Hierarchical Self-awareness and Authority for Scalable Self-integrating Systems

Ada Diaconescu
Télécom ParisTech, LTCI, IMT
ada.diaconescu@telecom-paristech.fr

Barry Porter and Roberto Rodrigues
Lancaster University
b.f.porter@lancaster.ac.uk

Evangelos Pournaras
ETH Zurich
epournaras@ethz.ch

***Abstract*—System self-integration from open sets of components provides the basis for open adaptability to unpredictable environments. *Hierarchical architectures* are essential for enabling such systems to *scale*, as they allow to compromise between processing detailed knowledge in parallel and coordinating parallel processes from a more abstract viewpoint; recursively. This position paper aims to bring to the fore the following key design aspect of such hierarchical systems: how should the *authority* of decision and action be assigned across hierarchical levels, with respect to the *self-awareness* capabilities of these levels? The difficulty lays in that all levels lack knowledge, which may be key to certain decisions, because lower levels have detailed knowledge but within a narrow scope (good for local customisation), and higher levels have a broader scope but no details (good for global coordination). We highlight the most obvious authority schemes available and discuss their advantages and shortcomings: top-down, bottom-up, and iterative (yoyo). We discuss three detailed application examples from our previous work on hierarchical systems, pointing-out the knowledge and authority schemes employed and the possible alternatives. This provides a basis for offering system designers the necessary understanding and tools for taking the appropriate decisions with respect to the distribution of self-awareness capabilities and authority of decision / action across hierarchical system levels.**

***Index Terms*—hierarchy, scalability, self-awareness, knowledge management, authority, self-integrating systems, self-optimisation, smart grids, poly-centric institutions.**

I. Introduction

Self-integration helps solve complex problems automatically by dividing them into smaller sub-problems and integrating ensuing partial solutions into global solutions. In open systems, such as smart grids, smart cities and poly-centric e-institutions, self-integration can ensure open-ended adaptability – by assembling an ever evolving set of components – for dealing with unpredictable environments. This includes the runtime discovery, selection, configuration, interconnection, coordination and management of system components.

Self-integration is a *control problem* – deciding which components to integrate in which contexts for meeting stakeholder *goals* [1]–[3]. Depending on the adaptation capabilities desired, control processes require a certain level of *self-awareness* [4]: acquiring *knowledge* about the system and its environment, and *reasoning* about that knowledge so as to *act* and achieve stakeholder *goals* (Fig. 1-a, Cf. section II).

In large-scale systems operating in complex environments, self-aware controllers must acquire and process increasingly large amounts of knowledge, which can become problematic

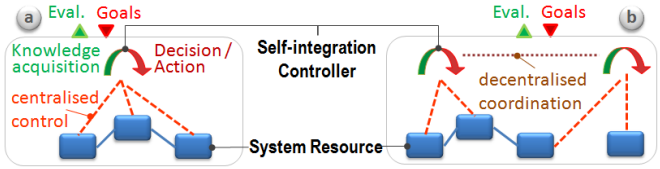


Fig. 1: (a) Centralised vs. (b) Decentralised Control

in terms of delays and resource consumption. Hence, self-integration controllers are often decentralised [3], [5], [6] (Fig. 1-b), to *parallelise* knowledge acquisition and processing (lowering execution times and distributing resource usage). Still, ensuring system coherence and optimisation requires *coordination* among such decentralised controllers. Moreover, each controller might self-adapt and evolve over time.

As system *scale* increases, the coordination of self-adaptive controllers becomes a large-scale self-integration problem in itself (where resources are local controllers). Often, this cannot be solved by mere local communication amongst neighbouring controllers, due to large convergence delays and communication overheads. Hence, the self-integration of controllers must, itself, be controlled and coordinated at a higher abstraction level (meta-control); which may, again, become a large-scale problem. This results in a recursive design process of controller division and coordination leading to *hierarchical* control architectures (Fig. 2) [7], [8]. The addition of control levels stops when the highest level is sufficiently simple, or small-scale, to be implemented via a fully-centralised or -decentralised controller (Fig. 1-a and -b, respectively).

Hence, hierarchical architectures enable control scalability by incrementally simplifying control complexity via *abstraction*, or loss of information, from lower control levels towards higher control levels. This means that higher control levels acquire knowledge from increasingly *wider scopes* (i.e. larger domains of visibility over managed resources), but with *lesser*

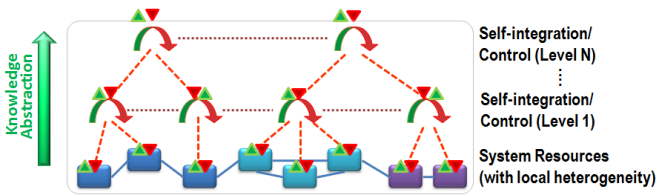


Fig. 2: Overview of Hierarchic Control

details, compared to lower control levels. Lower levels, in turn, have *narrower scopes* and hence can retain *more detailed* local knowledge (to reach the same complexity).

This implies that higher control levels take decisions with broader action scopes (i.e. impacting larger system parts), based on knowledge that is broader yet less detailed, compared to lower control levels. Such decisions are well-suited for global optimisation and coordination, yet cannot provide guarantees about the local optimisation or even the viability of decided adaptations (since they lack local details that may be essential). Hence, there is lack of knowledge at all levels: the lower levels have the details but only for a limited scope; and the higher levels have the overall view but no details. When it comes to decision-making and actions, *a trade-off must be made between knowledge scope and knowledge detail*.

This situation often leads to *conflicts* between local optimisation and global coordination in large-scale systems, potentially jeopardising the viability of the integrated system. This is particularly problematic in systems integrating conflicting local interests, typically stemming from high heterogeneity of local environments (e.g. smart grid and smart city federations).

The conflict potential of this inherent self-awareness trade-off among control levels raises serious questions about the *authority* assignment (Cf. sec. II) within control hierarchies. Most hierarchical systems adopt **top-down** authority schemes, with higher level decisions overriding or constraining lower-level decisions (e.g. most human organisations and institutions). This paper aims to investigates alternative approaches, discussing **bottom-up** authority schemes (i.e. where high-level decisions are mere recommendations); and **yoyo** decision processes (i.e. iterative top-down *and* bottom-up decisions).

We illustrate such alternatives in socio-technical systems via three use cases that employ hierarchic architectures: component deployment in a distributed environment (IV-A); self-management of sharing economies (IV-B); and, rule-management in poly-centric e-institutions (IV-C).

The purpose of this analysis is to raise awareness of the advantages and shortcomings of hierarchical control schemes, in terms of *knowledge* and *decision* distribution, and to highlight the implications of this on alternative *authority* schemes across hierarchical levels. Future work will analyse further hirarchical system examples and aim to extract generic principles and recommendations for control system designs.

II. Hierarchical Self-awareness and Authority

A. Knowledge Abstraction and Authority Legitimacy

Self-aware computing systems are control systems that must acquire *knowledge* (e.g. models) about controlled resources and their environment; and *reason* on this knowledge to decide on control *actions* for reaching stakeholder *goals* [4]. Hence, the focus of self-awareness (compared to other self-* functions) is on *knowledge* management, and its impacts on reasoning and actions. Knowledge acquisition can rely on runtime monitoring, learning and external sources. Within hierarchical architectures, knowledge is impacted via loss of information from lower to higher levels (Cf. sec. I).

Hence, hierarchies enable self-aware systems to scale by recursively abstracting away the details of information that is processed when taking decisions over increasingly large system scopes (Fig. 2). Other advantages include the parallelisation, reusability and robustness of partial control solutions (self-integrating *sub*-systems) for creating complete solutions (entire self-integrating systems) [1] – out of this paper's scope.

This cross-level loss of information raises concerns about the *legitimacy of authority* at higher hierarchical levels. *Authority*, rooted in the Latin *auctoritas*, generally referred to one's ability to exercise influence, impose their will, or act. In recent governance systems, it has been equated with *legitimate power* – where power refers to the ability to perform an act, and legitimacy to the justification and recognition of that power as acceptable, by both its holders and its subjects (e.g. by law, rationality, tradition, or higher/divine power).

Authority is particularly important when *conflicts* can arise among stakeholders [2], [9], in terms of (incompatible decisions leading to) incompatible actions. In this sense, authority can be interpreted as a form of *priority* for resolving conflicting (decisions and) actions. Such priority can be instantiated in various manners, e.g. the right to veto or override decisions taken by organisms with less authority; or, the right to define constraints on the decisions, and associated processes, of such lesser organisms. Authority has been extensively studied in various areas – e.g. [10], [11], [12] – out of the paper's scope.

Within the limited context of hierarchical self-aware socio-technical systems, we consider authority to be some form of priority over decisions and actions, which comes into play for resolving conflicts between cross-level controllers (at different abstraction levels). Priority assignments to hierarchy levels can be set by design (and tuned via static or runtime configurations), and potentially updated at runtime. Legitimacy here comes from the acceptance of the system's design, update processes and provided functions, by users and regulatory institutions. This general view of authority may have to be refined, or defined more formally, case-by-case.

B. Hierarchical Authority Schemes

Considering the necessary trade-off between knowledge scope and detail in hierarchical systems (Cf. sec. I) the question of authority assignments for taking decisions and actions within hierarchies becomes essential. E.g., in which cases should global decisions (higher-level) override local ones (lower-level), to help coordinate sub-systems for ensuring overall coherence and optimisation? And, in which cases should global decisions be mere guidelines for local optimisation and decentralised coordination among sub-systems? We argue that the answer to this question depends on the targeted system and its environment, and hence that it is important to understand and consider key principles for selecting a suitable answer, case-by-case. We aim here to identify the main types of authority distribution schemes and to provide concrete examples for illustrating their benefits and shortcomings.

One approach employed rather widely in human-made systems and organisations is the **top-down scheme**, where higher

levels have authority over lower levels (even if lower levels do retain some autonomy). Here, global decisions constrain, or override, local decisions – e.g. a federal governments decree overrides provincial laws; global trade agreements constrain local markets; global resource schedulers limit local ones. This does not preclude higher-levels from getting feedback from lower-levels, and adjusting their decisions accordingly. Still, the higher-levels have the 'final word' – i.e. priority, and the means to enforce it. The advantage here is that global knowledge and decisions help overall coordination, coherence and optimisation, which are essential for viability in heterogeneous systems. At the same time, global decisions based on global knowledge may also overlook key local details, and hence impose unsustainable conditions on certain sub-systems. When multiple sub-systems are affected, the stability and sustainability of the entire system may be jeopardised.

An alternative scheme is **bottom-up** authority assignment. Here, global knowledge, as well as coordination or optimisation plans, represent mere indicators, or recommendations, for low-level controllers, which retain complete autonomy (over their decisions and actions). A well-known example here is that of free markets: prices represent signals carrying aggregate knowledge, which guide individual behaviours and implicitly their coordination [13], [14]. Surely, since humans (advanced self-aware agents) can find innovative ways to extract new kinds of information from such generic signals (e.g. interpreting price changes as predictors of the intentions or needs of others) they can exploit such knowledge for local benefits in ways that can destabilise the economic system (e.g. herding/racing investments leading to economic bubbles and crashes) [15]. This may indeed suggest that in the context of human societies the theory of decentralised economy should not necessarily translate into a theory of unregulated market-oriented governance; but rather allow for certain stabilising regulations[1] (Cf. yoyo scheme below).

To avoid this shortcoming, such bottom-up approach may apply for systems where self-ware agents feature limited innovation capabilities or limited self-interests. For instance, in ant colonies, individuals take local decisions based on their own knowledge (local) and on aggregate knowledge (global) stored in the local environment (e.g. pheromone traces). Yet, there is no master decision taker using global knowledge to control everyone – indeed, some ants may explore outside the pheromone trails. Similarly, within single organisms (e.g. humans), central control (central nervous system) may take global decisions based on global knowledge (e.g. where to go next), but autonomic sub-systems cannot be directly controlled by such decisions (e.g. one can hardly stop their heart from beating just by thinking about it – and quite luckily so).

Hence, this approach may apply well to technical systems, where agents are unable to innovate beyond the learning boundaries they were designed for. A notable example is the *blackboard*-oriented approach in multi-agent systems [16]. Here, a publicly available blackboard (centralised or hierarchical) computes and publishes global knowledge based on local contributions from individual agents. Each agent can then use the global knowledge to tune their local behaviour. This approach has also been used in smart grid systems to coordinate decentralised prosumers (local behaviour) via global aggregate knowledge (frequency of the electric grid, which all prosumers impact and have access to) [17].

The two alternative schemes discussed above can be combined into a variety of **iterative** decision processing protocols, or **yoyo** schemes. Here, decisions may propagate top-down initially, but then enable feedback and decision amendments to propagate bottom-up, leading to potential changes in the original decisions and the re-distribution of these in a top-down manner. The process can be repeated iteratively (though sometimes one or few iterations may suffice (e.g. IV-B); may start with a bottom-up phase; and may take diverse yoyo paths through the hierarchy. In open environments, sub-systems that were not designed to integrate with each-other may feature incompatible authority schemes – e.g. one assuming top-down authority and the other bottom-up authority. This may happen for instance when human organisations merge and the legitimacy of a federative authority coordinating them is challenged. Future work will further investigate such cases.

We illustrate the above schemes via three hierarchical applications – in sec. IV – highlighting the use of knowledge abstraction and authority schemes, and discussing alternatives.

III. Related Work

The problem of assigning knowledge (or self-awareness) and authority (or decision priority) in distributed systems has been studied extensively, in various research domains, as these two aspects are essential for coordinating and resolving conflicts amongst distributed entities. Different solutions were identified in technical systems, such as organisation types for multi-agent systems [18]; design patterns for conflict resolution in autonomic systems [9]; and reusable architectures for self-organising systems [19]. Swarm intelligence and other decentralised approaches do manage scalability issues (without hierarchical organisation), yet we are unaware of any such solution that scales to millions of components, which could be highly distributed and heterogeneous. Reusable designs for scalable self-* systems notably include hierarchical or holonic architectures, such as generic Holonic MAS (HMAS) [20] and goal-oriented holarchies [2]; as well as application-specific solutions (Cf sec. IV). Within the realm of human societies, distinct organisation types were set in place to deal with political [21] and economic [22] [13] issues. These approaches have different ways of dealing with knowledge and authority (defining these aspects either implicitly or explicitly).

For instance, in HMAS [20], each agent set in a hierarchy level is represented by one agent in the level above, recursively. This implies that each agent representative acquires knowledge from the agents it represents (abstraction); has the authority to negotiate coordination issues with other agent representatives at the higher level; and has decision / action authority over

[1]This paper does *not* aim to take any position on the subject, but merely to discuss it as a theoretical example

the coordination of agents that it represents at the lower level. Several alternatives can be considered here, e.g. where the decisions of representative agents are mere suggestions for coordinating lower-level agents, which maintain full autonomy.

In the GoTT generic architecture [2], each hierarchical level is defined as an abstraction for the higher level, while leaving open the implementation details (e.g. aggregation function, de/centralised design) and the authority of decision/action. GoTT also emphasises the importance of partial-isolation among self-* sub-systems, and of timing differences between self-* processes at subsequent hierarchical levels – both impacting the behaviour of the knowledge abstraction function.

In the context of resource sharing in human communities, Elinor Ostrom has identified eight core design principles [22] for successful institutions that can ensure the sustainability of common resources (and avoid the tragedy of the commons). Importantly, while the first six principles (P1-6) concern the design of a single institution, the remaining two principles address scalability issues: (P8) defines encapsulated institutions (or hierarchies); and (P7) specifies the necessary balance of authority between higher and lower hierarchy levels (where any community must have the right to self-organise internally, hence avoiding complete interference from higher authorities).

P7 can be interpreted and implemented in various ways. One option is to have higher-level authorities *constrain* the scope within which lower-level communities can self-organise – if taken to an extreme, such constraints may override the right to self-organise all-together, or reduce it to an extent where it becomes irrelevant. Another option is to have higher authorities play merely consultative roles, and keep full autonomy at the community level – this, in turn, may jeopardise coordination amongst communities and risk global tragedy of the commons (e.g. difficulty to reach international agreements on issues such as global pollution). Intermediate options may be optimal, with iterative top-down and bottom-up feedback (yoyo) being codified as a formal process between embedded institutions and managed like any other institution rule (P1-6).

Further relevant examples shall be discussed in section IV. A comprehensive study of such approaches is beyond this paper's scope. This position paper aims to assert the key role of this particular design aspect – the assignment of authority to decide/act, with respect to the distribution of knowledge, considering the inherent limitation of knowledge availability.

While these aspects are relevant within any organisation (e.g. centralised or decentralised) we focus here on hierarchical control systems, and more precisely on the knowledge and authority distribution across hierarchical levels. This is because hierarchies are employed in large-scale systems to deal with the problem of knowledge management given limited resources [7], [8], and system designers may adopt them while overlooking their inherent knowledge-related limitations, the consequent compromises required, and the necessary implications on decision authority schemes. These considerations become particularly important in open, self-integrating, self-optimising systems, where high dynamicity raises additional knowledge management concerns.

IV. Use Cases

A. Hierarchical Component Assembly in Distributed Systems

This use case concerns a distributed component self-assembly system [23]. It aims to identify a composition of components, sourced from a large library, which achieves a desired task, and then continuously find higher-performance variations by adapting to alternative compositions which better fit the current deployment environment. Adaptations come in three forms: (i) a *local adaptation* to an alternative component implementing a given concept (such as an alternative sorting algorithm or buffering strategy); (ii) *relocating* components to other available host machines, where the additional hardware resources of the remote host may result in higher overall performance for I/O-bound tasks; or (iii) *replicating* components across multiple available hosts, with an injected load balancer to distribute load across replicas and a consistency management module to deal with any replicated state. Real-time machine learning observes the current performance of the system via particular metrics of interest, together with observations of the current characteristics of the deployment environment, and applies an explore/exploit strategy which drives the suite of possible adaptations to discover where the higher-performance system designs are for each set of environment conditions encountered.

A specific example is a datacentre software infrastructure (Fig. 3), which receives requests from users and should serve responses as quickly as possible (reduce latency) while using a minimal set of resources (maximise energy efficiency), all while user behaviour is continuously changing to cause stress in different parts of the system [6]. While conceptually simple, in practice this kind of system can involve hundreds of subsystems which are highly challenging even for expert human engineers to design and maintain [23], making it a good target for automated design with runtime learning.

Component-based applications feature directed graph topologies, with nodes being components and links being component dependencies (e.g. required / provided services). Hence, client requests are processed via various *call paths*, starting with an entry component (root) in the dependency graph, then forwarded through component dependencies, recursively, until reaching components with no further dependencies (leaves). A component can have several implementation variants which encode the same behaviour but have different performance characteristics under different deployment conditions; each such variant may feature different dependencies on other components which alter the graph below that component.

In this use case, hierarchy with respect to **knowledge abstraction** occurs in terms of the various performance parameters that are monitored throughout the component graph. That is, components closer to the roots of the directed graph (i.e., higher in client call paths) collect aggregate information about the performance of components closer to the leaves of the graph (i.e., lower in the call path). E.g., the delay of a root component for handling a client request subsumes the delays of all the components that it depends upon for

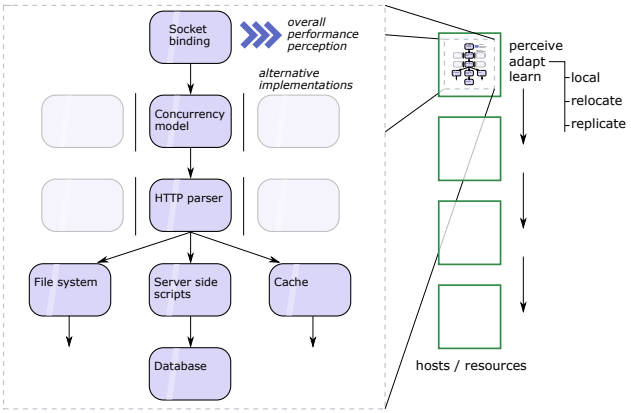


Fig. 3: A self-deploying, self-assembling, self-optimising system, which uses real-time machine learning to make decisions on local adaptations, relocating components to other resources, or replicating components across resources.

handling that request. Similarly, components in a call path can forward monitoring information on their memory and CPU consumption, recursively, from leaf components towards the root. Abstraction here occurs if each component only sends to its parent component the aggregate information of the entire sub-tree of which it is a root – e.g. each component sends the sum of its own consumption and the consumptions collected from the components it depends on.

In terms of **authority** assignment, higher levels of the component graph manage lower levels in cases where the performance of the higher level provides a broader view of the system. The learning agent at a particular host H may therefore choose to offload one of its components (including the dependency sub-graph of that component) to a different host, give that host time to find a 'good' composition for that sub-graph, and then measure the performance difference at H. If this decision made a positive impact, H may stay with it, otherwise H may decide to try something else. This approach does take into account feedback from lower levels by observing performance data; even once a system converges to a stable state, perturbations observed at lower levels due to changes in the deployment environment result in informative feedback for the upper level to make choices. However, all of this feedback is advisory, and the final authority over what to do next rests with the higher levels of the component graph.

As an alternative to this, we may use hybrid authority and knowledge propagation schemes. Consider a scenario in which we have *multiple* self-integrating distributed systems (as above), each of which can only see its own isolated perspective, but where all those systems are operating over shared resources. In addition, we assume that the shared resources themselves have an orthogonal intent or reward system, for example being paid to take computational work. Here, the purely top-down method of authority has the potential to break down into an oscillating or chaotic shared ecosystem: a resource may take on tasks from multiple client systems but in doing so cause each system to observe poor performance as a result of contention, in turn causing those self-* systems to move tasks from that resource to other resources, and so on. In this scenario we can imagine two kinds of authority.

The first is top-down authority, but with resources able to provide a 'soft no' with extended reasons as to why they may be a poor choice, acting as a kind of knowledge conduit between multiple otherwise isolated self-* systems. The higher level of authority may then be able to take this extra knowledge into account in its decision making, after which it can either proceed anyway and ignore the soft no, or take an alternative decision. This implies that higher levels take initial decisions based on abstract knowledge (lacking local resource details), but are then provided with feedback in the form of more detailed knowledge (limited to what is relevant with respect to their initial decision), case-by-case.

The second is hybrid authority, in which a 'hard no' can be given by resources; this may still contain additional useful knowledge for the higher levels, but cannot be overridden. As a consequence of empowering the lower level, there is a risk of 'artificially' claiming that resources are exhausted for any new systems, and could perhaps be even further complemented by a kind of shared knowledge among the resources themselves to coordinate when one of them is willing to take on more work when no other resources are willing to do so.

B. Hierarchical Self-management of Sharing Economies

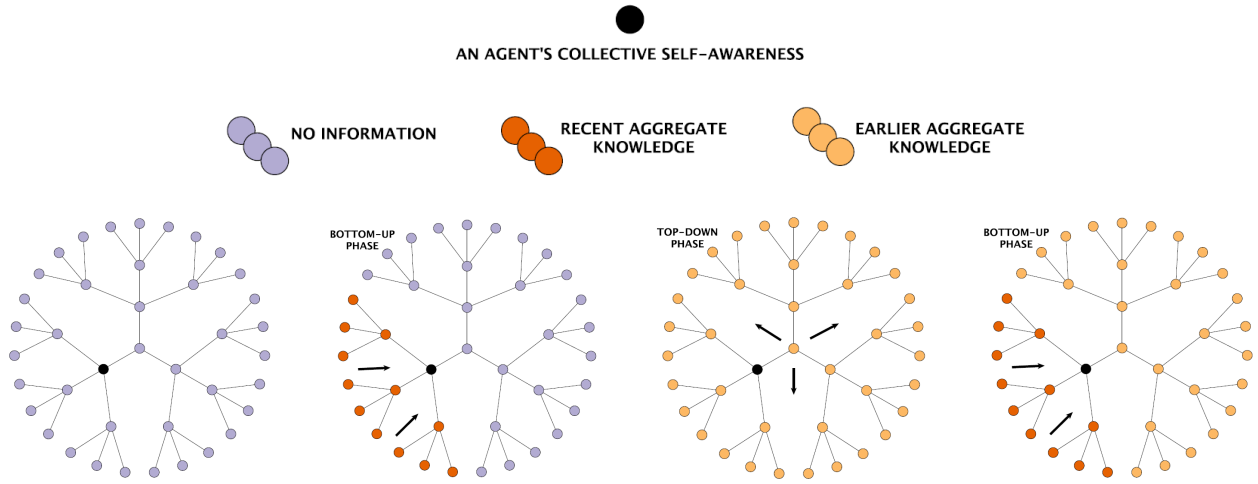
This use-case concerns the scheduling and resource allocation in scenarios of sharing economies within Smart Cities:

- Energy self-management to match supply-demand;
- Charging electric vehicles to match supply-demand;
- Load-balancing of bike sharing stations.

All these different scenarios face the same foundational computational challenge: at a local level, autonomous agents (e.g. individuals using personal assistants in their smart phones; or ubiquitous smart controllers) schedule or allocate their resources (e.g. switch on/off their appliances; plug-in/out their electrical vehicles; or choose stations from which to pick-up or return their bikes). At a global level, these choices have a collective impact on the system reliability and efficiency. For instance, power peaks can cause blackouts. Over/under-loaded parking stations require manual bike relocations, which increase operational costs. When agents choose among autonomously generated *plans* that schedule or allocate their resources, the computational problem of selecting the optimal plan for each agent, so as to minimize a quadratic cost function (e.g. system balancing or matching resources) is a combinatorial optimization problem known to be NP hard.

A generic and fully decentralized hierarchical learning approach has been introduced to address such computational problems: EPOS[2], the *Economic Planning and Optimized Selections* [24], [25]. The deep hierarchical tree structures of EPOS orchestrate the (remote) interactions of agents (nodes) that exchange knowledge in a peer-to-peer fashion via available communication channels (links). The role of the hier-

[2]Available at http://epos-net.org (last accessed: May 2018)



(a) Initialization. No information is available to the agent.

(b) Decision-making is made based on the aggregate decisions of the agents underneath.

(c) An aggregate global solution is found and propagated in the network.

(d) Decision-making is repeated to improve the earlier aggregate global solution.

Fig. 4: The decentralized coordination and learning paradigm of EPOS: Iterative bottom-up and top-down knowledge exchange forms the agents' collective self-awareness based on which iterative coordination and learning is performed. The hierarchical structure aggregates knowledge from the agents underneath (tree branch) and from earlier learning iterations.

archy is twofold: (i) to perform an efficient aggregation of information, i.e. aggregation of the selected plans (bottom-up) – this is in contrast to unstructured networks that require complex mechanisms and a long convergence time for a distributed estimations of aggregates [26]; and, (ii) to perform coordinated decision-making by taking into account the aggregate decisions of the agents lower in the hierarchy (bottom-up), as well as earlier decisions propagated over the tree topology (top-down). Therefore, EPOS draws parallels with back-propagation learning in neural networks, nevertheless, EPOS is a decentralized networked system of autonomous agents. Figure 4 illustrates the coordination concept of EPOS.

With respect to **knowledge abstraction**, EPOS aggregates resource scheduling plans in a bottom-up manner (Fig. 4-b). This means that initially, scheduling decisions are taken at lower hierarchy levels first, and then taken into account (via aggregates) by higher hierarchy levels (bottom-up). Once the aggregate decision-making process reaches the top of the hierarchy, a global plan is formed and propagated back throughout the hierarchy (top-down) – each level adjusts its scheduling plan accordingly (Fig. 4-c). This leads to an **iterative decision-making process (yoyo)** (Fig. 4-d), starting with locally optimised decisions (bottom-up) and adjusting these based on global optimisation feedback (top-down).

Authority-wise, higher levels have priority over lower levels, as lower levels adjust their local plans according to the global optimised plan received from the top. Still, lower levels play, indirectly, a key role in selecting the global optimum, by only sending upwards the aggregate plans that improve the earlier global plan. Furthermore, as detailed below, lower levels may also choose the extent to which they adjust local plans based on global optimisation plans from the top.

EPOS does not require any centralized authority. Therefore, the hierarchy does not impose different agent roles and serves exclusively the computational aspects of the scheduling coordination problem. However, the hierarchy does introduce a priority over the decision-making of the agents. It is shown that changing the positioning of the agents in the hierarchy leads to a different traversal of the optimization space and therefore the agents' topology has an impact on performance [5], [24].

For instance, consider the following extreme scenario of a large industrial consumer positioned at the root of the tree, while all the other agents underneath are smaller residential consumers. The plan selections of the root are performed at the last step of the bottom-up phase and therefore the chances of over-fitting are high, i.e. the larger energy consumption of the industrial consumer has a large oscillatory impact over the aggregate consumption of the residential consumers, hence causing power peaks in the total consumption.

Agents may also have preferences over alternative plans. For instance, turning on or off home appliances at specific times may cause discomfort to residents, impacting their lifestyle and daily activities [27]. Therefore, a selected plan may serve the system-wide objective (e.g. decrease power peaks to avoid blackouts), but may oppose the residents' local preferences. Forcefully having to adhere to the selected plan can undermine agents' autonomy, freedom of choice as well as system trust. To address such cases, EPOS encodes the priority over local or system-wide objectives in a local parameter[3] selected by the residents and incentivized by system operators or third party stakeholders, e.g. power utility companies providing (monetary) rewards to sacrifice comfort for global optimality.

[3]This is the λ parameter that encodes the extent to which agents choose locally preferred plans over globally preferred ones [24].

This parameter biases the algorithm and weights local vs. global objectives. While satisfying the individuals' preferences may not suit certain critical application scenarios (e.g. risk of blackouts in smart grids) it may work well in resource sharing systems where members are willing to pay the extra optimisation costs (e.g. more expensive membership fees for bike rentals, to cover bike relocations between overloaded and under-loaded stations, allowing, in turn, to leave and pick-up bikes at the users' most convenient stations).

This concept can be taken a step further by introducing the dispersion of discomfort among agents as a measure of fairness [27]. Tragedies of the commons can be prevented, for instance, by making sure that in case of a power grid emergency all residents contribute to demand response, i.e. experience equal discomfort.

C. Hierarchical Rule Management in Poly-centric Institutions

This use case concerns the *governance* of socio-technical systems (Fig. 5) – e.g. smart grids, cities, vehicular networks, the Internet of Things, and of People. Governance involves defining and updating *rules*, and evaluating their impacts with respect to high-level *goals*. Goals may be global (e.g. system stability, sustainability and fairness) and local (e.g. individual gains and development). *Rules* aim to constrain the behaviour of system agents (human or technical). To define rules, one must assess the system's current state and dynamics, and predict how rule changes would alter agent behaviours in a way that would bring the system states closer to the goals.

At large scales, the need for *hierarchical governance* stems from the sheer number of resources and agents to consider, as well as from their increasing heterogeneity, and diversity of their environments. Democratic hierarchies are typically implemented via representation, where each governance sub-system at a lower-level is represented by one or several governance agents at the higher-level. In other words, the role of agents at a higher level is to represent the interests (local goals) of the lower-level sub-system which has appointed them (bottom-up). In autocratic hierarchies, the role of agents at higher governance levels is to ensure that top-down rules are enforced successfully into lower-level sub-systems, which are assigned as their responsibility (top-down).

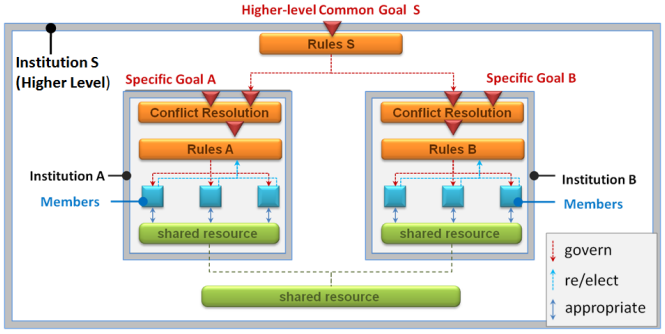


Fig. 5: Hierarchical / embedded institutions

An extensive analysis of governance hierarchy types is beyond this paper's scope. Still, in all cases, **knowledge abstraction** increases from lower to higher levels. For human governance, knowledge loss occurs *not only* as a necessity for scalability, but also because representatives can only acquire information via communication (indirect) rather than experience (direct). Indeed, even in relatively small human communities sharing a relatively uniform environment and adopting a decentralised governance scheme (e.g. democracy in ancient Athens [28]) collective decisions must be taken based on aggregate knowledge – about individual members, their views and priorities – which, to some extent, must also be acquired via communication (as no individual can have the exact same experience as the others). In such small-scale cases, knowledge aggregation can be decentralised (e.g. based on peer-to-peer exchanges and gossiping), rather than collected by specially-assigned representatives. As before, the larger the scale the higher the knowledge abstraction (and loss of details, which may be critical for collective decisions).

Authority-wise, most large-scale governance hierarchies feature top-down schemes (irrespectively of whether higher level agents are elected representatives or appointed managers). In smart grids for instance, price regulations are dictated by high-level government and electricity companies. Alternative authority schemes may feature bottom-up price setting – e.g. via decentralised markets and auctions [29]. Further alternatives may consider grid power as a common pool resource, where agents store their production in a shared pool, from which they can consume according to community rules. The community itself has the right to self-govern, i.e. to manage its own resource sharing rules [30].

Hierarchy-wise, a community's right to self-govern is typically constrained by a higher-level authority, e.g. federal government, which has the 'final word'. An alternative scheme may give communities authority over (at least) local matters. In the smart grid example, a higher government authority may impose technical standards on local power grids (e.g. electric power voltage and current frequency) but may not intervene on price regulations within such local sub-grids, which would be managed by local communities (except when power is exchanged with the national grid). Both alternatives above are consistent with Ostrom's P7 principle (Cf. sec. III) [22].

V. Conclusions and Future Work

This position paper aimed to highlight the importance of the following system design question: how should the *authority* (or priority) of decision and action be placed within a *hierarchical* system with respect to the *self-awareness* capabilities of each hierarchical level? This has been an ongoing question for (human) societies, leading to many studies and (socio-economic and political) solutions. It is certainly worth raising this question again for technical, and socio-technical systems, not at least because technical agents differ from human 'agents', and technically-'enriched' societies differ from traditional ones. It may also help rethink, reconsider or justify some current human organisations; rather than taking them for granted.

We focus on hierarchical (socio-)technical systems because of their ability to help ensure *scalability* with respect to knowledge management, in systems that pursue one or several *goals* (i.e. control systems). Centralised and fully decentralised designs are special cases of hierarchy. Self-integrating systems are of particular interest because of their increased dynamicity, which exacerbates the scalability and knowledge management issue. Hence, system self-integration brings about the issues of *self-integration of self-awareness* and *of authority*.

We argued that hierarchical designs help scalability by progressive knowledge *abstraction* (i.e. information loss) as knowledge *scope* (i.e. domain) increases, from lower to higher hierarchy levels. This means that the resources necessary to collect and process system knowledge can remain limited (or more-or-less constant) at each level. This also means that self-awareness capabilities are necessarily limited at all levels. The question of authority assignment considering this limitation becomes significant. We categorised solutions into three main schemes: *top-down*, *bottom-up* and *iterative* (*yoyo*).

Top-down authority schemes favour coordination among (self-integrated) system parts, which is best for global system goals; yet may miss system details and fail to customise solutions for local goals, which, if essential, may jeopardise the entire system. Conversely, *bottom-up* authority schemes favour local goals, yet may fail to coordinate with the rest of the system, possibly jeopardising other local and global objectives. Various *iterative* schemes (yoyo) combine several top-down and bottom-up phases, each one bringing more context-sensitive knowledge to hierarchical levels, as feedback to their previously proposed decisions. This helps avoid bloating the system with unnecessary information that is only used on rare occasions; and also helps improve privacy by sharing minimum information, and only when necessary.

We illustrated these concepts via three hierarchical applications from our previous work: i) component self-assembly in distributed systems; ii) resource self-management in sharing economies; and, iii) rule self-management in poly-centric institutions. For each, we highlighted cross-level abstraction and adaptation, discussed authority schemes and alternatives.

Future work will focus on consolidating and refining the authority schemes, including a study of hierarchical topologies (span of control) and their impact on self-awareness and authority in various contexts. We will also study conflicts among incompatible authority schemes in self-integrating systems.

Since most modern systems are large-scale socio-cyber-physical systems, often connected to other socio-cyber-physical systems (leading to ever larger-scales), seriously considering the positioning and interrelation among authority and knowledge management centres becomes rather urgent.

VI. Acknowledgements

Many thanks to Prof. Jeremy Pitt for all the insightful discussions and pointers to relevant work on these topics.

References

[1] A. Diaconescu et al., "Goal-oriented holonics for complex system (self-)integration: Concepts and case studies," in *IEEE Intl Cnf SASO 2016, Augsburg, Germany*, 2016, pp. 100–109. [Online]. Available: http://dx.doi.org/10.1109/SASO.2016.16

[2] A. Diaconescu, "Goal-oriented holonic systems," in *Organic Computing*, C. Müller-Schloer and S. Tomforde, Eds. Springer, 2017.

[3] A. Diaconescu, P. Mata, and K. Bellman, "Self-integrating organic control systems: from crayfish to smart homes," in *Intl Wrksp SAOS'18, Braunschweig, Germany*, 2018.

[4] S. Kounev et al., *The Notion of Self-aware Computing*. Cham: Springer International Publishing, 2017, pp. 3–16.

[5] E. Pournaras et al., "Decentralized planning of energy demand for the management of robustness and discomfort," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 4, pp. 2280–2289, 2014.

[6] R. R. Filho and B. Porter, "Experiments with a machine-centric approach to realise distributed emergent software systems," in *Proceedings of the 15th International Workshop on Adaptive and Reflective Middleware*, ser. ARM 2016. New York, NY, USA: ACM, 2016, pp. 1:1–1:6.

[7] H. A. Simon, "The architecture of complexity," *American Philosophical Society*, vol. 106, 1962.

[8] A. M. Uhrmacher, D. Degenring, and B. Zeigler, *Discrete Event Multi-level Models for Systems Biology*. Springer, 2005, pp. 66–89.

[9] S. Frey, A. Diaconescu, and I. Demeure, "Architectural integration patterns for autonomic management systems," in *Engineering of Autonomic and Autonomous Systems (EASe)*. IEEE, 2012.

[10] M. Sergot, "A computational theory of normative positions," *ACM Trans. Comput. Logic*, vol. 2, no. 4, pp. 581–622, Oct. 2001.

[11] C. Castelfranchi, *Social Power: A Point missed in Multi-Agent, DAI and HCI*. Elsevier, 1990, p. 4962.

[12] A. Jones and M. Sergot, "A formal characterisation of institutionalised power," 1996.

[13] F. A. Hayek, *The Road to Serfdom*. Univ. of Chicago Press, 1944.

[14] F. A. Hayek", "The use of knowledge in society," *American Economic Review*, vol. 35, no. 4, pp. 519–30, 1945.

[15] S. Bowles, A. Klrman, and R. Sethi, "The market algorithm and the scope of government: Reflections on hayek," *VoxEU columns*, 2017. [Online]. Available: https://voxeu.org/article/reflections-hayek

[16] B. Hayes-Roth, "A blackboard architecture for control," *Artif. Intell.*, vol. 26, no. 3, pp. 251–321, Aug. 1985.

[17] J. Beal, J. Berliner, and K. Hunter, "Fast precise distributed control for energy demand management," in *IEEE Intl. Cnf. Self-Adaptive and Self-Organizing Systems*, Sept 2012, pp. 187–192.

[18] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *Knowl. Eng. Rev.*, vol. 19, no. 4, pp. 281–316, Dec. 2004.

[19] B. H. C. Cheng et al., *Software Engineering for Self-Adaptive Systems: A Research Roadmap*. Springer, 2009, pp. 1–26.

[20] K. Fischer, "Holonic multiagent systems - theory and applications," in *Artificial Intelligence: Progress in Artificial Intelligence*. Springer-Verlag, 1999, pp. 34–48.

[21] G. A. Almond, "Comparative political systems," *The Journal of Politics*, no. 3, pp. 391–409, 1956.

[22] E. Ostrom, *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, 1990.

[23] B. Porter, M. Grieves, R. Rodrigues Filho, and D. Leslie, "Rex: A development platform and online learning approach for runtime emergent software systems," in *Symposium on Operating Systems Design and Implementation*. USENIX, November 2016, pp. 333–348.

[24] P. Pilgerstorfer and E. Pournaras, "Self-adaptive learning in decentralized combinatorial optimization: a design paradigm for sharing economies," in *Intl. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2017, pp. 54–64.

[25] E. Pournaras, M. Yao, and D. Helbing, "Self-regulating supply–demand systems," *Future Generation Comp. Systems*, vol. 76, pp. 73–91, 2017.

[26] E. Pournaras, J. Nikolic, A. Omerzel, and D. Helbing, "Engineering democratization in internet of things data analytics," in *Advanced Information Networking and Applications (AINA), IEEE Intl. Conf.* IEEE, 2017, pp. 994–1003.

[27] E. Pournaras, M. Vasirani, R. Kooij, and K. Aberer, "Measuring and controlling unfairness in decentralized planning of energy demand," in *IEEE, Energy Conference (ENERGYCON), 2014*, 2014, pp. 1255–1262.

[28] J. Ober, *Democracy and Knowledge*. Princeton Univ. Press, 2008.

[29] J. K. Kok, C. J. Warmer, and I. G. Kamphuis., "Powermatcher: Multiagent control in the electricity infrastructure," in *Intl. Cnf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, 2005.

[30] A. Diaconescu and J. Pitt, *Holonic Institutions for Multi-scale Polycentric Self-governance*. Springer Intl. Pub., 2015, pp. 19–35.