Goal-oriented Holonics for Complex System (Self-)Integration: Concepts and Case Studies

Ada Diaconescu, Telecom ParisTech, diacones@enst.fr
Sylvain Frey, Lancaster University, s.frey@lancaster.ac.uk
Christian Müller-Schloer, Leibniz University, cms@sra.uni-hannover.de
Jeremy Pitt, Imperial College London, j.pitt@imperial.ac.uk
Sven Tomforde, Kassel University, stomforde@uni-kassel.de

***Abstract*—System integration from sub-systems has always been a major engineering problem, which is progressively exacerbated by (1) sub-systems becoming more diverse, self-* and autonomous (2) systems operating in open environments, with third-party sub-systems joining and leaving unpredictably, (3) system (self-)integration being an ongoing process, increasingly needed at runtime. The fact that this problem occurs more and more often, as systems are built increasingly by composing existing sub-systems, requires rigorous, reusable integration solutions to replace ad-hoc approaches. In a complex world of uncertainty and change the new system integration paradigm must feature two main characteristics: support for a system-of-systems approach to manage complexity, and support for a high-level relation between sub-systems to manage diversity, uncertainty and dynamics. We propose a conceptual modelling solution combining holonic principles with goal-based relations. We highlight the key properties of holonic designs that support a systems-of-systems approach. We then specify the high-level relations between holonic sub-systems as goal-oriented requests and replies. Argumentation is grounded via concrete examples from existing complex systems. The proposed paradigm paves the way for future methodologies and tools for designing the next generation of socio-technical and cyber-physical systems.**

I. Introduction

The complexity of Information and Communication Technology (ICT) systems, and hence of their engineering and management processes, stems from aiming to fulfil multiple (conflicting) objectives, for different authorities, at various scales and time spans; while consisting of a large number of diverse and dynamic entities; and having to operate in complex environments. System development is increasingly replaced by system composition from existing entities (e.g. components, services, or entire systems). This process is being progressively pushed into the runtime for adapting systems to evolving goals and dynamic environments [1]. The task is more challenging still as integrated entities are more autonomous, may join and leave the system unpredictably, or update their behaviour, e.g. via self-adaptation or self-expression [2].

In a complex world of perpetual change and uncertainty, a new *integration paradigm* is needed to enable systems to self-compose and self-adapt at runtime, potentially from self-* sub-systems. As no paradigm occurs in a vacuum, we converge concepts from *software engineering (SE)* – e.g. requirements engineering, component and service models, and systems of systems; *artificial intelligence (AI)* and *multi-agent systems (MAS)* – e.g. problem solving and dynamic organisation of autonomous entities; and *complexity science* – e.g. holonic systems; to propose a novel paradigm based on a *conceptual model that helps understand, analyse, design and communicate about engineered complex systems, in a uniform way, and at a high abstraction level, which nevertheless lends itself to a transformation into concrete system architectures*.

The proposed solution relies on three key aspects: *holonic structures*, *goal-oriented relations*, and a *resolution method* for finding holonic structures that meet stakeholder goals. *Holonic structures* are encapsulated hierarchies with particular properties that help manage their complexity. We identify these as: *internal semi-isolation*, *external abstraction* and *graduated reactivity* across holonic levels. *Goals* represent abstract points-of-reference that self-* systems aim to achieve. Stakeholders - e.g. users, owners, designers or other systems - request *target goals* from a system, which aims to find the *system-of-systems* composition that achieves the goals. Each system in the composition *provides* target goals to its requesters and *requires* from other systems *cause goals* (i.e. which help provide its target goals). Finding the system composition that *resolves* a target goal is a recursive search process (involving intra-system goal transformations and inter-system goal requests and replies). Each step adds more systems to the composition, transforming target goals into cause goals, until reaching a set of leaf systems that can achieve their target goals without further cause goal requests. This process consists of (self-)sub-processes that can propagate top-down and/or bottom-up, in parallel, through the system of systems.

We contend that the structure of the system of systems should be holonic [3], [4], in order to address the *limited rationality* problem [5] of its (self-)integration search process (by reducing the combinatorial space of system compositions and offering *satisficing*, or "good enough" solutions, rather than optimal ones that arrive too late). It can also help prevent oscillations among the self-* processes of different systems (by tuning their dynamics to achieve progressive reactivity).

We illustrate and support this conceptual proposal via four concrete case study examples from a decentralised Community Energy System. The proposal is hence limited to concepts that we could identify and generalise from previously published engineered systems. In previous work we explored particular aspects of this broad challenge - e.g. a holonic architecture in smart grids [7] and self-governing communities [8]; goal-oriented self-growing systems [9]; conflict-resolution design patterns [10]; and, self-integrating feedback loops [11].

We do not claim here to offer a complete solution to the broad problem of system self-integration, but to propose a conceptual way to: i) structure the problem so as it can be split into (top-down) and/or composed from (bottom-up) sub-problems; and, ii) allow for alternative solutions to be explored for each sub-problem, recursively and in parallel. Many questions remain on the precise definition of this process, which will also vary, case by case. Here, we define this process generally, based on cross-domain observations, and argue that if it produced holonic solutions, then it would be more efficient and create more viable systems, when complexity raises.

Concretely, the purpose of the goal-oriented holonic model is, firstly, to provide a viable base to help users, analysts and engineers to better understand, represent, analyse, design and communicate about complex systems, in an uniform way, at a high-abstraction level. Secondly, it is to provide a foundation for future methodologies, tools and technologies for developing and controlling self-integrating systems of (self-*) systems.

II. Conceptual Model Overview

To facilitate the explanation of the goal-oriented holonic model, we take a 'holonic' approach: first provide an overview of the main concepts and their interrelations (this section); then detail each one in a dedicated section (sections III to V), and finally exemplify them in section VI. When designing actual systems, further specification is needed for many concerns e.g. conflicts, negotiations, self-* processes, organisations, standards, knowledge and learning. These represent research areas in themselves, and are out of the paper's scope. We aim here instead to provide a conceptual model for facilitating the understanding and design of solutions for (self-)integrating these aspects (dynamically) in a decentralised fashion.

The conceptual model relies on three main elements: i) holonic structures, ii) goal-oriented systems and relations, and iii) a goal-resolution process that (self-)integrates goal-oriented systems into holonic structures that meet stakeholder goals. These concepts are merged into *goal-oriented (self-)integrated holonic systems*. In brief, if H_{GT0} is a holonic system-of-systems that achieves a set of target goals G_{T0}, then when this set changes to G_{T1}, the goal resolution process Π_{GT1} must find a new holonic structure H_{GT1} that achieves G_{T1}. Conceptually, the system of systems undergoes a series of state transitions: $H_{GT0} \overset{\Pi_{GT1}}{\rightarrow} H_{GT1} ... \overset{\Pi_{GTn}}{\rightarrow} H_{GTn}$. We briefly describe these conceptual aspects next.

Firstly, we adopt *holonic* principles (III) as key to the structuring and dynamics of viable complex systems, as argued by Simon [3], [5] and Koestler [4]. The relevance of these works to self-adaptive and self-organising systems was also identified in [12]. This paper analyses Simon's principles from an engineering perspective, to identify the reasons behind their criticality. From a purely topological outlook, a holonic system H is composed of sub-systems, recursively; with no self-similarity required across levels (like composition in SE). Still, importantly, they also feature particular properties that ensure their viability when complexity rises. We identify these as: *semi-isolation* of a holon's internals from its external environment; *abstract* external perception of a holon via an aggregate of its state and behaviour; and, *graduated reactivity* across holonic levels. We discuss the implications and ways of achieving these when engineering complex systems.

Secondly, we consider goals (IV) as first class entities for system abstraction, since they represent reference points to reach in self-* systems where everything else can change; even if goals may also change. Goals also offer a uniform way to model interactions between highly diverse systems, via goal requests and replies, and hence enable their (self-)integration into systems of systems. Based on such goal-oriented relations, the system of systems that achieves the stakeholder's target goals G_T can be modelled as a directed *goal dependency graph (GDG)*, or network: $N_{GT} =< G_T, P >$ with target goals G_T as the root nodes and P a set or paths that lead to the achievement of G_T. A path is a list of goals $p =< g_0, g_1, ..., g_n >$, the first one being the target goal; and, for any adjacent pair $[g_t, g_c]$ that is a sub-path of p, and for any $p \in P$, we have that cause goal g_c causes target goal g_t. Several paths connecting different cause goals to target goals may be interrelated (e.g. contribute to the target goal together). Some cause goals can belong to several paths leading to different target goals; and may impact other targeted goals negatively (conflict). Goals at 'lower' abstraction levels can cause goals at 'higher' levels and vice-versa. Each path must be mapped to an actual system implementing it (i.e. linking cause goals to target goals); a system can be mapped to several paths. Since paths are contained within each other, so can systems encapsulate other systems. This results in a system of systems interrelated by the GDG, denoted as H_GT (IV-A).

Thirdly, we define an abstract *system (self-)integration process*, which, when given a set of target goals G_T, must *resolve* them by *finding* a system of systems composition H_{GT} – i.e. generating a GDG N_{GT} and its system mappings (to simplify, we ignore mappings in the following as tightly linked to the paths they implement). We use the append operator $++$ to return a concatenated path of two input paths; and the $last(p)$ operator to return the last goal of path p (leaf). Then we inductively define the GDG creation process as follows. In the first step $P^0 := < G_T >$; the GDG only contains the root nodes G_T and so we only have singleton paths containing these nodes. At each next step, a process π takes the last goal of each path, computes their cause goals, and appends them to the paths: $P^{i+1} := \{p + + < g_c >| \exists p \in P^i . \exists g_c \in G_c . last(p) = G_t \wedge \pi(G_t) = G_c\}$. At the end of each step, the last cause goals become target goals for the next step (i.e. systems that can target them were found). The process repeats until it produces no more cause goals to extend the paths: $P^{n+1} = \{\}$, where n is the number of steps for creating the GDG (longest path, or chain). The complete set of paths P is the union of all paths in steps $0 \leq i \leq n, P = \cup P^i$. This process can be executed top-down (e.g. via goal splitting and refinement) or bottom-up (e.g. via self-organisation and abstraction); and often both ways, in a vertical multi-level feedback-loop (i.e. a yoyo process [13], [14]). From this process we can define the set of processes Π_{GT} that describes each state transition, which transforms one holonic structure into a new one H_{GT}, that achieves G_T.

Intuitively, when a G_T is requested from a system, it generates a set of *request* chains, each system mapping its target goals to cause goals and requesting these from other systems, as above, hence forming the GDG N_{GT}. Once this is completed and the resulting system of systems H_{GT} executes, it results in the corresponding set of *reply* chains, where accomplished cause goals actually cause the target goals. When a cause goal fails to achieve a target goal (as per runtime evaluation), the request process restarts from that point. Each (self-*) system in H_{GT} performs these processes in parallel.

We refer to this process as *goal resolution*. In AI, it is a problem-solving process, e.g. [15]. In "traditional" SE, it is a manual offline process transforming system goals into technical requirements [16], design and implementation. In service-oriented systems, it relies on dynamic service binding (matching provided/required interfaces). In MAS, it is similar to mapping of goals to organisations (i.e. roles and relations), and then of roles to agents that can fulfil them [18], [19], [17].

III. Holonic Structures. Properties and Benefits

This section introduces holonic structures, as observed in *natural systems*, and emphasises key properties ensuring their viability in complex environments (drawing on Simon's work). We then show how goal-oriented designs (section IV) can help achieve these properties in engineered systems (section V).

A. Holonic System Overview

Holonic systems feature an encapsulated hierarchy structure where systems (or holons) are both composed of sub-systems and part of supra-systems [3], [5]. Each holon is a semi-autonomous entity playing a double role [4]: a self-sufficient whole controlling its parts; and a dependent part of a supra-system. In some cases, sub-holons can simultaneously be part of several holons. In addition to these structural characteristics, holonic designs may feature specific properties that are key to system development and adaptation when complexity rises.

B. Properties and Benefits of Holonic Structures

1) Viable holonic complexity from simplicity: Holonic systems are more likely to achieve *viable* structural and functional complexity (i.e. they can achieve their goals or survive) than other organisations, since their complexity can be built progressively based on combinations of simpler, viable, structures and functions; and rebuilt from intermediate composites, when current formations fail, rather than restarting from scratch (Cf. the watchmaker's analogy in [3] or [5]).

In engineered systems, this allows designers to concentrate on one component at the time and to reuse basic components and intermediate composites across systems. Complexity is built by integrating more complicated composites. Similarly, in natural systems, evolution only has to "come up with" stable compositions based on simpler ones, and then find ways to combine these into new composites; recursively. This is much faster than evolving complex systems directly from basic elements. When this leads to dead-ends, composites can be dismantled and alternative ones tried-out at each level; via a partial roll-back process rather than starting from scratch. In self-* systems, this can enable the progressive integration of self-* processes into coherent stable composites.

2) Holonic encapsulation and semi-isolation: The internal entities of each holon can be *partially isolated* from the external environment (e.g. peers or supra-holons). Their interactions with the environment are limited to a well-defined range of exchanges (e.g. in/out-put types). This creates a semi-controlled environment within each holon, diminishing environmental unpredictability for internal entities and hence facilitating successful adaptation to a limited inner state space [6]. Of course, this limits the holons adaptability; and if the isolation is breached, the holon can be corrupted or destroyed (like most systems). When holons can function autonomously, in complete isolation, if needed, their robustness and resilience can further improve. Here, a holon can be integrated within a supra-holon (in order to benefit from it, possibly at a cost) when possible, but can also survive as a standalone system when needed - e.g. the supra-holon fails. This allows lower-level holons to survive and self-organise into a more suitable supra-holon; rather than restarting from scratch. Isolation also stops cascading failures from propagating through the system.

3) Holonic abstraction: Each holon is influenced by other holons only in a coarse manner, via an *aggregate* of their states and behaviours. While *encapsulation* limits external influence on a holon's internals, *abstraction* protects external components from the holon's details, which are exposed in aggregate form. This means that a holon can use or rely on another holon's aggregate effects, or functions, irrespectively of how these are obtained (from its internals). This can make holons less sensitive to changes in other holons internals, and render their integration more stable. It also facilitates the development and co-existence of holons with diverse structures and implementations, as only their aggregate effects matter. This helps system robustness, as diversity increases chances of survival in unpredictable environments. It also allows for local optimisations to specific contexts. Finally, it helps external observers to represent and reason about the system, as each holonic level can be specified separately, via the abstractions of its contained holons and their interrelations.

4) Progressive reactivity across holonic levels : In some holonic systems, sub-holons are more tightly coupled among themselves within a holon (more links and/or stronger influences) than with sub-holons in other holons. The same applies to the holons within a supra-holon, with respect to external holons. Consequently, changes and reactions within a sub-holon propagate faster within the containing holon than between holons. The same applies between holonic levels, with lower levels featuring higher change rates than higher levels.

This property can help limit chain reactions and oscillations through the entire holonic system, since each holon may stabilise after an internal change faster than this change can propagate to other holons. If the holons stable aggregate state does not change, then no impact is felt on the other holons.

If the holons aggregate state does change, then the other holons must adapt to it, but only after this new state is stable. Similarly, higher levels only adapt to aggregate changes in the lower levels, once they have stabilised. In some cases, lower levels subsequently detect and adapt to changes in the higher levels, which they have caused in the first place i.e. causing a yoyo effect. Yet, when these dynamics hold, such oscillations occur over longer periods and may not cause major instability.

IV. Goals as First Class Design Elements

This section discusses the benefits of goals as first class modelling entities, and introduces high-level goal specifications and transformations. Section V shows how these can help achieve holonic properties (section III) in engineered systems.

A. *Reference points in a world of change*

When everything within and without a self-* system can change, from its internal resources and integration architecture to its external environment, we need to set a reference point of the system. In engineered systems, we argue that this reference point is the stakeholder's *goals* for that system. Therefore, goals are first class elements in the proposed conceptual model (which simply means that they are key notions at the model's abstraction level). For this, firstly, goals need to be defined more clearly (goal specification, IV-B); and secondly, a method must be provided for achieving them (goal resolution, IV-C).

Goal specifications have been studied extensively in SE to define requirements [16], [20]; in MAS to define agent objectives [17]; or in AI to define problems to solve [15]. We converge the goal concepts from these areas and propose the following informal high-level definition. *A goal is an evaluable property that should be achieved, or a verifiable statement that can be deemed true (or not), of a state or behaviour of a system under consideration.* Goals can be defined at different abstraction levels, from high-level declarative statements (e.g. functional or qualitative services, economic targets, social values, constraints, policies and norms) to low-level procedural specifications (e.g. technical requirements, plans, architectural styles and method calls). Hence, the term *goal* is here an umbrella label to include a variety of concerns, signifying anything that the system's stakeholder cares about - i.e. the system's “raison d’łtre”. In contrast to engineered systems, natural systems appear to have as their *default* goal that of their very *existence*, survival, and derived sub-goals (like the zero for natural numbers). This can also occur in socio-technical systems that self-organise by adding individually-motivated links between pre-existing systems with no prior global purpose (e.g. global markets or social networks).

The method for achieving the goal (resolution) must be determined at runtime Π_{GT}, by finding the system of systems H_{GT} interrelated by the goal dependency graph (GDG) that achieves G_T (section II). Figure 1 offers a generic view of a goal-oriented (self-*) system, which can be (self-)integrated with other systems to form H_{GT} (Figure 2). Each goal and its evaluation is depicted explicitly, via a dedicated input and output port (implying nothing about their actual definition).

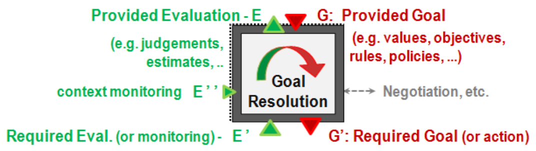

Provided Evaluation - E
G: Provided Goal
(e.g. judgements, estimates, ..
(e.g. values, objectives, rules, policies, ...)
context monitoring E''
Goal Resolution
Negotiation, etc.
Required Eval. (or monitoring) - E'
G': Required Goal (or action)

Fig. 1. Generic design of a goal-oriented self-* system

Each system has a set of *provided goals* (which it can pursue) and a set of *required goals* (which it needs from other systems in to reach its provided goals). When a system activates one of its provided goals (i.e. required by an external entity) this becomes a *target goal* for that system. It must then *resolve* the target goal by finding the internal actions and the required goals to activate (which become *cause goals*). This approach is similar to the BDI model in MAS [21], where beliefs map to knowledge, desires to target goals and intentions to cause goals. Certainly, the system has additional ports, e.g. context monitoring, negotiations or entity transit in/out of the system.

This architectural view is sufficiently generic to represent a complex system (e.g. with many stakeholders, goals, self-* functions and components), a system part (e.g. one control loop or component), a human actor or organisation, or combinations of these. An engineered system is a composition of (sub-)systems that fit this generic design. The (self-)process Π_{GT} that integrates (and disintegrates) these into compositions H_{GT} that fulfil target goals G_T is composed of the processes π of the systems involved. Each one aims to minimise the difference between its evaluation and its target goals, by requiring and evaluating cause goals from other systems, selected at runtime. Internal processes can range from basic reflexes (with/out self-learning), to intelligent and self-aware [22] processes or humans in the loop. This variety does not change the nature of the goal resolution process, only its efficiency, and hence can be modelled via a single abstraction.

B. *Defining goals in a world of change*

At the level of abstraction of interest here, goals should be well-defined yet minimal (i.e. specify precisely what the stakeholder wants to achieve; and not more). This focuses on *what* to achieve rather than *how*, leaving maximal flexibility to the (self-*) sub-systems. Many formalisms exist for goal-specifications, e.g. modelling of agent intentions, abilities, commitments, or desires (e.g. i* [20]); requirement engineering models (e.g. Kaos/Objectiver from Respect-IT[1]), objective

[1]Respect-IT: Requirements Engineering & Specification Techniques for IT

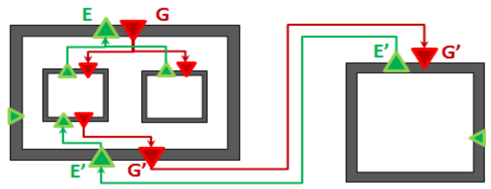


Fig. 2. Example of goal-oriented system of systems (holonic)

specification standards (e.g. IEEE-Std-830/1993 standards), rules, policies, constraint-oriented languages, or domain-specific formalisms (e.g. [9]). All these are refinements that should fit into the generic goal model discussed here.

Generally, a goal definition should comprise at least three parts ([7] for details): $G = (V_f, S_R, S_T)$, with V_f a *viability function* (which encapsulates both what to achieve e.g. 20 - 23C temperature; and its evaluation criteria - e.g. given a temperature measure, indicate the degree to which the goal has been achieved); S_R a *resource scope* (where to achieve it - over which set of resources is V_f defined and evaluated, e.g. in a smart home); and S_T a *time scope* (when to achieve it - over which periods is V_f evaluated, e.g. daily from 7pm to 11pm). Note that the viability function could return a binary value, a utility measure, or other semantics, e.g. too hot or too cold. Additional attributes may be defined, like priorities - to help resolve conflicting goals; or rewards and sanctions - in case goal-aware systems can choose to pursue a goal or not.

Many goal formalisms also include links between goals (e.g. positive / negative contribution, AND / OR refinement, satisfying / denying, dependency, or conflict links e.g. [16]); and between goals and the entities that fulfil them (e.g. agent responsibility or wish links to goals). These are compatible with the proposed model, yet separated from goal specifications and determined at runtime (resolution process). Hence, target goals are defined by human stakeholders, while cause goals are derived from these; either automatically or via human intervention; directly or via trial-and-error; top-down or bottom-up, or both. When users define several goals that have incompatible viability functions and intersecting scopes, *conflicts* may arise and have to be handled (use case VI-B).

C. Resolving goals in a world of change

For each system, resolving its target goal(s) G_T implies: 1) identifying the cause goals G_C that lead to G_T; 2) finding systems that provide G_C, and sending them goal requests; 3) evaluating G_C as provided by these systems (going to step 2 if they fail to provide); and 4) evaluating the extent to which G_C really achieve G_T (going to step 1 in case of failure).

These processes are propagated recursively, each step identifying further cause goals and the systems that can resolve them. Each system resolves its target goals either by its own means (self-action) or by transforming them into further cause goals; or both (partial self-action, partial goal transformation). In holonic systems the resolution process can map target goals to cause goals provided by *internal* sub-systems, recursively. This process may be implemented via various techniques, yet at the level of abstraction of interest here we merely focus on the goal transformation types that occur as systems achieve target goals collectively via mutual cause goal requests.

In previous work [7] we have identified two basic goal transformations, and respective reverse-transformations: *translation / inverse-translation* - changing the *type* of any goal element V_f, S_R or S_T (e.g. translating a thermostat's temperature target, into an electric heater's power configuration); and *splitting / composing* - changing the *values* of any goal element (e.g. dividing a thermostat's temperature goal over the entire house, all the time, into temperature goals for each room for each minute). Combining these operations can result in a plan with several interrelated cause goals (out of scope). Refining and formalising goal operations are topics of future research, as is conflict detection and resolution - e.g. if the scopes of a thermostat's temperature goal and a power manager's cost-saving goal intersect over an electric heater then it may lead to incoherent configurations. Introducing a system self-goal, that may conflict with external target goals allows to model conflicts between egoistic and pro-social behaviour.

V. Goal-oriented Holonic Systems

This section aims to discuss the benefits, implications and recommendations for designing (self-*) systems that feature holonic properties (III) based on goal-oriented concepts (IV).

A. Overview of goal-oriented holonic systems

Since technical complex systems are engineered for a purpose, merging the goal-oriented paradigm with the holonic structuration principles - giving rise to *goal-oriented holonic systems* - can capitalise on the benefits of both approaches. Each (self-)integrated holon has at least one target goal (or mere existence by default), achieved via a composition (linear or non-linear) of cause goals provided by its sub-holons. When holons belong to several supra-holons they may receive conflicting goal requests. Holonic designs may have several *semantic implications* concerning holons at higher levels with respect to lower levels: i) *authority* - higher priority of goal requests; ii) *scope* - larger field of view and/or action; and iii) *abstraction* - coarser granularity for observation, modelling and action. These aspects are typically interrelated.

We focus next on how system engineers can aim to achieve the properties identified in section III for holonic designs.

B. Engineering for Holonic Benefits

1) Complexity from simplicity via explicit goals: Compartmenting system functionality into interrelated self-encapsulated components is a well-known "divide and conquer" engineering technique. It is also the source of severe system integration issues, especially during runtime. Making holons goal-oriented, with well-defined provided and required goals, facilitates this task via (dynamic) goal-based system interconnections (goal matching). Goal-orientation allows for both: i) top-down goal translation and splitting into finer-grain goals, plans and actions; and ii) bottom-up goal composition and (re)definition. Both processes may occur simultaneously, within each holon, and between holons. Bottom-up processes may lead to the formation of smaller-scale sub-systems, that provide intermediate goals, which can then shorten top-down resolution processes that require these as their cause goals (goal publication, finding and dynamic matching are out of the paper's scope). Each holon evaluates and adapts its internal sub-holon organisation and implementation so as to meet its target goals (e.g. replace failing sub-holons or reorganise). Advanced holons may also justify the cause of goal failures to their requesters and suggest alternative goals or external help.

2) Encapsulation and semi-isolation via border control: In engineered systems semi-isolation can be achieved by encapsulating holons into a special-purpose container, or *membrane*, which defines a clear *boundary* between the holon's internals and its external media [6]. *Border control* mediates and regulates the holon's inputs and outputs - e.g. acceptable incoming and outgoing goal requests, evaluations, or sub-holons. The membrane can also transform the holon's target goals, resolve their conflicts, and distribute the results to internal sub-holons (future work will develop this aspect).

Semi-isolation allows the holon's internals to fine-tune and stabilise their self-integration processes, for target goal resolution, within the space delimited by well-defined provided and required goals, and influences from the execution context. This enables holons to develop diverse internal configurations that best meet the target goals with local resources and within local environments (also taking into account their self-goals). In open systems, semi-isolation can prevent the integration of unknown entities that may be malevolent; hence keeping high internal trust levels and a more efficient interaction between accepted members [33]. Surely, if the membrane is breached, these guarantees no longer hold and the holon should take appropriate action (e.g. reinforcing the membrane; triggering an immune response; signalling to external entities; self-disassembling; and preventing failure propagation).

3) Abstraction via goals: Goal-oriented holons can be abstracted as entities that reach well-defined goals, in certain contexts, without worrying about how they achieve this. The holon's success or failure, and its usefulness within a supra-holon's organisation, is determined merely by its provided goals and their evaluation. These represent aggregates, or abstract models, of the holon's capabilities, state and behaviour.

This goal-oriented abstraction helps human administrators, designers or autonomic managers to model, analyse and communicate about complex holonic systems, by focusing on one holonic level, in local context, at the time. It also helps to engineer holonic systems since each holon can be developed and maintained quasi-independently from the others, only taking into account their aggregate goal-based influences. It also facilitates the analysis of inter-holon goal dependencies, to assess the satisfiability of target goals.

The difficult system integration process can, in principle, be automated and moved into runtime, since the system itself may be able to search for successful holonic compositions via trial and evaluation processes, at increasing holonic levels. This search process is a research topic in itself (outside the paper's scope). We reemphasise however that it is much facilitated by holonic structuring, since intermediate composites attaining intermediate goals can be reused as intermediate search results.

Combined with the previous feature (semi-isolation via border control), the ability to represent and interact with holons via goal-oriented abstractions is key to supporting interrelations between *heterogeneous* holons, via standardised goal models. It hence facilitates their integration into coherent supra-holons. This allows dealing with local issues locally, and global issues globally, while (dynamically) finding a balance between the two. In authority-based hierarchies, this helps to balance the power between a holon's supra and sub-holons [8].

4) Progressive reactivity via tuning of self- processes:* In engineered holonic systems, the self-* processes within holons must be designed and tuned with respect to each other so as to ensure progressive reactivity among holons and across levels [3]. This can be achieved by having goal-requesting holons obtain goal evaluations (and hence react to these) less frequently than the rate at which goal-providing holons change internally. This may be achieved as evaluations rely on aggregate measures that take longer to collect and compute than the measure-producing processes. This means that the frequency of reactions increases as holons are farther from the roots of the goal dependency graph. Among reflex systems, progressive reactivity may be achieved by interconnecting a holon's sub-holons via topologies that favour communication and change propagation; and sub-holons in different holons and levels via more seldom links (e.g. community network). This can be facilitated by border control, which limits link formation across the holonic membrane; and hence encourages internal connections. Such tuning should allow self-* processes to stabilise within each holon, before triggering self-* processes in peer-holons and supra-holons. Achieving such dynamic properties is a rich research topic in itself (future work); here we merely identify it as a key engineering objective.

VI. Case Studies

We illustrate the proposed concepts via four interrelated case studies, based on previous work with smart micro-grids. This validates the conceptual model's applicability to concrete systems integrating other self-* systems. Briefly, smart houses in a neighbourhood are equipped with energy producing and consumption devices, or *prosumers* - e.g. thermostats, lamps, solar panels and batteries. Devices are autonomic and can reach simple goals, e.g. temperature, luminosity, or power prosumption. Houses are equipped with controllers that coordinate devices to achieve higher-level goals, like comfort and power targets. Houses are connected to a smart micro-grid, which must balance prosumption, avoid peaks, and compensate for any imbalances by prosuming from the main grid.

Each case study highlights a typical issue and illustrates a conceptual solution based on the proposed model: A) multi-layer translation from goals to rules to rule-enforcement; B) goal conflict resolution; C) top-down facilitation of bottom-up coordination; D) bottom-up goal definition and top-down goal enforcement. Solutions are not new, they merely ground the recommendations we make in the paper to existing cases. For each case study we 1) offer a brief description and 2) highlight the goal-oriented, holonic properties proposed in the paper. Goal definitions and transformations are conceptual; the exact formalisms have to be specified case by case. This aims to demonstrate the benefits of the presented modelling method.

A. Multi-level translation of single goal in a smart home

1) Overview and selected scenario: the first use case focuses on a single-goal smart home. It depicts a typical case of multi-level goal transformation, where each level is implemented via a distinct self-* process (holon). In most systems, a descriptive target goal must be translated into a prescriptive rule set, which must then be enforced into a (self-managed) production system. Each transformation can self-adapt at runtime, including the target goal by the user.

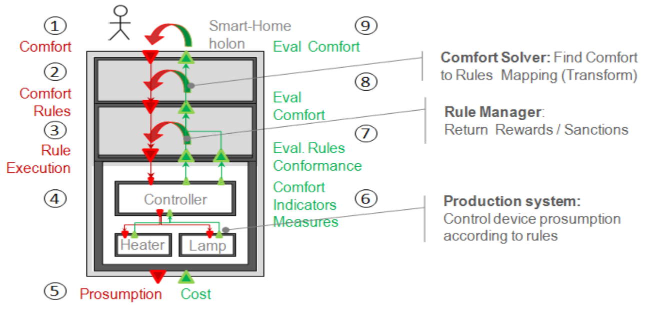
Figure 3 depicts a concrete example of such case. Initially, the user sets a comfort goal for the smart-home (step 1): $G_{Cmf} = comfort, home, forever)$. A Comfort Solver translates and splits this goal into rules for the home devices $G_R = (rules, devices, intervals)$, (step 2); also mapping the comfort goal to intermediate goals, e.g. temperature, $G_{Tmp} = (T, rooms, intervals)$, or luminosity, $G_L = (L, rooms, intervals)$; and setting their priorities (to simplify, shown as one translation from comfort to rules in Figure 3). Rules G_R are sent to a Rule Manager to enforce them. It sends them to the Production system for execution (step 3), where a centralised home Controller manages device prosumption (step 4) to meet the rules (e.g. [7]). Device usage results in an aggregated prosumption for the entire house (step 5, further discussed in case studies C and D). The Controller provides comfort indicators and is monitored for rule conformance (6). The Rule Manager returns rewards and sanctions to the production system, based on monitored rule conformance (7); the Controller may adapt its strategy or exclude non-conforming devices accordingly. Comfort indicators are sent to the Comfort Solver, which aggregates them into comfort evaluation estimates (8); then adjusts the rules G_R to better achieve the goal G_{Cmf}. It also forwards the comfort evaluation to the user, possibly indicating the causes of failures. Finally, the user changes the goal G_{Cmf} to a more realistic one (9).

2) Goal-based interactions & holonic properties: Complexity from simplicity is reached here by integrating several goal-oriented self-* loops: comfort goal adjustment by the human (formed by steps 9 and 1, for evaluation and action, respectively); translation of goals to rules (steps 8 and 2); rule enforcement (7 and 3); and rule-based device management (6 and 4) – this one being itself a hierarchy of goal-oriented self-* loops (Controller and smart devices). All self-* loops run and self-adapt in parallel depending on mutual feedback. Each is designed as a goal-oriented holon, connected together via a goal dependency graph with G_{Cmf} at its root.

Semi-isolation is achieved within each holon by limiting the kinds of goals it can be required to pursue (e.g. G_{Cmf} for the ComfortSolver, well-defined rules for the Rule Manager and the Controller). Hence, they can each optimise their internals for these types of goals. In the Production system, semi-isolation is also achieved by screening devices before they join the smart-home, e.g. to ensure they are trustworthy and can be managed by the Controller. Furthermore, external entities like grid controllers have no visibility over these devices, for privacy preservation; unless special permission is given.

Abstraction is reached by representing each holonic level via its goals: a home is seen from the grid only via its aggregate prosumption; the home Controller by the Rule Manager only via its conformance to rules; devices by the Controller only via their ability to reach given goals (e.g. temperature and luminosity); the user sees the smart home via its ability to reach comfort. This facilitates the modelling and management of each level: e.g., the Controller uses knowledge on how well devices achieve goals and not on how they achieve them. This also allows each holon to reach its goals via a specific internal organisation, favouring diversity and local optimisation.

Progressive reactivity is tuned so that higher-level self-* loops (e.g. user updating G_{Cmf}) are slower than intermediate ones (e.g. translation of G_{Cmf} to G_R; and rule enforcement), which are in turn slower than lower-level ones (e.g. device controls). Indeed, if the user changed G_{Cmf}, based on perceived temperature, faster than thermostats could reach a comfortable temperature, then oscillations may occur. Also, when devices of known types (e.g. temperature or luminosity regulation) join or leave the home, only the Controller reacts to take them into account (detect and manage them). The rest of the system remains unchanged, unless devices cause changes that are visible at the aggregate level, like breaking the rules.



1
Comfort
2
Comfort Rules
3
Rule Execution
4
5
Prosumption
Cost
Smart-Home holon
9
Eval Comfort
8
Eval Comfort
7
Eval. Rules Conformance
Comfort Indicators Measures
6
Controller
Heater
Lamp
Comfort Solver: Find Comfort to Rules Mapping (Transform)
Rule Manager: Return Rewards / Sanctions
Production system: Control device prosumption according to rules

Fig. 3. Multi-level transformation of goals to rules and actions

B. *Multi-goal conflict within a smart home*

1) Overview and selected scenario: this use case introduces an additional goal from an external stakeholder (e.g. grid manager), creating a conflict with an internal goal. Concretely, in case study A, this can represent power prosumption rules that conflict with the comfort rules. This is generalised here to external and internal rules, showing *where* the conflict is addressed from an architectural outlook, to produce a set of coherent rules for the Rule Manager. The same applies to cases with multiple (external and internal) conflicting goals; at different holonic levels. Figure 4 illustrates the two goals as rule inputs to the same Multi-goal Manager holon and the self-* process that resolves them.

2) Goal-based interactions & Holonic Properties: Complexity from simplicity is reached by inserting this new holon between the Comfort Solver and the Rule Manager (in case

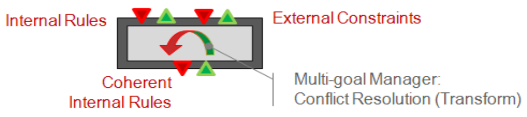


Fig. 4. Conflict resolution between internal and external goals

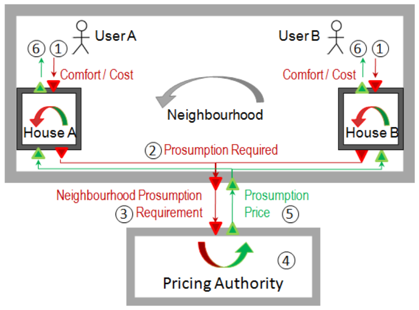


Fig. 5. Top-down facilitation of bottom-up coordination

A). Depending on its complexity, the Multi-goal Manager can be implemented as a monolithic algorithm with/out self-* capabilities, various design patterns [10] or a holonic system.

Semi-isolation ensures that the Multi-goal Manager only has to deal with conflicts inherent in the predefined types of its input goals and hence can optimise for this limited domain.

Abstraction makes this holon (re)usable as a conflict-resolution function, irrespectively of its internal design.

Progressive reactivity should be tuned so that the conflict-resolution implementation reacts to changes in the conflicting goal inputs faster than these can change; and slower than the lower-level holon to which the coherent goals are sent.

C. Market-oriented energy distribution

1) Overview and selected scenario: this use case increases abstraction to the neighbourhood level, where smart houses are mere prosumers, and illustrates a market-oriented approach for energy distribution via offer-and-demand price regulations. This represents a top-down goal definition and enforcement, via price regulation, which can impact smart home internals by introducing new conflicts (case B). This triggers bottom-up adaptation in overall prosumption, which can in turn readjust the prices (yoyo effect). It is based on an example solution from the smart grid literature (i.e. PowerMatcher [23]).

Figure 5 depicts a specific scenario within this case study: each house is requested to reach a user-defined comfort goal (1). We model each house as a holon, together integrated into a neighbourhood supra-holon. The smart houses (i.e. House A and B) translate their comfort goals into prosumption requirements (2), as in case study A. The neighbourhood holon aggregates these and forwards the result to a Pricing Authority (3), which calculates global energy prices (4), which are fed-back to the houses (5) and to the users (6). Each house (internal self-* loop) decides to buy/sell energy at the given price or to update requirements; users may update comfort goals, as before. Afterwards the actual prosumptions are performed, measured and balanced, if needed, from the external grid – the overall system goal is grid stability (not shown for simplicity).

2) Goal-based interactions & Holonic Properties: Complexity from simplicity is reached by integrating self-* prosumers (structured as in A) and regulating their behaviour to meet grid- and house-level goals. *Semi-isolation* is achieved as smart home internals only interact with the environment via energy prosumptions, bidding requests and prices; their internal self-* policies do not depend directly on those of other households, only via the global prices. Similarly, the Pricing Authority's internal strategies can optimise its goals (e.g. profit), based on prosumption requests, while abiding legal regulations. *Abstraction* is achieved by representing each home, and neighbourhood, only via their aggregate bidding requests and resulting prosumptions. This facilitates the management of large numbers of diverse households, which may join or leave the grid without impacting the global management scheme. *Progressive reactivity* is tuned so that the times of the bidding and price-setting self-* processes, and of the actual prosumption, amount to sufficiently small intervals (0.5s) for allowing grid controllers to react (i.e. to achieve balance).

D. Self-governing energy commons

1) Overview and selected scenario: this case study discusses an alternative to energy sharing based on self-governance. It shows how global goals, e.g. fairness, can be defined by community members (bottom-up), to balance individual goals, e.g. everyone wants to appear better off than their neighbours, yet also avoid race conditions leading to resource depletion. Global goals can then be formalised as rules to regulate member behaviour (top-down) [8]. Evaluations can then lead to redefining collective goals and rules (yoyo).

Figure 6 illustrates the process for two users (the multi-layer pattern for translating goals to rules is similar to case A and hence simplified here). Each user u in the shared neighbourhood n wishes to have more or at least as much comfort as their neighbours (at every instant t): $G_u = (better, homeu, t)$, (1). They soon realise this leads to escalating conflicts and hence negotiate (2) to reach equal comfort over medium intervals m, via a collective goal: $G_n = (fairness, n, m)$, (3). G_n is translated into prosumption rules: $G_{PR} = (rulesPR, n, t)$, (4), and enforced into each household (5). Conformance to rules is then assessed (6) and rewards / sanctions distributed. The effectiveness of the rules to reach the fairness goal is assessed (7); and the rules updated accordingly. Fairness evaluations (based on house comfort measurements) are made available to the collective (8). Finally, each user evaluates their position with respect to the overall fairness (9); and may renegotiate (step 2); or leave the collective.

2) Goal-based interactions & Holonic Properties: Complexity from simplicity occurs here by integrating smart houses within a self-governing community, with multiple goals, where

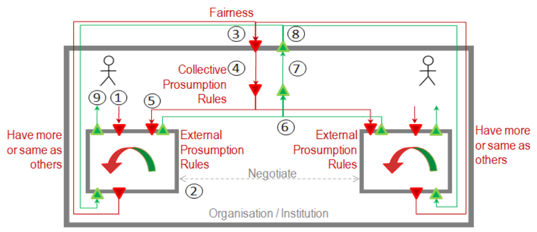


Fig. 6. Bottom-up collective goal definition, enforced top-down

houses are already self-* systems of systems (A). Semi-isolation is achieved, firstly, by identifying the community members that share the collective goal; and secondly, by having external authorities allow the community to self-organise and self-govern, without interfering in their local regulations - as long as they meet external policies and norms, e.g. constitutional laws. If individual goals evolve and are no longer represented by the collective goal, the community holon may be dismantled, yet the smart homes remain.

Abstraction is achieved at the collective level by modelling each house only via its aggregate energy consumption and appropriation, and its conformance to the prosumption rules. The translation of the fairness goal into rules, as well as their enforcement and monitoring, can be achieved by member representatives, by electronic institutions, or both - yet this is not visible at the presented abstraction level. Also, individual users see the entire collective only via its fairness indicator, without access to details of other house profiles. At a higher abstraction level, different communities in the grid may self-organise based on different rules, resulting in diversity and plurality across the integrated community grid system.

Progressive reactivity should be tuned so that member prosumptions are faster than the evaluation of their rule conformance; which is faster than the evaluation (and update) of prosumption rules; faster than the fairness goal management.

VII. Related Work

We have already discussed related work on holonic systems and goal-oriented engineering when introducing these in sections III and IV. Here we focus on their current use in engineered systems. This is a broad subject and we can only focus here on the topics most directly linked to our proposal - component and service oriented architectures; systems of systems; multi-agent systems; and domain-specific applications.

Component oriented technologies aim to develop systems based on reusable units of composition (e.g. CCM, EJB, .NET, Fractal). Components are defined and assembled based on low-level provided and required interfaces; and encapsulated into containers to mediate interactions and separate functional from non-functional logic. Typically, all system components belong to a single stakeholder who assembles them (mostly offline) to meet technical requirements. ***Service oriented approaches*** add dynamic service discovery, binding and sometimes Service Level Agreement negotiations among different stakeholders (e.g. Web services, iPOJO[2], Spring[3]). Still, integration relies on low-level interface-matching and there's little support for dealing with failed compositions (that do not reach goals). At larger scales, ***Systems of Systems (SoS)*** [25] require further support for multi-authority, multi-goal, heterogeneity, scalability and local adaptability. For instance, [26] propose a holon-oriented approach for systematic composition, where systems (holons) are specified via services and properties they offer and require. Then they focus on a concrete *code-generation approach* for system composition (offline and online).

[2]https://felix.apache.org/documentation/subprojects/apache-felix-ipojo.html
[3]https://spring.io

Holonic multi-agent systems (HMAS) have been proposed to manage MAS complexity [17], [18], [19], [27]. MAS are goal-oriented, yet typically couple goal-definitions to the organisations that fulfil them; even if agents are then mapped dynamically to organisation roles. In HMAS roles can be played by entire agent organisations, recursively. A central agent (gatekeeper or head) represents each level when exchanging with organisations at other levels. We are unaware of more generic proposals allowing for decentralised representation. In MAS for ***problem solving***, hierarchical planning [24] enables planners to concentrate on major decisions first (higher-level holons), and elaborate their details separately, later on (lower-level holons). This is similar to [15], where Wrappings decouple problem definitions from the code selected to solve them; and each Wrapping can become a new problem, recursively.

Finally, ***application-specific holonic designs***, featuring different degrees of the properties highlighted here, have been proposed for various areas, including traffic control [28], manufacturing [29], [30] and smart grids [31], [32], [7]. While they provide valuable experience reports they are ad-hoc application-specific solutions offering little reusable support.

While these approaches are compatible with our proposal, none of them alone offers all necessary concepts for a reusable self-integration model for complex systems. We select the most useful features form each and merge them into a comprehensive model: component-provided and -required interfaces merged with agent goal-orientation to increase abstraction and diversity support; dynamic service binding merged with goal-oriented self-* mechanisms for managing failed compositions; specialised component containers as means of achieving holonic semi-isolation principles; and, additional holonic-inspired considerations on the dynamics of multi-level self-* systems (limited support from other areas). New technologies must be built on top of agent and service-oriented platforms to provide reusable support for higher-level paradigms and functions for dynamic (self-)integration – e.g. goal specifications and operations; goal-oriented self-integration mechanisms; and cross-level dynamics analysis and tuning facilities.

VIII. Conclusions and Future Work

This work was motivated by the well-observed trends in ICT evolution from mainframe computing to Internet of Things and "disappeared" computer (where computing components are so closely interwoven with the social fabric that they become undetectable); from batch computing to cyber-physical and socio-technical systems; and from functional and non-functional (QoS) to super-functional (e.g. loyalty and fairness) requirements. These trends have brought about the necessity to rethink the principles of system integration, and furthermore, to bring integration into the runtime and automate it.

The paper proposed a novel (self-)integration paradigm, based on a conceptual model, including: 1) goals as high-level entities for modelling, in an uniform way, the reference points of highly-diverse self-* systems, and of their interrelations; 2) holonic design principles highlighting the key structural and dynamic properties of viable self-* systems of systems;

and 3) an abstract, decentralised (self-)integration process for generating goal-oriented holonic systems, capable of meeting stakeholder goals, in complex environments.

The conceptual model extracts and generalises existing specifications, models and design principles from related literature and our previous work. We did not aim to provide a complete, concrete solution to the challenging question of how to self-integrate systems of systems that achieve their goals. This is a broad multi-faceted topic requiring extensive future research. Instead, the conceptual model proposed here indicates the generic form and properties that the (self-)integration process and the systems it produces should take. The purpose of the goal-oriented holonic model is to provide a viable foundation to future methodologies, tools and technologies for analysing, developing and controlling self-integrating systems of (self-*) systems. The core contributions are:

- Identifying viability-enhancing design principles and their key role for engineering complex systems (motivated by holonic systems observed in nature);
- Showing how using goals and goal-oriented relations as first class modelling entities can help represent and (self-)integrate highly diverse systems in a uniform manner;
- Merging holonic principles with goal-orientation to offer a novel (self-)integration paradigm and high-level process for reaching holonic properties in engineered systems.
- Illustrating and supporting the viability of the proposed conceptual model via four sample case studies from previous work on decentralised community energy systems.

In future work we aim address membrane designs; formalisation of goals and transformations; conflict resolution; and the dynamics of cross-level goal-resolution processes (yoyo). Many other challenges remain, including risk analysis, specific formalisms, self-integration algorithms, learning and security. We believe that the proposed conceptual model offers a foundation for integrating such contributions and support future methodologies, tools and technologies for designing the next generation of socio-technical and cyber-physical systems.

References

[1] F. Zambonelli et al., *On Self-Adaptation, Self-Expression, and Self-Awareness in Autonomic Service Component Ensembles*, Intl. Cnf on Self-Adaptive and Self-Organizing Systems Workshops, 2011.

[2] G. Cabri and F. Zambonelli, *Towards Self-Aware and Self-Composing Services*, "The Computer After Me", Imperial College Press, 2014

[3] H. A. Simon, *The architecture of complexity*, In Proc. of the American Philosophical Society, 1962.

[4] A. Koestler, *The Ghost in the Machine*, Hutchinson Publisher, 1967.

[5] H. A. Simon, *The Sciences Of The Artificial*, MIT Press, Cambridge, Mass, 1st edition, 1969.

[6] G. Jagers op Akkerhuis, *The operator hierarchy, a chain of closures linking matter, life and artificial intelligence*, Ph.D. Dissertation, 2010.

[7] S. Frey, A. Diaconescu, D. Menga and I. Demeure, *A Generic Holonic Control Architecture for Heterogeneous Multi-Scale and Multi-Objective Smart Micro-Grids*, ACM Transactions on Autonomous and Adaptive Systems (TAAS), Volume 10, Issue 2, June 2015, pp 9:1-9:21

[8] A. Diaconescu and J. Pitt, *Holonic Institutions for Multi-Scale Polycentric Self-Governance*, in Coordination, Organizations, Institutions, and Norms in Agent Systems X, 9372 (LNCS), 2015, 19-35

[9] B. Debbabi, A. Diaconescu and P. Lalanda, *Controlling self-organising software applications with archetypes*, Intl Cnf on Self-Adaptive and Self-Organizing Systems (SASO), Lyon, France, 2012

[10] S. Frey, A. Diaconescu, I. Demeure, *Architectural Integration Patterns for Autonomic Management Systems*, Intl Cnf and Wrkps on the Engineering of Autonomic and Autonomous Systems (EASe), 2012

[11] Y. Maurel, P. Lalanda and A. Diaconescu, *Towards a service-oriented component model for autonomic management*, Intl Cnf on Services Computing (SCC), Washington, DC, USA, 2011

[12] P. Valckenaers, H. Van Brussel and T. Holvoet, *Fundamentals of Holonic Systems and Their Implications for Self-Adaptive and Self-Organizing Systems*, SASO Workshops. IEEE CS, 168173, 2008.

[13] M. Amoui, M. Derakhshanmanesh, J. Ebert and L. Tahvildari, *Achieving dynamic adaptation via management and interpretation of runtime models*, Systems and Software, 85(12), pp 2720 2737, 2012.

[14] S. Tomforde and C. Müller-Schloer, *Incremental Design of Adaptive Systems*, Jrnl of Ambient Intelligence & Smart Environments, 1, 2013

[15] C. Landauer, K. Bellman, *New architectures for constructed complex systems*, Applied Mathematics&Computation, 120(1-3), 2001, 149163

[16] A. van Lamsweerde, *Goal-oriented requirements engineering: a guided tour*, Intl. Symp. on Requirements Engineering, 2001, 249-262

[17] J. Jiang, V. Dignum and Y-H Tan, *An Agent-Based Inter-organizational Collaboration Framework: OperA+*, Coordination, Organizations, Institutions, and Norms in Agent System VII, 2012, 7254, LNCS, 58-74

[18] K. Fischer, M. Schillo and J. Siekmann, *Holonic Multiagent Systems: A Foundation for Organisation of Multiagent Systems*, HoloMAS, 2003

[19] S. Rodrigues et al. *An Analysis and Design Concept for Self-organization in Holonic Multi-agent Systems*, in Engineering Self-Organising Systems, (LNCS), Vol. 4335. Springer, 2006, 1527.

[20] E. Yu, P. Giorgini and N. Maiden and J. Mylopoulos, *Social Modeling for Requirements Engineering: An Introduction*, in Social Modeling for Requirements Engineering, MIT Press, 2010

[21] A. S. Rao and M. P. George, *Modeling rational agents within a BDI architecture*, Intl. Cnf. on Principles of Knowledge Representation and Reasoning, CA, US, 1991, pp. 201-203, Morgan Kaufmann publishers

[22] S. Kounev, X. Zhu, J. O. Kephart and M. Kwiatkowska, *Model-driven Algorithms and Architectures for Self-Aware Computing Systems*, Report from Dagstuhl Seminar 15041, Vol. 5, Iss. 1, pp. 164196, 2015

[23] J. K. Kok, C. J. Warmer and I. G. Kamphuis, *PowerMatcher: Multiagent Control in the Electricity Infrastructure*, Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), 2005

[24] L. Nolle, K. Wong, A. Hopgood, *DARBS: A Distributed Blackboard System*, Research and Development in Intelligent Systems XVIII, M. Bramer, F. Coenen, A. Preece, Springer London, 161170, 2002

[25] A. Sage and C. Cuppan, *On the Systems Engineering and Management of Systems of Systems and Federations of Systems*, Information-Knowledge-Systems Management Journal. 2(4), 2001.

[26] G. Blair, et al, *Holons: towards a systematic approach to composing systems of systems*, Workshop on Adaptive and Reflective Middleware (ARM), Vancouver, BC, CA, 2015, (6 pages).

[27] M. Cossentino, S. Galland, N. Gaud, V. Hilaire and A. Kouka, *An organisational approach to engineer emergence within holarchies*, Int. Journ. of Agent Oriented Software Engineering, 4, 2010.

[28] K. Fischer, *Holonic Multiagent Systems Theory and Applications*, in Progress in Artificial Intelligence, Pedro Barahona and JosJ. Alferes (Eds.). LNCS, Vol. 1695. Springer Berlin Heidelberg, 3448, 1999.

[29] J. Christensen, *Holonic manufacturing systems - initial architecture and standard directions*. In Proc. of the 1st European Conference on Holonic Manufacturing Systems, Hannover, December 1994.

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

[31] J. Lassig, B. Satzger and O. Kramer, *Hierarchically Structured Energy Markets as Novel Smart Grid Control Approach*, In KI 2011: Advances in Artificial Intelligence. LNCS, Vol. 7006, 2011.

[32] A. Schiendorfer, J-P. Steghofer and W. Reif, *Synthesis and Abstraction of Constraint Models for Hierarchical Resource Allocation Problems*, Intl Cnf on Agents and Artificial Intelligence (ICAART), 2, 2014.

[33] S. Edenhofer et al., *Trust Communities: An Open, Self-Organised Social Infrastructure of Autonomous Agents*, Book chapter in Trustworthy Open Self-Organising Systems, Autonomic Systems series, Vol. 7, 2016, Springer International Publishing, pp 125-150