on the market.

The design process has been described by various authors and from a variety of points of view. Descriptive models explain how design is done, systematic prescriptive type of models show how design should be done, cognitive models explain the designer thinking process. The correlation with the cognitive and system sciences methods as artificial intelligence, AI or artificial life, AL, methods start to be emphasized in design activities.

Interaction between subjective knowledge and objective world was discussed in significant descriptive theories of design. Yoshikawa (1981) proposed the theory called General Design Theory GDT in which the interaction between a designer and an objective world is formulated as a continuous map between two topological spaces Extended GDT (Tomiyama and Yoshikawa 1987) includes major developments of the GDT. Correlated to such approach are the axiomatic approach of design and other mathematical theories of design (Reich 1995, Braha and Reich 2003).

The systematic design approach describes the engineering activities as a sequence of phases as for instance: clarification of task, conceptual design, embodiment layout and detailed design (Pahl and Beitz 1996). Related to this approach is the universal design theory that view design as a finite number of abstraction levels and a set of structures stages to follow (Grabowski et al. 1998). These methodologies are in the same time descriptive and prescriptive focusing mainly on how design should be done as a procedure.

The cognitive design approach, identify and represents the cognitive or mental activities in design. A class of models refers to logical frameworks for design (Coyne 1988, Takeda et al. 1990). The design process is regarded as an evolutionary process that is the design is improved step by step. The design grammar and the analogies between language and design have been emphasized by Coyne (Coyne 1988). These grammars are interesting for conceptual design work, more concerned with reasoning in terms of engineering concept than physical parts (Pahl and Beitz 1996).

Inspired also by cognitive methods the so-called conceptual cycles of Sowa (Sowa 2000) may be considered as an illustration of Piaget (Piaget 1971) developmental theory that found applications in design modeling.

5.1.2 Framework for Engineering Design

Although engineering design has always been associated with human creativity and skill, the generation of designs can be formalized in a structured manner. Such a formalization of design synthesis enables automatic design synthesis through computation.

The SKUP framework capability for engineering design and relation with GDT will be evaluated in what follows.

For design modeling the states S, in SKUP will be associated to real artifacts or to solutions of the design. The conditions K are associated to symbolic or ideal design, to specifications or knowledge.

For evolvability we need to consider that the designer system K and the designed one S, the operators U and the possibilities P are vectors or tensors. This means multiple conditioning levels or in other terms multiple time scales. The wave equation, WE, may generate and organize the space of conditions K. The multiple scales allow combining sequential and iterative steps in design. Large designs can be decomposed into small designs each having similar structures.

The operator U characterizes the capability to pass from plans or symbols K to reality based on previous reality and on new plans. In one of the simplest case U describes the concatenation of successively realized stage of the plans.

The states S capability to reinitiate some of the plans and to modify the symbolic description is characterized by the possibility P. P describes the selective activation or deactivation of conditions pertaining to K.

Evolvable designs should be based on the complementarity of upward and downward causation. The operator U, from the SKUP is associated to the upward design. Typically upward design is sequential and easier to model. The design may start with the first-order wave equation, WE solutions.

These generate the design schemes.

GDT formulates engineering design as a process initiated by ideal specifications such as functional requirements in the function space. This corresponds to the conditions space K, in the SKUP. The designer is able to match partial structural information in the attributes space corresponding to S.

The operator U is linked to deduction stages in GDT.

The possibilities P from the SKUP are associated to downward design. Complementing the upward typically discrete design, it is the downward design working with continuous parameters states. The appropriate downward dynamical design based on specifications or on dynamical parameter measurements may contribute to the closure and finally the design scheme evolvability. The downward design is local and parallel.

The possibilities P are associated to the abduction step in GDT.

Abduction is the method to integrate knowledge that satisfies the two aspects of the evolutionary design that is creates new product and expands knowledge.

U and P have complementary roles.

The conventional design cycle is focused upon registering and compensating the deviation from established, specified goals. Conventional design is about this single loop design cycle and is adaptive. But in complex situations it happens that this cycle is not achievable in its present form. Developing the conventional design, the evolvable design focuses upon goal formation. The general goal formation is included in the design cycle not outside to the cycle. In other words the evolvable design should be more focused to what happens inside the design cycle.

The transformations within the design cycles may be more important. The multi-scale transformations allow accelerating designs and potentially make the design evolvable. In conventional design, the partial goals formation that is, the design schemes set up is external to the design cycle. The design schemes are established too early and are fixed in most cases. Modifications of design schemes, if performed, are too late. As the design scheme formation is more included in the multi scale design cycle, the design itself may become more evolvable. Dynamic markets require that the design have the capability to adapt itself to an environment within its life span.

Elements of the general SKUP schema may be identified in the study of evolutionary stimulation in conceptual design. The process design starts from pre-inventive structure, in K, and follows a cycle including generating design rules, U, and exploration and understanding of the results, that is P following S. The new state in K includes the evolutionary design. A practical example is the cyclic process of chemical product design (Gani 2004). In this case K denotes the conceptual designs, S the computer aided design, CAD solutions, U-the methods and constraint selection, P-the result analysis and verification. The K-process corresponds to the evolution from pre-designs to post-designs.

5.1.3 Multiple Scales Evolvable Designs

Evolvable designs, ED, should have an adjustable or modifiable architecture since they contain interacting components designs subjected to continuous reorganization after confronting the reality.

It was observed that design steps consist of typical elementary processes. These form the so called design cycle. Design cycle solves a small design problem or divides it into smaller sub-problems. This observation led to a model which is the repetition of design cycles at different scales (Yoshikawa 1981, Takeda et al. 1990).

Yoshikawa (1981) considered that the so-called ontogenetic design can be decomposed into small design cycles. Each cycle has sub-processes focusing on research, development, testing and evaluation aspects.

Following GDT Let us consider that the design cycle has four basic processes or actions: R-research (includes problem identification and suggesting key concepts to solve the problem). D-development (includes developing alternatives from the key concepts by using design knowledge. T-testing (includes evaluation of alternatives, simulations).

E-evaluation and adaptation (includes selection of a candidate for adaptation and modification, conclusions).

K is the symbolic description of the system, in this particular case, the planned choices for elementary design processes such as research-R, development-D, testing-T and evaluations-E for different conditioning levels.

For sub-processes the elements of R are R1, R2, R3 and R4, the elements of D are D1, D2, D3 and D4 and so on. All pertains to K at different conditioning levels.

K elements may be grouped in matrices of DOE, as generated by the wave equation, WE.

Examples of DOE are the matrices associated to Latin-squares or to Walsh-Hadamard designs (Iordache 2009). These should be more efficient than the unstructured designs of experiments as that utilized by Reich (Reich 1995).

Table 5.1 consists of several levels of conditions.

Table 5.1 Array of conditions for RDTE

D22		D23		D32		D33		T22		T23		T32		T33
	D2				**D3**				**T2**				**T3**	
D21		D24		D31		D34		T21		T24		T31		T34
			D								**T**			
D12		D13		D42		D43		T12		T13		T42		T43
	D1				**D4**				**T1**				**T4**	
D11		D14		D41		D44		T11		T14		T41		T44
R22		R23		R32		R33		E22		E23		E32		E33
	R2				**R3**				**E2**				**E3**	
R21		R24		R31		R34		E21		E24		E31		E34
			R								**E**			
R12		R13		R42		R43		E12		E13		E42		E43
	R1				**R4**				**E1**				**E4**	
R11		R14		R41		R44		E11		E14		E41		E44

Here the notations are: R-Research, D-Development, T-Testing, and E-Evaluation.

Then R, D, T and E describe the conditions at the conditioning level m=0.

Then R1, R2, R3, R4 are the conditions of R corresponding to the level m=1. Then R11, R12, R13, R14 are the sub-conditions of R1 and corresponds to the level m=2. They were represented as elements of a cyclic loop.

Table 5.1 contains the semantic network off all the possible conditions K, that is, the selected factors, to be grouped in DOEs.

The design includes the act of redesign which is defined as the act of successive improvements or changes made to a previously implemented design.

This suggests to identify elementary RDTE actions having components as R11-research, r, R12-design, d, R13-test, t and R14-evaluation, e. This reality level is indexed here by n=0 or by m=2 if considered as a conditioning level. It is possible that some components of the elementary actions are unrelated.

The coupling of several RDTE actions corresponds to the reality level n=1 or with different notations to the conditioning level m=1. The resulting actions are R1, R2 and so on.

For the action R2 we have specific components: R21-research, r, R22-design, d, R23-test, t and R24-evaluation, e. Coupling the information resulting from different actions corresponds to the reality level n=2 that is, to the conditioning level m=0. At this level the R, D, T and E are the four summarizing actions. The reality level n=2 is unconditional in this case study.

One outcome of the complexity is that currently the designer may not have time to adequately explore all the design alternatives and select the best alternative.

Consequently, PSM framework will include only some of the conditions K and the corresponding states S, also.

The detailed PSM framework is presented in the following.

5.1.4 Schema for Multiple Scales

Consider only a fragment of the Table 5.1. For three level evolution, m=0, m=1, m=2 the SKUP consists of the vectors S = (s^0, s^1, s^2); K = (k^0, k^1, k^2); U = (u^0, u^1, u^2); P = (p^0, p^1, p^2).

Table 5.2 illustrates the PSM framework, with conditions and states for two-levels only, m=0 and m=1. The conditions at the level m=0 are R, D, T and E. Let R=k^0_0, D=k^0_1, T=k^0_2 and E=k^0_3. The upper index refers to level while the lower index refers to the time step. It should be emphasized that the time steps at different levels may be different and this is a key feature for different levels of evolvability. The states and conditions at the level m=0 are represented by a loop with high thickness border cells.

The system initial state is s^0_0. With possibility $p^0(k^0_0|s^0_0)$ the condition k^0_0 is selected. This condition is a digit symbolizing a specific research R. This may be a matrix corresponding to R-DOE. Based on this, the operator s^0_1=$u^0(k^0_0$, $s^0_0)$ allows the transition to the new state s^0_1. This state is the realization of the research. Then with possibility $p^0(k^0_1|\, s^0_1)$ the new condition, k^0_1 arises. This condition symbolized by a digit corresponds to the selection of developments D. In the new condition, the operator $u^0(k^0_1,\ s^0_1) = s^0_2$ allows the system to reach the state s^0_2. This corresponds to the completion of design and materials.

Observe that: s^0_1=$u^0(k^0_0$, $s^0_0)$ implies s^0_2=$u^0(k^0_1$, $u^0(k^0_0$, $s^0_0))$.

With possibility $p^0(k^0_2|s^0_2)$ the testing, k^0_2 that is T, is selected and finally the new state results s^0_3=$u^0(k^0_2$, $s^0_2)$ results. This may corresponds to the processed product. It represents the succession of realized design, materials and processes.

Observe that: $s^0_3 = u^0(k^0_2,\ u^0(k^0_1,\ u^0(k^0_0,\ s^0_0)))$.

This result will be modified at the level m=0 in the condition E denoted by k^0_3. After this the state is s^0_4=$u^0(k^0_3$, $s^0_3)$=$u^0(k^0_3$, $u^0(k^0_2$, $u^0(k^0_1$, $u^0(k^0_0$, $s^0_0))))$.

The states are resulting not necessarily in a recursive way since, in practical cases the operators may varies with the step.

The states at the level m=0 are: s^0_0, s^0_1, s^0_2, s^0_3, s^0_4. The interpretation of the high thickness border cells trajectory is the process description as follows: from the state s^0_0 through condition k^0_0 towards the state s^0_1, then through condition k^0_1 towards the state s^0_2, and so on. Any trajectory is a succession of states and conditions. The role of initial state specification s^0_0 is outlined by such presentations.

If the experiment analysis shows that the factor E is the more significant factor, the analysis may be continued at the level m=1 for different test conditions E1=k_0^1, E2=k_1^1, E3=k_2^1, E4=k_3^1. This means to perform four different evaluations.

The states and the conditions at the level m=1 are in medium thickness border cells. The system initial state at the level m=1 is s_0^1. With possibility $p^1(k_0^1|\, s_0^1)$ the condition k_0^1 arises. The condition is a digit symbolizing a specific test. Based on this, the operator $u^1(k_0^1,\, s_0^1)=s_1^1$ describes the transition to the new state s_1^1. Then with possibility $p^1(k_1^1|\, s_1^1)$ the new condition, k_1^1 arises. In the new condition, the operator $u^1\ (k_1^1,\, s_1^1) = s_2^1$ allows the system to reach the state s_2^1.

Observe that: $s_2^1=u^1(k_1^1,\, u^1(k_0^1,\, s_0^1))$ and $s_3^1=u^1(k_2^1,\, u^1(k_1^1,\, u^1(k_0^1,\, s_0^1)))$.

The states at the level m=1 are: s_0^1, s_1^1, s_2^1, s_3^1. The conditioning at the level m=1 is represented by the loop E1E2E3E4 tat is: k_0^1, k_1^1, k_2^1, k_3^1.

The interpretation of the medium thickness border cell trajectory is as follows: from the initial state s_0^1 through condition k_0^1 to the state s_1^1, then through condition k_1^1 to the state s_2^1, and so on.

Due to representation restrictions Table 5.2 illustrates only two successive levels, in this case m=0 and m=1. Suppose that a more detailed study is necessary. The same framework may be used to outlines levels m=1 and m=2.

Observe that generally K=((k_0^0, k_1^0,...), ((k_0^1, k_1^1,...)), S=((s_0^0, s_1^0,...), ($s_0^1 s_1^1$,...));

U = (u^0, u^1, u^2,...,), P = (p^0, p^1, p^2,...,)

The design activity is confronted with knowledge data-base acquisition and development. Modern products and technologies should surpass the complexity emerging by non-linear interactions of large number of rules, restrictions or objectives. In the same time it is a need to construct larger-scale knowledge bases to put together data bases and models from different domains. This represents just a part of the designer activity that includes also, searching data base, analysis, product improving and so forth. An evolvable CAD, ECAD is the targeted strategy to confront complexity and the objective of this research. The entire design system should be evolvable. This involves problem solving, support of designer and interface. Local databases store all knowledge about the behavior of the agent and the community to whom the agents belongs. The information stored in these databases involves constraints, objectives, procedures, rules and experience, and organizational structures and techniques. It may be organized by logical rules. The CYC project, the KIF-knowledge integrated format and KIEF-knowledge intensive engineering framework represent major attempts in this direction (Takeda et al. 1995). These projects collect knowledge and provide mechanisms for sharing and reusing knowledge.

Table 5.2 Two levels schema for engineering design

				s_2^0				
		D				**T**		
s_1^0						s_2^1		s_3^0
					E2		E3	
		R		s_1^1		**E**		s_3^1
					E1		E4	
				s_0^0		s_0^1		

5.1.5 Perspectives

5.1.5.1 Three Levels Evolutionary Designs

Most of the current computer aided design, CAD systems employ the hierarchical decomposition strategy that is a form of analysis thought process, corresponding to categorical product interpretation of tensor product for the K model. Such a strategy can lead to refinements of existing design but do not leads always to new, evolutionary designs. Moreover it happens that the design becomes too large.

The switch from categorical product to coproduct controls the size of the search space and allows the emergence of new designs. Similar ideas have been emerged in the study of divergence/convergence strategies in engineering design. The divergence step, correlated to categorical product, implies understanding the problem and creating solutions. The convergence step, correlated to coproduct, selects among solutions for further development.

The process is illustrated in Fig. 5.1.

A general categorical presentation of the architecture was presented in Fig. 4.9.

In this case the notations are: K1-cognitive design, conventional (convergence), K2-evolutionary design (divergence), and S-design entities.

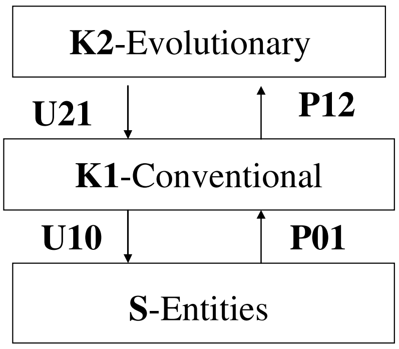

Fig. 5.1 Three levels hierarchical framework for engineering design

5.1.5.2 Four Realms: Meta-meta-models for Engineering Design

Difficulties arising in designing complex industrial products and operations require evolvable engineering design schemes. Evolvability in turn, is based on closure, at several complexity levels, for instance the closure between the dynamical laws of the material aspects and the complementary symbolic aspects of a physical organization. Full evolvability and autonomy requires the integrative closure that is a link between top and bottom levels.

Studies of evolvable and autonomous systems in complexity conditions challenged the traditional engineering fixed design methodology. Traditional engineering is based on a clear distinction between the design and the production phase and requires a system's performance and construction to be specified. For complexity conditions this is neither possible nor recommendable. Implementation of cyber-physical systems engineering concepts (Lee 2007, Carreras et al. 2009) outlined the need for a design for complexity that is a design for systems that fundamentally and continually adapt and evolve in a changing environment.

One problem that may benefits from this new type of model is the easy formulation of schedules in cases when the number of manufacturing cells and interconnected operations increases. The wave model offered the minimal number of cells and offers feasible schedules (Black 1991).

The challenge is to effectively build a fully evolvable design scheme based on integrative closure. This implies complementary and continuous back and forth between the wave equation, WE, results, that is, a specified design scheme, and the physical data of design process.

For this, the evolvable computer aided design, ECAD, should be able to face hard restrictions with respect to measurement analysis. The uses of ECAD lie primarily in exploring beyond the scope of conventional designs tools not competing with them. The main concerns are related to robustness, results analysis and scalability.

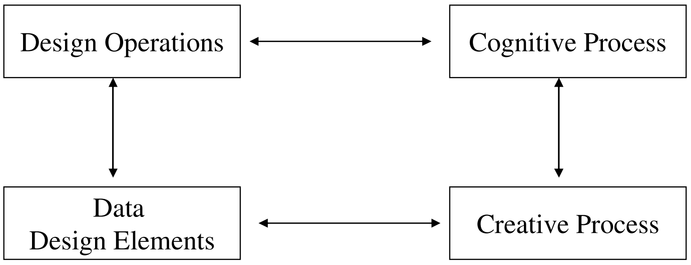

Fig. 5.2 Creative conceptual designs

The elements of the four realms schema are easy identified in the study of creative stimulation in conceptual design (Benami and Jin 2002). In this case the Data are operated at Design Operation level. To ensure evolvability, this level should be interconnected to Cognitive and then Creative levels (Fig. 5.2).

Another four level approach to ED may be detected in the use of meta-meta-model and meta-model concepts for integrated modeling in design (Tomiyama et al. 1989, Kiriyama et al. 1992).

In this case the levels to be considered are: first level quantitative features, second level qualitative relationships, third level translation and interaction of concepts and fourth level the meta-meta-model (Fig. 5.3).

The meta-meta-model represents relation between the aspect models.

Engineering of complex systems focuses on meta-designing of genotype associated to concepts instead of directly designing the phenotype associated to quantitative aspects.

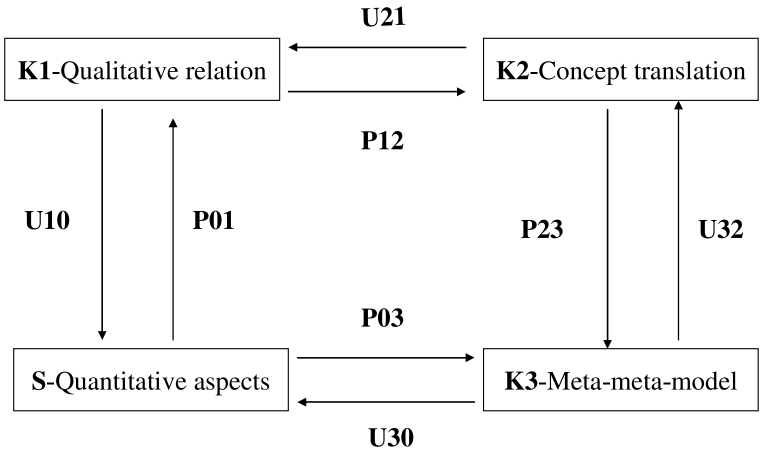

Fig. 5.3 Meta-meta-model for evolvable engineering design

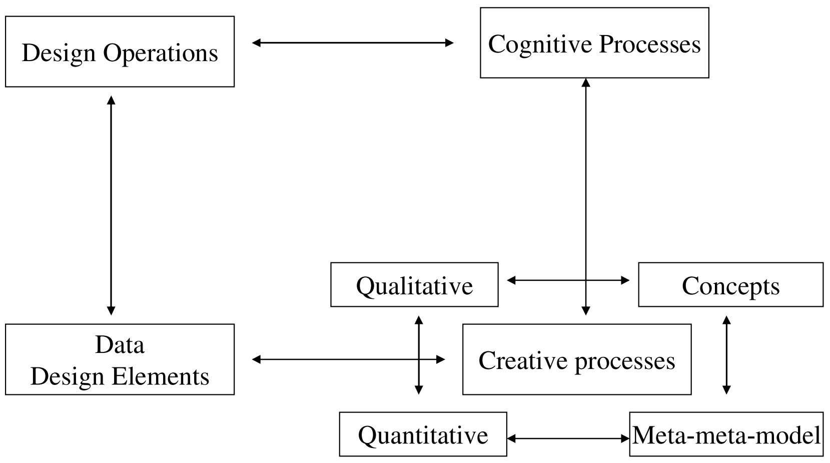

Fig. 5.4 Four realms and sub-realms for creativity

The integrative closure including a link between K3-Meta-meta-model and S-Quantitative aspects is critical for evolvability and autonomy of the design system.

The entire structure shown in Fig. 5.2 may be just one realization of the creative process of four realm diagram shown in Fig. 5.3. This aspect is clarified by the Fig. 5.4 representing a kind of superposition of Fig. 5.2 and Fig. 5.3.

The achievement of a particular creative design parallels and recapitulates the history of general design systems from data to creative stage. It is an ontogenetic versus phylogenetic relationship.

The relation between ontogenetic design and phylogenetic design has been discussed also by Braha and Maimon (1998). Purposeful adaptation of artificial thing, that is, the ontogenetic design was seen as an interface between the inner environment of artifacts and the outer environment, the surroundings in which it operates.

Ontogenetic design evolution refers to the design process that share the characteristic of observed evolutionary phenomenon which occurs between the time when a problem is assigned to the designer and the time the design is passed to the manufacturer.

During this period the design evolves and changes from the initial form to an acceptable form, towards a fit between the design and the requirements.

5.1.5.3 n-Graphs for Multiple Scale Engineering Design

Fig. 5.5 shows a representation of multiple scales frames presented in Table 5.1 using n-graphs. The n-graphs are computing tools able to describe asynchronous systems with multiple entrances and exits (Appendix A5). Asynchrony allows faster processing.

Different scales are associated to different levels in n-graph.

The reality level n=0 corresponds to the 0-graphs or sets. They represent r, d, t or e individual, undefined and uncorrelated actions, elementary process or objects. The level n=1 correspond the 1-graphs. These are directed graphs including the morphisms that is, the relations between r, d, t and e.

The morphisms are 1-cells. Their coupling allows the initiation of engineering design.

A 1^{st} order evolutionary step is represented by the transition to the level n=1.

This level is associated to 1-categories.

The level n=2 corresponds to the 2-graphs. These are graphs plus the 2-cells between paths of same source and target. These 2-cells express relation between relations, in particular the natural association of the quadruple r, d, t, e elements in just one macro actions denoted here by D-design, R-research, E-evaluation, or T-test.

There exist two different compositions of the 2-cells. The vertical corresponds to sequential 2-cells, while the horizontal corresponds to parallel 2-cells.

A 2^{nd} order evolutionary step is represented by the transition to the level n=2.

This level is associated to 2-categories.

The level n=3 corresponds to the 3-graphs. These are 2-graphs that include 3-cells that is, the relations between 2-cells. Fig. 5.5 shows the cells association as an evaluation action **E̲**. The 3-graphs represent graphs modification and should be subjected to conditions of natural transformations too.

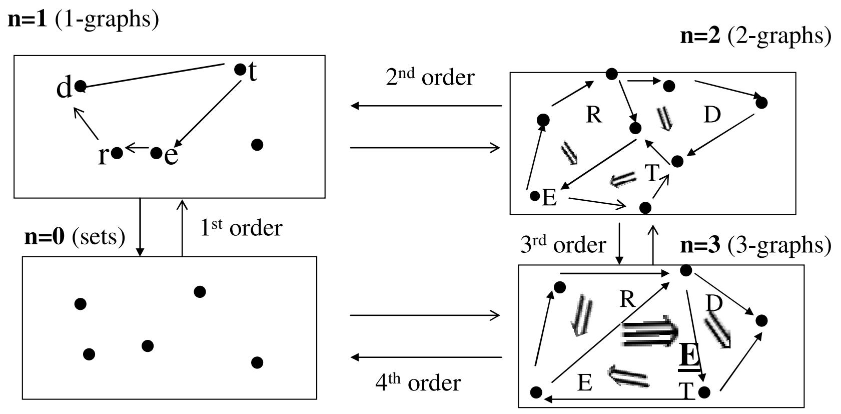

Fig. 5.5 n-graphs for multiple scale engineering design

A 3^{rd} order evolutionary step is represented by the transition to the level n=3.

This level is associated to 3-categories. The 4^{th} order step represents a challenge for engineering design.

5.1.5.4 Nested Frameworks for RDTE

Fig. 5.6 shows a categorical framework for RDTE as presented in Table 5.1

The four realms are identified as K0-Research, K1-Design, K2-Tests and K3-Evaluations. Fig. 5.6 outlines the possibility of integrative closure since it includes the link between K0 and K3. This allows evolvability and autonomy.

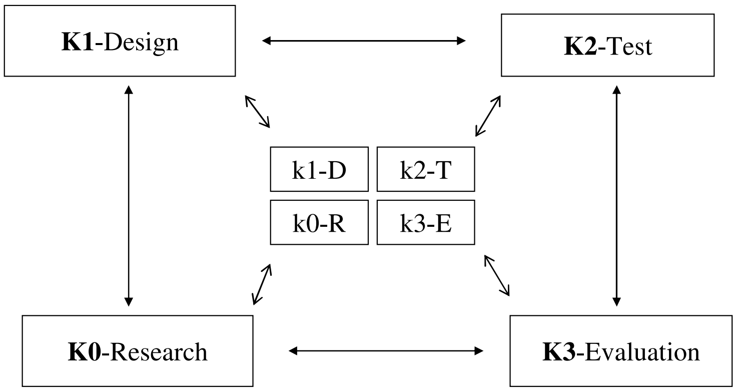

Fig. 5.6 Nested frameworks for RDTE

Theoretically the architecture is not confined to four realms.

Fig. 5.6 shows nested and self-similar architectures.

A similar four realm cyclic structure is repeated starting from the whole central system that may be built by four sub-realms denoted here by k0, k1, k2 and k3.

Fig. 5.6 emphasizes the integrative closure as a process that can develop self-similar patterns in design.

5.2 Case Based Reasoning

5.2.1 Case Based Reasoning Method

Conventional CBR is a problem solving paradigm that solves a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation (Aamodt and Plaza 1994, Aha et al. 2001). More specifically, CBR uses a database of problems to resolve new problems. The database can be built through the knowledge engineering (KE) process or it can be collected from previous cases.

CBR traces its roots to the studies of learning and memory. Schank (1982) developed a theory of learning and reminding based on retaining of experience in a dynamic evolving memory structure. This model of dynamic memory was the basis for some of the earliest CBR systems.

In a problem-solving system, each case would describe a problem and a solution to that problem. The reasoning engine solves new problems by adapting relevant cases from the library. Moreover, CBR can learn from previous experiences. When a problem is solved, the case-based reasoning engine can add the problem description and the solution to the case library. The new case that in general represented as a pair <problem, solution> is immediately available and can be considered as a new piece of knowledge.

The CBR process can be represented by a schematic cycle, as shown in Fig. 5.7 Aamodt and Plaza (1994) have described CBR typically as cyclical process comprising the four steps:

1. Recall the most similar cases
   During this process, the CBR engine searches the database to find the most approximate case to the current situation.
2. Reuse the cases to attempt to solve the problem
   This process includes using the retrieved case and adapting it to the new situation. At the end of this process, the user might propose a solution.
3. Revise the proposed solution if necessary
   Since the proposed solution could be inadequate, this process can correct the first proposed solution.
4. Retain the new solution as a part of a new case

This process enables CBR to learn and create a new solution and a new case that should be added to the case base.

It should be noted that the Recall process in CBR is different from the process in a database. If we want to query data, the database only retrieves some data using an exact matching while a CBR can retrieve data using an approximate matching.

As shown in Fig. 5.7, the CBR cycle starts with the description of a new problem, which can be solved by recalling previous cases and reusing solved cases, if possible, revising the solution and giving a suggested solution, retaining the restored case and incorporating it into the case base.

However, this cycle rarely occurs without human intervention that is usually involved in the retain step. Many application systems and tools act as a case retrieval system, such as help desk systems and customer support systems.

The CBR provides support for applications if the input data tend to repeat similar patterns from time to time. When the factors recur, the studied system

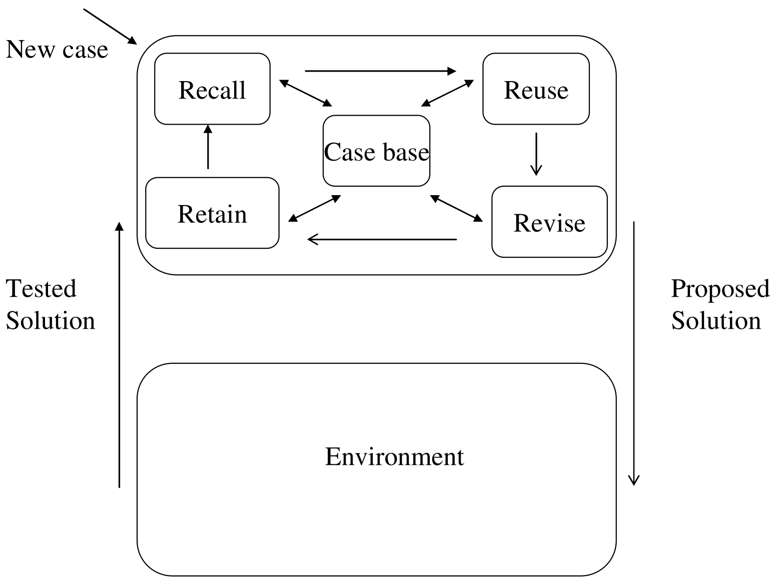

Fig. 5.7 CBR basic framework

is likely to display regularly repetitive patterns. This repetitiveness explains why it is reasonably to apply CBR in complex problem solving situations.

Traditional CBR have limited potential. For example in common versions, CBR involves just one user and don't answer in real-time to explosive amount of user data, to the unexpected cases, or to non-linear interacting cases and questions.

It is a need to implement CBR frameworks in which answers to multiple questions are gathered from multiple information sources, in real time.

For continuously addressing multiple-goals, multiple arrays of CBR cells systems are needed. For such arrays it is difficult to arrange the architecture or scheme of problem-solving, to schedule, to elaborate and to run rules, to adjust them to continuous changing environment.

Problem solving methodologies as case-based reasoning CBR are confronted with high complexity situation due to chaotic or random character of data, and to severe time restrictions. The method to confront the high complexity is that of evolvability. This implies improving the conventional passive CBR, to an evolvable one, ECBR. ECBR should be active and autonomous, able to take control of its environment, able of responding to random unexpected problems and to large amounts of knowledge in real-time.

ECBR schemes may be generated by the developed here partial differential model, WE, and compared to existing schemes. Schemes with variable number of cells and multi-scale schemes are resulting. Connections between the problem solving schemes, and the designs of experiments are of interest.

Applicability in domains including: process diagnosis and failure analysis, financial analysis, data mining, sensor operation and interpretation is expected.

5.2.2 Case Based Reasoning Frameworks

Concepts from theoretical biology, developed to characterize evolvability in artificial and natural systems represent the inspiration source for ECBR building.

The wave equation, WE, generates schemes with variable number of stages and multi-scale schemes. Connections between these problem solving schemes, and the well-known designs of experiments will be outlined. Cyclic problem solving arrays with evolvability based on multi-scale schemes organized by self-similar replication at different conditioning levels will be presented more in detail.

As biology suggests, the evolvable knowledge schemes should be embodied or situated.

The PSM framework outlines the active interaction between conditions and real states.

The classic scheme of problem solving in CBR is a cycle with four steps (Aamodt and Plaza 1994, Melendez et al. 2001). The CBR cycle steps are: Recall, Reuse, Revise, and Retain. The steps will be denoted by "c", "u", "v", and "t".

The four steps represent a CBR cell.

A platform or a scheme in which in any time step any of the four operations is activated is of interest. The advantage is continuous data input and output for the scheme. The scheme should contain four CBR cells indexed here by #0, #1, #2 and #3.

The functioning of the cells at successive time steps is represented in Table 5.3.

This table is in fact a solution of the wave equation were the following identifications are of use: 0-c, 1-u, 2-v, and 3-t. Here c, u, v and t represent the cell condition.

T is the time step, 0, 1, 2 or 3 and Z denotes the space of operations, 0, 1, 2 or 3.

Z describes also the travel along the classification scheme.

Observe that the positions of the c-recall, u-reuse, v-revise, t-retain are changed in the direction of circular information flow at a regularly point in time. At any given time for any cell, only one of the inputs corresponding to c-recall, u-reuse, v-revise, or t-retain is in the active mode. The 4 cells allow continuous input and output.

5.2.3 Schema Modification

Let us restrict as a first example, to the C(4) solution shown in Table 5.3.

Observe that the above examined cycling operations schedules are in fact designs of experiments, DOE. Table 5.3 contains a Latin square. Running DOE based scheme allows fast identification of significant data for classification regime acceleration.

The DOE factors are time steps, cell and operations. The time is multiple of the same time-step.

Standard DOE table may be developed by indicating the conditions associated to any element of the 4x4 Latin square (Table 5.4).

Experimental results of DOE application may be the interesting object selection, the efficiency, the resolution, and so forth. Typical control tasks in classifications are to obtain the highest throughput of the cell, highest efficiency or to reduce time consumption.

The DOE selects the significant results and also the significant factors by standard ANOVA calculations. This is in fact Fourier analysis over the real field, for the functioning parameters. The evaluation of performances may be based on real data or on real-field dynamical model of the process.

Next step will be to reorganize the scheme or to reproduce the experiment in the direction of beneficial results. The new experiment means a new DOE based on modulo-m algebra calculation and the WE model. Physically this means to generate a new, modified classification scheme. This scheme may be one with a different number of cells or a device with the same number of cells but with modified parameters.

Suppose for instance that the experiment underlined in Table 5.4 gives the worst result (cell #2, at the third step 3, lumped operation "u-reuse"). Suppose that u-reuse is the operation offering the expected product or result and that cell factor is the only significant factor. In that case the cell may be changed with a modified one, possibly from the same array of cells. The classification scheme is supposed to be redundant. If all other factors are significant, an improvement strategy may consists in the modification of the cell (#2 by #2'), of the time step, 3 by 3', of the operation u-reuse by u'-reuse followed by the introduction of a new operation for instance s-restore in a new cell #4 (Table 5.5). This kind of situation appears in case-base maintenance situations. For maintenance, restore step which selects and applies maintenance operations is necessary.

Schemes as presented Table 5.3 or Table 5.5 are solutions of the wave equation, WE. The classification scheme evolution appears in fact as a continuous

Table 5.3 Scheduling for CBR

Z\T	0	1	2	3
#0	c	u	v	t
#1	u	v	t	c
#2	v	t	c	u
#3	t	c	u	v

Table 5.4 DOE associated to CBR

Exp	Cell	Time Step	Operation
1	#0	0	c
2	#0	1	u
3	#0	2	v
4	#0	3	t
5	#1	0	u
6	#1	1	v
7	#1	2	t
8	#1	3	c
9	#2	0	v
10	#2	1	t
11	#2	2	c
12	$\underline{\#2}$	$\underline{3}$	$\underline{u}$
13	#3	0	t
14	#3	1	c
15	#3	2	u
16	#3	3	v

Table 5.5 Modified CBR

Z\T	0	1	2	$\underline{3'}$	4
#0	c	$\underline{u'}$	v	t	s
#1	$\underline{u'}$	v	t	s	c
$\underline{\#2'}$	v	t	s	c	$\underline{u'}$
#3	t	s	c	$\underline{u'}$	v
#4	s	c	$\underline{u'}$	v	t

oscillation between the WE generated DOE schemes and the real field evaluations of the resulting data. The complementarity or disjoint-ness between the finite-field scheme and, the real field data represents the key mechanism for evolvable classification.

Data mining schemes in which the duration of some steps is higher than that of other steps are frequently encountered.

5.2.4 Multiple Scales

A multi-scale scheme is considered in what follows. We may limit the Table 5.3 to vectors containing distinct elements only.

They represent solutions at specified time or stage in the problem solving development.

Table 5.6 Singlets y_0

Z\T	0	1
0	c	u
1	t	v

They results as solution of the WE too.

Denote by Y(T) = y_0 = (c, u, v, t).

Since there are only four elements it is possible to represent $y_0$as a 2x2 matrix as that shown in Table 5.6. It results as a kind of cyclic folding of the vector y_0. Obviously other type of folding may be of interest.

Let us consider more scales in the scheme of conditions. The method used in Sect. 2.2.3 is used in the following. The doublets are resulting by direct product and concatenation $y_0 \times y_1$. Table 5.7 represents a direct product of 2x2-matrices, Y (T) = $y_0 \times y_1$ with $y_0 = y_1$ The new letters have been put adjacent to the first, to the right side. The initial problem was solved by splitting in 4 steps (c, u, v, t). However any of these steps is a new problem that may be solved by the same algorithm.

In this way the sub-problems cc, cu, cv, ct and so on are resulting.

They have significance as described by their notation and as suggested by the task-method decomposition a possible interpretation is as follows (Aamodt and Plaza 1994): cc-identify features, cu-initially match, cv-search, ct-select, tc-extract, tu-index, tv-adjust indexes, tt-update knowledge and so on.

The transition from a level to another may be triggered by the presence of data of interest in the expected product, by specific shapes of the recorded signals, and so forth The higher-level problem solving operations should take place with a timing that ensures and support the cyclic functioning at the previous level. The problem solving scheme should have an adjustable or unsettled construction since it contains interacting modules subjected to continuous reorganization after confronting the reality. It is not only a spatial scheme but a temporal one as well.

Table 5.7 Doublets $y_0 \times y_1$

Z\T	00		01		10		11
00	cc		cu		uc		uu
		c				u	
01	ct		cv		ut		uv
10	tc		tu		vc		vu
		t				v	
11	tt		tv		vt		vv

Due to the size of search space the multi-scale scheme is confronted to the apparent improbability of chance to produce any successful solution of the problem solving. But in fact the problem solving trajectory in the multi-scale scheme is not a blind search. The multi-scale scheme allows modifying the searchable domain and the search velocity by adding more levels and scales to the search process. Any new problem solving level appears as adding more sensors and effectors to the system. Moreover interaction with real data accelerates scheme construction and discovering new solutions.

5.2.5 Schema for Multiple Scales

The elements of the SKUP for multi-scale scheme will be presented in what follows. This is of help for classification schemes design and processes visualization.

A section of the general Table 5.7, illustrating PSM structure at two levels only, m=0 and m=1 will be considered. The SKUP elements are:

$\mathrm{S} = (\mathrm{s}^0, \mathrm{s}^1)$; $\mathrm{K} = (\mathrm{k}^0, \mathrm{k}^1)$; $\mathrm{U} = (\mathrm{u}^0, \mathrm{u}^1)$; $\mathrm{P} = (\mathrm{p}^0, \mathrm{p}^1)$

The scheme includes at the first level m=0 the operations c, u, v, t and at the second level, m=1, the operations vc, vu, vv and vt. In this particular case, the second level is resulting by a separate re-cycle processing after operation v-revise.

The states and conditions at the level m=0 are in the high thickness border cells.

The states and the conditions at the level m=1 are in medium thickness border cells.

Table 5.8 includes the conditions K and the real valued states S. The conditions at the level m=0 are t=k_0^0, c=k_1^0, u=k_2^0 and v=k_3^0. The upper index refers to level while the lower index refers to the time step. Time steps at different levels are different. The system initial state is s_0^0. With possibility $\mathrm{p}^0(k_0^0|s_0^0)$ the condition k_0^0 is selected. This is a digit symbolizing a specific operation t-retain. Based on this, the operator $\mathrm{u}^0(k_0^0, s_0^0) = s_1^0$ allows the transition to the new state s_1^0. Then with possibility $\mathrm{p}^0(k_1^0| s_1^0)$ the new condition, k_1^0 arises. This condition symbolized by a digit corresponds to the selection of c-recall. In the new condition the operator $\mathrm{u}^0(k_1^0, s_1^0) = s_2^0$ allows the system reach the state s_2^0. With possibility $\mathrm{p}^0(k_2^0|s_2^0)$ the operation, k_2^0 that is u-reuse, is selected and finally the product $\mathrm{u}^0(k_2^0, s_2^0) = s_3^0$ results. It will be operated at the level m=0 in the condition v-revise denoted by k_3^0. Then the state is s_4^0. The states at the level m=0 are represented by the square: s_0^0, s_1^0, s_2^0, s_3^0. The conditions at the level m=0 are represented by the square t-retain, c-recall, u-reuse, v-revise, tat is: k_0^0, k_1^0, k_2^0, k_3^0.

If experiments shows that v-revise is the critical operation, the classification may be limited at the level m=1 for the operations vt=k_0^1, vc=k_1^1, vu=k_2^1 and vv=k_3^1.

Table 5.8 Two-level schema for CBR

				s_2^0				
		c				**u**		
s_1^0						s_2^1		s_3^0
					vc		vu	
		t		s_1^1		**v**		s_3^1
					vt		vv	
				s_0^0		s_0^1		

The system initial state at the level m=1 is s_0^1. With possibility $p^1(k_0^1|s_0^1)$ the condition k_0^1 arises. This is a digit symbolizing a specific operation. Based on this the operator $u^1(k_0^1, s_0^1) = s_1^1$ describes the transition to the new state s_1^1 and so on. Each condition supposes the selection of other condition for operations.

The states at the level m=1 are represented by the square: s_0^1, s_1^1, s_2^1, s_3^1. The conditions at the level m=0 are represented by the square vt, vc, vu, vv, tat is: k_0^1, k_1^1, k_2^1, k_3^1.

The potentialities are defined by vectors as $P = (p^0, p^1)$. The component $p(k^m)$ is an evaluation of the condition k^m. An example of evaluation is to take $p(k^m)$ equal to 0 or 1. The value zero corresponds to situation in which that condition is ineffective while the value 1, corresponds to active conditions.

Transfer between different levels may be controlled by external criteria.

K elements, representing the symbolic conditions indicating the types of operations at two levels are in fact cyclic classification schemes. S appears as sequences of more or less classified problems. Operators U characterize the capability to pass from intended conditions of classification to the reality of classification steps. The possibility P describes the capability of states S to reactivate the classification scheme and to modify the symbolic K description that is the classification scheme elements. P shows the activation of some areas of the operations shown in Table 5.8 and the inactivation of others.

5.2.6 Perspectives

5.2.6.1 Three Levels Evolvable CBR

Elements of adaptability and evolvability may be detected for some knowledge systems presented in literature. For instance, conversational CBR, received substantial attention (Aha et al. 2001). This essentially interactive CBR, involves the refinement of diagnoses through interaction with the user or other CBRs.

These systems attempt to find the quickest ways to increase the accuracy of diagnosis through estimating information gain.

Another research is that of active knowledge systems with conceptual graphs (Li and Yang 1999, Delugach 2003). In that case the concept may play the role of factors.

Difficulties arising in cyclic operations in complex situations require adaptable and evolvable classification schemes. These are in turn, based on closure concepts implies the disjoint or complementary description and closure between the dynamical laws of the material aspects tat is real data and the symbolic aspects tat is condition data of the physical organization. This is the concept of semantic closure, restricted to two levels architectures.

The challenge is to effectively build entirely evolvable classification schemes based on complementarity and continuous back and forth between the wave model results that is a specified classification scheme, and the physical or real data of classification process itself.

A number of useful areas of applications have been identified for evolvable classification technologies.

Main examples are process and quality control, diagnosis and failure analysis, engineering design, financial analysis, emergency situations, Data mining with augmented semantics, evolvable agent operations, temporal reasoning, sensor operation and interpretation.

Evolvable knowledge schemes offer the prospect of devices to suit a particular individual. Evolvable technologies have the potential to adapt autonomously to changes in their environment. This could be useful for situations where real-time control over systems is not possible such as for space applications. Evolvable control systems are required in such cases. Evolvable knowledge devices may be of help in the study of central concepts as self-repair and development.

An open question is if this kind of evolvable knowledge systems can be implemented in reality. A generic architecture and algorithms should be proposed so that the particular system builder do not starts from beginning and may become evolvable.

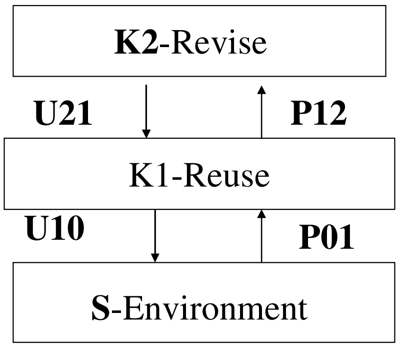

Fig. 5.8 Three levels framework for evolvable CBR

Fig. 5.8 and Fig. 5.9 show the categorical framework for a CBR system with three hierarchical levels or three realms.

S represents the real system.

The elements of the SKUP categorical framework are as follows:

S-environment representing the processes

K1 and K2-corresponds to the CBR cycle

U10: K1→S action

P01: S→K1 sense of data and monitor

K is structured to provide an approximation of what it is considered as a dynamic memory model that basically consists of retaining experiences as cases for further reuse.

Cases are registers containing a description of a problem and its solution.

The elements of the categorical framework may be: K1-Reuse, K2-Revise, U21-Reuse action P12-Retain procedure.

It is possible to run on different time scales for different SKUPs. Several K1, K2 cycles may be performed before the coupling in the larger SKUP loop. K1 and K2 cycles negotiate among themselves as to which should be active. This allows anticipative control of the process.

The system interacts with its environment, through its data base that acquires new cases in response to changes in the environment and through the actions that it performs.

The framework shown in Fig. 5.9 allows cognitive evolvability and autonomy.

5.2.6.2 Nested Frameworks for Evolvable CBR

Applications of CBR methodology for autonomous service failure diagnosis have been proposed (Montani and Anglano, 2006). This kind of CBR approach allowed self-healing in software systems.

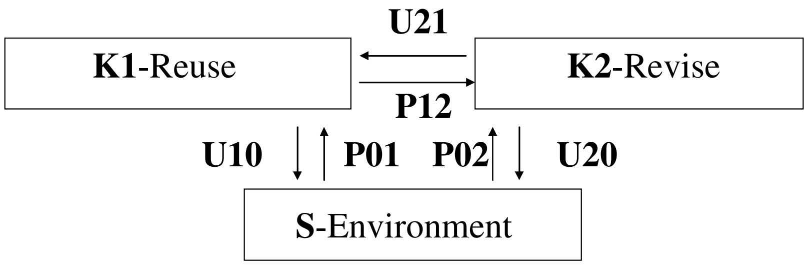

Fig. 5.9 Three realms framework for evolvable CBR

Fig. 5.10 shows a four realms categorical framework for CBR as presented in Table 5.8. The four realms are K0-Retain-T, K1-Recall-C, K2-Reuse-U and K3-Revise-V.

K0 reflects the environment response.

The architecture shown in Fig. 5.10 outlines the possibility of integrative closure including the link between K0 and K3 and allowing evolvability and autonomy.

This link may be established by implementing autonomic computing paradigm (Kephart and Chess 2003). This studies methods for increasing environment-awareness and automatic responsiveness. Autonomic computing methods promise to facilitate CBR tasks and facilitate information capture (Montani and Anglano 2006).

Theoretically the cyclic architecture is not confined to four realms.

Fig. 5.10 shows nested and self-similar architectures.

A similar structure is repeated starting from the whole central system that may be built by four sub-realms denoted here by k0, k1, k2 and k3.

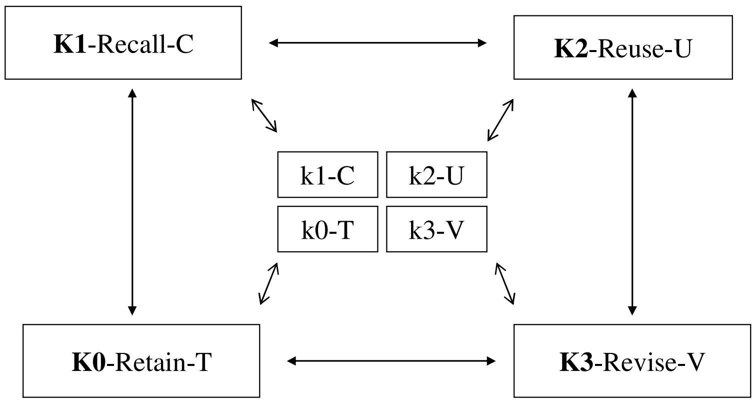

Fig. 5.10 Nested frameworks for evolvable CBR

5.3 Failure Analysis

5.3.1 Complexity Challenges

Failure analysis, FA, failure mode and effect analysis, FMEA, root cause analysis, RCA are useful quality and reliability tools in different industries.

FA and RCA are structured analytic methodology used primarily to examine the underlying contributors to an adverse event or condition. FMEA focuses on prevention and proactive risk management as RCA is concentrated on the occurrence of adverse events.

FMEA differs from FA in that it is a structured methodology used to evaluate a process prior to its implementation. Its purpose is to identify on an a priori basis the ways in which that process might potentially fail, with the goal in mind being to eliminate or reduce the likelihood and outcome severity of such a failure.

The complexity advent imposes significant modification of basic concepts and methods such as FA, FMEA, RCA, reliability and quality systems, problem solving methodologies, testing strategy, time concepts and frames.

Some of the difficulties of conventional FA methods are as follows:

- They focus on short-term customer satisfaction not on process improvement.
- FA is fixed not reviewed during the life of the product.
- In several companies, FA are developed too late and don't improve the processes.
- FA are not conceived as dynamic tool that will be developed.
- FA is not able to identify complex failure modes involving multiple failures or subsystems, or to discover expected failure intervals of particular failure modes.
- FA don't take into account the timing and scheduling in failure analysis and process improvement.

The constructivist strategy to confront complexity frontier is based on evolvability. Evolvable failure analysis, EFA are presented as the approach to meet the requirements imposed to the industry in high complexity domains.

An EFA system is an FA system that has the characteristics:

- Addresses multiple problems, tasks, failures that can be correlated to the real system
- Can change autonomously both the FA scheme as the real process dynamic behavior
- Is capable to control and to take advantage of the unexpected events of their environment in increasingly complex ways
- Have emergent, not entirely pre-programmed, behavior
- Shows multiple scale, parallel, evolution potentialities
- Can incorporate and accommodate new information

5.3.2 Basic Framework for Diagnosis

The two level basic categorical frameworks are able to gather some of the elements of adaptability for failure analysis (Fig. 5.11).

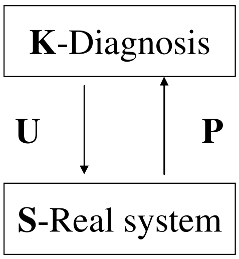

Fig. 5.11 Two levels diagnosis

The starting step is a failure analysis scheme in K. It is based on the expertise for several case studies. This summarizes the experiment and the possible factors.

To this summary we may associate a tree-like diagram being in fact similar to the standard fishbone root-causes. Obviously we may start from the existing root-cause diagrams. Then a comparison with reality S is proposed.

In this step couples of factors are selected and tested.

It is a process described by the operators U showing how the real state for a given scheme in K.

In the next step the significant factors may be grouped.

It is a process allowing establishing affinity diagrams based on similarities.

It results a structure similar to the root-cause diagram but having reversed direction arrows. It looks more like a decision diagram.

This step is associated to possibilities P.

Finally a structure in K is resulting.

Iordache (2009) presented in detail examples of failure analysis based on SKUP frame and DOE resulting as solution of the wave equation WE.

The basic SKUP shown in Fig. 5.11 may be perceived in other failure analysis method.

The SKUP steps corresponds to the Boyd's OODA loop (Dettmer 2003).

In this case the observe part is linked to S, the decide part is associated to K, the orient part is linked to P and the act part is liked to U.

The same elements may be identified in the Goldratt, constraint management model (Dettmer 2003).

In this case the mismatches analyze part is linked to S, the creation of transformation design and plan part is associated to K, the review the strategy and the definition the new paradigm is linked to possibilities P and the strategy deployment part is liked to the operators U.

5.3.3 Perspectives

5.3.3.1 Three Levels Frameworks

Recent work has pointed out that diagnosis strategies represent a necessary tool for complex systems diagnosis. Nejdl et al. (1995) introduced a formal meta-language to express strategic knowledge in an explicit way.

Fig.5.12 shows the categorical framework for a diagnosis system with three levels.

S represents the real system.

K1 and K2 are the two cognitive levels. K1 represents the diagnosis level.

K2 represents the meta-level of strategies. The strategies are defined at the meta-level.

U10:K1→S describes the actions towards the real level while P01: S→K1, summarizes the observations from S evaluations.

The information change between the basic level and the meta-level of diagnosis is characterized by the operator U21 and the possibilities P12.

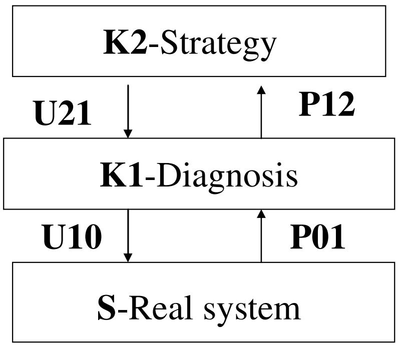

Fig. 5.12 Three levels hierarchical diagnosis

A similar failure-driven driven modeling approach that incorporates ideas from developmental learning is due to Sakhanenko et al. (2007). It is based on the architecture with several levels of control and of learning for adapting and evolving models to represent changing and evolving situations (Piaget 1970).

The loop S, K1, P01, U10 is linked to assimilation mechanism while the loop K1, K2, P12, U21 is linked to accommodation mechanism.

Assimilation supposes integrating new information into pre-existing structures. Accommodation supposes changing and building new structures to understand information.

5.3.3.2 Four Realms Frameworks

A developed four-level categorical approach for security of distribution information systems was presented by Sisiaridis et al. (2008). The four levels correspond to Data, Schema, Construct and Concept (Fig. 5.13). The improvement is representing by the integrative closure allowing the emergence and autonomous testing of new concepts.

Restricting the levels interactions to the operators U10, U21, U32 leave the choice of control to the users and are appropriate for low-level security risks. The bottom-up approach, emphasizing the possibilities P01, P12 and P23 allows risk analysis and are more suited to high level security risks.

The signification of the functors U and possibilities P is explicit. U10, U21, U32 and U30 corresponds to implementation operations.

Observe that: U10: K1-Schema→S-Data, U21:K2-Constructs→K1-Schema, U32: K3-Concepts→K2-Constructs, and U30: K3-Concepts→S-Data.

P01, P12, P23 and P03 are synthesis steps.

P01: S-Data→K1-Schema, P12: K1-Schema→K2-Constructs, P23: K2-Constructs→K3-Concepts, and P03: S-Data→K3-Concepts.

Fig. 5.13 emphasizes the role of integrative closure via U30 and P03. This interconnection may make the system quite evolvable and autonomous.

The link via U30 and P03 may be established by implementing pervasive computing (Estrin et al. 2002). In a case of failure analysis and self-healing, as sensors are installed on a device, the information can be automatically captured during preventive maintenance. It may be possible to broaden the range of environmental and device information captured and transmitted automatically. Pervasive computing methods facilitate information capture and failure analysis tasks.

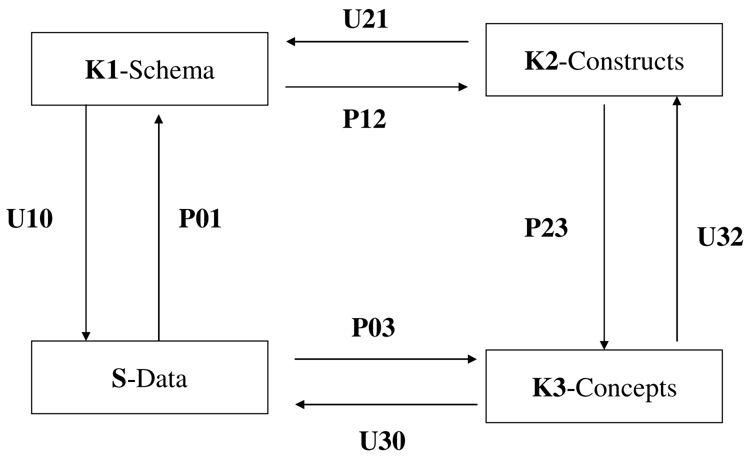

Fig. 5.13 Four realms network for security of distribution information systems

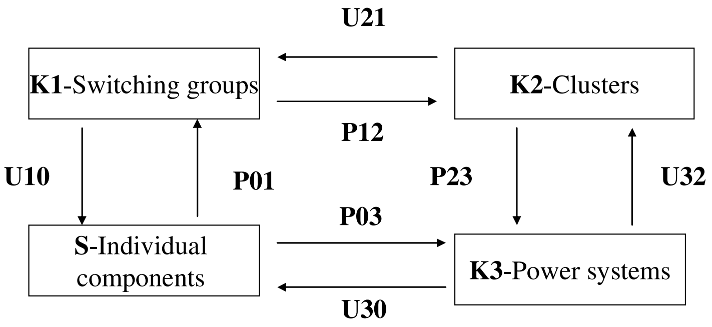

Fig. 5.14 Four realms network for failure diagnosis

Another example of evolved failure analysis making use of the four-level architectures is shown in Fig. 5.14 (Rayudu et al. 2000).

The first reality level represents behavior of individual components and their present status. The second level, characterizes the switching groups and this refers for instance to isolators, protective relays, circuits breakers, and so forth.

The representation of entities bounded by a set of switching groups called clusters make the third level. The cluster level incorporates behavior knowledge concerning connected switching groups and the operational equipment between them.

The fourth level represents the whole network in terms of clusters. This level encompasses the strategic problem solving knowledge related to the complete power network. It is an integrative closure for failure diagnosis, allowing system evolvability, self-repairing and autonomy. The operators U and P describe the testing procedures and the action in case of failure. The possibilities P describe the testing procedures and the information transfer between levels.

5.4 Multi Agent Manufacturing Systems

5.4.1 Multi Agent Systems

Agents are participants as individual elements within a complex system. Each agent may have its own set of internal states, skills, rules, and strategies that determine its behavior. Agents generally exist in a hierarchy. For example, an employee in a corporation interacts with other agents at a higher hierarchical level in the organizational environment. The agents receive information from within and outside their environment. Agents may develop their own schema through interaction and find regularities in the data and compress these perceived regularities into internal models that are used as the basis for action. An agent may be defined as a device or a self-directed program object which has its own value system and the means to solve certain tasks independently and then communicate its solution to a larger problem solving organization.

The main categories of agents are:

- Autonomous agents, capable of effective independent actions
- Objective directed agents, when autonomous actions are directed towards the achievement of defined tasks
- Intelligent agents, with ability to learn and adapt
- Cooperative agents, assisting other agents to perform a task

Examples of agents are neurons in brain, antibodies in case of immune systems, ants in colonies, wolfs in packs, investors in the case of stock market, people in social networks, and so forth. In each case agents have relatively limited set of rules, and the complexity of the collective behavior emerges from the large number of interactions among each other and their environment. There is constant action and reaction to what other agents are doing, thus nothing in the system is essentially fixed.

The distributed manufacturing environments and the flexibility and reaction to disturbances requirements are crucial reasons for moving to new organization paradigms.

Next generation of manufacturing control systems comprises the high adaptation and reaction to the occurrence of disturbances and to environment changes. On the other hand these control systems should optimize the global performance of the system which requires a global view of the entire system. These requirements imply the development of new manufacturing control systems with more autonomy, robustness against disturbances, able to handle to the changes and disturbances much better than the actual systems. New paradigms should focus on the ability to respond promptly and correctly to external changes, without external interventions. Distributed manufacturing architectures, multi-agent-based manufacturing systems represent a potential answer to complexity challenges (Parunak and Brueckner 2001).

Table 5.9 compares the multi agent systems MAS, with conventional hierarchical approaches. The autonomous multi agent systems may have some disadvantages. Theoretical optima cannot be guaranteed. Predictions can be made only at the aggregate level. Systems of autonomous agents can become computationally unstable. On the other hand, an autonomous approach appears to offer significantly advantages over conventional systems. Because each agent is close to the point of contact with the real world, the system computational state tracks the state of the world closely, without need for a centralized database.

Multi agent systems offer a way to relax the constraints of centralized planned, sequential control. They offer production systems that are decentralized rather than centralized, emergent rather than planned and concurrent rather than sequential.

Table 5.9 Multi-agent versus conventional systems

Characteristics	Conventional	Multi agent systems
Model source	Military	Biology, economy
Optimum	Yes	No
Prediction level	Individual	Aggregate
Computational stability	High	Low
Match to reality	Low	High
Requires central data	Yes	No
Response to change	Fragile	Relatively robust
System reconfiguration	Hard	Easy
Calculus	Complicated, long	Simple, short
Time required to schedule	Slow	Real time
Processing	Sequential	Concurrent, parallel

5.4.2 Frameworks for Manufacturing Systems

PSM and EDOE methodology offers suggestions for agent based architectures for manufacturing applications (Iordache 2009).

The knowledge processor is a knowledge base system that stores and processes the necessary knowledge for an agent to play the role the agent society has designed for it.

The typical conceptual model of an agent comprises four components surrounding the knowledge processor. The four elements are:

- Perception, a channel for an agent to receive information from the external world
- Actuator, an interface for an agent to modify or influence the states of an agent community
- Communication, a mechanism for an agent to exchange views with other members in the agent society
- Objectives, a list of roles for an agent to play

In terms of EDOE, the knowledge processor plays the role of the center, K.

These local databases store all knowledge about the behavior of the agent and the community were the agents belongs. The information stored in these databases involves constraints, objectives, procedures, rules and experience, and organizational structures and techniques. It may be organized by logical rules.

The four factors of the center K are: Communication C, Perception P, Objectives O and Actuator A.

Modules have to be associated to component DOE.

The communication module deals with the need to regulate the interaction between distributed agents and defines a communication language.

The communication module may have sub-modules such as: contents C1, message C2, physical information C3 and so on. These are analogous to the factors of component designs.

It results that the generic architecture for an agent may be represented as an EDOE.

The multi-agent system is an open and distributed system that is formed by a group of agents combined with each other through a network of cooperatively solving a common problem. Often, several agents do same simple thing. It is possible that other agents don't use all modules defined in the generic architecture.

The architecture is in fact a complex EDOE framework and can be operated according to the EDOE methodology.

It is easy to identify the SKUP elements.

The product of implementing multi-agent architecture is described by S.

The elements of K are denoted by C, P, O, A at the level m=0 and C1, C2, C3 and so on at the level m=1. Operators U describe the scheduling, while possibilities P take into account the execution step. The overall system performance is not globally planned but emerges through the dynamic interaction of the agents in real time.

Operators U and possibilities P express that the system does not alternate between cycles of central scheduling and final execution. Rather than this mechanism, the schedule emerges from the concurrent independent decisions of the local agents.

Elements of the SKUP may be detected in other multi-agent systems.

For instance, in the study of coordination, the agents with their specific roles represents the elements of K, the so-called rational actions are associated to the elements of U, the perceptions are elements of P while the states S are represented in this case by pheromones (Parunak and Brueckner 2001).

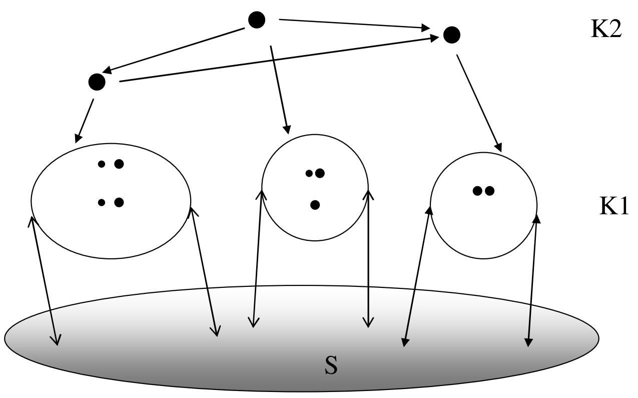

Fig. 5.15 Three levels hierarchical framework for multi-agent-system

Fig. 5.15 shows a three levels hierarchical framework for multi-agent-system.

A general presentation of this architecture was shown in Fig. 4.9.

In this case the notations are: S-environment, K1-agents and K2-meta agents.

5.4.2.1 Holonic Manufacturing Systems

The holonic manufacturing system is the paradigm that translates the holon concept developed for living systems and social organizations into a set of appropriate concepts for manufacturing industries (Tharumarajah et al. 1996, Valckenaers et al. 1997, Ulieru 2002).

The term holon describes the basic unit of organization in living organisms and social organizations. The holon can represent a physical or logical activity such as a machine on order or a human operator. The holon can be broken into several other holons, a procedure which allows the reduction of problem complexity. A manufacturing holon comprises a control part and an optional physical processing part. Multiple holons may dynamically aggregate into a single-higher level holon.

The holarchy is a system of holons that can co-operate to achieve an objective. The holonic manufacturing system is a holarchy that integrates the entire range of manufacturing activities.

The holonic manufacturing systems paradigm is part of the next generation of distributed control and introduces the hierarchical control within a heterarchical structure. This innovation makes available the combination of robustness against disturbances, presented in heterarchical control with the stability and global performances optimization presented in hierarchical control. The implementation of this concept requires that decision power must be distributed between the central and local holons that is there exists a switch between hierarchical and heterarchical control. In categorical terms this corresponds to a switch from product to coproduct constructions. A categorical presentation of the architecture was shown in Fig. 4.9. Usually the categorical product is associated to cognitive or interactive steps while the categorical coproduct is associated to reactive steps.

The function of central holon is to advise the local holons. When disturbances occur the autonomy of holons increase, while during normal functioning, the autonomy of local holons decreases and they follows the central holons input as for example the scheduling plans.

The holonic manufacturing system design starts with a forward control step in which the definition of all appropriate holons and their responsibility is established. In comparison with traditional methodologies a rather vague responsibility than a precise function for each holon is established. This facilitates the backward control. Complementary controls ensure systems evolvability.

Ulieru et al. (2002) studied the four layer holonic control architecture.

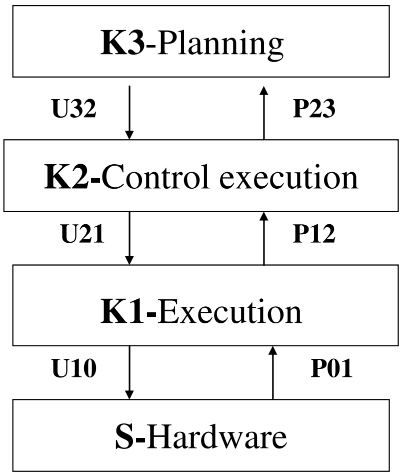

Fig. 5.16 Four levels hierarchical framework for holonic control

The level S corresponds to resources. It represents the physical platform.

The K1 layer is concerned with the execution of the application.

In this case the K2 layer achieves the arranging for the distribution of applications across multiple resources.

The K3 layer is concerned with planning issues such as reconfiguration or execution control.

This architecture reflects the multi-resolution structure of the holonic enterprise. As we move down the layers shown in Fig. 5.16, time scales become shorter.

Higher layers are more sophisticated but slower, while lower layers are fast and light-weight.

The multi layer holonic assembly system was studied also by Sugi et al. (2003). The holarchy of the system consists of execution holons layer identified as K1, assembly operation holons layer identified as K2 and top management operation holon identified here as K3 layer. If the management holon is ordered to assembly a specified product, this assembly task is decomposed into subtasks for lower management holons. An operation holon secures appropriate execution holons, which corresponds to real manufacturing devices, using the contract net protocol. Then the operation holon makes the execution holons execute a job such as assembling parts. The decentralized nature of the system enabled to realize plug and produce, a system function that supports easy addition and removal of manufacturing devices. Sugi et al. (2003) developed techniques for plug and produce such as distributed resource allocation method for installation of new robots and an automated calibration for mutual positional relationship between an existing robot and a newly added one.

Plug and produce function is related to interoperability at several levels and n-categorical modeling.

What is missing for hierarchical holonic architecture as that shown in Fig. 5.16 is the interaction of the top layer K3 and real execution layer S, that is, the integrative closure.

The four-level approach to holonic systems presented in Fig. 5.17 challenge this difficulty. It represents the mandatory development of the current hierarchical approach (Naumenko and Wegmann 2003, Baina and Morel 2006).

In this case the lowest level denoted here by K0 presents different subjects for modeling, each of them called a universe of discourse. The next level K1 contains viewpoints, for instance mechanisms and models.

The next level K2 focuses on meta-models and the highest level K3 focuses on meta-meta-models.

The meta-meta-model should be designed to allow unification under a common framework.

Each application can be considered as a specific use of a viewpoint defined in the K1 level which is based on meta-models described in K2.

The universe of discourses K0 concerns the manufacturing enterprise.

To describe this universe it is possible to use holonic views K1 that are instantiations of the meta-model K2 defining specific type of holons and their relationship within the context. K2 is an instantiation of K3. K3 level corresponds to the meta-meta-model.

Application interoperability can be resolved by achieving the K3→K0 connection and the integrative closure. This integrative closure makes the system evolvable.

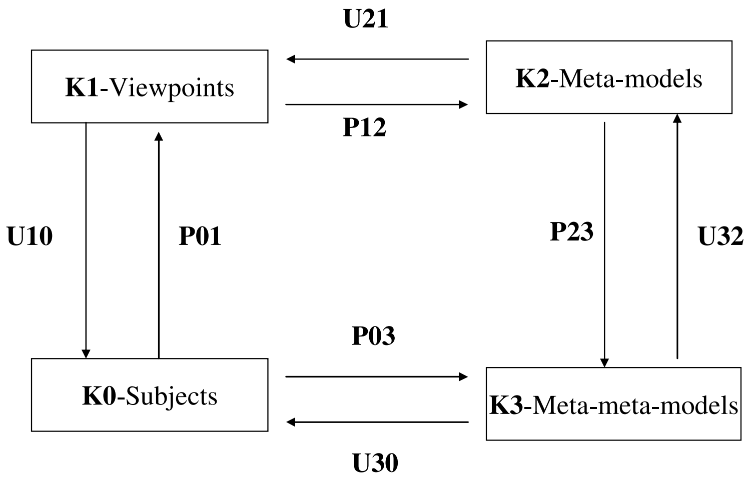

Fig. 5.17 Four realms multi-agent-system modeling

5.4.3 Evolvable Manufacturing Systems

The globalization of markets, shortening of product life cycles, decrease of dimensions for products and outsourcing were identified as major threats for industry. Answers to such threats were paradigms as evolvable assembly system, EAS (Onori 2002), evolvable production systems EPS (Onori et al. 2006, Frei et al. 2007), and evolvable manufacturing systems EMS (Iordache 2009).

The design process of assembly systems, EAS has been modeled by a hierarchy of four levels: S-Environment, K1-Domain knowledge, K2-Inference knowledge and K3-Task knowledge (Lohse et al., 2005). The domain knowledge level defines all the specific concepts needed by the inferences. The inference knowledge level defines what inferences are needed to fulfill the reasoning task. The task knowledge level defines the reasoning tasks required to achieve a specific goal. All three levels of knowledge, K1, K2 and K3 have been modeled in a Protégé interface to allow the dynamic definition and adaptation of the assembly system design process.

As discussed the four levels hierarchy doesn't allows complete evolvability and autonomy.

EPS represents a concrete solution to the requirements from the market such as stated within the agile, reconfigurable and distributed approaches. They include high responsiveness, low down-times, ability to handle small series with many variants, and on the fly changeability. Together with ontology-based process specific modules, a distributed control system using the multiple agent paradigm allows to quickly and cost effectively adapt to ever changing production requirements.

EPS have similarities with the bionic, fractal, holonic, biological and reconfigurable manufacturing systems, but there exists major differences too.

Besides considering system morphology, EPS strongly links product, processes, system and environment by the means of detailed ontologies.

EPS focuses on self-organization and implies the ability of complex systems to co-evolve with continuously changing requirements. EPS are expected to allow the user to build any required system and to modify this at wish.

Some features of the production systems necessary to achieve evolvability are:

- Modularity since many small, dedicated units that can easily be integrated into different systems/cells
- Process-orientation for units
- Variable granularity and fluidity process related. This implies multiple scales
- Control system, distributive
- Interoperability
- Multi-agent technology to capture emergent behavior

Evolvable systems may be considered as a natural development of flexible and reconfigurable manufacturing systems (Table 5.10).

Table 5.10 Comparison of different management systems

Criterion\System	**Specialized**	**Flexible**	**Reconfigurable**	**Evolvable**
Skills	One	Set of fixed skills	More skills adapted	No particular product focus
Flexibility	Low	Discrete	Continuous	Emergent
Capability	High efficiency for one situation	Cope with different situations	Cope with differences. Can be adapted	Very agile
Concerns	Rigid	Cannot cope with new	Unexpected are not coped	Difficult to define a generic mechanism

Table 5.10 suggests considering the different stage in the historical development of manufacturing systems as the necessary stages in categorification.

The first stage corresponds to specialized manufacturing, to single installation and in the same time to sets or 0-categories (Appendix A4).

A 1st order evolutionary step is represented by the transition to flexible manufacturing systems.

Flexibility approach allows doing diverse tasks with the same installation.

This is associated to 1-categories.

A 2nd order evolutionary step is represented by the transition to reconfigurable manufacturing systems.

Reconfiguration is supposed to make use of several installations. It is linked to the 2-categories. Reconfigurable manufacturing systems incorporate principles of modularity, integrability, flexibility, scalability, convertibility and diagnosability. Some flexible and reconfigurable systems failed because they don't take into account that if any system is to be flexible then its constituents need to be far more flexible.

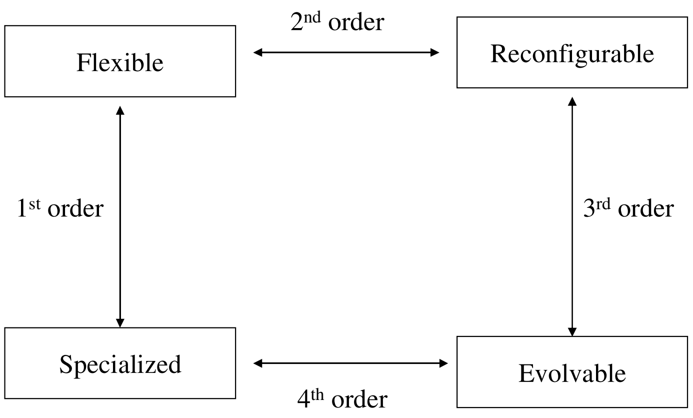

Fig. 5.18 Four stages for evolvable manufacturing systems

A 3^{rd} order evolutionary step is represented by the transition to evolvable manufacturing systems.

Evolvability achieves the full flexibility and is related to the 3-categories concept implementation.

Observe that EAS, EPS, EMS considers the production unit as an artificially living entity and emphasizes on evolution rather than adaptation.

Usually the adaptability implies an adjustment on the time scale of the life cycle of the organism. It characterizes 1-category frames. But this is not enough to challenge the high complexity. Evolvability should imply the capacity for genetic-like change to invade new life-like cycles on several time-scales, by higher categorification steps.

In a dynamic environment, the lineage that adapts first wins. Fewer mutations steps mean faster evolution. The request is for some production or management systems built to minimize the number of mutations required to find improvements.

Fig. 5.18 reinforces the idea of categorification process by imposing to different realms to be categories. By successive categorification the legacy equipment and associated software will still be utilizable. An n-graph model may be naturally associated to the framework shown in Fig. 5.18. Categorical issue implies that EMS achieves specific fluidity properties. It should have fluidity at different levels of complexity. Consider that the production line is composed from several components that can be plugged in or out. These are 1-cells and the corresponding fluidity is the so-called fine fluidity or 1-fluidity corresponding to flexible manufacturing and to 1-categories.

When a manufacturing line is composed of several cells and these cells are modules or 2-cells that can be plugged in or out this is the thin fluidity or 2-fluidity. It corresponds to reconfigurable manufacturing and to 2-categories.

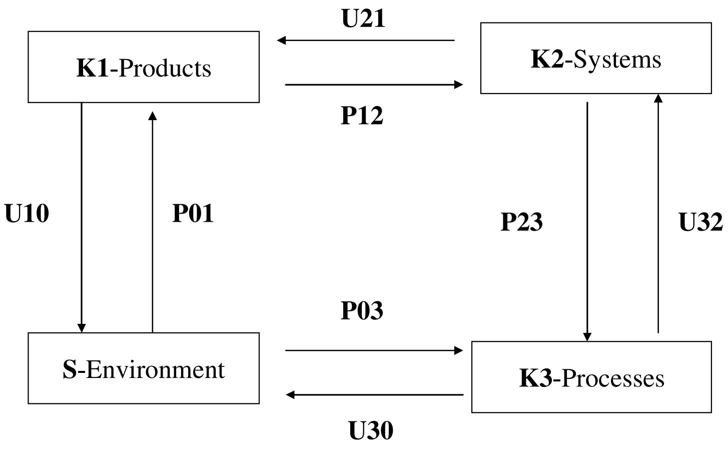

Fig. 5.19 Four sub-realms network for evolvable manufacturing systems

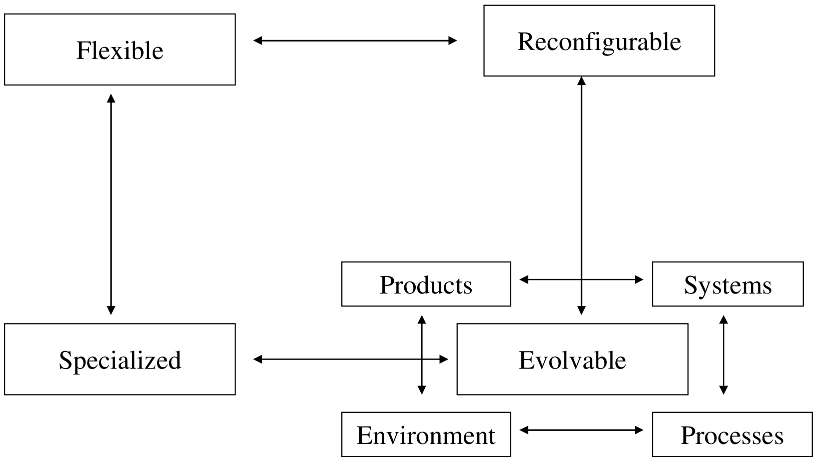

Fig. 5.20 Four realms and sub-realms for evolvability

The thick fluidity or 3-fluidity will refer to the whole system that is 3-cells to be plugged in or out. This corresponds to evolvable manufacturing and to 3-categories.

The autonomic and organic computing (Kephart and Chess 2003, Bauer and Kasinger 2006) were identified as fundamental concepts for achieving evolvable manufacturing systems. Although autonomic computing was designed for software systems, the related ideas can be projected into a modular production system. Automatic computing in this context refers to computing elements disseminated throughout the production system which beyond the normal mechanical, electrical and sensorial units includes computational power.

Organic computing focuses on completing the closure by studying the 4^{th} order evolutionary step.

Fig. 5.19 outlines a four sub-realms network for evolvable manufacturing systems.

The environment refers to real and artificial aspects, including the available materials.

Products sub-realm denotes the products and product related activities. Production sub-realm denotes the production system skills, modules. Processes sub-realm refers to all processes, for example assembly.

The signification of the functors U and possibilities P is explicit.

U10, U21, U32 and U30 corresponds to top-bottom implementation operations

In this case U30=U10oU21oU32.

P01, P12, P23 and P03 are bottom-top synthesis steps. Observe that: P03=P01oP12oP23.

Fig. 5.19 emphasizes the role of integrative closure via U30 and P03.

Onori (2002) highlighted the interaction between products and systems illustrated by a generic product life cycle view.

The entire structure shown in Fig. 5.19 may be just one realization of the four stage diagram shown in Fig. 5.18. This aspect is clarified by the Fig. 5.20.

It is a superposition of Fig. 5.18 and Fig. 5.19.

Observe that the construction of a specific evolvable manufacture parallels and recapitulates the general history of manufacturing systems from specialized to evolvable.

5.4.4 Belief Desire Intention Agents

The belief desire intentions, BDI, agent introduced a formal meta-language to express agent rationality in an explicit way. BDI architecture is one of numerous architectures that enact deliberative agents. The BDI agent architecture is an attempt to encapsulate the hidden complexity of the inner functioning of an individual agent.

The agent shown in Fig. 5.21 is structured in four elements: beliefs, goals, plans and intentions (Rao and Georgeff 1991). We will refer to a BGPI structure for BDI agent.

Fig. 5.22 outlines a possible categorical framework for a procedural reasoning system with three levels.

S represents the environment.

K1 and K2 are the two cognitive levels. In this case K1-includes goals while K2-includes plans. The strategies are defined at the K2-level.

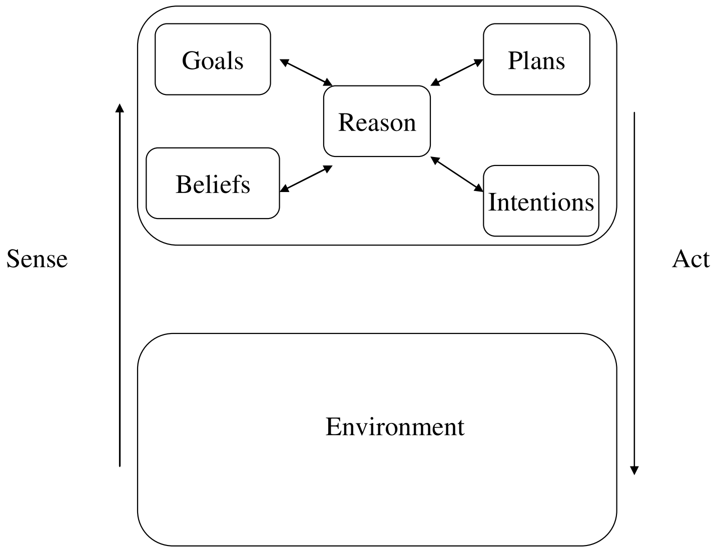

Fig. 5.21 Structure of BDI agents

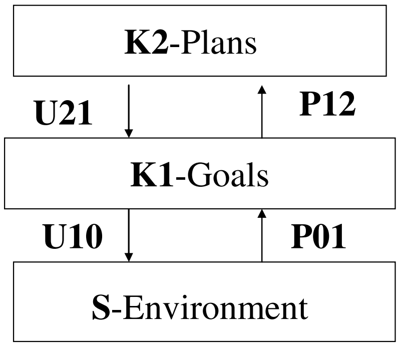

Fig. 5.22 Three levels hierarchical framework for cognitive agents

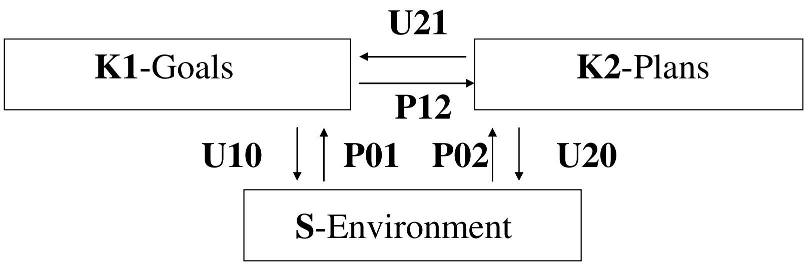

Fig. 5.23 Three realms framework for cognitive agents

U10: K1→S describes the actions from goals towards the environment level while P01: S→K1 summarizes the sensed info about S and forwards this toward goals.

The information change between the basic level and the K2 of strategies is characterized by the operator U21 and the possibilities P12.

The BDI architecture with three realms, and links between K2 and S is shown in Fig 5.23.

The framework shown in Fig. 5.23 allows cognitive evolvability and autonomy.

5.4.5 Multiple Levels Cognitive Architecture

5.4.5.1 Multiple Levels Agents

Innovative multiple-scale agent architectures have been proposed by Goschnick (Goshnick 2003, Goschnick and Sterling 2002).

Based on Jung analytical psychology (Jung 1997) Goschnick developed a cognitive architecture named Shadow Board. This implies:

- Decomposition of a user's multiplicity of roles into a hierarchy of sub-agency
- Relaxing of the autonomy of the sub-agents under control of an autonomous agent-the so called Ego/Self agent which is autonomous
- Wrapping of the external services and agencies including the web services and utilizing them as if they were internal sub-agents
- Ability to apply ontology at the localized level

The Shadow architecture may be considered as resulting by WE equation.

Table 5.11 shows the agent based structure of the cognitive architecture at different levels.

The notations are: B-beliefs, G-goals, P-plans, and I-intentions.

The central agent $\begin{matrix} G & P \\ B & I \end{matrix}$ is the so-called Ego/Self agent (Goshnick 2003).

It is autonomous in the sense that its parts B, G, P, I don't depend on others agents.

This central agent should be considered as a whole.

It is an executive decision maker. Decisions are based on the knowledge of sub-selves B, G, P and I.

Table 5.11 Array of conditions for BGPI multi-agent system

G22		G23		G32		G33		P22		P23		P32		P33
	G2				**G3**				**P2**				**P3**	
G21		G24		G31		G34		P21		P24		P31		P34
			G								**P**			
G12		G13		G42		G43		P12		P13		P42		P43
	G1				**G4**				**P1**				**P4**	
G11		G14		G41		G44		P11		P14		P41		P44
							G P B I							
B22		B23		B32		B33		I22		I23		I32		I33
	B2				**B3**				**I2**				**I3**	
B21		B24		B31		B34		I21		I24		I31		I34
			B								**I**			
B12		B13		B42		B43		I12		I13		I42		I43
	B1				**B4**				**I1**				**I4**	
B11		B14		B41		B44		I11		I14		I41		I44

The Ego/Self agent is able to call the four main agents, B, G, P and I individually or in team. The sub-selves are again divided in sub-sub-selves as for instance B is divided in four agents B1, B2, B3 and B4. Then the splitting in four is continued one level more. From B1 it results B11, B12, B13 and B14 and so on. They may be identified with an elementary BDI agents B11-belief, b, B12-goal, g, B13-plans, p and B14-intentions, i. Consider that this elementary level is indexed by n=0. It is possible that the elementary agents are unrelated.

The coupling in agents corresponds to the level n=1. The resulting agents are B1, B2 and so on.

For the agent B2 we have specific activities: B21-belief, b, B22-goal, g, B23-plans, p and B24-intentions, i.

The coupling of information corresponds to the level n=2. At this level the B, G, P and I are the four agents. The reality level n=3 corresponds to the Ego/Self agent.

All levels have been illustrated by Table 5.11.

5.4.5.2 n-Graphs for Multiple Levels BGPI

Fig. 5.24 shows a representation of multiple scales frames using n-graphs.

The reality level n=0 corresponds to the 0-graphs or sets. They are represented by b, g, p or i individual uncorrelated objects. The level n=1 correspond the 1-graphs. These are directed graphs including the morphisms that is, the connections between b, g, p and i.

The morphisms are 1-cells. Their coupling allows the functioning of agents.

The level n=2 corresponds to the 2-graphs. These are graphs plus the 2-cells between paths of same source and target. These 2-cells express the

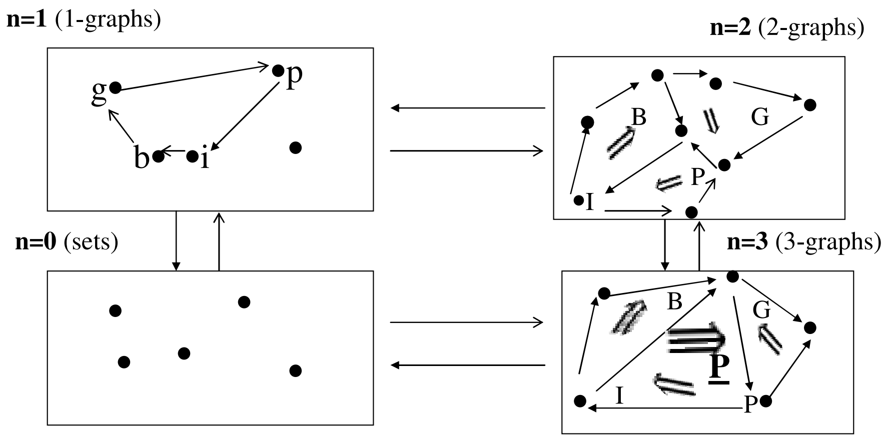

Fig. 5.24 n-graphs for multi-scale framework

natural association of the quadruple b, g, p, i elements in just one macro agent denoted here by B-belief, G-goal, P-plans or I-intention. The level n=3 corresponds to the 3-graphs. These are 2-graphs that include the 3-cells that is, the cells between 2-cells. Fig. 5.24 shows a complete association as a plan **P**. The 3-graphs represent graphs modification and are subjected to conditions of natural transformations too.

5.4.5.3 Nested Frameworks for BGPI

Fig. 5.25 shows a categorical presentation for BGPI architecture as presented in a different form in Table 5.11.

In this presentation K0 includes environment and the Beliefs, K1-Goals, K2-Plans, K3-Intentions.

The architecture shown in Fig. 5.25 outlines the possibility of integrative closure allowed by the link between K0 and K3 opening the road for evolvability and autonomy.

The architecture is not confined to four realms.

Fig. 5.25 shows also nested and self-similar architectures.

A similar structure is repeated starting from the whole central system built by four sub-realms denoted here by k0, k1, k2 and k3.

It should be noted that similar architectures are of interest for autonomic and organic computing (Trumler et al. 2004, IBM 2005, Bauer and Kasinger 2006).

The logical structure of an autonomic element is similar to that of BDI or BGPI agents.

For autonomic computing, the BGPI structure is replaced by the so-called MAPE loop whose elements are M-Monitor, A-Analyze, P-Plans, E-Execute.

Autonomic computing systems are composed of four levels identified as S-Managed resources, K1-Autonomic managers, K2-Orchestred autonomic

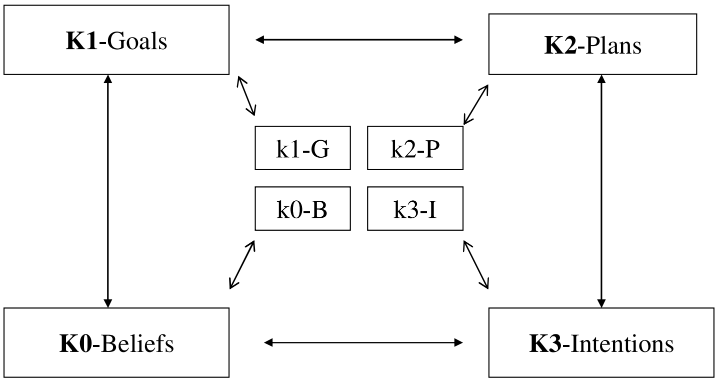

Fig. 5.25 Nested frameworks for BGPI

managers, K3-Manual manager. The closed loop in which K3 is replaced by an automatic device was presented by IBM (2005).

For the organic computing middleware architecture (Trumler et al. 2004), the four levels may be identified as: S-Transport interface, K1-Event dispatcher, K2-Service interface and proxy, K3-Organic manager. In the middleware architecture the organic manager is, linked to the levels below it.

References

Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)

Aha, D.W., Breslow, L.A., Munoz-Avilla, H.: Conversational case-based reasoning. Applied Intelligence 14(1), 9–32 (2001)

Baina, S., Morel, G.: Product centric holons for synchronization and interoperability in manufacturing environments. In: 12th IFAC Symposium on Information Control Problems in Manufacturing, St-Etienne, France (2006)

Bar-Yam, Y.: When Systems Engineering Fails-Toward Complex Systems Engineering. In: International Conference on Systems, Man & cybernetics, vol. 2, pp. 2021–2028. IEEE Press, Piscataway (2003)

Bauer, B., Kasinger, H.: AOSE and organic computing-how can they benefits from each other. Position paper. AOSE III. Springer, Heidelberg (2006)

Benami, O., Jin, Y.: Creative stimulation in conceptual design. In: Proceedings of DETC 2002 ASME 2002 Design Engineering Technical Conference, Montreal, Canada, vol. 20, pp. 1–13 (2002)

Black, J.: The Design of the Factory with a Future. McGraw-Hill, New York (1991)

Braha, D., Maimon, O.: A mathematical theory of design: foundations, algorithms and applications. Kluwer, Boston (1998)

Braha, D., Reich, Y.: Topological structures for modeling engineering design processes. Res. Eng. Design 14, 185–199 (2003)

Carreras, I., Miorandi, D., Saint-Paul, R., Chlamtac, I.: Bottom-up design patterns and the energy web. IEEE Transactions on Systems, Man Cybernetics, Part A, Special issue on Engineering Cyber-Physical Systems (2009)

Coyne, R.: Logic Models of Design. Pitman, London (1988)

Delugach, H.S.: Towards Building Active Knowledge Systems With Conceptual Graphs. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 296–308. Springer, Heidelberg (2003)

Dettmer, H.W.: Strategic Navigation: A Systems Approach to Business Strategy. ASQ Quality Press (2003)

Estrin, D., Culler, D., Pister, K., Sukhatme, G.: Connecting the physical world with pervasive networks. IEEE Pervasive Computing, 59–69 (2002)

Frei, R., Barata, J., Di Marzo Serugendo, G.: A Complexity Theory Approach to Evolvable Production Systems. In: Sapaty, P., Filipe, J. (eds.) Proceedings of the International Workshop on Multi-Agent Robotic Systems (MARS 2007), pp. 44–53. INSTICC Press, Portugal (2007)

Gani, R.: Chemical product design: challenges and opportunities. Comp. & Chem. Engng. 28, 2441–2457 (2004)

Goschnick, S.B.: Enacting an Agent-based Digital Self in a 24x7 Web Services World. In: Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E. (eds.) ISMIS 2003. LNCS (LNAI), vol. 2871, pp. 187–196. Springer, Heidelberg (2003)

Goschnick, S.B., Sterling, L.: Psychology-based Agent Architecture for Whole-of-user Interface to the Web. In: Proceedings of HF2002 Human Factors Conference: Design for the Whole Person - Integrating Physical, Cognitive and Social Aspects, Melbourne (2002)

Grabowski, H., Rude, S., Klein, G. (eds.): Universal Design Theory. Shaker Verlag, Aachen (1998)

IBM, An architectural blueprint for automatic computing (2005)

Iordache, O.: Evolvable Designs of Experiments. Applications for Circuits. J. Wiley VCH, Weinheim (2009)

Jung, C.G.: Man and his symbols. Dell Publishing Company, NewYork (1997)

Kephart, J.O., Chess, D.M.: The vision of autonomic computing. IEEE Computer 36(1), 41–50 (2003)

Kiriyama, T., Tomiyama, T., Yoshikawa, H.: Qualitative Reasoning in Conceptual Design with Physical Features. In: Faltings, B., Struss, P. (eds.) Recent Advances in Qualitative Physics, pp. 375–386. MIT Press, Cambridge (1992)

Lee, E.A.: Computing foundations and practice for cyber-physical systems: A preliminary report. Tech Report Univ of California Berkeley/EECS-2007-72 (2007)

Li, S., Yang, Q.: Active CBR, Integrating case-based reasoning and active database, TR-1999-03, School of Computing Science, Simon Fraser University, Burnaby BC, Canada (1999)

Lohse, N., Valtchanov, G., Ratchev, S., Onori, M., Barata, J.: Towards a Unified Assembly System Design Ontology using Protégé. In: Proceedings of the 8th Intl. Protégé Conference, Madrid, Spain (2005)

Melendez, J., Colomer, J., Macaya, D.: Case based reasoning methodology for supervision. In: Procceedings of the European Control Conference, ECC 2001, Oporto, Portugal, pp. 1600–1605 (2001)

Montani, S., Anglano, C.: Case-Based Reasoning for autonomous service failure diagnosis and remediation in software systems. In: Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A. (eds.) ECCBR 2006. LNCS (LNAI), vol. 4106, pp. 489–503. Springer, Heidelberg (2006)

Naumenko, A., Wegmann, A.: Two approach in system modeling and their illustration with MDA and RM-ODP. In: ICEIS 2003, the 5th International Conference on Enterprise Information System (2003)

Nejdl, W., Froehlich, P., Schroeder, M.: A formal framework for representing diagnosis strategies in model-based diagnosis systems. In: Int. Joint Conf. on Artif. Int., IJCAI, vol. 95, pp. 1721–1727. Morgan Kaufmann Publishers, Inc., San Francisco (1995)

Onori, M.: Evolvable Assembly Systems-A New Paradigm. In: IST 2002 33rd International Symposium on Robotics, Stockholm, pp. 617–621 (2002)

Onori, M., Barata, J., Frei, R.: Evolvable Assembly System Basic Principles. BASYS Niagara Falls, Canada (2006)

Pahl, P.G., Beitz, W.: Engineering design, a systematic approach. Springer, London (1996)

Parunak, H.V.D., Brueckner, S.: Entropy and Self-Organization. In: Multi-agent Systems, Proceedings of the Fifth International Conference on Autonomous Agents, pp. 124–130. ACM Press, New York (2001)

Pattee, H.H.: Causation, control and the evolution of complexity. In: Anderson, P.B., et al. (eds.) Downward Causation, pp. 63–77. Aarhus University Press, Aarhus (2000)

Piaget, J.: Genetic Epistemology. Columbia University Press, New York (1970)

Piaget, J.: The construction of Reality in the Child. Ballantine Books, New York (1971)

Rao, A., Georgeff, M.: Modelling rational agents with a BDI architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds.) Proceedings of Knowledge Representation and Reasoning. Morgan Kaufman Publishers, San Mateo (1991)

Rayudu, R.K., Samarasinghe, S., Maharaj, A.: A co-operative hybrid algorithm for fault diagnosis in power transmission. IEEE Journal of power Systems Engineering, 1939–1944 (2000)

Reich, Y.: A critical review of General Design Theory. Res. Eng. Des. 7, 1–18 (1995)

Sakhanenko, N.A., Luger, G.F., Stern, C.R.: Managing Dynamic Contexts Using Failure-Driven Stochastic Models. In: Wilson, D., Sutcliffe, G. (eds.) Proceedings of the Florida Artificial Intelligence Research Society of AAAI, FLAIRS-2, pp. 466–472. AAAOI Press (2007)

Schank, R.: Dynamic memory: a theory of reminding and learning in computers and people. Cambridge University Press, Cambridge (1982)

Sisiaridis, D., Rossiter, N., Heather, M.A.: Holistic Security Architecture for Distributed Information Systems - A Categorical Approach. In: European Meeting on Cybernetics and Systems Research, Symposium Mathematical Methods in Cybernetics and Systems Theory, EMCSR-2008, University Vienna, pp. 52–57 (2008)

Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove (2000)

Sugi, M., Maeda, Y., Aiyama, Y., Harada, T., Arai, T.: A Holonic architecture for easy reconfiguration of robotic assembly systems. IEEE Trans. on Robotics and Automation 19(3), 457–564 (2003)

Takeda, H., Tomiyama, T., Yoshikawa, H., Veerkamp, P.J.: Modeling design processes. Technical Report CS-R9059, Centre for Mathematics and Computer Science (CWI), Amsterdam, Netherlands (1990)

Takeda, H., Iino, K., Nishida, T.: Agent organization and communication with multiple ontologies. International Journal of Cooperative Information Systems 4, 321–337 (1995)

Tharumarajah, A., Wells, A.J., Nemes, L.: Comparison of the bionic, fractal and holonic manufacturing systems concepts. International Journal of Computer Integrated Manufacturing (9), 217–226 (1996)

Tomiyama, T., Yoshikawa, H.: Extended General Design Theory. In: Design Theory for CAD, Proceedings from IFIP WG 5.2, Amsterdam (1987)

Tomiyama, T., Kiriyama, T., Takeda, H., Xue, D.: Metamodel: A key to intelligent CAD systems. Research in Engineering Design 1(1), 19–34 (1989)

Trumler, W., Bagci, F., Petzold, J., Ungerer, T.: Towards an organic middleware for the smart dooplate project. GI Jahrestagung 2004(2), 626–630 (2004)

Ulieru, M.: Emergence of holonic enterprises from multi-agents systems: A fuzzy evolutionary approach. In: Loia, V. (ed.) Soft Computing Agents, pp. 187–215. IOP Press, Amsterdam (2002)

Ulieru, M., Brennan, R.W., Walker, S.S.: The holonic enterprise: a model for Internet-enabled global manufacturing supply chain and workflow management. Integrated Manufacturing Systems 13(8), 538–550 (2002)

Valckenaers, P., Van Brussel, H., Bongaerts, L., Wyns, J.: Holonic Manufacturing Systems. Integr. Comput-Aided Eng. 4(3), 191–201 (1997)

Yoshikawa, H.: General Design Theory and a CAD system. In: Man-Machine Communications in CAD/CAM, Proceedings, IFIP W.G5.2, Tokyo, pp. 35–38. North-Holland, Amsterdam (1981)