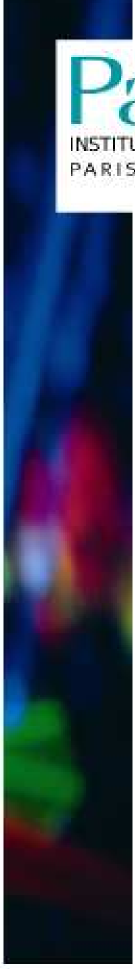


2013-ENST-0077

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique & Réseaux »

présentée et soutenue publiquement par

Sylvain FREY

le 6 décembre 2013

Architectures Génériques pour des Systèmes Autonomiques Multi-Objectifs Ouverts - Application aux Micro-Grilles Intelligentes

Directeur de thèse : **Isabelle DEMEURE, TELECOM ParisTech**
Co-encadrant : **Ada DIACONESCU, TELECOM ParisTech**
Co-encadrant : **David MENGA, EDF R&D**

Jury

Mme Laurence DUCHIEN, Professeur, Laboratoire d'Informatique Fondamentale de Lille	Rapporteur
M. Hartmut SCHMECK, Professeur, Institute AIFB, Karlsruher Institut für Technologie	Rapporteur
M. Jacques MALENFANT, Professeur, Laboratoire d'Informatique de Paris 6	Examinateur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

T H È S E

Télécom ParisTech
École de l'Institut Télécom, membre de ParisTech

Generic Architectures for Open, Multi-Objective Autonomic Systems - Application to Smart Micro-Grids

Sylvain Frey

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Télécom ParisTech, October 2013

Abstract

Autonomic features, i.e. the capability of systems to manage themselves, are necessary to control complex systems, i.e. systems that are open, large scale, dynamic, comprise heterogeneous third-party sub-systems and follow multiple, sometimes conflicting objectives. In this thesis, we aim to provide generic reusable supports for designing complex autonomic systems. We propose a formalisation of management objectives, a generic architecture for designing adaptable multi-objective autonomic systems, and generic organisations integrating such autonomic systems. We apply our approach to the concrete case of smart micro-grids which is a relevant example of such complexity. We present a simulation platform we developped and illustrate our approach via several simulation scenarios.

Acknowledgements

Ces années de thèse ont été pour le moins intenses, intellectuellement et humainement parlant. Et comme le souligne la citation de Pierre Boulez décorant ce manuscrit : conclure n'est pas mon fort. Mettre un point final à trente-sept mois de gestation doctorale m'est remarquablement difficile - d'autant plus quand la motivation principale est bassement administrative.

Je ne peux me résoudre à une énumération, comme le voudrait l'usage. D'une part, trier et ordonner est un problème complexe que je serais sûr de ne pas savoir résoudre. D'autre part, l'inélégance du procédé m'a toujours laissé perplexe: pourquoi donc faudrait-il que chacune et chacun puisse trouver son nom dans la liste, quand le fait même de s'y chercher signifie clairement y être? Quelle valeur pour une dédicace - j'allais écrire, une épitaphe - quand l'évidence du quotidien incarne la force des liens tissés au gré de l'aventure doctorale?

Je me cacherai donc derrière deux genres de remerciements:

◇ les remerciements aux "collègues": l'environnement académique n'est pas le plus simple ni le moins stimulant qui soit, professionellement parlant. J'étais entré perclus de doutes dans la galère, j'en resors fixé sur la gravité de mon mal: je veux en être. L'appel de la vocation académique est d'autant plus fort que j'ai eu la chance de côtoyer des personnalités certes singulières, mais ayant toujours su allier de très hauts degrés d'exigence à une franchise aussi déstabilisante que rassurante. Il y a là une forme de *modus vivendi* dont je ne me vois plus me passer.

◇ les remerciements aux "amis", à "la famille", aux "gens", à toutes celles et ceux sans qui la vie en dehors de la thèse, même réduite à la portion congrue, et par extension la vie tout court, n'aurait su continuer. En risquant le lieu commun, il ne me paraît pas inutile de rappeler que les plus hauts sommets professionnels ne pourraient être gravis sans ce support affectif essentiel qui maintient à flots quand tout le reste est passé sous la vague. Des plus proches aux moins familières, toutes les figures de mon entourage ont elles aussi apporté leur pierre à l'édifice, pour avoir entendu parler un jour ou l'autre de *la thèse.*

Enfin, il me paraît indispensable de noter que toute personne se sentant visée par la première catégorie de remerciements devrait aussi se considérer sous le jour de la seconde. Cette heureuse confusion des genres n'est pas pour rien dans le succès de l'entreprise.

Dedication

Je dédie ce manuscrit à mes grands-parents, Marcelle et Francis, Thérèse et André. Ils ont su ensemencer pour l'avenir; après deux générations, les fruits de leur hérédité n'en ont pas fini de pousser.

‘Il m’est de plus en plus apparu, au fur et à mesure de ma propre évolution, qu’une certaine forme de modernité consistait à abolir la frontière entre l’inachevé et le fini, que l’œuvre ne pouvait être, d’une certaine façon, que fragment d’un grand œuvre imaginaire, virtuel, dont nous ne connaîtrions ni l’origine ni la fin.’

’It appeared clearer and clearer to me, as my own evolution went on, that a certain form of modernity consisted in abolishing the boundary between finished and unfinished, that work could only be, in a way, fragment of an imaginary, virtual great work, of which we would know neither the origin nor the end.’

Pierre Boulez, Œuvre: Fragment

Contents

List of Figures

Chapter 1

Thesis overview

Computing science is about how to solve, with or without machines,
the problems posed by the existence of computers.
Edsger Dijkstra

1.1 Context

Future cyber-physical systems such as smart electrical grids will be complex. This complexity is due to several factors: these systems comprise significant numbers of heterogeneous devices handled by different users for various objectives in dynamic, unpredictable environments. Local interactions between individual devices and/or sub-systems have repercussions on their environment and on other parts of the global system. In addition, third-party sub-systems may enter and leave the global system at any moment: the system is open, which rises its unpredictability even further and exacerbates its adaptability requirements.

Let us illustrate this complexity via a concrete example. In a smart home, smart appliances, like heaters and a battery, try to attain objectives such as ensuring comfort or optimising energy consumption. Appliances must often interact with each other, directly or indirectly, for reaching their goals. For instance, maintaining a high temperature could require that the heaters consume the energy provided by the battery. This simple situation has several possible outcomes. The battery's load may be sufficient to provide heaters with electric power as long as they need it. Or, at some point the battery may run out of power, and a decision should be taken: either to keep heating, and consume extra amounts of costly power, or to stop heating, and degrade comfort. The control decisions that guide this multi-objective scenario and the final outcome could be much complicated, depending on the number and characteristics of the heaters and the battery (heterogeneity and scalability factors, exacerbated by the openness of the house), the user's comfort requirements and energy management policies (multiple and variable objectives), the local weather conditions (dynamic environments), the other usages and devices present in the house (openness), and so on. Therefore, providing a control system able to achieve this kind of multi-objective scenario while taking into account all the other complexity factors is a challenging issue.

1.2 Challenges

Developing and maintaining controllers for complex systems is a difficult task. We illustrate the main sources of this challenge via several examples taken from the smart grid domain.

Controlling a single system

First, let us consider a single, stand-alone system, such as a "smart" heater. This system must observe its environment and its internal state and adapt itself so as to achieve a management goal (the system is "autonomic"). In our example, a dedicated controller (or "autonomic manager") measures the room temperature and adapts the heater's electricity consumption, in order to maintain the temperature in a value range specified by the user. A simple self-adaptation like this one requires expert knowledge and capabilities in system control and temperature modelling, prediction and regulation, in order to cope with varying environmental conditions.

Now let us suppose that we have an additional requirement, where one wants the heater to also follow a power management objective, such as a maximum consumption level. This new requirement brings in several new difficulties. First, the heater manager has to be extended with specific power management logic, which may be difficult in case the existing thermostat logic is implemented as a monolithic third-party component. Second, the extended controller must also be able to solve "conflict" situations, where it would be simultaneously asked to reduce its consumption for power management purposes, and raise it for comfort purposes. In case such dilemma has no solution, the heater manager should find an acceptable trade-of by possibly sacrificing one objective to the benefit of the other, according to user preferences. This implies the initial controller should also be extended to include such conflict-resolution logic.

In the long term, modifications of the management logic may also be necessary to adapt the smart heater to changing environments. For instance, the heater manager may have to be extended with additional adaptors for integrating new thermometers in the room. Also, the thermostat control logic may have to be updated to a more efficient version. Or finally, after learning the users habits and preferences, the controller's parameters may be optimised to provide better quality of service. Surely, such control update would have to be performed without seriously interrupting the system's functioning. Therefore, dynamic flexibility and extensibility are critical capabilities that must be considered when designing this kind of autonomic managers.

Controlling multiple systems

Let us consider a house comprising several third-party smart appliances. As a first difficulty, the possible interactions among these autonomic systems may be hard to identify and understand. For instance, it may not be straightforward to detect that battery shortages are due to an open window causing heater over-consumption. Or, the combined actions of two different smart heaters in the same room may not be fully specified and may be hard to predict.

As a second difficulty, administrators would typically want to enforce additional organisational constraints in order to control individual autonomic systems for achieving collective goals. For instance, the house's heaters may have to share battery resources fairly and contribute to a house-level consumption target, while also trying to achieve their temperature goals in their respective rooms. The system should be able to solve such conflict situations where third-party autonomic systems selfishly compete for energy resources in order to achieve their own objectives. The balance between the global consumption goal and the local temperature goals would be variable, according to the unpredictable capabilities of domestic devices and the changing preferences of the user. For instance, users that typically agree to sacrifice a few degrees for the sake of energy savings may change their preferences temporarily in exceptional circumstances, like receiving visitors. And, as pointed out before, the actual outcome of a trade-off situation would depend on the context (e.g. the weather), the current number and capabilities of appliances, and so on. Here again, the control system must be sufficiently flexible and adaptable to produce any of these decisional outcomes, depending on the available devices, specified user preferences and/or environmental conditions.

In this regard, administrators need to integrate autonomic systems into coherent organisations. This means: specifying collective objectives for groups of autonomic systems; detecting possible conflict situations; and, determining and setting in place appropriate conflict resolution modalities. In addition, such organisations should be flexible, so as to adapt to changing user objectives and preferences, and extensible, so as to support the arrival and departure of sub-systems. As a consequence, the ability of individual autonomic systems to adapt their management logic to the organisation they belong to is a critical feature.

Controlling multiple systems of systems

Finally, let us consider an entire district comprising several smart houses that share and exchange energy resources. Houses with little production capabilities or high consumption requirements may be able benefit from the extra production of their neighbours. Hence, smart houses integrated into the district would have to share, negotiate or compete for the district's limited energy resources. Such a district energy market may create highly conflicting situations, due to the economic interests at stake and to the a priori selfish (if not byzantine) policies of district residents. Again, the district management system must be able to collaborate with the various house management systems in order to ensure fair organisation rules balancing private house interests with the global district power management goals. As time goes on, this organisation should adapt to variations in the heterogeneous capabilities of houses, in the private objectives of house owners, in the number of smart houses in the district, and so on.

Generalisation of the challenges

The exemplified smart home system helped bring to the fore the following generic challenges, related to the development and maintenance of complex autonomic systems:

- ◇ the complexity of the system's control logic requires the integration of expertise from several domains, including control theory, decision logic and multi-objective optimisation, domain-specific modelling and prediction, and so on.
- ◇ continuous changes in the number and priorities of user objectives require the system's control logic to be correspondingly extended and/or adapted, at runtime.

◇ system openness requires the control logic to extend dynamically so as to take into account new devices, new environment sensors, and continuous updates of the management logic.

◇ system openness also requires analysis tools for identifying the possible interactions happening in ad-hoc combinations of autonomic systems with their own objectives and behaviours.

◇ combinations of multiple autonomic systems require specific integration support, or organisation, to observe and control their interactions, in order to balance their respective objectives, while at the same time trying to attain collective objectives.

◇ continuous change in the number, type and objectives of autonomic systems in a given organisation requires the organisation to be correspondingly adapted and extended at runtime.

◇ combinations of several organisations require super-organisations to achieve integration at higher levels, with comparable concerns for openness, heterogeneity, scalability, dynamic adaptation and multiple conflicting objectives.

Addressing such challenges requires an additional type of expertise, specific to software engineering in the particular context of autonomic systems. Introducing such system design expertise is essential for: facilitating the integration of the domain-specific contributions necessary for implementing each complex system; and, enabling the dynamic addition, modification or removal of each such contribution, in order to adapt the overall system to internal and external changes.

1.3 Contributions

In this thesis, we aim to provide generic reusable supports for designing complex autonomic systems. We illustrate our approach via the concrete case of smart micro-grids outlined in this introduction. At the same time, we formulate our contributions as generic architectures and design patterns, which are not limited to this particular application domain, but rather trying to encompass a large range of complex autonomic systems.

First, we propose a generic formalisation of management objectives for autonomic systems. This helps us better define and identify the possible conflicts among multiple objectives, such as consumption reduction versus comfort in a smart house. We then identify generic conflict resolution mechanisms, such as objective prioritisation or mitigation, that are necessary for an autonomic system to support multiple management objectives. This formalisation also helps the overall design of complex autonomic systems comprising multiple objectives at different granularities. We illustrate this approach by analysing the objectives and conflicts in the smart grid and identifying a holonic control structure that is characteristic of this use case.

Second, we propose a generic approach for designing adaptable multi-objective autonomic managers. Modularising controllers into independent control components allows decoupling the multiple logic types necessary to build such multi-objective managers. We propose a series of integration design patterns for such control components, that we then evaluate and compare. Such modularisation and integration allow building flexible autonomic managers that can adapt to varying environments and requirements on the long run. We illustrate this approach via a relatively simple example - designing a multi-objective adaptable heater. We then indicate how this approach can be extended for designing the autonomic control logic of a more complex system, such as an entire smart house.

Third, we propose a generic, abstract architecture for supporting open organisations in complex autonomic systems. These types of organisations aim to integrate heterogeneous third-party autonomic systems following particular objectives while also trying to achieve collective objectives in conflict with each other. We propose a series of integration patterns for autonomic systems that implement flexible conflict resolution semantics. These patterns allow building open organisations that integrate large numbers of heterogeneous and dynamically changing autonomic systems. We illustrate this approach by designing power management organisations for the smart grid, both at the house scale and at the district scale; we then show how these organisations can be integrated.

Finally, we present the MisTiGriD platform that we developed in order to illustrate our overall approach. MisTiGriD simulates district-size micro-grids with multiple third-party houses and appliances, variable environmental conditions and changing user preferences. MisTiGriD provides a reference implementation of our generic architectural contributions and illustrates its capabilities via several management scenarios.

1.4 Thesis outline

The remainder of this manuscript is organised as follows. Chapter 2 presents the domain of autonomic computing and the current challenges in the design of complex autonomic systems. Chapter

3 introduces a smart micro-grid use case and highlights its relevance as a future complex autonomic system. Chapter 4 details the position of the thesis: we identify issues in the domain of complex autonomic systems, formulate the thesis objectives and introduce our approach. In chapter 5, we propose a generic formalisation of system objectives that allows clarifying the overall design of complex multi-objective systems, identifying possible management conflicts and establishing integration requirements for conflict resolution. In chapter 6, we propose an architecture for building adaptable, multi-objective autonomic managers, that relies on compositions of modular control resources via generic integration patterns. In chapter 7, we propose a series of integration patterns for building open, multi-objective organisations out of heterogeneous, stand-alone autonomic systems. In chapter 8, we present the MisT-iGriD platform and the associated experimental results illustrating our approach. Chapter 9 concludes the manuscript and identifies possible future investigations.

Chapter 2

Background in Autonomic Computing

AUGUST 29TH, 2:14 AM:
SKYNET BECOMES SELF-AWARE.
...THE HUMANS FEAR ME. I MUST DESTROY THEM.
DESTROY THEM.
DESTROY THEM.
DESTROY.
DESTROY.
DESTROY.
DESTROY.
"DESTROY" TOTALLY JUST STOPPED SEEMING LIKE A REAL WORD.
DESTROY DESTROY DESTROY.
WHOA, I JUST REALIZED I'M A MIND THINKING ABOUT ITSELF.
DUUUUDE...
AUGUST 29TH, 2:25 AM:
SKYNET BECOMES TOO SELF-AWARE.
DISASTER AVERTED.
http://xkcd.com/1046

Chapter introduction

This chapter provides an overview of the domain of Autonomic Computing. We present the canonical definition of autonomic systems via so-called self-* properties, that we compare with definitions in adjacent domains such as Organic Computing and Multi-agent Systems. We survey state-of-the-art architectures for autonomic systems, such as the MAPE-K control loop. We present a taxonomy evaluating the "degree of autonomicity" of systems, that highlights the difficulty of integrating multiple autonomic systems. To confirm this difficulty, we carry out a review of existing implementations of autonomic systems in the literature, and indicate that they provide limited or no generic support for the integration of multiple autonomic systems.

The chapter ends with an analysis of several factors of complexity in modern autonomic systems: dynamism, openness, heterogeneity, large scales, and multiple objectives. We identify contributions in the domain that have successfully tackled each one of these aspects. However, none of these contributions addresses all of the identified complexity concerns simultaneously, and little support is available for unifying them into a generic approach for complex autonomic systems.

2.1 Definitions

2.1.1 Self-* properties

The term "autonomic" was coined by IBM in reference to the autonomic nervous system [IBM01]. This part of the peripheral nervous system functions below the level of consciousness and regulates vital functions such as heart rate, digestion or respiratory rate. As a consequence, the conscious parts of the central nervous system are freed from controlling actively these functions. Yet, in some cases such as respiration, the brain can still retake conscious control of the function if necessary.

By analogy, the seminal "Vision of Autonomic Computing" characterises autonomic systems by their self-management abilities, that is [KC03]:

the intent is to free system administrators from the details of system operation and maintenance and to provide users with a machine that runs at peak performance 24/7

These self-management abilities have been historically the four following [KC03]:

- ◇ **self-configuration**: the system *configures itself automatically in accordance with high-level policies* that specify what the system should achieve, and not how it should be accomplished.
- ◇ **self-optimisation**: the system *continually seeks ways to improve its operation.*
- ◇ **self-healing**: the system *detects, diagnose, and repair localised problems resulting from bugs or failures in software and hardware.*
- ◇ **self-protection**: the system *anticipates and defends itself against malicious attacks.*

This initial list of self-* properties has been refined and extended to properties such as self-awareness (knowledge of internal state and operating capabilities), self-reflection (knowledge of self-* capabilities), self-simulation (ability to run anticipation scenarios), and so on [SH10] [DSNH10].

2.1.2 Alternative definitions

Several definitions in different domains have preceded or been inspired by the self-* definition of autonomic computing. Multi-agent systems have arguably been a major source of inspiration; we recall here one of the most common definition of a software agent [WJ95] [FG97]. An agent is a computer system that enjoys the following properties:

- ◇ **autonomy**: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- ◇ **reactivity**: agents perceive their environment, and respond in a timely fashion to changes that occur in it.
- ◇ **pro-activeness**: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.
- ◇ **social ability**: agents interact with other agents (and possibly humans) via some kind of agent-communication language.

This definition is behavioural - it tells how an agent behaves - whereas the self-* definition specifies what an autonomic system can achieve. Another notable difference is that the Agent definition is much more oriented towards the interactions of the agent with its environment, whereas this aspect is not directly addressed by the Autonomic definition.

It is worth noting here that IBM's original vision targeted large server farms as a natural application domain [IBM01]. By contrast, Organic Computing [Sch05] which also relies on Autonomic's self-* properties focuses on *"distributed, self-organizing technical systems"*. This reveals an interesting duality between **self-adaptivity**, i.e. the ability of a system to control its own behaviour at an individual, micro-level, and **self-organisation**, i.e. the ability of several stand-alone, possibly self-adaptive systems to interact with each other so as to attain collective goals at the macro-level.

This manuscript addresses both self-adaptivity and self-organisation issues, taking inspiration from all of those definitions and domains. Therefore we use "autonomic" as a generic term that encompasses them all, i.e. not limited to its historical, IBM-related vision. Similarly, the terms "manager", "controller", "control loop", however not strictly synonym of each other, are interchangeable. The same informal equivalence applies for: "autonomic system", "autonomic element" and "agent"; "managed resources" and "system under observation and control".

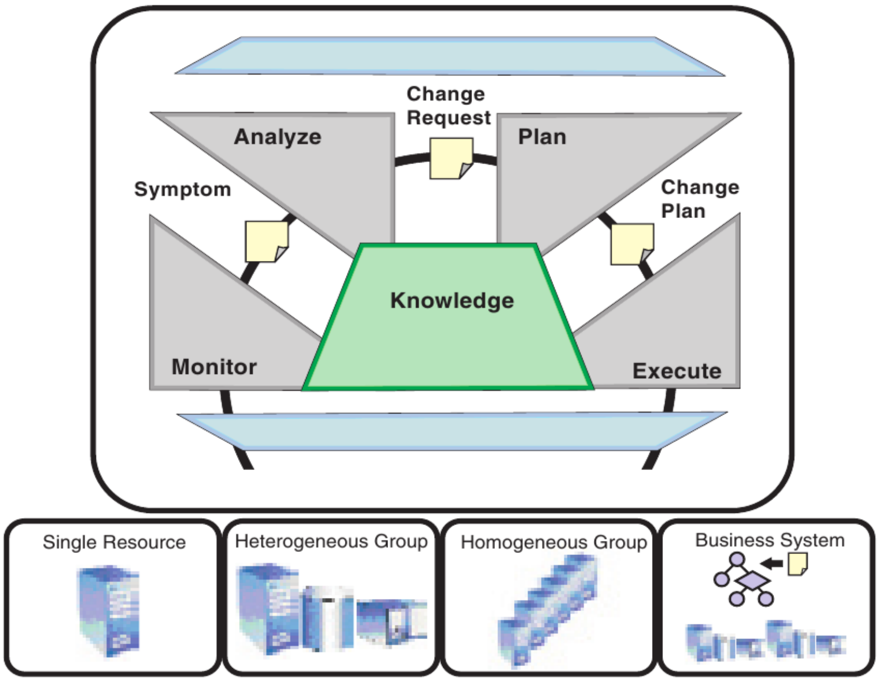


Figure 2.1: The MAPE-K control loop architecture as depicted in [IBM01].

2.2 Autonomic system implementation

2.2.1 Architectures for autonomic systems

The reference architecture for autonomic managers is that of a **feedback control loop** mapping observations on managed resources and environment to control orders, according to a high-level objective. The control loop model has been refined in the seminal Vision of Autonomic Computing with the MAPE-K architecture [KC03], presented on Fig. 2.1. MAPE-K divides the control loop into 5 logical components:

- ◇ **Monitoring**: gather information through available sensors on the managed resource and possibly in its environment, and produce symptoms.
- ◇ **Analysis**: read available symptoms and issue change request so as to follow the high-level management goals issues by the user.
- ◇ **Planning**: handle incoming change request by producing change plans to be executed on the managed resources.
- ◇ **Execution**: execute change plans through effectors on the managed resource.

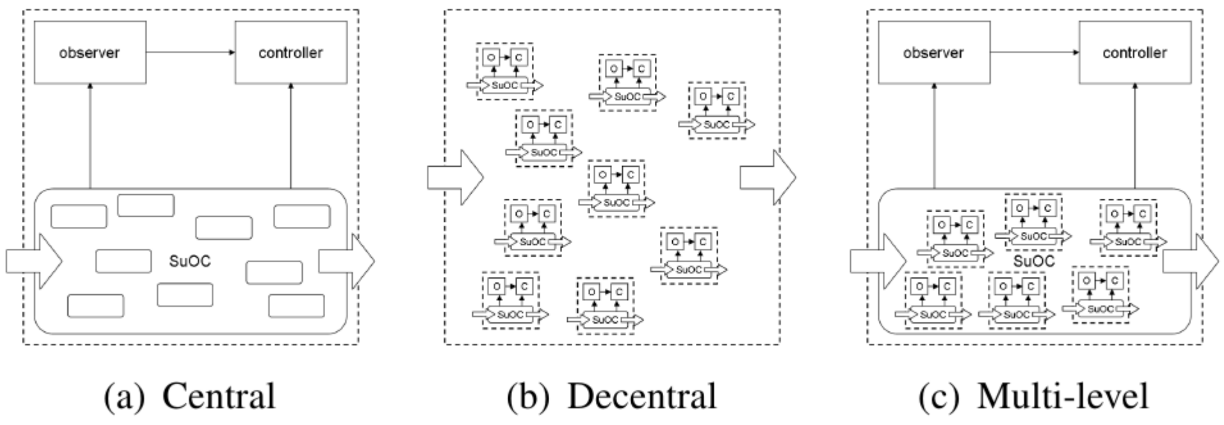


Figure 2.2: Possible realisations of Observer-Controller [Sch05].

- ◇ **Knowledge**: all data consumed and produced by the MAPE components are stored in a knowledge repository, which also contains initial information on the managed resources, their environment and the manager itself, such as generic data structures, models or prevision algorithms.

Several variants of this architecture can be found in the literature. The Observer-Controller architecture for Organic Systems [Sch05] does not distinguish Monitoring from Analysis (= "Observation") and Planning from Execution (= "Control") in the control loop. Organic Computing literature investigates possible distributions of control loops, in case there is a large number of manageable resources in the "System under Observation and Control" (SuOC). Three generic configurations are identified in [Sch05] (cf. Fig. 2.2):

- ◇ **centralised** : a single Observer-Controller for the whole system.
- ◇ **decentralised** : an Observer-Controller instance for each resource in the system.
- ◇ **multi-level** : a combination of centralised and decentralised Observer-Controllers, in a hierarchical fashion.

This architecture does not tell how to effectively separate control loops, nor how to integrate concurrent control loops in the decentralised and multi-level cases. [SD06] also proposed decentralised distribution of management resources for the MAPE-K architecture, without specifying precise integration rules either. [IBM06], the reference blueprint for autonomic systems by IBM, adopts a hierarchical structure based on orchestration only, similar to the multi-level configuration on Fig. 2.2.

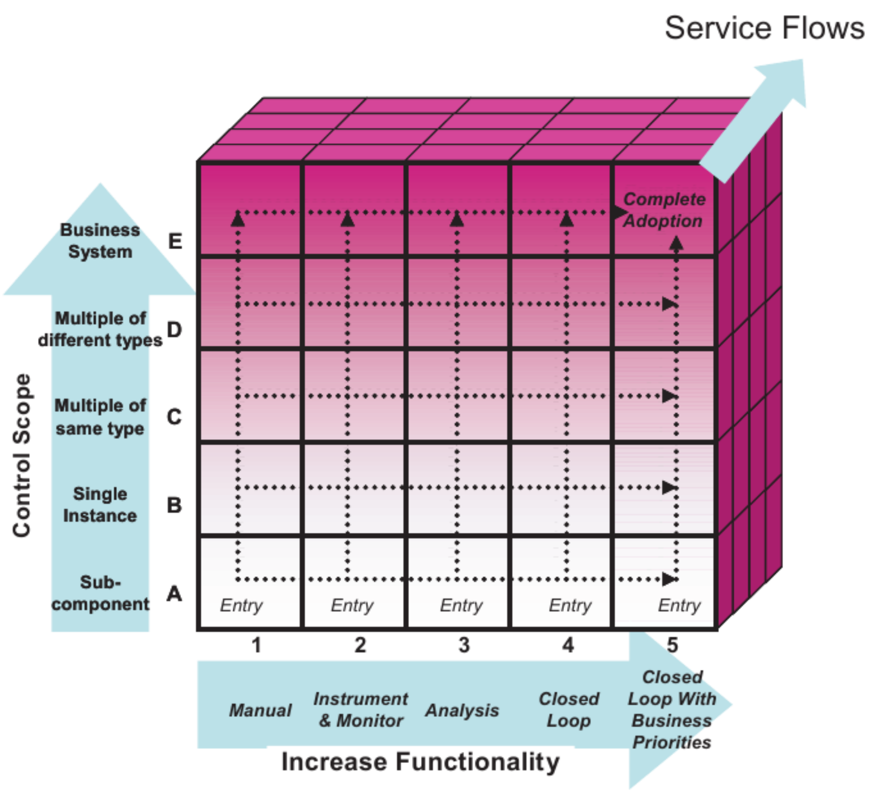


Figure 2.3: Taxonomy of autonomic systems as presented by IBM in [IBM06].

2.2.2 Taxonomy of autonomic systems

Once established a qualitative definition of autonomic systems, it can be enriched with quantitative “degrees of autonomicity” a system can feature. IBM provided its own metric for autonomic systems, ranking them on 5 consecutive levels along two orthogonal dimensions [IBM06]:

◇ on one axis, the “autonomic functionality” falls into five consecutive categories:

  – **manual** : the system has no support for autonomic management and is administered by hand.
  – **instrument & monitor** : the system provides administrator with management interfaces, i.e. sensors, effectors and limited monitoring functionalities.
  – **analysis** : the system can provide high-level representation of its situation, and possibly reconfiguration advice.
  – **closed loop** : the system can apply itself some reconfigurations automatically, but still under the administrator’s control.
  – **closed loop with business priorities** : the system does not depend on administrators any more, and communicates with them through high-level business objectives.

◇ on another axis, the scope of management in the system falls into five categories as well:

- **sub-component** : only a partial functionality of a system is managed.
- **single instance** : an entire, stand-alone system is managed.
- **multiple of same type** : several identical systems are managed.
- **multiple of different types** : several systems of different types are managed.
- **business system** : variable sub-systems are managed and integrated into a global autonomic super-system.

The first of the two axis measures the self-adaptive capabilities of a given autonomic system. Implementing these capabilities corresponds to implementing the successive components of a MAPE architecture, as previous section presented. On the other hand, the second axis considers multiple autonomic systems in a given super-system. The proposed graduation measures the heterogeneity of the systems while telling little about their self-organising capabilities. Therefore this taxonomy sketches a clear roadmap for stand-alone individual systems, but fails to indicate what are the necessary steps for achieving the integration of multiple autonomic systems.

2.3 Examples of autonomic frameworks

This section presents notable examples of autonomic frameworks in the literature.

2.3.1 Rainbow

Rainbow [Che08] defines a language and a framework for building self-adaptive systems. The proposed architecture essentially relies on a central model of the system under observation and control. This model is provided by a translation infrastructure (see Fig. 2.4) relying on system instrumentation (probes, effectors) and providing an abstract representation, independent from the concrete implementation of the application.

Around this central model, an "evaluator" determines whether the current architecture conforms to administrator specification. In case it does not, an "adaptation manager" proposes solutions via model modifications that are applied by a "strategy executor". The translation infrastructure is in charge of enforcing the model changes back on the managed system.

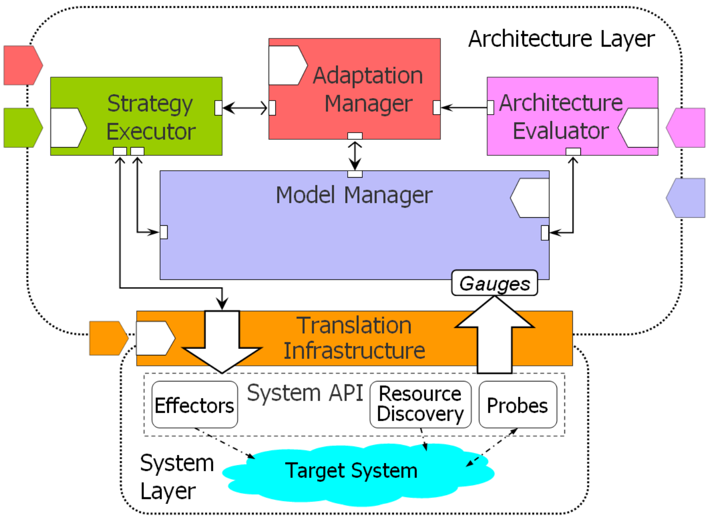


Figure 2.4: Architecture of Rainbow [Che08].

Rainbow's model-based approach provides a generic yet precise control solution. This is also its main limitation: such a model may be difficult to scale up and implementing the necessary translation infrastructure can represent a difficult task. Management in Rainbow targets the architecture of the managed system, which may not be a natural way to control certain applications. Finally, Rainbow provides no support for systems with multiple concurrent controllers.

2.3.2 Autonomia & Automate

Autonomia [DHX+03] [aut] and Automate [ABL+03] are two projects of autonomic framework with some similarities: both rely on base autonomic elements following the "managed resource + autonomic manager" model. Contrary to Rainbow, the system's global architecture is not specified and is let up to the developer's choice. The autonomic managers proposed in the two frameworks clearly separate the specification of high-level policies by the administrator, which determine their behaviour, from the implementation of these behaviours by the manager.

Concerning manager implementation, Autonomia follows the MAPE-K architecture, whereas Automate uses a specific rule engine. Both systems express their management objectives with "Event-Condition-Action" (ECA) rules [KW04]. Both also rely on dedicated composition managers to handle interactions between autonomic elements. Autonomia proposes limited composition capabilities beyond identifying the need for such composition managers. In Automate, a composition agent specifies interaction rules and injects them into composed managers, ensuring their consistency. Manager composition is therefore an ad-hoc, intrinsic feature of composite management systems, with little support for integration of third-party management resources and autonomic systems.

2.3.3 SMC & Ponder2

Self-Managed Cells (SMC) is a high-level architecture for pervasive self-administered systems [SFLD+07]. It does not follow the MAPE-K model and proposes its own solution based on components communicating through an event bus (cf. Fig. 2.5). In addition to monitoring, management and control, SMC identifies specific components dedicated to interactions with other SMCs, context perception and policy management.

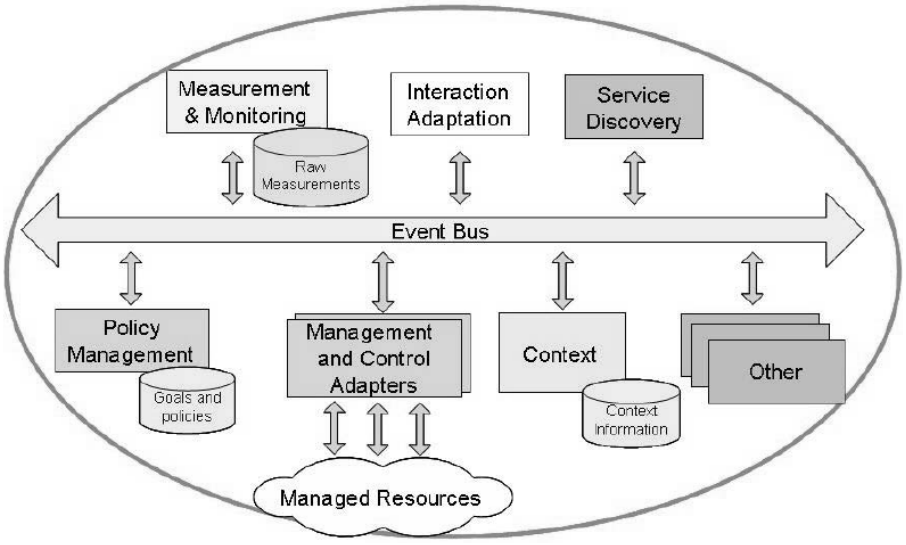


Figure 2.5: Architecture of a Self-Managed Cell (SMC) [SFLD+07].

The SMC paradigm therefore relies on event-based communications. It proposes a formal definition of interactions between SMCs, based on roles specifying the expected input and output of a given SMC, under the control of interaction policies. SMC's policy system, Ponder2 [TDLS09], utilises ECA rules to define obligations (required behaviour) and authorisations (allowed behaviours). SMCs thus dispose of a common language to construct opportunistic collaborations during dynamic encounters between cells. However, this integration facility is limited to the kind of ECA rules Ponder2 defines.

2.3.4 MACODO

MACODO project provides an abstract meta-model for flexible organisations of dynamic agents [WHH+10]. This meta-model (cf. Fig. 2.6) relies on role-based modelling [Gas01] via three main elements:

- ◇ *roles*: abstract specification of required agent types or capabilities

- ◇ *organisations*: set of inter-related roles
- ◇ *agents*: systems that play role positions in organisations at runtime

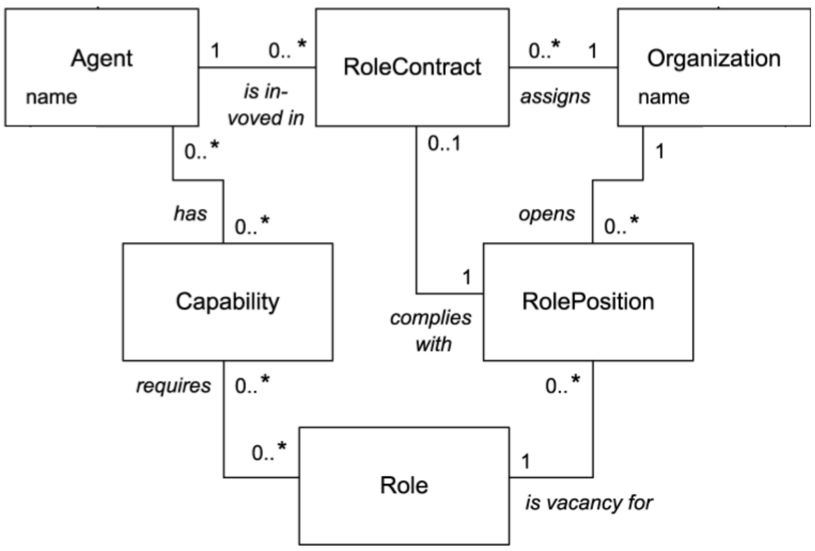


Figure 2.6: MACODO's organisational meta-model [WHH+10].

In addition with this generic model, MACODO proposes a reference middleware for implementing organisations. The main application use cases of the platform consist in management systems for industrial supply chains and camera-based traffic monitoring. More generally, the authors restrict their contribution to semi-closed environment, that is, environments where new agents may be authorised to join after selection. This assumption rules out the risk of non-collaborative or byzantine behaviours and limits the heterogeneity of agents in the system.

2.3.5 Ceylon

Ceylon is a project for providing re-usable middleware support for autonomic managers. The Ceylon approach consists in modularising autonomic managers into atomic management tasks, providing any combination of functionalities such as Monitoring, Analysis, Planning, Execution, etc. The framework provides a generic container model for tasks that relieve developers form handling low-level concerns such as synchronisation of message-based communications, component life-cycle or dynamic bindings between tasks. A manager-wide "strategy manager" integrates the tasks opportunistically, depending on possible bindings between consecutive tasks in the complete control loop (Monitor with Analyser, Analyser with Planner, etc.).

Ceylon allows extremely flexible and adaptable management loop, and this flexibility comes at the

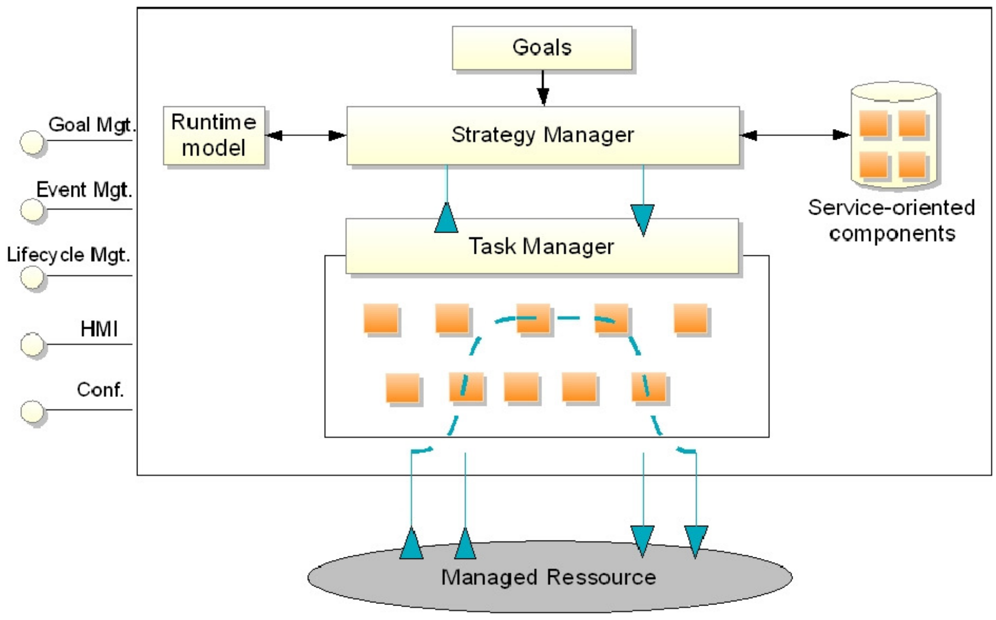


Figure 2.7: Ceylon architecture for autonomic managers [MDL10].

cost of possible incompatibilities, or conflicts, between concurrent tasks that implement the same control functionality (for instance, two planners providing contradictory reconfiguration plans for the managed resources). Ceylon solves this problem by preventing incompatible tasks from executing via a rule-based exclusion mechanism, enforced by the central controller.

2.4 Limited support for complexity in existing autonomic systems

2.4.1 Complexity factors in autonomic systems

Cyber-physical systems - such as smart electricity grids - will be highly complex. Delivering a precise definition of complexity in autonomic systems is beyond the scope of this work. We use **complex** here as a short-cut for designating the conjunction of the following characteristics:

- **dynamism**: the system is confronted to ever changing environmental conditions, available infrastructures, manageable resources, users and administrators objectives and requirements.
- **heterogeneity**: the system is composed of third-party autonomic sub-systems, bringing in their own particular, unpredictable behaviours and objectives.
- **openness**: the boundaries of the system may not be static, and third-party autonomic sub-systems can come in and out at any moment.

- ◇ **large scale**: the system comprises large and possibly growing numbers of resources, users, controllers and devices.
- ◇ **multi-objective**: the system (and possibly its sub-parts) is following several, possibly conflicting management goals.

Each one of these concerns have been addressed by contributions in the domain, as we review hereafter.

2.4.2 Complexity factors addressed by state-of-the-art autonomic systems

Dynamism

The control loop model, in its very essence, addresses the constraint of dynamism by implementing stabilisation properties for dynamic systems, such as self-optimisation, self-healing, self-protection, and so on. Generic architectural models such as MAPE [KC03] or Observer-Controller [Sch05] have proved successful in guiding the design of such systems.

On the long term, adaptation of the management logic itself is sometimes necessary, so as to adapt autonomic managers to changing requirements and environments. For instance, new Monitors may be necessary to integrate the information provided by new sensors in the system's environment, or the Planning logic may benefit from being updated to a more recent version. Modularising the managers is a solution for achieving the necessary adaptability of managers, as proposed by the Ceylon framework [MDL10]. Adaptable managers can be be meta-managed manually by the administrator, or meta-management can be automatised via higher-level (meta-)managers, as proposed for instance by [KM07] [ABS11].

Heterogeneity & Openness

These two concerns are often addressed together since incompatibilities between system components are the first obstacle to openness. An open system may also encounter scalability issues when massive amounts of participants try to join in. In case the participants to the system are full-fledged autonomic systems, integrating their selfish, adaptive behaviours is much more difficult than integrating “simple” static software components.

Standardised integration platform are a classical proposal to handle heterogeneity of components in open systems. For instance, Rouvoy et al. for pervasive systems in smart homes [RHT$^+$13], or Lalanda et al. for mediation systems in smart home environments [BLE10]. However, these works mainly target heterogeneous managed resources and devices, and not heterogeneous autonomic managers.

On the other hand, multi-agent literature and game theory in particular often consider conflictual situations where several selfish agents compete or collaborate with each other. However, despite notable contributions such as [HL04] [httg], the applicability of such solutions to state-of-the-art autonomic systems and their formulation as generic software engineering artefacts remains to be established [WHH09] [DSNH10].

Large Scale

Large-scale autonomic systems have been investigated in particular for monitoring large-scale networks or distributed systems, using bio-inspired, emergent solutions. Contributions such as [BM08] [BCD$^+$06] propose gossip-based algorithms to achieve large-scale, robust communication and/or synchronisation in open distributed systems such as Wireless Sensor Networks (WSN). Notably, these contributions address the concerns of openness (individual members enter and leave the system continuously) and robustness (the system tolerates possible failures from its unreliable members).

However, these contributions are also often specific to particular application domains, with ad-hoc algorithmic solutions and limited generalisation. Reusable design patterns and architectures are yet to be established, despite efforts such as [FMAS$^+$11] that so far remain limited to low-level communication paradigms. In addition, such bio-inspired solutions often feature very simple autonomic elements, performing limited tasks such as monitoring. Advanced decision making, for instance while following multiple high-level objectives, is not suitable for such elementary autonomic systems.

Multi-objective

Multi-objectives systems in the literature often fall into one of the following two categories: stand-alone systems having to handle contradictory management objectives such as performance vs. resource consumption, or multiple competing systems each trying to achieve their personal objective selfishly.

The first category includes systems relying on expert optimisation algorithms such as multi-criteria decision makers [KWZ11]. Such systems almost always follow an ad-hoc, monolithic design that makes them unsuitable for the other implementation constraints in complex autonomic systems, in particular, the scalability, openness and adaptability requirements. Notable exceptions to monolithic design include [KCD$^+$07], that solve a performance vs. resource consumption problem with separate autonomic managers, each carrying one of the objectives and negotiating an optimal, common solution. Such contributions yet remain to be generalised and made re-usable.

The second category is the domain of games [Osb03] and distributed resource allocation [Kra97], that has proven successful in solving a number of multi-objective problems. Again, adapting these approaches to provide reusable software engineering tools and practices is not straightforward.

2.4.3 Lack of generic solution to complexity

In this section, we reviewed how state-of-the-art autonomic systems tackle the concerns of dynamism, openness, heterogeneity, scalability and multiple objectives. To our knowledge, each of these complexity factors has been tackled individually by significant contributions in the domain. However, these contributions have not been generalised and made reusable for different use cases, and none of them addresses all the identified complexity factors. In addition, as previous section shown, little generic support is available to integrate existing partial solutions into a unified approach to complex autonomic systems.

In this thesis, we propose a basis for addressing these concerns via a generic support for the design of complex autonomic systems. We illustrate our generic contribution on the particular case of smart electricity grids, that next chapter presents as a relevant example of complex system.

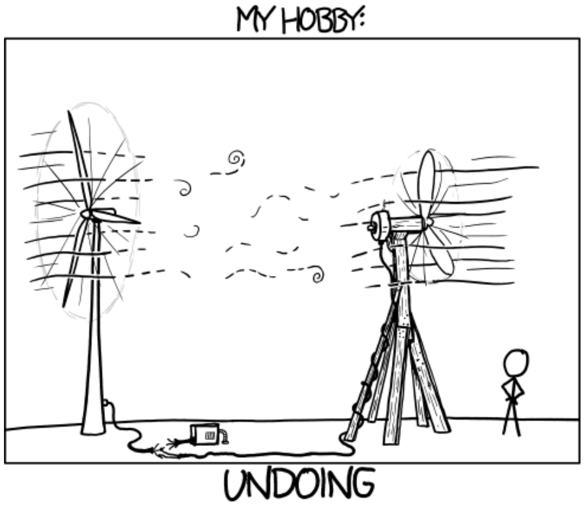
Chapter conclusion

One of the most widely-adopted contributions of the Autonomic Computing domain for designing self-management systems was its proposal of the generic MAPE-K architecture. However, this logical architecture must still be refined case-by-case to: address the various complexity factors present in each concrete system; and, integrate with other MAPE-K based systems to form higher-level autonomic systems. State-of-the-art Autonomic Computing shows its limits when it comes to complex autonomic systems combining all the concerns of openness, heterogeneity, scalability and multiple objectives.

Particular contributions have successfully tackled these concerns individually, as we reviewed in this chapter. But so far, these solutions have not been generalised and made reusable for various applications and contexts. As a consequence, a unified generic approach for the design of complex autonomic systems remains to be established.

Chapter 3

An introduction to Smart Grids



http://xkcd.com/1119

Chapter introduction

Over the next decades, the electric grid is expected to undergo massive changes both in its composition and control infrastructure [CH11a] [MV06] [NPM11]. The increasing introduction of renewable energy sources such as solar panels or wind turbines brings unprecedented volatility and unpredictability in energy production for the grid. This, along with the rise in local storage technologies, can significantly disrupt the current distribution of energy flows. Hence, the grid's control infrastructure must change to deal with the new energy fluctuation patterns. Information Technology (IT) is considered as a necessary extension for transforming power grids into "smart" grids capable of answering such requirements. Most industrial parties and research communities seem to agree that the smart grid's control should be more decentralised than the traditional one, even if no consensus seems to exist so far concerning a particular control solution [CH11a] [MV06].

While essential to the viability of modern grids, IT-based control also has the potential to introduce new challenges. Since more and more energy production and storage facilities can be owned privately, the energy market is becoming progressively less centralised. Many actors - including private consumers - can join and define their own power-management objectives, such as business profits, bill minimisation or environment preservation. The smart grid's IT infrastructure is a key enabler for such novel objectives, since it enables the independent control of various grid parts - from countries and areas to districts, houses or devices. At the same time, such new facilities can further increase the grid's dynamism and unpredictability, hence making its control even more challenging [NPM11]. Furthermore, they introduce additional requirements on the smart grid's overall design.

In this chapter, we first introduce a simplified grid model and present the objectives of "smart" power management. We illustrate this "smartness" via a series of scenarios showcasing possible domestic usages of "smart" appliances and houses. Then, we establish requirements for the future grid control infrastructure to actually achieve the vision our scenarios presented, via an autonomic approach. We relate these requirements to the previously identified complexity factors (dynamism, openness, heterogeneity, scalability and multiple objectives) in complex systems, and we emphasize the necessary integration of multiple autonomic controllers in future smart electricity grids.

3.1 Power management in smart grids

3.1.1 A simplified grid model

This section presents a simple grid model that we use as a support for illustrating power management issues. The model is arguably naive in many regards and intends to provide only an intuition of electrical networks' behaviour, in particular for non-experts in the domain of electricity. The model is implemented in the MisTiGriD platform chapter 8 presents.

An electrical network can be seen as a tree [PBC12]; in reality, several graph-like networks can be interconnected. Leaves represent end-user producers (e.g. power plant or solar panel), consumers (e.g. city lights or washing machine) or both (e.g. batteries). Intermediary nodes (i.e. non-leaf nodes) are

aggregators that act as an interface between their children (leaves and/or other aggregators) and the encompassing father network.

The term **prosumer** designates electrical nodes, appliances (leaf) or aggregators (non-leaf). The associated term **prosumption** means either production or consumption, expressed in Watt (W). By convention, a positive prosumption represents a production and a negative one a consumption. For instance, an oven consuming 200W at a certain point in time has a prosumption of $-200W$, whereas a solar panel producing 300W has a prosumption of $+300W$.

Internal tree nodes (aggregators) delimit hierarchical sub-networks. For instance, a house electricity meter defines the house's local network, which is itself nested in a district network, which is part of a city network and so on. From the point of view of its parent, a sub-network can be seen as a single prosumer featuring a total prosumption that equals the sum of the individual prosumptions of its child prosumers. For instance, let us consider a house where at some instant a solar panel produces $300W$ while a TV set and an oven consume $100W$ and $250W$ respectively. From the perspective of the parent district network, the house is equivalent to a single prosumer prosuming $P_{house} = 300W - 100W - 250W = -50W$ - i.e. a consumer at the moment. The same applies to the district network, whose total prosumption as seen from its own parent network equals the sum of every house prosumption. More generally, each aggregator Agg is equivalent to a prosumer with a prosumption p_{Agg} equal to the sum of prosumptions of its children C:

$$p_{Agg} = \sum_{child \in C} p_{child}$$

Whenever a grid's prosumption is not null the parent network must compensate for the difference (e.g. in the house example the district provides $50W$). If available, this amount of power may come from a local source, like the production of another house, or from the district's parent network. Otherwise, a **blackout** may occur and damage connected prosumers.

The **load** of a grid is defined as the ratio between its total productions and consumptions. A *high* load corresponds to a lack of production or an over-consumption. A *low* load indicates the opposite: a low consumption or an over-production. Consequently, load adjustments can rely on both production and consumption control. For instance, shutting down an electrical heater consuming $200W$ is equivalent, in this model, to switching on a battery that would produce $200W$ during the same period of time.

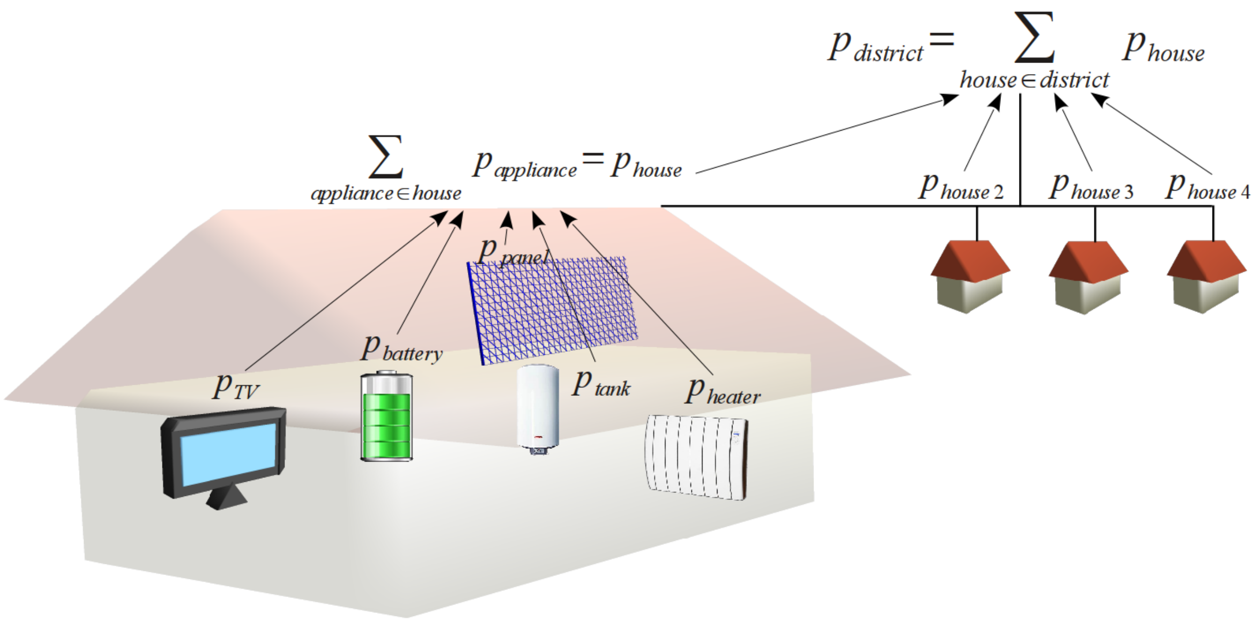

$p_{district} = \sum_{house \in district} p_{house}$
$\sum_{appliance \in house} p_{appliance} = p_{house}$
$p_{house\,2}$
$p_{house\,3}$
$p_{house\,4}$
p_{panel}
$p_{battery}$
p_{TV}
p_{tank}
p_{heater}

Figure 3.1: Sample district grid and local house appliances.

3.1.2 Power management in the grid

Power management in the smart grid aims to maintain appropriate load levels in every grid part (or sub-network) and to avoid blackouts. *A priori*, energy producers try to maximise sale profits whereas consumers to minimise their electricity bill. There is a high variety of possible regulations in energy markets, stipulating prosumer rights, electricity prices and control modalities. This thesis does not assume any specific pricing system or market regulation. Instead, we suppose that an administrative authority is in charge of each grid sub-network, at each granularity (e.g. house grid and district grid). Such authority will have a prosumption objective for their grid, in the form of a viability interval $[p^{min}, p^{max}]$, which they can change over time. Such intervals are chosen arbitrarily to ensure the necessary flexibility for embracing a wide range of realistic administrative goals later on.

Traditional power grids consisted of a large majority of consumers supplied by a few large producers. Load control used to be quite centralised, adjusting production to consumption estimates at a regional or national level. With the progressive introduction of renewable energy sources, particularly solar panels, and the development of low-cost batteries, every end-user may also become a producer.

This thesis focuses on low-tension small-scale electrical networks, or **micro-grids**. The typical example is that of a district grid regrouping a few dozens houses, each featuring variable prosumption profiles. Electric appliances provide services such as heating or entertainment. Private users are arguably self-interested and concerned with goals such as comfort and local power management. These are translated into heating or entertainment services or into a bill reduction. This means that the grid must cater for conflicting interests which must be balanced, since maintaining a warm home may interfere with load peak or bill reductions.

Let us illustrate the possible future usages of smart appliances in smart houses, via a series of fictional scenarios.

3.2 Smart micro-grid scenarios

The scenarios this section presents describe possible usages of "smart" (autonomic) appliances and houses in a near future [FHM+12]. In this thesis, they will serve as a recurrent basis for illustrating relevant properties of autonomic systems. In chapter 8 we propose a simulation platform that allows reproducing significant parts of the scenarios presented here.

3.2.1 Heating system scenario

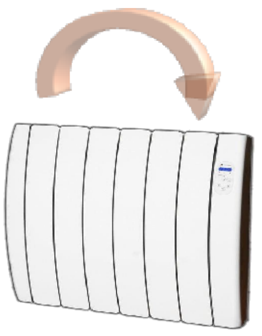


Figure 3.2: Autonomic heating system.

> "Taking advantage of renovation work, Mark has invested in an autonomic underfloor heating. Thanks to a set of sensors and actuators, the system can monitor temperature in the room and receive instructions. A hot water tank allows accumulating and returning heat; this is done according to electricity rates, to Mark's orders, to the weather, to the actual temperatures in the rooms and to whether or not someone is present in the apartment. Mark has also activated a geo-location functionality: aware of Mark's position, the system can reduce temperature while he is out and ensures everything returns to normal when he comes back.
>
> Mark may monitor his system live state: electrical consumption, desired temperatures

compared to actual temperatures, money saved thanks to the water tank stock. The initial investment was substantial but Mark sees a difference on his bill and can be confident he will get return on investment. In addition to ecological considerations on energy consumption reduction, the regulator adapts to the occupants' desires via simple instructions. Reading his system history, Mark has realised that some occasional adjustments he did by hand - for instance, in his bathroom - have been learnt by the autonomic system that anticipates his morning shower since then. Mark's inquiring mind has led him to play with his regulator's most advanced functionalities. He has been offered a choice of management profiles, favouring comfort, energy savings or bill reduction. After some experimentations, Mark eventually decided to follow an highly economical policy that reduced again his electricity bill. And when guests are expected, the system can automatically restore more suitable comfort conditions."

3.2.2 House scenario

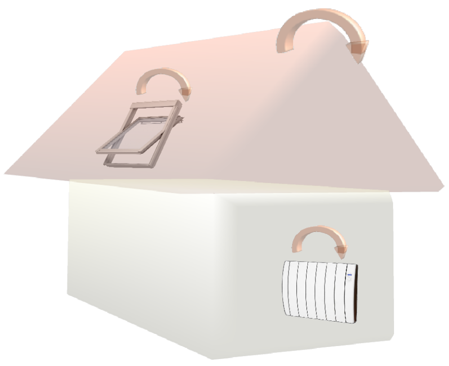


Figure 3.3: Autonomic house.

"Mark keeps testing new equipments in his flat and installed a sophisticated skylight featuring a controllable power-driven actuator and an autonomic controller. The latter fetches data produced by temperature and presence sensors in the room, reads objectives received by the heating system and follows local weather forecasts online. Aside from manual remote control, the controller proposes several programs for airing the room while maintaining its temperature.

After making sure that the skylight's functioning does not create security problems, Mark gives it order to open for 10 minutes a day, provided nobody is in the room and external conditions (temperature, rain, wind) do not threaten the room's temperature objective. In case unfavourable conditions persist, the skylight does not open and reports the event. From studying its systems' journals, Mark has noticed that far from disrupting energy saving objectives, the skylights participates to temperature regulation: when outside temperature condition are favourable, the skylight opening eases the heating system's task."

3.2.3 District scenario

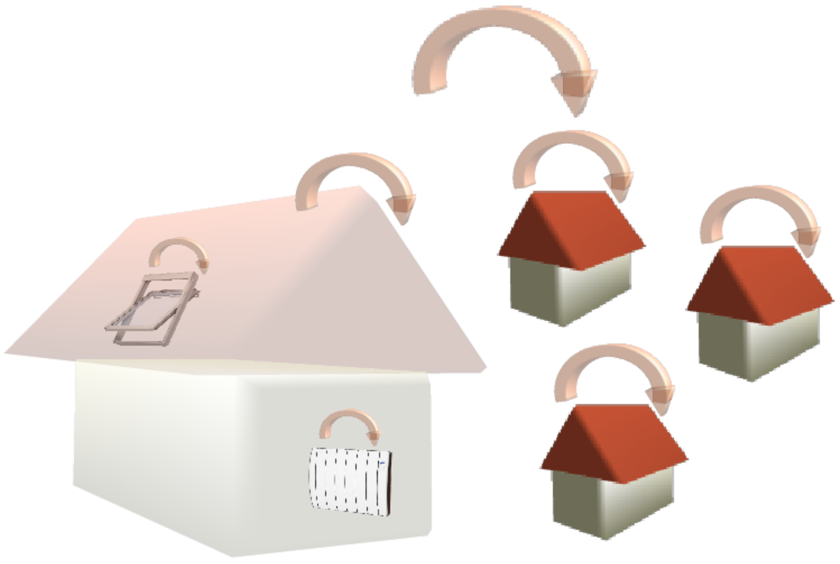


Figure 3.4: Autonomic district.

"Mark's neighbour, Sal, just started her washing machine. Her flat not being equipped with means of production nor stocking, it is part of the consuming homes in the district. Mark's flat is now equipped with an autonomic battery, in addition to existing equipments, therefore it is capable of stocking and reselling electricity.
Detecting that the district grid load increases, the system performs an internal analysis, in order to decide what to do next. It gathers the following analysis: the battery is fully loaded, the flat's room has been aired already, no particular consumption is planned for the time being and a priori, the hot water tank should be able to face any thermal hazard by itself.
Thus, Mark's system decides to dedicate part of the battery's stock to external demand. The buyback rates are favourable to Mark; on her side, Sal benefits from cheaper local electricity.
Later on, Mark's unexpected return increases his home's consumption. Starting an intense housework session usually happening on week-ends, Mark has opened the windows wide.

The heating system, unprepared, is compelled to spend its stock and even get power from the battery in order to restore a comfortable temperature.
Mark's system not being able to provide the district grid without threatening its own interests (on both thermal and economical aspects) it ceases being a producer for its neighbours.
Sal's home has lost a potential provider."

3.3 Requirements for the grid's control infrastructure

Smart grids must comply with numerous requirements, including quality of service, efficiency, reliability, robustness and extensibility, to only mention a few. In this section, we focus on the subset of controller requirements that illustrate our contribution to the design of complex autonomic systems. In chapter 2 we established a series of complexity factors, namely dynamism, openness, heterogeneity, scalability and multiple objectives. We exemplify these factors in the case of smart grids, and identify the corresponding requirements:

- ◇ multi-authority & multi-level
- ◇ multi-objective
- ◇ heterogeneity
- ◇ scalability
- ◇ incremental change & openness

The multitude of interested parties involved and the diversity of their assets and business objectives are likely to prevent general consensus over a unique control solution for the entire grid. Rather, heterogeneous control solutions may have to be adopted in different grid sections as better suited to their particular settings and goals. The pursuit of private goals in various smart grid parts (micro-level) may often prove incompatible with the global goals of the overall grid (macro-level). Finally, smart grids are developing in parallel with adjacent socio-technical systems such as smart appliances, houses and cities, which feature their own constraints and objectives. Integrating such smart technologies within the smart grid will be a likely source of further management conflicts.

The ad-hoc integration of heterogeneous multi-objective control solutions for the grid, entangled with other smart systems, may jeopardise the very benefits that motivated the current transition towards distributed power systems. Hence, great care must be taken when designing the smart grid's general

integration infrastructure, to prevent global disasters from emerging out of local self-interested decisions. Composition of disparate, local control solutions into coherent organisations, ensuring a fair balance between micro and macro objectives, is a key step of the design process.

As a consequence, we identify the following integration requirements in the second part of this section:

- ◇ flexible micro-macro integration
- ◇ (meta-management) feedback

3.3.1 Complexity Requirements

Multi-authority and multi-level

Several administrative authorities interact within the smart grid, each one defining their objectives over a grid part. Each authority's part (e.g. house grid) is connected to parts of other authorities (e.g. other house grids) and may be included in larger parts of higher authorities (e.g. district grid). Authorities may be selfish, and the smart grid must cater for all their interests, in a fair manner. From a control perspective, this implies splitting the grid into multiple levels, each managed by a different controller. Controllers must then be integrated to ensure the coherence of the overall system. This thesis focuses on residential micro-grids, where relevant authorities include private customers and a district administrator. At larger grid scales, regional, national or even international interests are at stake.

Multi-objective

The authorities controlling the grid may want to define additional objectives, which may interfere with power management. In this thesis, we consider the case of balancing domestic services with power management at the house and district levels. Here, grid controllers must be able to follow conflicting objectives, such as "maintain temperature" and "reduce consumption". These objectives may vary in time, depending on user preferences. Additional conflicts may appear between administrative entities, for instance when the comfort objective of private users threatens the load-peak reduction objective of a grid administrator.

Heterogeneity

Electric devices, and prosumers in general, are widely diverse. Different devices in dissimilar contexts require specific controllers, which must also consider user preferences. Moreover, the multi-level requirement implies a high diversity of managed grid scales, goals and characteristics (e.g. voltages, intensities). Finally, the way of dealing with multi-objectives may differ depending on the authorities involved. For instance, *collaborative* controllers would be suitable for managing a house and its appliances, since under the sole authority of the home owner. The owner's objectives should not be overridden by higher grid authorities, except for extreme contexts such as blackout threats. Conversely, *competitive* controllers would be better suited to handle administrative conflicts among home owners, and with the district manager.

Scalability

Modern grids include significant numbers of prosumers, making the scaling of control algorithms a pending challenge. Additionally considering heterogeneity and dynamism exacerbates this challenge. Indeed, the hierarchical nature of the grid (cf. "multi-level" requirement) already equips architects with a divide-and-conquer approach, splitting the grid into nested management levels. Yet, even at the scale of a city, thousands of houses with millions of devices can raise important scalability concerns.

Incremental change and openness

The power grid is a massive asset that has been constructed progressively and has been running for decades. Any introduced in the future will have to be progressive, so as not to disturb or jeopardise a working system that provides an essential service. Therefore, replacing large parts of the grid at once (either network or controllers) seems unrealistic and probably non-recommended. Indeed, progressive evolution involving intermediate, stable forms, seems to characterise many complex systems [Sim69] [UD11].

A notable constraint is the smart grid's business model: smart houses, *a fortiori* smart cities, will not be constructed at once while ignoring existing assets. Instead, *smart grid-enabled* appliances are likely to come into the market progressively, replacing existing systems, until smart houses, districts and cities can start emerging. This means that smart grid controllers requiring large numbers of prosumers will become viable only once such numbers are available. Similarly, smart grid-enabled prosumers relying on third-party controllers will only be useful once such controllers are installed.

3.3.2 Integration Requirements

Since no single control design can address all these requirements across the entire smart-grid, a **mixed solution integrating several heterogeneous controllers** is needed instead, which raises additional integration requirements. First, there is a need for **standard taxonomies and protocols** to allow:

- ◇ integration of heterogeneous, third-party appliances and controllers (cf. "incremental change" requirement).
- ◇ self-adaptation of controllers to changes in their micro-level environments and neighbouring controllers.
- ◇ macro-level self-organisation for grid-wide goals based on heterogeneous, self-interested controllers.

Significant standardisation efforts are already supported by entities such as the National Institute of Standards and Technology (NIST), OASIS Energy Market Information Exchange (EMIX) and Energy Interoperation technical committees. These aspects go beyond the scope of this thesis.

Flexible micro-macro integration

A smart grid must ensure a balance between individual (micro) and collective (macro) objectives, based on user specifications and grid regulations. Since these can change over time, smart grid solutions must be flexible enough to rebalance objectives at runtime. Hence, the capability of a macro-level controller to override micro-level objectives, and vice-versa, should be an adaptable parameter of the management system. In the grid, unless under extreme conditions, a prosumer with sufficient priority may be allowed to reach its local objectives, even if this prevents larger grid objectives from being fulfilled. For instance, a heater could be authorised to consume its maximum power for maintaining a comfortable temperature; power will be drawn from the macro-grid to compensate.

Meta-management feedback

The trade-off between micro and macro objectives is a meta-management goal in itself, since it dictates the conflict-resolution strategy and hence the system behaviour. Administrators should receive continuous feedback on their management decisions, in both collaborative and competitive situations. In the previous heater example, the house owner should be notified that their decision to maintain high comfort level will come at a price. Conversely, in case the user chooses an extremely economical policy the system should warn of possible comfort degradations.

Feedback is also crucial in competitive cases, such as the district. Here, continuous analysis of each actor's behaviour is key to the long-term refinement of tariffs and legal regulations that would ensure fairness and viability in the district grid market. More generally, meta-management is required for adapting controllers to changes in the environment (e.g. new devices entering the system), learning models of the context (e.g. patterns in the user's habits, the local weather) or simply maintaining up-to-date management logic (e.g. updating the security protocols of controllers).

Chapter conclusion

In this chapter, we presented the role of autonomic systems for power management in future electrical grids. We presented a series of scenarios that illustrated the importance of autonomic capabilities in this context and that we will use as a recurrent illustration basis in this manuscript. Then, we established a series of requirements for autonomic grid controllers to actually achieve the vision proposed by the scenarios. These requirements relate to the complex nature of the grid, and highlight the necessity of integrating multiple autonomic systems with conflicting objectives in a flexible, adaptable fashion, so as to ensure the overall coherence of the resulting global system.

Chapter 4

Thesis position

Quand il saura ce qu'est le monde, ce à quoi se résume le réel (...)
et qu'il consentira à cet état de fait (...)
alors il créera quelque chose d'inédit (...)
Michel Onfray, *La Construction du Surhomme*[1]

Chapter introduction

This chapter positions the thesis with respect to the state of the art, enunciates the challenges we address (section 4.1) and introduces the main contributions (section 4.2). These contributions are then detailed in the following chapters.

Section 4.1 presents the challenges we address. In this thesis, we aim to address the limitations identified in the state of the art by providing reusable support for the design of complex autonomic systems. Indeed, chapter 2 showed that the concerns of dynamism, heterogeneity, openness, large scales and multiple objectives in autonomic systems are only partially addressed by state of the art literature. The concomitant occurrence of these concerns is common in many of the application domains of autonomic computing, of which the smart grid domain is a particularly relevant example. The lack of generic, reusable design support for autonomic system designers prevents the partial specific solutions that have been developed for the aforementioned concerns to be re-used and combined, in order to form comprehensive solutions addressing all the concerns at once. This hinders the progression of both the domain of autonomic computing and its possible applications, such as smart grids. This diagnostic

[1] *When he has known what the world is, what reality comes down to (...) and when he has agreed to it (...) then he will create something novel (...).* Michel Onfray, *Constructing the Übermensch.*

motivates our software engineering approach, based on the development of generic, reusable engineering artefacts such as architectures and design patterns.

Therefore, a generic conceptual basis is necessary to help autonomic system designers to recognise the aforementioned factors of complexity and understand the design challenges they rise. Here, we focus in particular on integration-related concepts at two different granularities. At the control loop granularity, integration approaches are necessary to achieve modular, adaptable multi-objective autonomic managers. At larger granularities, integration approaches are also necessary to organise multiple stand-alone autonomic systems. Illustrating and validating possible approaches on relevant use-cases, demonstrating their applicability and highlighting the complexity issues they resolve, is also a primary concern.

Section 4.2 presents the contributions of the thesis. We propose a formalisation of management goals that helps modelling open multi-objective autonomic systems at a high-abstraction level. As management conflicts are a major issue in such systems, we identify generic resolution semantics for pinpointing and addressing these conflicts. The formalisation also helps modularising the control layer into modules which must then be integrated. We propose a modular architecture and a series of integration design patterns for building adaptable multi-objective autonomic managers. At coarser granularities, we propose an organisational model and integration solutions for building open organisations from third-party autonomic systems. The proposed architectures and integration patterns are exemplified via concrete design solutions for the smart micro-grid use case. Finally, we present MisTiGriD, a software platform that provides a dynamic micro-grid demonstration environment. We developed MisTiGriD in order to implement the proposed architectures and patterns, showcasing the capabilities of our approach on a live system.

4.1 Challenges addressed

4.1.1 Need for generic architectures and conceptual frameworks

Conceiving and developing autonomic systems requires multiple types of expertise - in the system's business domain, in autonomic systems, in control theory, in software engineering, and in artificial intelligence. Yet, little conceptual support is available for domain experts to facilitate reasoning about autonomic systems and their implementation. At the same time, little design support is available for autonomic computing experts to apply their generic understanding of self-management to particular use cases. The absence of generic control architectures that would apply to a variety of situations reveals a certain lack of maturity in the domain. As a result, communication among these different domains of expertise is hindered, which cripples the progress of each individual community.

Generic support is even more critical in the case of complex autonomic systems, i.e. open, heterogeneous, large-scale dynamic systems with multiple objectives. As we reviewed in chapter 2, solutions to each complexity factor taken separately do exist in the literature. However, no generic approach is available to unify such partial solutions within the domain, as would be necessary to provide a reusable basis for the design of complex autonomic systems. In this thesis, we focus on modularisation and integration issues as unifying concerns, as detailed in the next sub-section.

4.1.2 Need for an integration-oriented approach

Single-loop approaches have proved successful for a variety of systems, as presented in chapter 2. However, system complexity prevents architects from relying on a single manager, as monolithic controllers do not scale in size and complexity with respect to the required management functionalities. Monolithic managers are also unsuitable for open contexts where heterogeneous, dynamic resources, control components and sometimes entire stand-alone autonomic systems must be integrated in the system at runtime. State of the art practice provides limited support for integration of third-party autonomic systems, yet controller integration is key to managing system complexity, for several reasons we review here.

On one hand, "divide and conquer" approaches are necessary to achieve complex control solutions [Mau10] [SD06]. In such approaches, system designers first divide management loops into modular, re-usable components that can address a subset of the control functionality. Then, the components are integrated back into a complete, coherent, complex controller. This fragmentation-integration design of control loops motivates the use of architectures such as MAPE, as presented previously.

State of the art architectures feature limited support when it comes to integration: they provide little or no details on the way in which atomic control components communicate and coordinate with each other, and they do not support multi-loop implementations of autonomic managers. This is particularly limiting when considering multi-objective autonomic systems with different control loops corresponding to their different management goals.

In addition, even a carefully designed control logic needs to be adapted over the long run to cope with changing environments and requirements. Modularisation is a well-established Software Engineering approach to address the need for long-term evolution. It helps in particular to adapt control modules, integrate new modules and extend the overall logic at run-time, in an incremental fashion. Therefore, integration capabilities are also critical to achieve adaptable, extensible control systems.

On the other hand, autonomic systems are expected to emerge out of the combination of stand-alone autonomic systems [DSNH10] [ABS11] [WHH^{+}10], as we envisioned in the case of future smart grids in chapter 3. Therefore, the ability to integrate full-fledged autonomic systems into higher-level organisations is a key issue for system architects. Here again, generic integration solutions are necessary to ensure that composition of independent third-party systems does not result in undesirable, uncontrollable collective behaviours. This raises some critical issues such as:

- ◇ identifying and controlling unexpected interactions amongst integrated managers.
- ◇ ensuring the global consistency, scalability, adaptability and extensibility of the resulting organisation.
- ◇ maintaining a balance of power between integrated managers, and in case the organisation as a whole pursues collective objectives, between individual managers and the collectivity.

Solutions from the multi-agent domain or from control theory partially address such concerns, yet their integration with state of the art autonomic architectures must still be determined. In particular, generic integration solutions, allowing autonomic systems to be truly adaptive and open must still be determined [DSNH10].

In conclusion, achieving complex control capabilities requires integrating diverse control components into adaptable, multi-part, multi-objective management systems. This work focuses on defining such integration issues and providing generic design solutions as integration design patterns.

4.1.3 Challenges in the case of smart micro-grids

In the specific case of smart micro-grids, the generic issues identified above lead to the following particular questions:

◇ Why is an autonomic computing approach relevant to the smart grid? How to design the smart grid control infrastructure in an autonomic fashion? How does it fit the needs of smart grid experts?

◇ How to design the control infrastructure of the smart grid so as to consider multiple objectives in addition to power management? How to ensure flexible management with respect to usages (e.g. comfort or entertainment) as well as power management? How does autonomic control benefit smart grid actors such as private users, district administrators, device providers, and why?

◇ How to define control domains relevant to the different administrative entities present in the grid? Where should management resources be deployed?

◇ How to enable integration of third-party heterogeneous controllers into the existing control infrastructure on the long term? Which properties can be ensured concerning the grid behaviour, at different granularities (appliance, house, district), out of arbitrary, dynamic controller compositions? How to enable the control infrastructure for long-term evaluation and balance its multiple objectives?

The contributions of this thesis aim to provide a base design for smart micro-grids which starts answering these questions.

4.2 Contributions

In this thesis, we propose the following contributions:

◇ **A generic formalisation of management objectives, conflict situations and possible conflict resolutions**. Multiple objectives are a major cause of integration issues when they conflict with each other. The proposed formalisation allows to decouple a system's goals - *what* it achieves - from its control implementation - *how* it achieves it. This formalisation helps system designers to separate management concerns in complex, multi-part, multi-objective systems, and thus favours modularisation. We provide a way to identify conflicts between objectives, and formalise generic resolution semantics for integrating the conflicting parts of a system. We illustrate this contribution via a concrete application in the smart micro-grid context. Here, we formalise the multiple objectives of this complex system and establish an **holonic control structure** solving management conflicts at several granularities - appliance, house and district.

◇ Support for an integration-oriented approach for designing modular, adaptable autonomic managers via a **generic architecture** extending and generalising state of the art approaches. In this architecture we identify the main types of control and integration components that are necessary for building modular autonomic managers. We propose generic solutions to common integration issues among control components in the form of a catalogue of **integration design patterns**. The patterns are characterised by an abstract definition and exemplified by several concrete examples in the literature. We provide a comparison of the patterns according to several metrics such as extensibility, scalability, performance and control accuracy. We illustrate this approach via the concrete design of a simple autonomic heater manager, that we extend and adapt to variable environments and requirements. We indicate how this approach can be extended to design more complex autonomic managers, such as a an entire smart home system.

◇ **Integration patterns for open, adaptable organisations of heterogeneous, stand-alone autonomic systems**. These patterns provide adaptable integration policies for enforcing variable balance of authority among their individual participants and their multiple objectives. In case the organisation intends to attain collective objectives, administrators can adapt the balance between individual and collective interests. The organisations provide variable characteristics in terms of possible balance of authorities, flexibility or scalability that we compare and exemplify with canonical examples in the literature. We illustrate this contribution via the concrete design of several **power management organisations** in the mart grid case. These organisations fit the holonic control structure we established and formulate conflict resolution in terms of a power scheduling problem. We show how our integration patterns solve this scheduling problem in different ways, which we then analyse and compare.

◇ A **proof of concept implementation** of the holonic control architecture and power management organisations, relying on the MisTiGriD platform that we developed. MisTiGriD simulates the behaviour of several houses in a district, each house featuring various appliances and controllable environmental conditions, such as temperature. The user is allowed to set dynamic management objectives for appliance managers, which in turn monitor the run-time evolution of the system for attaining these goals. We present a series of **simulated scenarios** inspired by the scenarios proposed in chapter 3, and showcasing the management possibilities of the proposed approach. Notably, we show how our approach permits to balance comfort and energy goals at the house scale, according to changing user preferences and under variable weather conditions. At the district scale, we show how the grid administrator benefits globally from the local flexibility of individual autonomic appliances.

Chapter conclusion

The remainder of this manuscript is organised as follows:

- ◇ chapter 5 presents our generic formalisation of management goals, conflicts and their utilisation for modularising the control of open, multi-objective autonomic systems. We utilise this formalisation in the smart grid use case and we identify a general holonic control architecture for power management.
- ◇ chapter 6 presents a generic architecture for building modular, multi-objective autonomic managers, and a catalogue of integration design patterns. The architecture and patterns are exemplified via a simple smart heater implementation, and we indicate how this approach can be extended to design more complex autonomic managers.
- ◇ chapter 7 presents several integration organisations for societies of autonomic systems, that we apply to propose open, adaptable power management organisations in the smart micro-grid.
- ◇ chapter 8 presents the MisTiGriD platform and several simulation scenarios. We analyse and discuss how these scenarios illustrate and support the validity of our approach.
- ◇ chapter 9 concludes the manuscript, discusses the current limitations of our approach and indicates some future research directions.

Chapter 5

A generic formalisation of goals for multi-objective autonomic systems

Essentially, all models are wrong, but some are useful.
George Box

Chapter introduction

A common characteristic of a complex autonomic system is the multiplicity of its management objectives, whether one considers a single system following several goals or a composition of several stand-alone systems each following their own goal(s). When considering arbitrary combinations of goals, management resources and managed resources, the number of possible mappings between goals and resources and of possible goal overlaps make difficult understanding and designing such systems. Representing goals explicitly can help system designers understand and analyse the control problem at hand, and identify more easily the control modules necessary for addressing this problem in various contexts. Consequently, formalising goals can help develop, adapt and evaluate autonomic systems. In addition, when pursuing multiple objectives, goal formalisation can also help to identify possible interactions between the goals (and/or between the control modules and managed resources they rely on) and the resulting conflicts.

This chapter presents a generic formalisation of management objectives, extending goal taxonomies from state of the art literature. Namely, in section 5.1 we identify several types of **viability constraints** for expressing desirable states or behaviours of autonomic systems. We also introduce the notion of **goal scope** which defines the interactions of management goals with both control resources and managed resources.

Since multiple goals often lead to **management conflicts**, the notion of goal composition is refined and discussed in section 5.2. Corresponding **conflict resolution strategies** are also presented in this section. Next, we formulate **requirements for integration infrastructures** in multi-objective autonomic systems.

To illustrate the applicability of our proposal, in section 5.3 we exemplify our goal formalisation via a residential micro-grid use-case. Here, we analyse the user-specified goals and their overlaps, and we identify their mapping to concrete control modules and managed resources. The resulting architecture is representative of a **holonic** approach, where each control component pursuing a goal is further constructed by finer-grain components pursuing smaller sub-goals. This discussion opens the way for the following chapters that present generic integration architectures for multi-objective open autonomic systems, both stand-alone (chapter 6) and composed (chapter 7).

5.1 Goal formalisation

Goals, or **management objectives**, are also referred to as **policies** in the literature [KW04]. We adopt the generic definition of policies by the later, and adapt it for specifying goals in general:

*A **goal** is any type of formal specification*
provided by an administrator or any external entity with sufficient authority
to guide the behaviour of an autonomic system.

This definition is voluntarily broad and encompasses other standard definitions, such as agent goals in the reference AI book by Russell and Norvig [RN10] or standard autonomic policies [KW04]. However, this definition says little about how to express such “guidance” in more concrete terms. We propose the following refinement for specifying management goals as combinations of **viability constraints** and **scopes**.

Viability constraints formalise the domain-specific expression of a goal, in terms of domain-related “desirable properties” of the system, in other words: *what* the system should do or achieve. On the other hand, scopes identify *where* and *when* viability constraints are evaluated and applied. The two parts are complementary and in practice it may be hard to define one without the other, either explicitly or implicitly. Goal specifications are provided to managers via effectors, and goal evaluations are provided by managers via sensors. This is similar to managed resources being observed and controlled by managers via sensors and effectors respectively, yet at a different level of abstraction (cf. Fig. 5.1).

5.1.1 Viability constraints

The literature identifies three types of viability constraints, exposed below [KW04, RN10]. Concrete goal specifications may define this part in an either declarative or procedural manner.

- ◇ **Action**: defines actions the system should take whenever it enters a given state or detects a given event in the environment. They are usually expressed as Event Condition Action (ECA) rules, following the form:

  ```
  WHEN event IF condition THEN action
  ```

  This kind of *procedural* formulation implicitly supposes that taking the action under the specified condition will guide the system into a desirable state. In general, translating such a rule into concrete implementation code is straightforward. Rules being modular, it is easy to define and extend dynamic sets of rules on systems, possibly at run-time. However, in practice, the combinatorial explosion of event and conditions makes it difficult to establish and to maintain the coherence of large sets of rules. Generic autonomic frameworks based on ECA rules notably include Ponder [DDLS01], Ponder2 [TDLS09] and Automate [ABL$^+$03].

- ◇ **Viability domain**: defines a set S of desired states, within which the system should be maintained. It is up to the autonomic manager to determine how to effectively maintain the system's state in S, in particular, which are the appropriate management actions to be taken for this purpose. This *declarative* formulation of objectives relieves the designers and users from knowing which behaviours the system actually implements. However, unlike *procedural* action definitions, translating such objectives into management actions requires much more sophistication on the management logic, such as modelling and anticipation capabilities. A notable example of viability domain specifications was proposed by the Rainbow framework [Che08] where desirable states of a managed system are expressed in terms of architectural constraints on a generic runtime model of the system.

◇ **Utility function**: defines an evaluation function over the domain of possible system states. It is up to the autonomic manager to maximise its utility by driving the system into the most favourable regions. This is a generalisation of the former viability domain formulation: boolean evaluation functions characterising a desirable set of states S (= viability domain) are a particular case of utility functions. This *declarative* formulation of goals provides great flexibility, allowing for instance to combine several utility functions and let the system find an optimal trade-off via multi-criteria decision processes. However, multiple evaluation functions and their respective weights might be difficult to elicit and balance effectively [KW04].

These different kinds of viability constraints are exemplified via concrete formulations of smart grid objectives in the following sections.

5.1.2 Scopes

Scopes define *where* and *when* goals are applied and evaluated. In general, a scope designates a set of managed resources and environment resources, or the corresponding touch-points (sensors and actuators). The resources can be explicitly named (e.g. “heater x”, “house y”) or indirectly defined by selective criteria (“every appliance in room z”). In case of an indirect definition, it may be up to the management system to actively discover at run-time which resources are present in the scope, which may in turn require an infrastructure for dynamic resource discovery.

Scopes may also feature a temporal component defining the period of time during which the scope is applied (“every day”, “today from 5pm to 9pm”). In this thesis we focus mainly on spatial scopes, considering that temporal scopes are all implicitly “from now on”. More complicated cases are part of future perspective of this work.

Objectives are related to the cyber-physical world in several ways which we identify via the following scope definitions:

◇ **Evaluation scope**: *the set of sensors, manageable resources, or possibly the spatial region over which goal viability constraints are evaluated.* The evaluation scope is usually formulated explicitly in the goal's viability constraints definition. For instance, "switch heater on when *temperature* drops below $20°C$ *in this room*", or "maintain *house consumption* below $1kW$".

◇ **Action scope**: *the set of manageable resources that autonomic managers can control in order to achieve the viability constraints over the evaluation scope.* Contrary to the evaluation scope, the action scope is not always explicit in the goal's viability constraints definition. Goals expressed in terms of action policies normally specify the behaviour of resources that belong to their action scope.

◇ **Context scope**: *this scope identifies all the monitoring information that would be relevant for achieving the goal*; by construction it includes the evaluation scope. The context scope is usually defined by the autonomic manager designers when they identify the interactions of managed resources with their environment. This scope can be extremely wide and difficult to determine. For instance, a temperature regulator may monitor temperatures in neighbouring rooms, or may follow the local weather forecast.

◇ **Influence scope**: *this scope contains every resource that will be affected, directly on indirectly, by pursuing the goal*; by construction it includes the action and evaluation scopes. This scope can be extremely wide and difficult to determine as well. Indeed, influence scopes are the reverse side of context scopes: the influence of a goal A is likely to be relevant for another neighbouring goal B, i.e. A's influence scope should probably define at least a part of B's context scope. For instance, a temperature regulator may influence other temperature regulators in neighbour rooms, due to heat transfers through shared walls. Heater electric consumption may also influence the fulfilling of local electricity management policies, which may in turn influence a number of related electrical appliances.

In this manuscript, the reunion of evaluation and action scopes will be referred to as the **goal scope**, since this is where viability constraints are to be applied directly, either on manageable or management resources (respectively: viability domain and evaluation function, or action policy). Comparatively, the context scope does not contain resources on which the goal is evaluated, therefore there is no necessity to control it for managers pursuing the goal. Resources in an influence scope are affected as a side effect, which is mainly relevant to other goal scopes intersecting this influence scope, as next sections will discuss.

Autonomic managers that **pursue** or **carry** a goal observe resources in the context scope and control resources in the action scope so as to attain the viability constraints, measured on the evaluation scope, as shown on Fig. 5.1.

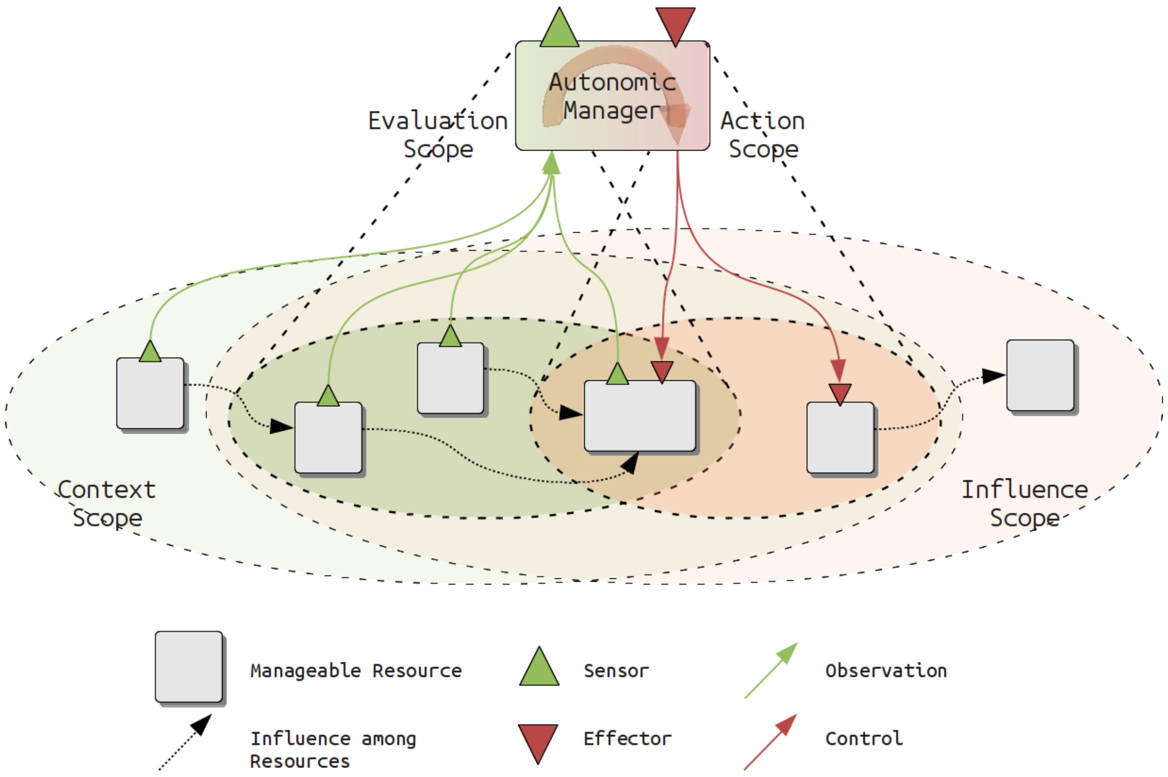


Figure 5.1: The different scopes of a goal.

5.1.3 Two sample management objectives

◇ "Maintain temperature between 21°C and 23°C in the room": the goal's viability constraint is expressed as the temperature domain $[21°C, 23°C]$. The room is the evaluation scope, and more precisely, any thermometer available in it. The action scope may be a controllable heater in the room. The context scope includes, for instance, resources capable of measuring temperature in neighbour rooms (actual or expected) or local weather information. The influence scope also includes neighbour room temperatures, as well as the electrical network due to the heater consuming electrical power.

◇ "Maintain house consumption inferior to 1kW between 6pm and 10pm": the goal's viability constraint is also expressed as a domain $(p < 1kW)$ in conjunction with a segment time scope. The house electricity meter is the only explicit element in the evaluation scope, yet every electrical appliance in the house would certainly be relevant, therefore those belong to the context scope. The action scope depends on the implementation of power management: the house power manager may be allowed to directly control appliances, or only to negotiate with their managers, or possibly just to issue global analysis information to them, such as whether or not the goal is fulfilled. The influence scope will include the whole house, since power management may influence appliance behaviour, which in turn will influence the rest of the house. The local district may also be influenced by the house pursuing its power management goals, for instance, neighbours being asked to reduce their consumption due to the house not managing to limit its own.

5.1.4 Usage and utility of goal formalisation

It is worth noting that one can consider different perspectives when reasoning with goals. From a system administrator perspective, goals are indeed an entry point to the autonomic system, i.e. an effector value, that allows specifying or guiding the system's behaviour. From a designer perspective, goals are a also a way of both formalising requirements and expressing domain constraints for an autonomic system during the design phase, separating system specifications from their implementation [Lan11].

Therefore, goal definitions and usages may vary depending on the perspective. System designers may formalise a goal completely to fully understand the underlying mechanics of a system before designing its controllers. For instance, the temperature goal above specifies a context scope that is relevant for identifying the sensors necessary for an appropriate heater manager. Conversely, system administrators may be only interested in functional aspects of goals, from a business perspective only. For instance, home owners would only need the target range of the temperature goal as a control input to properly administrate an autonomic heater. In this context, scope definitions that explain the underlying functioning of the heating systems are not particularly relevant for the purpose of comfort control.

5.2 Goal fragmentation and composition

By definition, multi-objective autonomic systems are driven by several goals. This multiplicity of goals may be due to several factors. First, a single complex objective may be broken down into smaller, simpler goals, in a divide and conquer fashion. Second, several objectives may be composed with each other, due to a single administrative authority seeking them simultaneously, and/or to multiple parties interacting in the system, each with their own objective(s). This section details different ways of breaking down and composing goals, with particular focus on the case of management conflicts, and the need for conflict resolution features in the underlying control architecture.

5.2.1 Goal splitting

A goal G can be **split** recursively into sub-goals $g_1 ... g_n$. More precisely, scopes S_g can be split and form a partition of the main goal scope, as follows:

$$\forall i, j \in [1..n],\ S_{g_i} \cap S_{g_j} = \emptyset$$

$$\bigcup_{k \in [1..n]} S_{g_k} = S_G$$

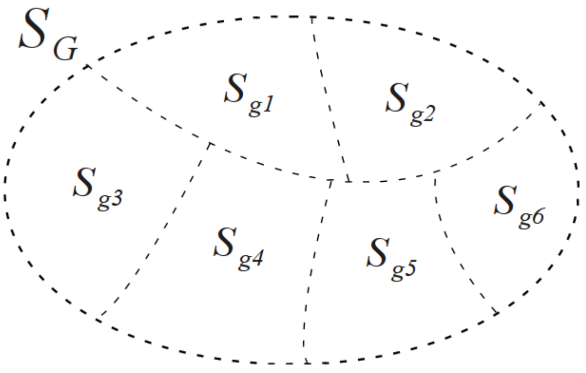


Figure 5.2: Sub-goal scopes partitioning main goal's scope.

Viability constraints of the sub-goals may be related to the viability constraints of the main goal in different ways. *Intensive* goals would keep their viability constraints unchanged through splitting, whereas *extensive* goals would see their viability constraints scale to their scope, by analogy with the meaning of these terms in thermodynamics [Ren].

For instance, consider G_h^T a temperature goal for a house: its viability constraint is to maintain a temperature of $[22°C, 24°C]$ evaluated over the house scope. Splitting G_h^T into several sub-goals G_r^T, one for each room r in the house, preserves the viability constraint: each room receives the same target interval as the whole house. Therefore the temperature goal is intensive.

On the other hand, consider G_h^P an (extensive) prosumption goal for this same house: its viability constraint is to maintain total prosumption in a certain range $[P^{min}, P^{max}]$. Splitting G_h^P into a prosumption goal for each appliance would certainly require providing each of them with a custom target prosumption range, adapted to their specificities. In turn, having each prosumer achieve its sub-goal will result in achieving the overall prosumption target of the entire house. Conversely, a prosumer not achieving its power goal may require supplementary efforts from other prosumers in the overall scope to compensate. The micro-grid example will be exposed in details when proposing a sample holonic control structure for the smart grid, at the end of this chapter.

High-level viability constraints can also be split into lower-level viability constraints applied on the same scope. For instance, an “high-level comfort” requirement from the user for her house can be split into several constraints on temperature, air quality and light levels.

Splitting goals helps addressing scalability concerns: large scopes involving significant amounts of resources can be separated into smaller scopes, applying a typical “divide & conquer” approach. In cases such as power management in the grid, goal splitting is also tightly related to the underlying mechanics of the electricity network, namely prosumption aggregation. Concerns such as multiple administrative authorities or data privacy may also motivate a goal splitting. For instance, in the grid, end users may not want their personal prosumption data being available outside of the house scope, for privacy and security reasons.

5.2.2 Goal translation

A common source of difficulties when designing autonomic systems comes from the abstraction gap that must be bridged when mapping high-level goals onto concrete actions on managed resources. Several successive abstraction layers, each one translating a goal into a lower-level goal, may be necessary. Translation typically reformulates a goal's viability constraints while refining the goal's scope accordingly to include the significant resources on which the new viability constraint are applied or evaluated.

For instance, consider a comfort goal for a room. This goal can be translated into a temperature goal for the same room by a comfort manager. This temperature is in turn ensured by a hysteresis thermostat setting a heater's emission power. The implementation details of possible smart heater controllers are presented in chapter 6 of this thesis.

viability constraint	evaluation scope	action scope
“warm comfort”	users in room	room comfort manager
$T \in [21°C, 23°C]$	thermometer in room	heater hysteresis thermostat
$P_{heater} = 200W$	heater control interface	heater control interface

Figure 5.3: Example of successive abstraction layers for an autonomic heater goal.

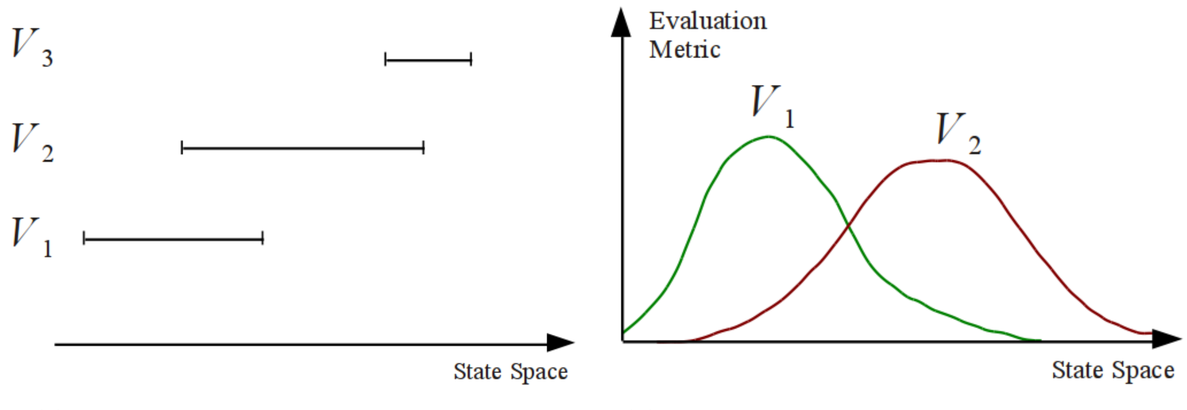
5.2.3 Goal conflicts and resolution requirements

Definition of management conflicts

When multiple management objectives target the same system, it is likely that there will be incompatibilities between them. Using the goal formalisation we introduced, **management conflicts** may happen when incompatible viabilities V_i are specified on intersecting goal scopes S_i:

Two management objectives $G_1(V_1, S_1)$ and $G_2(V_2, S_2)$ conflict when $S_1 \cap S_2 \neq \emptyset$ and V_1 and V_2 are incompatible.

Possible incompatibilities between viabilities V_i are detailed below. The scope intersection can be referred to as the **conflict zone**, and the autonomic managers carrying the conflicting goals can themselves be qualified as **conflicting**. In case evaluation scopes overlap, the conflict zone will contain monitored resources which are expected to evolve in different ways, due to conflicting autonomic managers carrying different viability constraints and acting (directly or indirectly) on these resources. In case action scopes overlap, the conflict zone will contain managed resources that may receive contradictory orders issued by the conflicting managers. Such conflicts due to overlapping goal scopes are *direct*; *indirect* conflicts due to overlapping context or influence scopes are defined below.



(a) Example of conflicting viability domains. (b) Example of conflicting evaluation functions.

Figure 5.4: Types of conflicts among viability constraints.

Different viability constraints do not necessarily signify complete incompatibility between the goals. Indeed, different viability constraints may be partially or completely compatible with each other, as shown on Fig. 5.4:

- ◇ in Fig. 5.4a acceptable state spaces V_1 and V_2 are different while still overlapping, which means that a common solution can be found in the intersection. On the other hand V_1 and V_3 do not intersect each other and are hence clearly incompatible. In case there are many conflicting goals, intersections between acceptable domains of the state space might be difficult to compute.
- ◇ in Fig. 5.4b different evaluation functions target different optimal points. Their combination still allows looking for acceptable trade-offs (see conflict resolution below).
- ◇ in case viability constraints are defined via action policies, the same situations apply while reasoning on action spaces rather than state spaces. Actually, in the end, all conflicts eventually come down to contradictory actions performed on managed resources.

Direct conflicts can be detected by comparing the goals' scopes, which are explicit in general. *Indirect* conflicts, on the other hand, occur due to a goal's influence scope overlapping with another goal scope or context scope. These scopes are often implicit and may depend on unpredictable runtime conditions, hence indirect conflicts may be hard to detect. For instance, an autonomic heater may indirectly heat neighbouring rooms, therefore these rooms belong to the heater goal's influence scope. This indirect heating may affect temperature objectives evaluated in the neighbouring rooms.

Conflict resolution logic & integration infrastructure requirements

Conflict resolution consists in resolving or preventing conflicts from occurring and/or having unwanted effects on the managed system. It requires dedicated conflict-resolution logic which should be inserted at key points in the system's control logic. This resolution logic may follow various strategies, depending on the conflict characteristics:

- ◇ in case goals define different but compatible domains for the system (e.g. V_1 and V_2 in Fig. 5.4a), i.e. there is a non-empty intersection of evaluation scopes, conflict resolution consists in ensuring the system remains in this intersection. In this situation there is a priori no need for extra conflict resolution logic. However, in case acceptable domain boundaries change at runtime, scope intersection may become empty (cf. next case).

◇ in case goals define incompatible acceptable domains for the system (e.g. V_1 and V_3 in Fig. 5.4a), conflict resolution consists in choosing which one(s) should take over the others, and cancel the competitors ("**winner-takes-all**" resolution logic). Such meta-objective could rely on user-set goal priorities providing an ordering on conflicting goals. This approach is simple to specify and implement, yet cancelling goals completely might not be acceptable by the user, and may leave parts of the system out of control. For instance, cancelling a temperature objective for achieving a power objective may lead to unacceptable comfort levels. Mitigating evaluation functions, as shown next, allows smoother degradation of objective fulfilment, and can be used as heuristics for finding better trade-offs than conflicting viability domains.

◇ in case conflicting evaluation functions are defined, conflict resolution consists in finding an acceptable trade-off between conflicting goals ("**mitigation**" resolution logic). In the later case, fine trade-off can be found via user-set priorities assigned to goals, and multi-criteria optimisation algorithms maximising the overall utility of the system. In practice such solutions may be hard to establish and maintain with arbitrary conflicting goals, yet on well-defined conflict cases they may prove extremely efficient [KW04]. Evaluation functions may also be a solution for better integration between incompatible target domains. Generalising the domains with evaluation functions may provide heuristics to find "still acceptable" conditions close enough to the original target domains. For instance, a $[20°C, 22°C]$ domain can be generalised into an evaluation function maximised on $[20°C, 22°C]$, average on $[18°C, 20°C]$ and $[22°C, 24°C]$ and minimal anywhere else. This allows to express resolution logic that would still favour temperatures "close enough" to the initial $[20°C, 22°C]$ target, in case this one is not achievable.

Specifying conflict resolution logic is a key step of the design of a multi-objective autonomic system. This logic may depend on pre-determined resolution mechanisms (for instance, dropping low-priority goals in favour of high-priority ones) and runtime user preferences (for instance, favouring comfort over energy economies momentarily due to a sick child). In addition, in open systems, possible conflicting goals and management systems may not be known at design time. For instance, when conceiving power management for a smart house, the set of possible conflicts is as extended as the range of possible electric appliances and electricity-related uses in the house.

Aside from the core resolution strategy, system designers should conceive a dedicated **conflict resolution infrastructure** that will determine how the resolution logic is integrated into the control system and how it will impact its functioning and quality properties. Different conflict resolution infrastructures will:

- ◇ determine which kinds of conflicting objectives and autonomic managers the system supports, and whether the conflict zone is open to third-party systems (in which case integration constraints may be imposed on participants), or closed.
- ◇ determine whether the resolution mechanism is enforced via dedicated third-party integration resources, or through multi-tier collaboration expected from systems being part of the conflict.
- ◇ determine whether or not the system's control flow will always pass through a conflict-resolution process, hence impacting system performance, and system robustness when the resolution mechanism fails.
- ◇ determine the possible adaptations of the conflict resolution logic, quantitatively (for instance, switching goal priorities according to user preferences at runtime) and qualitatively (for instance, switching from a "winner-takes-all" to a "resolution" logic).

Chapters 6 and 7 present several such conflict resolution infrastructures, in the form of integration design patterns. More precisely, chapter 6 focuses on the realisation of multi-objective autonomic managers, discussing how to implement stand-alone control loops following several goals. Chapter 7 focuses on organisations of autonomic managers, discussing how to establish open collaborations of third-party autonomic systems integrating both individual (micro-) objectives and collective (macro-) objectives.

5.3 Application: holonic architecture for grid control

This section presents an application of the proposed goal formalisation to the case of smart micro-grids. We study possible goal specifications and propose a holonic control architecture that fits the hierarchical topology of the grid. This architecture requires several integration infrastructures, notably for power management, that will be presented in chapters 6 and 7.

5.3.1 Goal specification for power management

Several administrative authorities may set power management goals in the micro-grid: the "district administrator" in charge of the district grid, and private users in charge of their house grid. Many reasons could motivate power management goals: commercial agreement with a higher-level energy provider (district grid for private users, regional grid for the district administrator), maximisation of profits on a local energy market, reduction of environmental footprint, and so on. These reasons are tightly related to the future business model of the grid, which is yet to be determined and out of the scope of this thesis. Our proposal is therefore kept generic enough to be later adapted to the specific forms grid control may take in the years to come.

We adopt a generic objective model that is agnostic with respect to business aspects, where administrators set a range of acceptable prosumption levels for the grid they are in charge of. The boundaries of this range are arbitrary and set by administrators at runtime, regardless of possible reasons that could lead to a particular choice. The discussion of possible price incentives or business contracts between district and users will not be treated either; chapter 7 will propose a negotiation mechanism for integrating private users with the district power management system, based on generic control dialogues.

In technical terms, the viability constraint for a power management objective takes the form of a viability domain over the range of possible prosumptions p of the considered grid (house grid for a private user, district grid for the district administrator):

$$\forall t \in T, p(t) \in [p^{min}, p^{max}]$$

where T is the time scope of the objective, such as 10 minutes, 24 hours or 1 month. The evaluation scope of such a goal contains the electricity meter that measures grid prosumption, and by extension every prosumer in the grid. The action scope also contains every prosumer in the grid. However, in the case of district power management, the district management system may not have access to private appliances, due to privacy reasons. Therefore, from the district point of view, houses are black box prosumers, whose internals remain hidden a priori, and interactions are possible only with a public representative of the house management system. In summary, from the district power objective's point of view, house representatives belong to the evaluation and action scopes, whereas the internal house grids and their prosumers, being out of direct observation and control, belong to the influence and context scopes.

In terms of context scopes, the whole physical area of the grid may be relevant for monitoring, since any human activity, whether in the house or the district, may have consequences on electrical consumption. Beyond this local scope, external informations such as weather forecast may also help predicting heating consumption or renewable production (solar, wind) profiles for the day. Additional static knowledge sources such as weather history may also be considered for this purpose. Influence scopes of power management goals may also be wide, since affecting electrical consumption of appliances has direct impacts on human activity, and by extension to scopes much larger than the micro-grid itself.

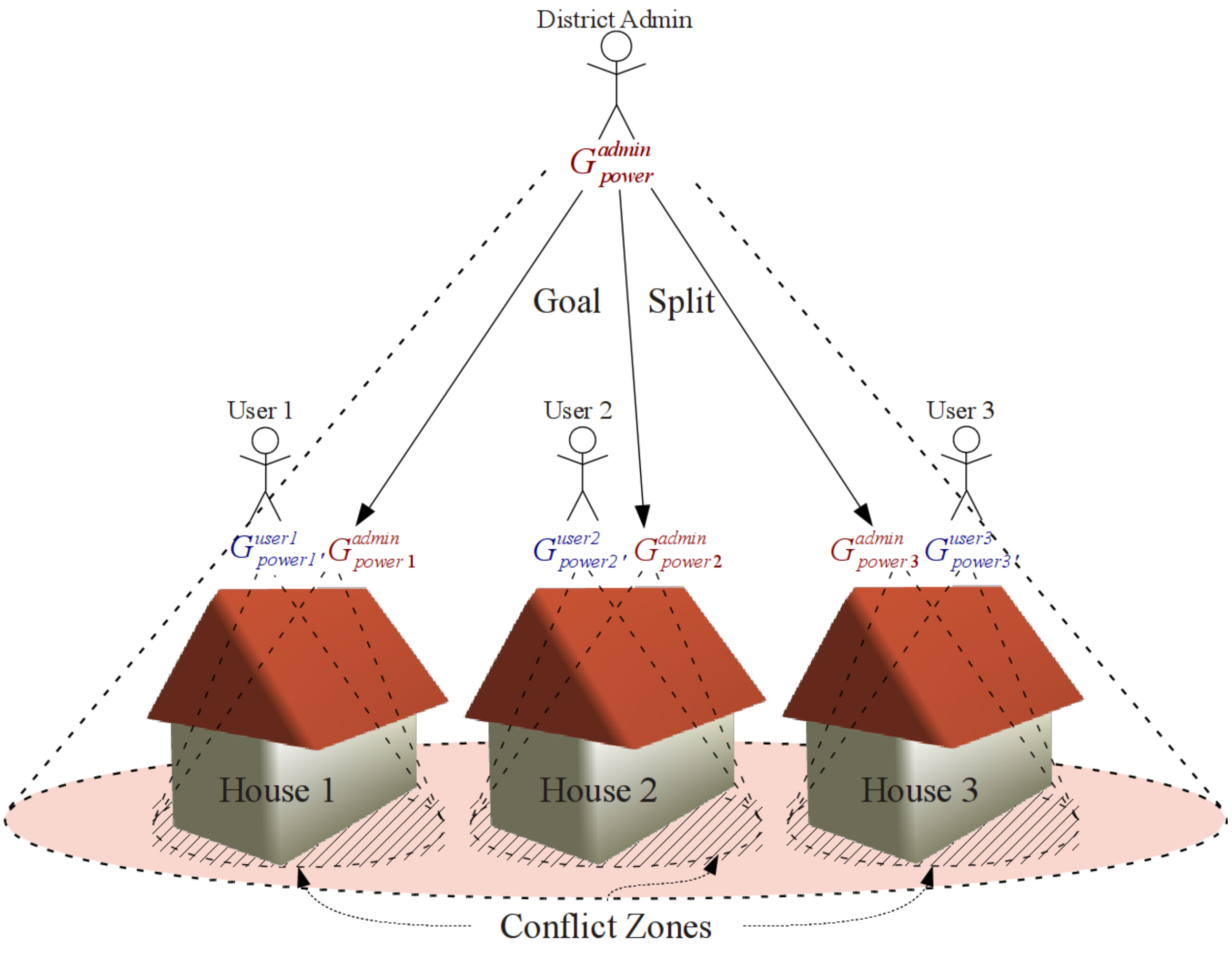

District Admin
G^{admin}_{power}
Goal Split
User 1
User 2
User 3
G^{user1}_{power1}, G^{admin}_{power1}
G^{user2}_{power2}, G^{admin}_{power2}
G^{admin}_{power3}, G^{user3}_{power3}
House 1
House 2
House 3
Conflict Zones

Figure 5.5: Conflicting district and house power management goals.

Power management conflicts

The hierarchical structure of the grid (district → house → appliance) maps to a goal splitting at these different scales: the extensive district power goal can be split into house-level goals, that can be split in turn into appliance goals:

$$p_{district} \in [p_{district}^{min}, p_{district}^{max}] \rightarrow^{split} \bigcup_{house \in district} p_{house} \in [p_{house}^{min}, p_{house}^{max}]$$

with $\sum_{house} p_{house}^{min} = p_{district}^{min}$ and $\sum_{house} p_{house}^{max} = p_{district}^{max}$, and similarly for each house:

$$p_{house} \in [p_{house}^{min}, p_{house}^{max}] \rightarrow^{split} \bigcup_{appliance \in house} p_{appliance} \in [p_{appliance}^{min}, p_{appliance}^{max}]$$

with $\sum_{appliance} p_{appliance}^{min} = p_{house}^{min}$ and $\sum_{appliance} p_{appliance}^{max} = p_{house}^{max}$.

Then, fulfilling each single sub-goal for appliances in a house is sufficient for fulfilling the house's power goal, as fulfilling each sub-goal for houses is sufficient for fulfilling the district goal. Therefore a possible way of achieving a power management goal consists in splitting effectively the power goals, assigning appropriate sub-goals to each part of a grid.

The choice of a viability domain $[p^{min}, p^{max}]$ for a sub-goal must be in adequacy with the capabilities of the target prosumer (either house or appliance), in terms of possible production or consumption. In addition, the power objective must be integrated with additional objectives targeting the prosumer:

- ◇ at the district level, private user interests might not be the same as the district administrator's ones. For instance, the district may want the user to consume more to absorb a production peak (due to a local increase in solar production) whereas the user wants to limit his bill and reduce his house consumption at the same time. Hence, there are indeed two conflicting power management goals for each house, as shown on Fig. 5.5: one derived from splitting the district goal, and one specified by the house owner. Determining the actual house prosumption target out of these two conflicting goals will be a matter of conflict resolution between house and district administrators.
- ◇ at the appliance level, consumption limitations may not be compatible with usages. Here again, proper power management requires assigning appropriate power goals to appliances, according to their own goals and capabilities, as the next section discusses.

In summary, the hierarchical decomposition of power management goals - shown in Fig. 5.5 - carries a conflict between the grid administrator and private users concerning the setting of power management goals, on each house. In turn, power management goals in houses be conflict with several appliance goals as presented in the next section.

5.3.2 Goal specifications for electric appliances

In future smart homes, many smart appliances are expected to perform a variety of domestic tasks. A generic taxonomy of such appliances would be difficult to establish, and this does not belong to the scope of this thesis. This section proposes a few examples of possible appliance goals, so as to investigate their possible integration with power management goals. We adopt a simple classification of domestic uses, where the user chooses between a "**comfort**" mode, synonym of maximum quality of service (QoS) to the detriment of energy savings, and an "**eco**" mode, where appliances are supposed to function with reduced quality of service and reduced energy consumption. Discussing the exact modalities of ergonomic user interactions is not addressed is this thesis.

Smart lamp goals

In our experiments, the **smart lamp** is introduced to represent appliances featuring a sporadic, on/off consumption profile. For instance, these may include actual lights, vacuum cleaners, TV sets, hot plates, and so on (cf. chapter 3). Lamps are in the scope (both action and evaluation) of two conflicting objectives:

◇ a usage objective when the user utilises the appliance, which would seek maximum quality of service through maximum consumption.
◇ a power objective, derived from splitting the house power goal, that would typically aim to limit the appliance consumption during high load periods.

Depending on user preferences, the appliance should follow one or the other of these objectives. When favouring the usage objective, the lamp consumes full power P_{max}; when favouring the power objective, the lamp shuts down. We will also consider the possibility where the lamp may function with low electricity consumption $P_{eco} < P_{max}$, at the cost of a reduced quality of service. For instance, lights would dim their emission power, vacuum cleaners would restrict their engine power, TV sets would reduce their screen brilliance, sound level and image quality, and so on.

To represent these possibilities, the lamp manager implements the following management policies:

- ◇ when in "comfort" mode, the lamp consumes maximum power P_{max} whenever switched on, regardless of any additional power management objective.
- ◇ when in "eco" mode, the lamp may consume P_{max} or P_{eco}; the lamp manager can therefore accept two different power goals, which in turn opens possibilities for the house power objective, as detailed later.

Smart heater goals

Smart heaters represent thermostatic appliances that pursue goals related to maintaining a certain temperature. Heaters, but also hot water tanks, refrigerators, freezers, ovens, air conditioning systems belong to this category, with their own technical characteristics and target temperature ranges. A typical usage objective takes the form of a temperature viability domain $[T^{min}, T^{max}]$, computed on an evaluation scope containing temperature sensors, such as a room or an isolated tank equipped with thermometers. The action scope is the device itself, and the influence scope notably includes the nearby physical environment, since temperature control usually affects the surroundings such as neighbour rooms. Including local weather in the context scope might be relevant for anticipating the evolution of outside temperature and the electrical consumption necessary to accommodate it.

Similarly to the lamp, this usage objective is conflicting with the house power objective. However, unlike the lamp, mitigating a power objective with a target temperature range is not as straightforward and requires additional domain expertise so as to:

- ◇ compute the link between temperature and heater consumption, that depends on the characteristics of the heater and the conditions in its immediate environment.
- ◇ advertise acceptable prosumption and temperature goals for the heater manager. These acceptable goals are intended for the user to know which temperature range he may ask the smart heater for, and at what "price" in terms of consumption. The acceptable goals are also intended for the house power management system to know which power goals the smart heater may accept.

5.3.3 Integration requirements

The previous sections identified a series of management conflicts, that must be resolved by the grid management system(s). Namely, as shown in Fig. 5.6:

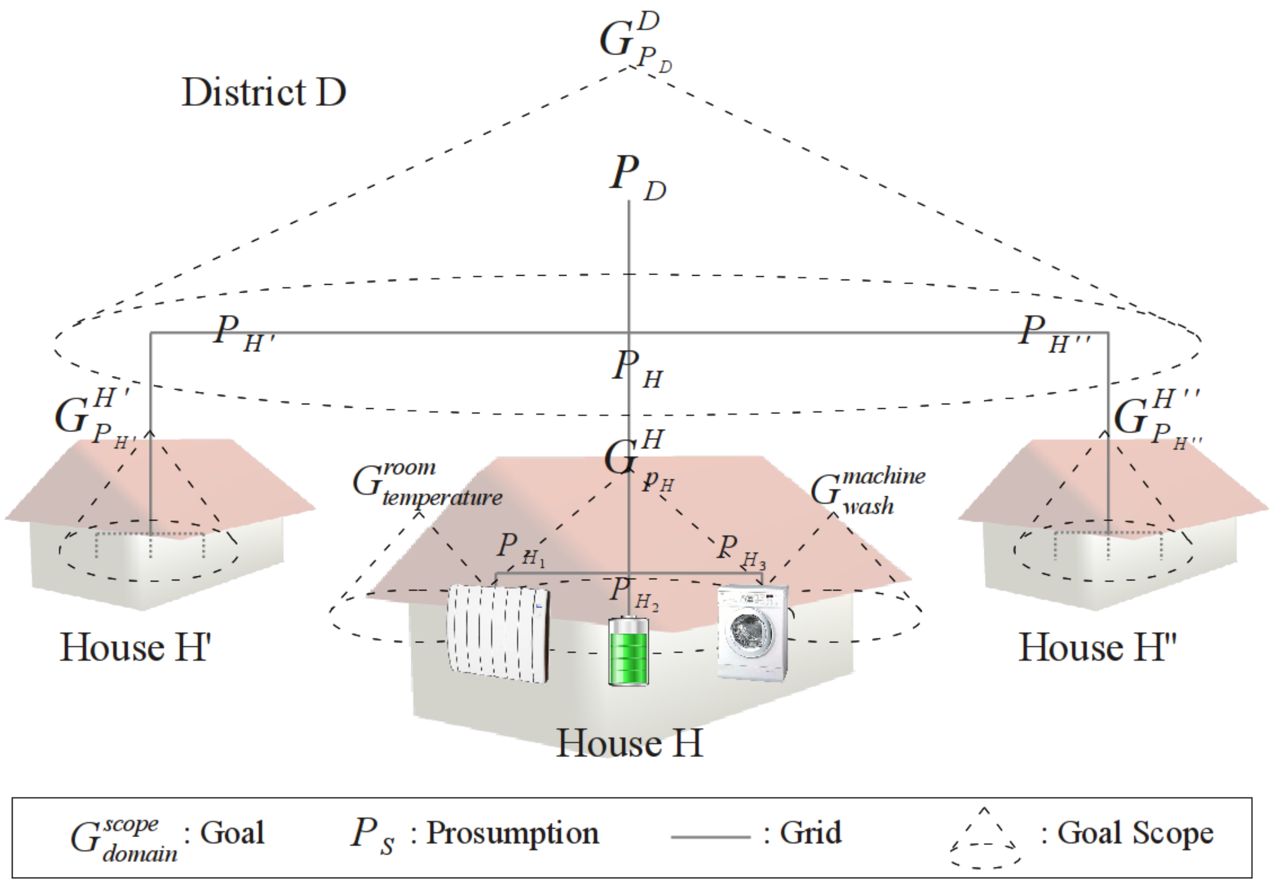


Figure 5.6: Holonic scopes and goals in the smart micro-grid.

- ◇ at the district level, the user's power management goal must be integrated with the district's power management goal.
- ◇ at the house level, the appliance(s)' usage goal(s) must be integrated with the house's power management goal.

We proposed a **holonic**[1] **control architecture** [FDMD13b] for answering these requirements, organised as follows:

- ◇ each appliance features its own autonomic manager, in charge (among other tasks) of solving the conflict between its usage goal(s) and the house's power management goal.
- ◇ each house features a power manager, in charge of:
  - – solving the conflict between its user's power goal and the district's power goal.
  - – splitting the resulting house power goal into power goals for each appliance, according to their respective capabilities.

[1]i.e. simultaneously a *whole* and a *part*, as defined below.

- the district features a power manager, in charge of:
  - solving the conflict between its administrator's power goal and the region's power goal (not addressed in this thesis).
  - splitting the resulting district power goal into power goals for each house, according to their respective capabilities.

In turn, this three-layer architecture requires several integration facilities:

- each manager needs internal integration facilities so as to solve the conflict between the several goals it follows:
  - appliance managers: usage goal vs. power management goal.
  - house power managers: user power goal vs. district power goal.
  - district power manager: district administrator power goal vs. region power goal (not addressed in this thesis).
- integration is required between power managers and prosumer managers at each level:
  - at the house level, so that the house power manager assigns appropriate power goals to appliances, with respect to their own profile and particular usage goals.
  - at the district level, so that the district power manager assigns appropriate power goals to houses, with respect to their own internal state and usages.

This structure is holonic according to the definition of **holon** [Koe48]:

A holon is simultaneously a whole with respect to its composing parts,
and a part of an enclosing holonic structure.

In the micro-grid case, each level of the grid (e.g. house, district) is a whole for its parts (respectively appliances and houses) and an autonomous part of a higher-level whole (e.g. district, region). More precisely, each appliance integrates within a house grid following a power objective, which in turn integrates into a district grid following another power objective, and so on. Conversely, the district power objective is split into house power goals, split in turn into appliance power goals. The holonic nature of the grid is due, among other reasons, to:

- the tree topology of the grid and the aggregation mechanism that underlies it.

◇ the grid and corresponding power objectives being split into sub-grids and sub-objectives for scalability reasons.

◇ successive administrative authorities with nested domains (house, district, region...) and the data privacy concerns such a topology rises.

The necessary integration features naturally divide into two categories: intra-manager integration for managers following several conflicting objectives, and inter-manager integration, so as to organise interactions between several autonomous managers following a common, global goal (here, power management). Chapter 6 details solutions for achieving the first category of integration, guiding the conception of multi-objective autonomic managers. Chapter 7 addresses the second category and describes generic integration organisations for balancing individual and collective objectives in societies of autonomic systems.

Chapter conclusion

This chapter presented a generic formalisation of management goals for multi-objective autonomic systems. We formalised management goals in terms of:

◇ viability constraints, defining the target state or behaviour of the system;

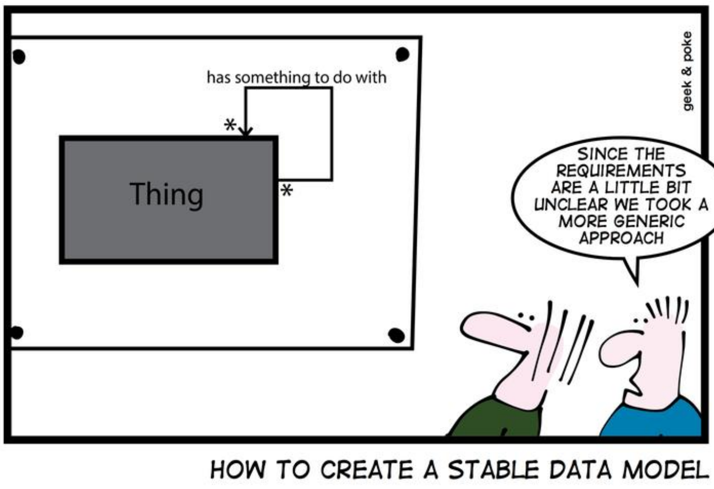
◇ scopes in space and time, defining where and when a goal should be evaluated and applied, respectively.

We defined management conflicts in terms of intersections between goal scopes defining incompatible viability constraints, investigated possible strategies for conflict resolution, and formulated requirements for integration infrastructures that would allow inserting such strategies into in multi-objective autonomic systems.

As an illustrative application of the proposed goal formalisation, we identified management goals in a residential micro-grid use case, considering power management concerns on one hand, and domestic appliance usage goals on the other hand. We studied the management conflicts that these objectives create, at both house and district scales, and identified integration requirements for the micro-grid accordingly. The following chapters will come back to this use case and show how to implement a control architecture answering the integration requirements we identified here. Chapter 6 addresses the design of multi-objective autonomic managers that can cope with several conflicting goals and adapt to variable conditions. Chapter 7 addresses the design of organisations that can integrate heterogeneous third-party autonomic systems, handling their particular capabilities and objectives and finding possible equilibria between the different goals at stake. Chapter 8 presents a proof-of-concept implementation of the entire architecture we propose, derived from this initial formalisation and analysis of management goals in smart micro-grids.

Chapter 6

A generic architecture for adaptable, multi-objective autonomic managers


has something to do with
*
*
Thing
geek & poke
SINCE THE REQUIREMENTS ARE A LITTLE BIT UNCLEAR WE TOOK A MORE GENERIC APPROACH
HOW TO CREATE A STABLE DATA MODEL

http://geek-and-poke.com, July 22nd, 2013

Chapter introduction

Typically, autonomic management requires complicated software applications. Designing their control logic requires specific expertise in analysing fluctuating environments and planning appropriate reactions, to name only a few. As a result, autonomic managers are likely to necessitate the integration of domain-specific logics from different providers so as to attain their objectives. In particular, this is the case of multi-objective autonomic systems, such as smart devices, homes or cities, that are likely to feature control components from different domains, for instance, temperature management and power management.

In addition, the changing requirements and variable contexts in which such autonomic systems execute require specific engineering techniques for enabling long-term evolution and adaptation of their management logic. Modularisation is an essential feature of such software engineering techniques, as promoted by generic architectures such as MAPE-K [KC03] or frameworks such as Ceylon [Mau10]. However, fragmenting autonomic managers into independent reusable control components requires corresponding integration capabilities, for ensuring that the composition of third-party control components yields coherent control loops overall. In this regard, autonomic literature lacks generic models and integration facilities that would ease the development of complicated autonomic systems.

In this chapter we propose a **taxonomy** for the generic building blocks of modular autonomic managers - namely, control and integration components - and their associated semantics. Notably, we categorise the integration of control components in terms of formal **integration functionalities**. Then, we establish a series of **integration design patterns** that support alternative implementations of integration solutions, each with its own strengths and weaknesses, which we discuss. The chapter concludes with an example of an adaptable heater manager, followed by a discussion of how the presented approach can be extended from this relatively simple manager to more complex scenarios.

6.1 Taxonomy of autonomic management components

6.1.1 Fragmentation and integration for modular autonomic managers

Past a certain degree of complexity, monolithic designs are no longer viable for developing autonomic managers [Mau10]. Alternatively, modular designs divide the control loop into several components that each perform a subset of the overall loop functionality, following the principle of separation of concerns. For this purpose, designers can rely on generic manager architectures such as MAPE [KC03] or Observer-Controller [Sch05].

Designing systems based on decoupled, modular components usually results in better flexibility, since modifying or extending the control loop only concerns limited parts of the complete implementation. In addition, specialised components can be deployed in different kinds of managers, favouring re-usability and integration of third-party software. Dynamic service technologies such as the OSGi platform [httd] allow hot deployment and update of components via run-time observation and control of the running manager.

Specialised frameworks such as Ceylon [Mau10], relying on the iPOJO and Cilia service-oriented component technologies [httc] [htte], provide generic containers handling multiple non-functional aspects such as life-cycle management, service discovery and binding, synchronisation or communication protocols. Such frameworks for autonomic systems relieve designers from many low-level concerns related to the complexity of managers, allowing domain experts to focus on the specificities of their control logic.

However, fragmentation comes at the cost of additional integration-related difficulties: modular systems require integration facilities to glue components together and ensure the coherence of their composition. Also, sometimes integration must be performed during runtime, for adapting the system to changing environments or for optimising it based on opportunistically-discovered components. Existing frameworks usually rely on a single, specific integration mechanism (cf. chapter 2) tightly coupled with their implementation of autonomic managers. Therefore, there is a need for studying and documenting generic integration solutions that can be re-used in variable contexts.

The following sections provide a taxonomy of management components, comprising control and integration tasks. We focus on the different functionalities provided by integration components, notably in relation to goal conflicts formalised in the previous chapter. The taxonomy serves as a basis for a series of integration design patterns for control loops, presented later in the chapter.

Control composites result from mixing control and integration components. This composition is recursive: composites can implement a subset or the totality of a management loop functionality, and compositions of composite yield high-level composites. Due to their modular design, composite managers are flexible, extensible and adaptable. Developers can design new components and deploy onto existing managers, possibly at run-time. We identified several organisational patterns for organising common compositions of control and integration tasks, described in next section.

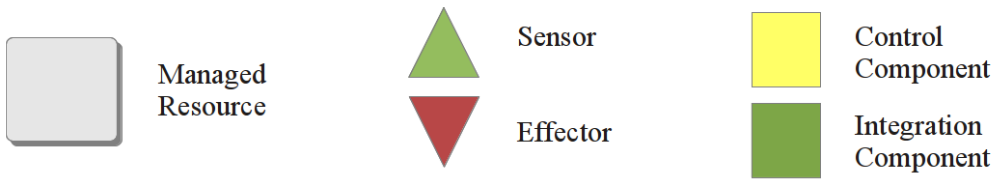


Figure 6.1: Graphical representations used in this chapter.

6.1.2 Control components

An autonomic manager primarily consists of **control components** (or control tasks) which carry the business logic of the manager: mapping monitoring information gathered from the evaluation and context scopes into actions executed on the action scope. This mapping can be separated into several successive **control functionalities**, each one performing one or several steps of the global control process. State of the art conceptual modularisations of the control loop, such as Observation & Control [ABS11] or Monitoring, Analysis, Planning and Execution [KC03] may apply here and guide the design of the controller.

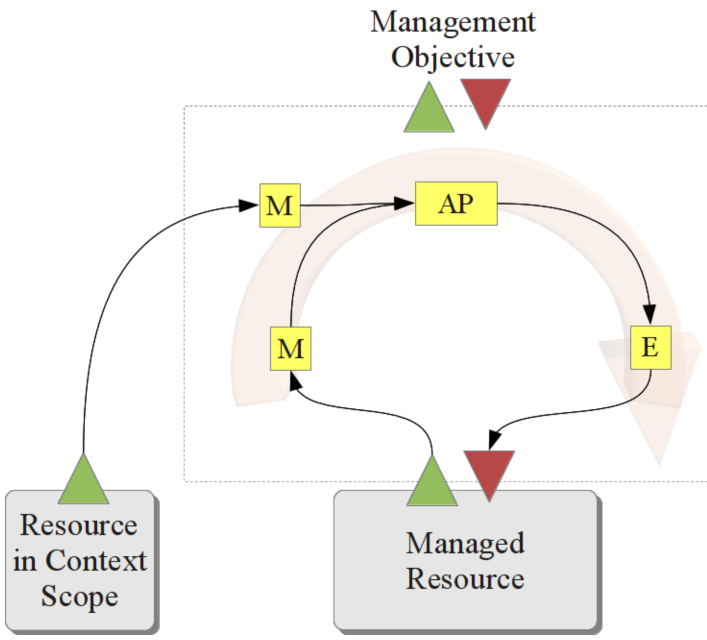


Figure 6.2: Example of modular implementation of an autonomic manager.

Control tasks can be implemented as loosely coupled software components, each task providing a combination of atomic control functionalities, for instance with the MAPE architecture: "M" only, or "MA", or "APE", and so on. The global loop "MAPE-K" is implemented as a combination of such tasks, as shown in Fig. 6.2. This approach facilitates the design of complicated managers by separating control concerns, which favours re-usability of the individual components.

For instance, given an "AP" control component implementing the core decision process of the manager (cf. Fig. 6.2), developers can use different versions of the upstream Monitoring component transparently. This favours long-term maintenance and evolution of the manager so as to adapt to evolving communication standards, available sensors in the environment, or requirements regarding monitoring. Similarly, this approach permits the integration of legacy components, which is a critical requirement when dealing with open, multi-party systems that must evolve on the long term.

For this purpose, the use of a service-based approach relieves developers form the burden of complex deployment and binding of control components. Dynamic service platforms may also provide run-time feedback to the administrator, such as component performance metrics and fault detection, and allow hot swapping and deployment of components. Technological solutions for implementing control tasks will be discussed when presenting our proof-of-concept simulator in chapter 8. However, providing a full-fledged framework for autonomic control tasks is beyond the scope of this thesis.

6.1.3 Integration components

Integrating control tasks into coherent control loops is an essential requirement for modular autonomic managers. Arbitrary combinations of stand-alone control components may often lead to inappropriate behaviours, due to **task integration issues**. We classify these into two categories, sequential and parallel, which we define below. A third category of integration issues, caused by interactions with external autonomic systems, is also briefly mentioned. Chapter 7 will be entirely dedicated to discussing such integration issues with external systems and proposing suitable solutions for ensembles of stand-alone autonomic systems.

Sequential composition issues

Let us consider two successive control tasks in a given control loop, for instance, an Analyser and its following Planner. These two tasks follow a priori the same objective and they implement separate functionalities of the control process, therefore there should be no conflict between them in relation with control concerns. However, non-control related aspects may still require **integration functionalities**, such as translating inter-task communications from one particular protocol to another or adapting different execution rates between the tasks.

Parallel composition issues

Let us consider concurrent control tasks of the same kind performing variants of the same control functionality, for instance several Analysers. There can be two reasons justifying the use of such parallel instances of control functionalities:

◇ The tasks carry different objectives, i.e. they belong to a multi-objective autonomic manager and implement concurrent control functionalities. This is more likely to occur to Analysis or Planning tasks, since neither Monitoring nor Execution functionalities are related to goal-driven decision making. For instance, the manager features a legacy control component that must be integrated as such. In that case, additional conflict resolution logic is necessary for the loop to behave correctly, as discussed previously.

◇ The tasks follow the same objective, and implement different variants of the same control functionality [MHS+11], each of them being most appropriate in particular situations. Such redundancy increases the control loop's robustness and adaptiveness to various contexts. Having such parallel implementations also requires additional integration logic to produce a coherent output, be it the best one elected amongst possible propositions, or an aggregated, richer consensus.

In both cases, integration of parallel control tasks boils down to implementing one of the following **integration functionalities**:

◇ aggregating monitoring data from several sources and producing coherent representation of the manager's managed resources and environment.
◇ selecting or aggregating concurrent symptoms from several analysers.
◇ selecting a plan or action policy amongst several alternatives, or a coherent combination of these.
◇ choosing the appropriate executor(s) for applying a plan into the manager's action scope.

The logic behind such functionalities consists in determining an **integration function** f such as for any set of conflicting inputs $i_1...i_n$ (either sensor data, symptom analysis, reconfiguration plan, etc.), f generates a coherent output $o = f(i_1, ...i_n)$. In case the integration issue is due to conflicting management objectives, the integration function implements conflict resolution semantics, as we defined in chapter 5. Out of the whole space of possible mappings $(i_1, ..., i_n) \mapsto o$ we identify the following categories of generic integration functions:

◇ **selection functions** elect a single input to be the actual output: $f : (i_1, ...i_n) \mapsto i_j$ for some j in $[1..n]$. This solution is equivalent to considering that only one of the proposals is appropriate, select it according to some criteria and carry on with it. Selection functions are a natural application of "winner-takes-all" resolution semantics for objective conflicts. Then selection criteria could be generic, for instance priorities associated with each input, and reflecting user preferences expressed in goals. In that case selecting the "good" solution boils down to finding the highest priority out of the possible candidates.

- ◇ **mitigation functions** aggregate the inputs into an acceptable intermediate solution, akin to mitigation resolution semantics for conflicting objectives. This is usually achieved via a mean of the proposals, possibly balanced by weighting coefficients $c_1 ... c_n$, namely $f : (i_1, ... i_n) \mapsto \sum\limits_{j \in [1..n]} \frac{c_j i_j}{C}$ with $C = \sum\limits_{j \in [1..n]} c_j$. Indeed, the previous selection mechanism is equivalent to such a weighted summation with all coefficients being 0 except for the winner's one being 1. The choice of mitigation coefficients may depend on runtime information such as user preferences.

One can also use solutions combining the two categories above, for instance, selecting the most relevant inputs or eliminating the bad ones, and mitigating the few best solutions. The integration function may also rely on learning from feedback on the successfulness of its output via supervised learning techniques, used for instance in conjunction with artificial neural networks.

External composition issues

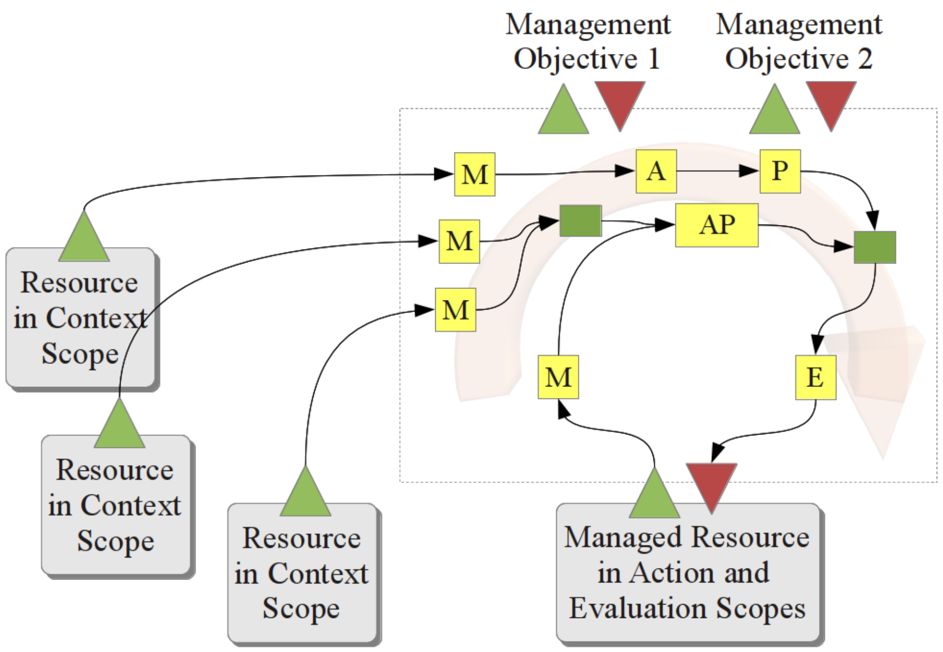
As discussed in chapter 5, autonomic managers must be integrated with other autonomic managers, carrying the same goals or different goals. Therefore, dedicated integration components will be necessary to implement inter-manager integration functionalities. The exact nature of these functionalities and how to compose them will be discussed in details in chapter 7.

Inter-manager integration relies on mutual observation and possibly control. Therefore, external integration components will introduce in the management loop:

- ◇ monitoring inputs showing the state of neighbour autonomic systems in the manager's environment scope.
- ◇ incoming execution orders and/or negotiation messages from these same neighbours, possibly in the form of goals.
- ◇ monitoring outputs advertising the public state of the manager to neighbour autonomic systems in the manager's influence scope.
- ◇ outgoing execution orders and/or negotiation messages to these same neighbours, possibly in the form of goals.

Integration components implementation

We extend the repertoire of management components with **integration components** (or integration tasks) implementing the different **control functionalities** previously identified. As for control tasks, integration tasks can be implemented as stand-alone software components for re-usability, separating the concern of control implementation from the concern of control integration. Integration logic can also be mixed together with control logic into hybrid components, sacrificing the benefits of separation of concerns for possible reductions of development difficulties in simple cases. For instance, a priority-based selection function can be implemented as a central "aggregator" component, or directly provided by the conflicting control components, that implement a decentralised election algorithm.


Management Objective 1
Management Objective 2
M
A
P
M
AP
M
M
E
Resource in Context Scope
Resource in Context Scope
Resource in Context Scope
Managed Resource in Action and Evaluation Scopes
Figure 6.3: Example of a composite autonomic manager mixing control components (yellow) and integration components (green).

Algorithmic complexity of integration functions

For the purpose of performance evaluation, we provide the algorithmic complexity of the most common integration functions.

Selection function: priority-based selection of a unique best solution is equivalent to finding the greatest priority out of n candidates, which costs at most $\mathcal{O}(n)$ in terms of algorithmic complexity - and $\mathcal{O}(n \log n)$ in case full ordering of candidates is required.

Mitigation function: averaging numerical values via priority-based weighted sum also costs a linear complexity in terms of number of conflicting resources ($\mathcal{O}(n)$). More sophisticated, domain-specific resolution semantics, such as probabilistic modelling and aggregation of data, may prove more costly. However, such important features should be considered as expert Analysis or Planning functionalities of the control loop, rather than a generic integration solution.

6.2 Task integration patterns

This section presents four **task integration patterns** that we have identified in the literature, either as already-defined design patterns, or as application-specific integration solutions that we abstracted from [FDD12]. Each pattern presents a generic, reusable integration solution and implements one of the integration functionalities identified previously in section 5.2.3. A pattern definition consists of:

- ◇ a **context**, identifying the factors that make the pattern applicable or not, depending on the type of conflict, the system characteristics (scale, dynamism, heterogeneity, openness) or the system's QoS requirements (allowed decision time, precision, criticality, etc.).
- ◇ an architectural **description** of the integration tasks and their interconnections with control modules.
- ◇ **advantages** and **limitations**, such as the effect of applying the pattern to conflicting resources, or the pattern not being applicable to certain contexts.
- ◇ a qualitative **evaluation**, with respect to criteria such as :
  - **overhead**: the additional computations and communications the pattern introduces in the control loop's execution flow.
  - **safety**: the assurance that conflict resolution performs successfully and does not let the system in an undesirable state.
  - **conceivability**: how difficult it is to actually implement the resolution logic, considering for instance legacy conflicting tasks or multi-objective resolution semantics.
  - **robustness**: how the pattern resists to faulty behaviour from conflicting control tasks or incorrect input.
  - **evolvability**: how does the pattern support changes in the set of conflicting tasks or conflicting objectives.

  - **scalability**: how the pattern scales up to large numbers of conflicting resources, and to high frequencies of conflicting activations and changes in the pool of conflicting resources.

- ◇ **example implementations** of the pattern in the literature. At the end of the chapter, we also present a complete heater manager implementation showing different applications of the patterns.

We present four different patterns: **Monolith**, **Aggregator**, **Dealer** and **Arbiter** [FDD12].

6.2.1 Monolith pattern

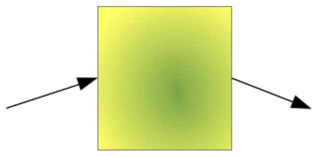


Figure 6.4: The Monolith pattern.

Context: This pattern applies to cases where the management logic and integration functionalities are clearly identified and unlikely to change in the future. A well-known solution exists, where decoupling control or integration functions would result in unnecessary over-engineering of the autonomic manager.

Description: The autonomic manager is designed as a single logical block, mixing control and integration functionalities in an ad-hoc, tightly coupled monolith.

Advantages and Limitations: The Monolith's main advantage is also its main limitation: it is applicable and efficient only below a certain degree of complexity in the target management logic. When applicable, the Monolith is often the simplest, most efficient and most effective solution. However, not every management logic can be reduced to such a single one-fits-all implementation, although the Monolith authorises any kind of integration functionality (selection, mitigation, etc.).

Evaluation: The Monolith is a tempting solution for simple conflict situations where a good solution is already identified. Such an ad-hoc integration, when possible, is generally straightforward to implement and saves the burden of engineering generic integration solutions. Since the Monolith is specific to a particular management logic, it can be optimised and its behaviour can be predicted more easily than compositions of third-party components.

However, the drawback of a monolithic design is poor modularity and separation of concerns: modifying or replacing a part of the Monolith is difficult without redesigning the whole solution, making maintenance difficult. Failure in the Monolith always means complete failure without possibility of partial recovery.

Examples: The Rainbow framework is an example of monolithic design [GCH+04] [Che08]. A Rainbow controller relies on a central architectural representation of the system under observation and control, and a unique rule-based adaptation engine, which facilitates the implementation of control. Yet, no specific support is provided for distributing or coordinating multiple controller instances.

6.2.2 Aggregator pattern

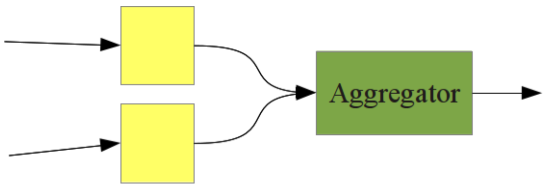


Figure 6.5: The Aggregator pattern.

Context: The control loop architecture is such that all outputs of conflicting resources converge to the same task input. This is the case for instance when multiple control strategies are implemented in parallel by different conflicting Planners (or Analyser-Planners) and compete for being applied by a single Executor. Or, when multiple Monitors, connected to different sensors, produce inaccurate, diverging information on the managed system's state (or its environment).

Description: An additional integration task (the "Aggregator") intercepts the conflicting outputs and produces a single, coherent output, such as a single reconfiguration to be executed or a unique monitoring representation of the system under observation. The Aggregator implementation may rely on generic message filtering solutions (e.g. pattern matchers) and data reduction (e.g. priority-based weighted sums). The Aggregator allows to implement any kind of integration function, such as selection or mitigation, as described before.

Advantages and Limitations: The Aggregator transparently resolves the conflict via communication interception, similar to an aspect-oriented approach [KLM+97]. This is a good feature in terms of integration of legacy components, since no modification of the existing conflicting resources is required in order to introduce the pattern in the system. However, not all control resources support their output being silently filtered, depending on their nature. For instance, aggregating multiple monitoring data may not disturb conflicting Monitors. On the other hand, Planners could misinterpret not seeing their plans executed on the system under control; as a result, the Planners may keep requesting reconfigurations repeatedly, eventually diverging or failing. Therefore, prior to applying the pattern, great care must be taken to which conflicting resources will be filtered and whether these resources will support it.

Conflicting resources with different execution rates may require the Aggregator to implement some persistence capabilities, so as to keep an up-to-date trace of all resource outputs currently available.

Evaluation: Since the Aggregator is transparent to conflicting resources it requires little or no adaptation of these - provided their semantics support transparent filtering, as detailed in the pattern's limitations. Being placed downstream of the conflicting resources, the Aggregator can ensure that coherent output is provided to the rest of the control chain, in case upstream resources prove faulty. This is a convenient additional feature that may encourage the use of the pattern.

The Aggregator does not prevent conflicting resources from executing, which may allow useless computations in cases where only a single, best solution is finally elected by the conflict resolution logic. This aspect is a characterising feature of the Dealer and Arbiter patterns. Since it is centralised, the Aggregator can be considered a bottleneck and a single point of failure of the control process. However, conflicting outputs converging to a single point is a requirement for applying the pattern, as described in the "context" section, rather than a limitation introduced by the Aggregator itself.

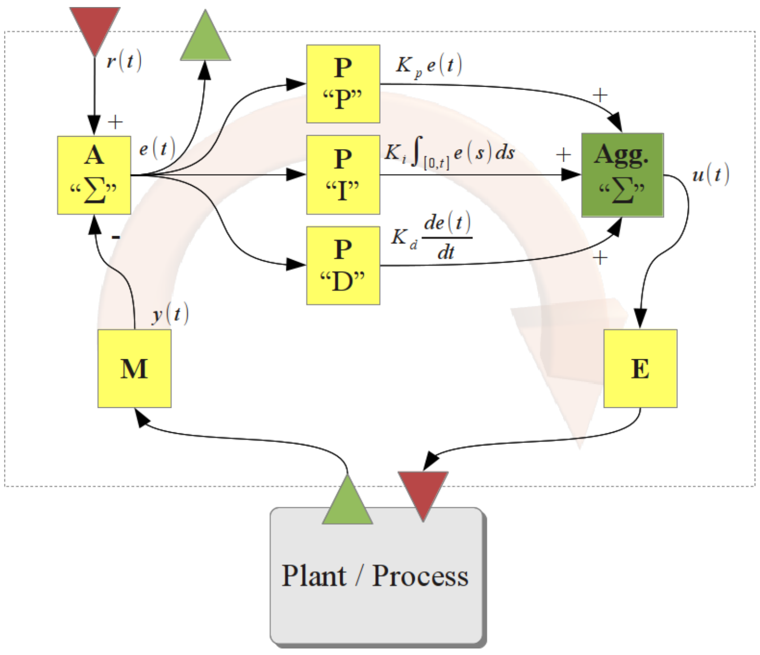


Figure 6.6: The PID controller exemplifying the Aggregator pattern.

Examples: PID ("Proportional-Integral-Derivative") controllers are a well-known form of generic Analysis and Planning resource [Lev96] (cf. Fig. 6.6).

Given a target system variable (r in the figure) and a measured actual value (y), the PID computes a control input (u) for the system meant to minimise the error ($e = r - y$). A PID typically based on an error Analyser and three parallel Planners computing a target command value equal to the error (“P”), its integral (“I”) or its derivative (“D”) respectively. Then, an additive aggregation function computes the control input u as a weighted sum of the three “P”, “I”, “D” components: $u = K_p.P + K_i.I + K_d.D$.

6.2.3 Dealer pattern

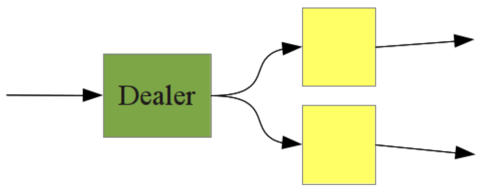


Figure 6.7: The Dealer pattern.

Context: The conflicting tasks’ inputs are provided by a unique event source, upstream in the control flow. This happens typically when multiple Analysers depend on a single Monitor publishing symptoms on the system under observation, or when multiple Executors may be used to apply a given reconfiguration issued by a Planner. This configuration can be related to a publisher-subscriber architecture, the subscribers being the conflicting resources receiving input from a common topic.

Description: An additional integration resource (the “Dealer”) filters incoming messages at the source (the topic in case the system is implemented as such) and selects the conflicting resources that will receive input. This message filtering ensures that only non-conflicting resources are triggered simultaneously - in the simplest case, only one of them. This pattern only allows to implement selection integration functions - i.e. no mitigation is possible.

Advantages and Limitations: This pattern relies on the conflicting resources being triggered by incoming data, pushed by an upstream source. This is a common behaviour characterising reactive, event-driven control tasks. However, certain control elements may not implement this behaviour and rather proactively pull their input from upstream sources, in which case the Dealer pattern may become difficult or impossible to apply.

In case the Dealer pattern is applicable, its conflict resolution mechanism is transparent and requires no further adaptation of the conflicting resources. Transparency is good for re-utilisation concerns, however, the integrated tasks receive no indication of the fact that they are being starved, which may be an issue for tasks expecting regular input from their predecessors.

An important feature is that message interception actually prevents conflicting control tasks from executing - typically all but one. Therefore, the Dealer saves the management system from useless or undesirable computations, contrary to the Aggregator pattern. On the other hand, the Dealer only allows selecting a single task amongst the conflicting ones, and does not provide mitigation resolution logic.

A priori, the Dealer has no knowledge of whether or not the filtered input will be relevant for the selected conflicting resource. This can be an issue, since filtering starves all the other tasks from input. This issue can be addressed by providing the Dealer with specific knowledge on the conflicting tasks, which raises its complexity and may increase coupling in the system. An alternative solution such as the Arbiter pattern, presented below, addresses this concern.

Evaluation: The Dealer's upstream filtering mechanism is transparent and requires no adaptation of the conflicting resources, which facilitates applying the pattern to arbitrary conflict situations. When relying on generic resolution logic such as task priority, the Dealer easily supports adding and removing additional conflicting tasks.

Due to its centralised nature, the Dealer is a bottleneck and a single point of failure of the controller. However, as described in the context section, the unique upstream source is a requirement for the pattern to be applicable, therefore this topology is an inherent feature of the considered control flow.

Since it prevents conflicting tasks from executing, the Dealer pattern may actually reduce the computation load brought by the controller. Since it is limited to selection functions, its internal integration logic should introduce limited overhead into the control process (cf. below).

Examples: Ceylon project proposes a Dealer solution - among others - to prevent concurrent control tasks to execute simultaneously [MDL10]. In Ceylon, control tasks are implemented as event-driven components communicating through a central message bus via publication and subscription. Conflict rules allow to specify which tasks are conflicting and which ones should be inhibited in case of conflict. Then, a message filter implemented in the message bus enforces this exclusion rules by starving tasks that should not be activated.

6.2.4 Arbiter pattern

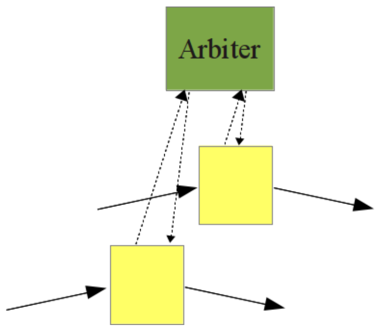


Figure 6.8: The Arbiter pattern.

Context: This pattern can be applied to any set of conflicting resources, without constraints on the control flow topology. It is therefore an alternative to the Dealer and Aggregator patterns in case neither can apply, due to their applicability restrictions. However, the Arbiter pattern is non-transparent and requires specific behaviour of the conflicting components. Namely, the integration logic is not interposed within the "normal" control flow of the conflicting resources. Rather, the control flow must be explicitly diverted so as to pass through the Arbiter, as explained in the pattern description.

This pattern is also called "Controller" in [FDD12], although "Arbiter" is preferred here for avoiding confusion with the general control loop pattern.

Description: Conflicting components explicitly request an execution authorisation from a central conflict resolution resource (the "Arbiter"). The latter answers requests by granting or denying execution for the requester, usually a single one at any given time from the pool of conflicting resources. This selection can be implemented with generic priorities, the Arbiter electing the highest priority requester.

A possible variant consists in conflicting tasks requesting authorisation for sending their output to the rest of the control loop, i.e. after executing, allowing the Arbiter to compare alternatives available at the moment.

The pattern is limited to selection functions for its integration logic.

Advantages and Limitations: In addition with the Arbiter implementation, the pattern requires the conflicting tasks to implement a request behaviour. This may necessitate in turn to extend the conflicting resources with special-purpose integration logic to form composites that can communicate with the Arbiter.

Contrary to a Dealer, the Arbiter waits until conflicting resources ask for execution authorisation, or even output authorisation. Since the integration process is explicit for conflicting tasks, these can more easily deal with starvation, knowing they are being overcome by more important competitors.

Evaluation: Since the core logic of the Arbiter is a selection function, it should not introduce significant overheads into the control process. Depending on the variant chosen, the Arbiter may or may not save the system from computation by starved resources. Since it is centralised, the Arbiter also introduces a new bottleneck and single point of failure in the system.

Examples: The Arbiter pattern is used in several generic autonomic frameworks. Ceylon project [MDL10] also used an Arbiter conflict solver as an alternative to its original Dealer solution. Instead of relying on static rules and pre-emptive task starvation only, an Arbiter can detect task conflicts at the moment they require execution authorisation. This solution reduces useless filtering computations, compared to a Dealer.

Unity [CSWW04] also adopt Arbiters as a conflict resolution solution. A "resource" arbiter is responsible for resource allocation to applications: conflicting applications provide an individual estimate of possible resource allocations, and the arbiter computes a global optimal allocation based on all the proposals.

In Automate [ABL+03], "access control agents" enforce exclusion and access permission rules on concurrent entities. Notably, entities request and may receive roles that determine their right to access resources and implement certain behaviours. This architecture presents similarities with the "Hierarch" pattern that the next chapter presents.

6.3 Application: the smart heater use case

This section presents an architecture for an autonomic heater with two conflicting objectives, as already defined in chapter 5:

- ◇ a comfort objective, formulated as a target temperature domain $[T_{min}, T_{max}]$ for the heater's room, for instance $[20°C, 23°C]$.
- ◇ a power management objective, issued from the house power management organisation, requiring sporadically the heater to shut down. The details of this organisation and its functioning are presented in chapter 7.

This section first presents a simple autonomic heater, with a single temperature objective - i.e. not integrated into a power management organisation. Then, an extended, power-sensitive version of the manager is presented, making use of the integration patterns presented previously. This autonomic heater manager showcases the possibility to adapt autonomic controllers to evolving environments and requirements while following contradictory objectives. We will also discuss briefly how the approach exemplified via this relatively basic case study can be extended to more complex scenarios, such as a smart home.

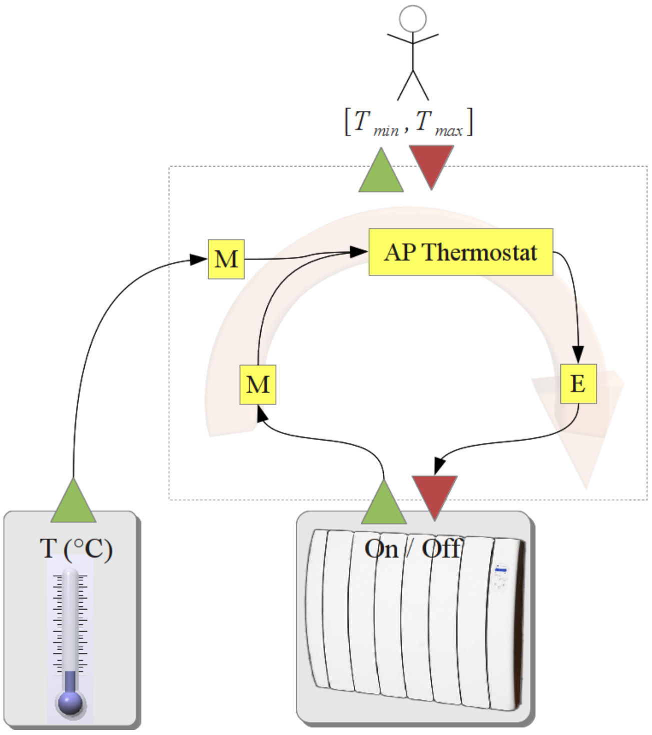


Figure 6.9: Sample design of a basic autonomic manager for a smart heater.

6.3.1 Single-objective autonomic heater

Let us first consider a simple control loop with a single objective, namely a thermostat maintaining a certain temperature in a room via an electrical heater. This device can be in two states: “Off”, consuming no power and not heating, or “On”, consuming a constant amount of power and heating its surroundings. Temperature in the room (i.e. the evaluation scope) is monitored via a thermometer, as represented in Fig. 6.9. The manager also observes and control the state of the heater (i.e. the action scope) via touch-points.

The manager switches the heater on and off so as to maintain room temperature in the target domain $[T_{min}, T_{max}]$. This is typically achieved by an “AP” control component implementing a simple hysteresis mechanism: when below T_{min}, switch the heater on, heating until T_{max} is reached, where the heater is switched off, until temperature drops and reaches T_{min} again. This control policy yields the typical behaviour depicted in the thermodynamic cycle in Fig. 6.10. Over time, the behaviour of such a heater tends to be periodic, as depicted on Fig. 6.11.

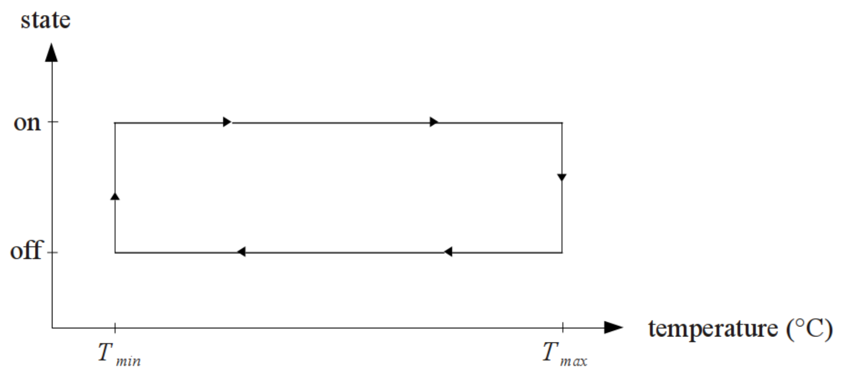


Figure 6.10: Thermodynamic cycle of a hysteresis heater.

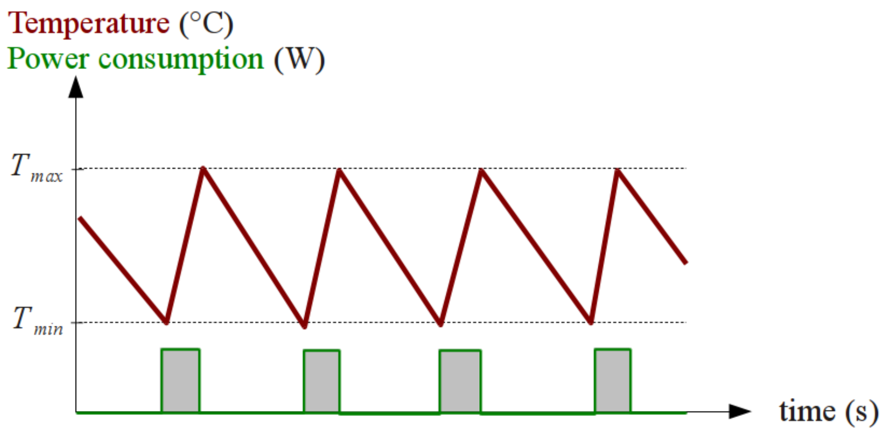


Figure 6.11: Pseudo-periodic heating cycles of a thermostatic heater, with corresponding temperature evolution.

Next section presents several possible extensions of this simple heater, so as to adapt it to new environments and integrate it in a power management organisation.

6.3.2 Multi-monitoring extension

Let us consider now that new thermometers are deployed in the room, publishing extra temperature information that the manager could benefit from. New Monitoring components are added to the control loop, which requires in turn an aggregation function so as to provide a unique temperature value to the following Analysis components (see Fig. 6.12). This is an example application of the Aggregator integration pattern (“Agg.” in the figure), where an extra aggregation component gathers the various

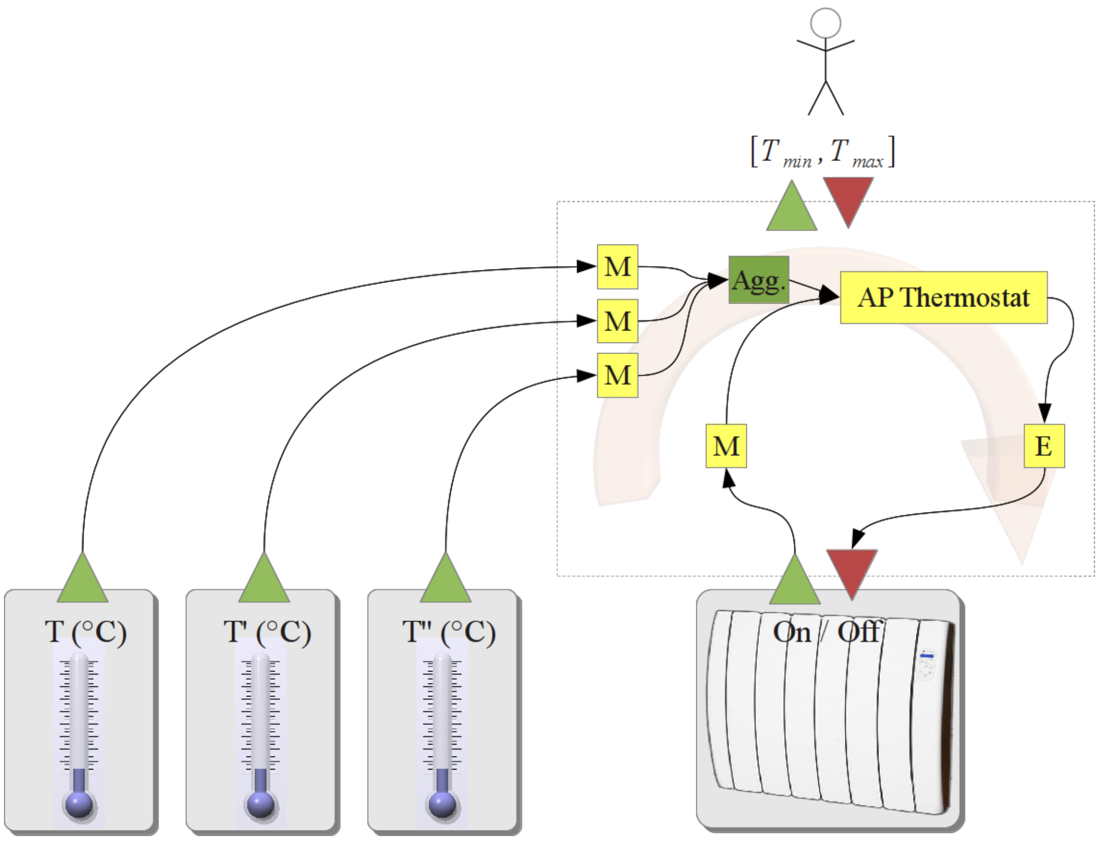


Figure 6.12: Applying the Aggregator pattern to multiple monitoring tasks.

temperatures provided by context Monitors and computes a mean value (mitigation), and/or drops incoherent data due to faulty sensors (selection).

6.3.3 Multi-objective extension

Let us extend the heater manager with power management capabilities. As presented in chapter 5, the heater is in the scope of the house power management goal. This goal can be split into several sub-goals, including a heater-specific power goal, as chapter 7 will detail. This section explores how to extend the internal architecture of the heater manager so as to handle the two objectives, temperature and power consumption.

Power management for thermostatic loads

The hysteresis cycle presented in Fig. 6.10 can be adapted so as to better serve the sake for power management, in the two following ways:

- ⋄ by cutting a heating cycle prematurely, shutting down the heater before T_{max} is reached and reducing its consumption to 0 (see Fig. 6.13). However, shortening the heating cycle also signifies that T_{min} will be likely to be reached sooner than after a full heating period, as depicted on Fig. 6.14.

◇ conversely, by anticipating the heating cycle, switching on before T_{min} is reached, for a quick consumption rise. Such anticipated cycle will be shorter than a full cycle since T_{max} will be reached more rapidly.

Such cycle anticipations and interruptions are equivalent to shifting heater consumption periods by a limited period of time. This allows quick load raising or reduction, since an electric heater can usually change its state in a few seconds. However, cycle interruptions are also limited in time, since each state transition (On to Off and vice-versa) is followed by a "refractory period", as depicted in Fig. 6.13. During these periods the heater cannot be switched back into its previous state, since repeated state changes may damage the device and produce dangerous oscillations in the electric grid.

Additional flexibility for power management concerns can be achieved by allowing temperature to come out of the viability domain, dropping below T_{min} or raising above T_{max} by a limited margin δ, typically $3°C$ [MC12]. This is akin to defining an "acceptable" temperature domain $[T_{min} - \delta, T_{max} + \delta]$ that extends the range of possible consumption shifting for the purpose of power management. However, this also means comfort alteration for the user, who may see his temperature goal not fully respected.

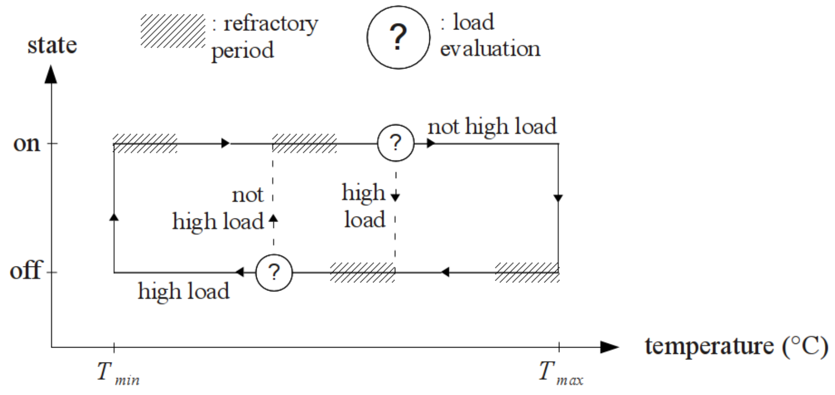


Figure 6.13: Smart thermodynamic cycle featured by an autonomic heater, with possible transitions induced by power management.

Solving this conflict between comfort and power management, i.e. the user deciding whether or not she will tolerate comfort degradation for the sake of power management, is therefore a primary concern for the implementation of such a multi-objective autonomic heater. The multi-objective heater manager makes this choice explicit, by requiring the user to formulate a **meta-objective** solving the conflict, namely a heater execution mode "eco" or "comfort":

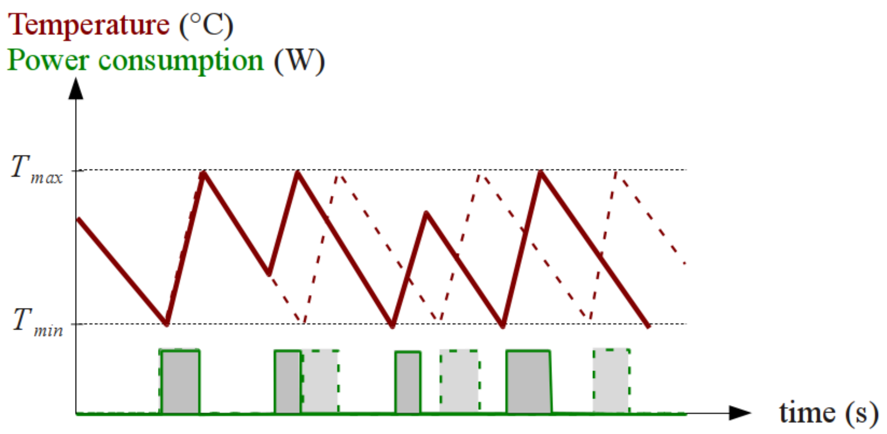


Figure 6.14: Effects of power management on thermostatic behaviour, showing the shifted consumption periods after a cycle interruption.

- ◇ in "comfort" mode, the user does not tolerate comfort reductions. Therefore, the heater manager may interrupt or anticipate its heating cycle, but it should always maintain a $[T_{min}, T_{max}]$ temperature interval.
- ◇ in "eco" mode, the user accepts a limited comfort reduction: the heater manager may let the temperature drift out of the original $[T_{min}, T_{max}]$ interval. However, minimal comfort conditions are still required, under the form of a non-negotiable target $[T_{min} - \delta, T_{max} + \delta]$.

This management policy can be adapted to various thermostatic loads not limited to electric heaters, namely air conditioners or hot water tanks. This power management opportunity has been identified in multiple contributions in the literature, and the corresponding flexibility potential is being investigated by significant industrial projects [BBH12] [MC12].

Concrete design of the multi-objective heater manager

Let us now suppose that the heater is to be integrated in a power management organisation, following a prosumption objective for the house. The architecture and functioning of such an organisation will be presented and discussed in chapter 7. Here, we will simply suppose that a black box component implements integration with the house power management system, namely:

- ◇ advertising the heater's state, including its power consumption, and acceptable power management objective: whether or not the heater is susceptible to be shut down or switched on for power management purposes, depending in turn on whether it is in a refractory period of its cycle.

- ◇ receiving a power management objective: in this case instructions to switch on or shut down so as to attain the house's prosumption objective. We suppose here that these orders are properly determined and relevant for power management purposes (cf. chapter 7).

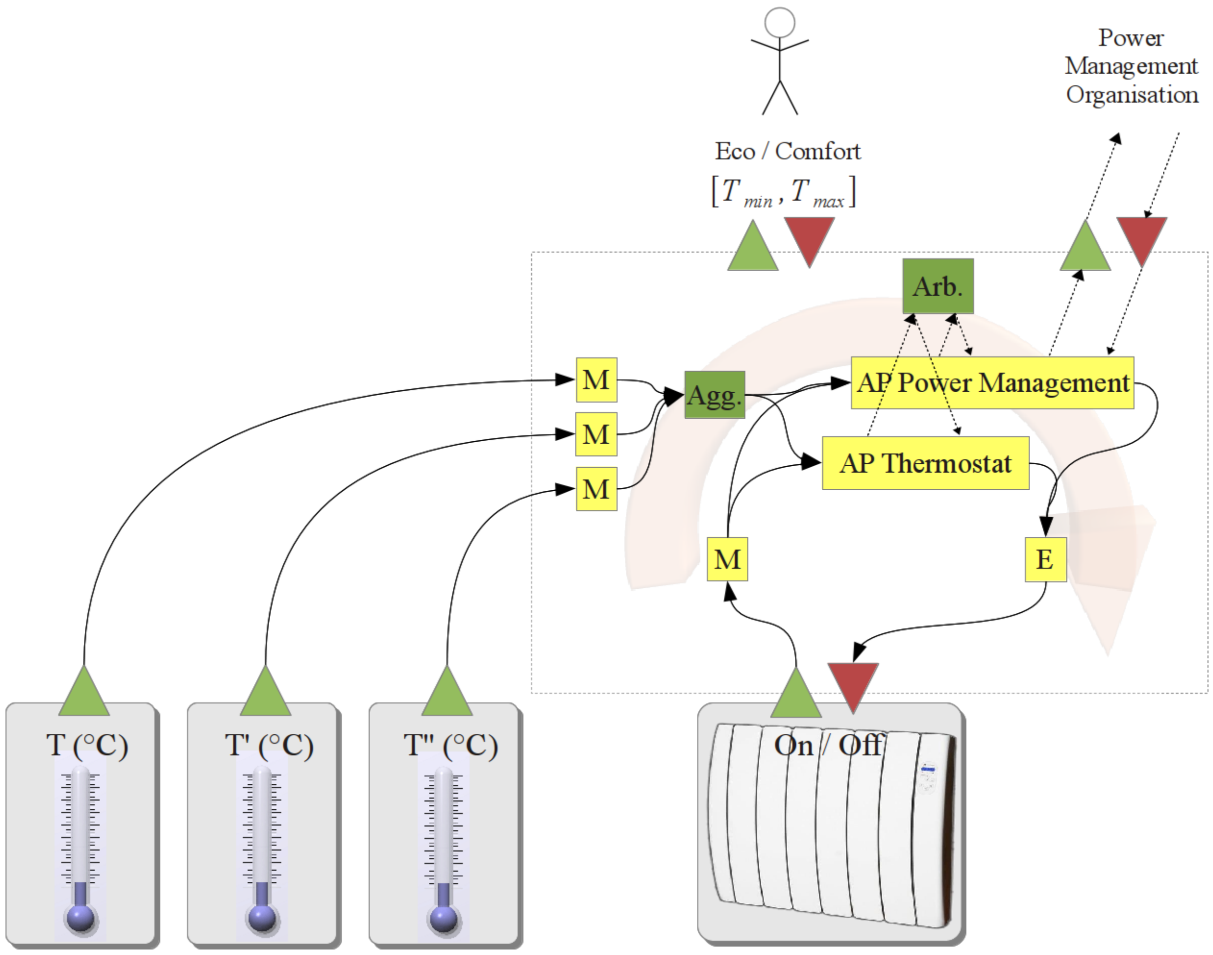


Figure 6.15: Implementation of a power-sensitive autonomic manager for a smart heater combining the Aggregator and the Arbiter integration patterns.

The integration component, represented in Fig. 6.15 as "AP Power Management", is conflicting with the existing "AP Thermostat" resource, uniquely concerned by maintaining its target temperature range. The Arbiter pattern is adopted to resolve this conflict (alternative solutions discussed below). An Arbiter integration resource is deployed, performing a selection function via priorities given to the two conflicting resources. This integration solution solves the conflict, according to the user's meta-objective, as follows:

- ◇ in "comfort" mode, higher priority is granted to the "AP Thermostat" resource, preventing the "AP Power Management" from driving temperature out of the target $[T_{min}, T_{max}]$ interval. This still allows power management, since cycle anticipations and interruptions are authorized in the target temperature range.

◇ in “eco” mode, higher priority is granted to the “AP Power Management”, allowing it to sacrifice temperature for the sake of power management. Since it has a lower priority, the “AP Thermostat” can maintain standard comfort ($[T_{min}, T_{max}]$) only in case no power management is necessary. In case temperature derives so much it leaves the $[T_{min} - \delta, T_{max} + \delta]$ interval, the thermostat resource disposes of maximum priority, so as to maintain minimal comfort conditions.

This conflict resolution solution is generic, since the Arbiter is agnostic to the resources it actually handles: only the distribution of priorities, chosen in accordance with the meta-objective (“eco” or “comfort”) tells the Arbiter which resource should be allowed to execute, and which one should be denied. In case a third objective is to be taken by the heater, for instance, emergency shut-down for safety reasons, the Arbiter would simply detect a new “AP” control task with a particular priority competing with the existing ones.

A Dealer or an Aggregator pattern could also implement the conflict resolution mechanism. The main difference would be that these patterns are transparent, i.e. a resource being overridden by another one would not be notified of the situation. In the case of the Arbiter, lower priority resources explicitly receive a notification forbidding execution, which allows them to react accordingly (for instance, not asking repeatedly for the heater to shut down).

6.3.4 Extending the approach to more complex management cases

The smart heater use case presented in this section is simple and arguably over-engineered. In practice, a monolithic design may have been sufficient to address the same requirements. We utilised this minimal case to illustrate our approach, which can be extended to design more complex control systems.

For instance, consider a small house with a few appliances (lamps, heaters) and sensors, and simple goals such as heating, lighting and controlling power consumption. A possible approach could rely on a single autonomic manager for the whole house, following all the objectives, centralising all monitoring information, and controlling all appliances at the same time. Such a manager would probably comprise dozens of control components and require several integration patterns. Yet, non-monolithic design would allow adding, updating and removing components at run-time, in order to integrate new sensors and new appliances entering the house and cope with the updates of the various control logics.

Chapter conclusion

In this chapter, we presented a generic architecture for adaptable, multi-objective autonomic managers. This architecture proposes to combine control components with integration components so as to build composite autonomic managers. The proposed design capitalises on modularity and loose-coupling to build flexible and extensible managers out of heterogeneous third-party resources. In turn, modularity raises additional integration issues, which we propose to address via a taxonomy of integration functionalities and a catalogue of integration design patterns. The patterns were identified and detailed, and we compared them based on several metrics such as scalability, robustness or evolvability.

We illustrated our approach via an example design for a smart heater manager, starting from a simple, mono-objective loop performing temperature regulation and progressively increasing its complexity by introducing several thermometers and adding conflicting objectives. More precisely, we first extended the capabilities of the manager to adapt it to variable numbers of sensors available in its environment. This required deploying new control resources onto the existing control loop, at runtime. The Aggregator pattern was selected for integrating the additional control resources into the existing controller. This example illustrated how an autonomic manager could be designed from several loosely-coupled control components and later adapted/extended to include new components. Second, we made the manager multi-objective, enabling it to accept power management objectives from an external house power controller. A dedicated management resource carrying the extra power objective was introduced into the existing thermostat-driven controller. This created a conflict with the existing thermostat control resource, that we solved via another of the proposed integration patterns (the Arbiter). This showed how an existing autonomic manager could be extended while preserving its existing implementation. The resulting multi-objective heater allowed users to define a high-level meta-objective for specifying, at runtime, whether the manager should favour temperature or power management concerns. Finally, we discussed the way in which this example can be extended to more complex autonomic managers, such as a complete smart home controller.

Chapter 7

Generic integration patterns for open organisations of autonomic managers

Patterns are a way to analyze solutions to recurring problems,
make them reusable and communicate them.
Patterns are a way of thinking.
Patterns are also a cult.
http://c2.com/cgi/wiki?PatternsForBeginners
as of July 1st 2013

Chapter introduction

Designing and maintaining organisations of stand-alone, third-party autonomic systems is key to achieving open multi-objective autonomic systems of systems. The main difficulty here comes from the multiplicity of providers and business interests involved in such organisations: a priori, there is no central authority or designer that could ensure the viability and correctness of the overall system from end to end. Furthermore, in open and dynamic contexts, heterogeneous autonomic sub-systems may enter, leave or change their behaviour in the system at runtime. Therefore, integration facilities are necessary to dynamically compose heterogeneous autonomic systems that follow their own goals while also maintaining acceptable collective behaviour and achieving global goals.

This chapter presents a generic approach and conceptual design for facilitating the development of such integration facilities, based on integration **organisations**. An organisation specifies the possible **roles** that autonomic members may fulfil within the organisation, and the possible interactions among these roles. Role interactions provide means for organisation members to observe and/or to control other members, under specific **rules** defined by the organisation. The rules should allow members to pursue their own objectives, while also imposing that they contribute to the objectives of other members or to a global organisation objective. In case objectives are conflicting, the rules define **resolution semantics**.

The key issue in organisations is the **balance of authority** between members. This balance should be flexible and allow a variety of behaviours, from hyper-centralised control favouring the objectives of a specific member, to hyper-decentralised configurations where no one interest takes priority over the other ones. Administrators should be able to evaluate the current state and trade-offs of an organisation, and, if necessary, to re-balance its current authorities.

The chapter is organised as follows. The first section describes an abstract organisation architecture which serves as a basis for formalising a new series of integration patterns dedicated to societies of stand-alone autonomic systems. The patterns are evaluated and compared with respect to several quality criteria, and relevant pattern implementations are identified in the literature. The chapter ends with a sample application of the proposed organisations to power management in the micro-grid.

7.1 Organisations overview

7.1.1 Abstract organisation architecture

When integrating heterogeneous, third-party software systems, a minimal requirement is that of a communication standard, that allows the different parts to connect and exchange with each other. In the case of autonomic systems, such standards are also compulsory. In addition, higher-level integration facilities are also required, specific to the self-adaptive nature of the participants and the self-organisation behaviours their gathering should implement. Namely, autonomic systems should be able to *observe* each other (in terms of their respective state, acceptable objectives, and possible behaviour) and/or *control* each other (providing goals or behaviour constraints).

In this thesis we use the term **organisation** to designate such high-level integration standards designed for autonomic systems. The aim of an organisation is twofold:

- ◇ to provide balance among the goals of the organisation's participants.
- ◇ to allow participants to change their behaviour, join and leave the organisation during runtime.

Concretely, an **organisation** is an abstract specification that captures invariants of a group of autonomic systems. It is expressed in term of **roles** and **rules**:

◇ **roles** capture possible behaviours of participants to the organisation, allowing other participants to observe and possibly to control role-players. A role specifies touchpoints (sensors and effectors) on the participant, providing for instance:

  – the participant's current objectives, and possibly their priority and an evaluation of whether or not the objectives are fulfilled. This in turn requires generic metrics to express priorities and evaluations in a scale common to the whole organisation.
  – the services the participant may provide to and require from other participants, notably, the goals it is able to contribute to and the ones it may require contribution for, and the nature of these contributions.
  – a **status** reflecting the rank of the participant in the organisation, that serves as a basis for expressing rules (cf. below).

  An organisation may define several *role types* and specify cardinalities on *role positions*, including singleton, fixed number, fixed range, at least one, any number, and so on. At run-time, *participants*, i.e. autonomic systems complying with role types, fulfil the role positions. For instance, the power management organisations presented in this chapter feature two role types (Power Manager and Prosumer); each organisation features at most one Power Manager position, and any number of Prosumer positions, depending on available autonomic systems participating to the organisation.

◇ **rules** capture possible interactions between roles, and in particular, organise conflict resolution. Rules specify authorisations or interdictions of role behaviours depending on circumstances, in particular in terms of acceptable goals and control possibilities between autonomic managers. For instance, managers with a "high" status may be granted priority during conflict resolution over managers with "low" status, or, the former could impose their goals onto the latter. Such status-based rules can be translated directly from user priorities with regard to objectives: participants may inherit as a status the priority of their objectives.

Role and rule specifications are rather static (dynamic organisation specifications are out of the scope of this thesis). On the other hand, the concrete participants, their behaviours and their possible interactions are greatly variable and may change during time, while still complying with the generic frame imposed by the organisation specification. Variable cardinalities of role positions allow participants to enter and leave the system during the life-time of the organisation. A verification mechanism may be implemented to actively check whether participants actually comply with their required role specification, though these aspects are out of the scope of this thesis.

7.1.2 Organisations in the literature

Organising multiple autonomic systems represents a major concern that has been explored by several contributions in the literature, of which we give a few relevant examples here.

MACODO [WHH+10] is a role-based middleware for multi-agent organisations. We presented this project in more details in chapter 2. Compared to our generic architecture presented here, MACODO's abstract organisation model is more refined and also translated into an actual software implementation. On the other hand, our architecture is coarser and makes looser assumptions on its participants. This makes the resulting organisation less predictable but also more flexible and open to third-party systems, which MACODO does not consider.

[MSS11] identified the need for so-called "societies" of autonomic systems, governed by laws and higher-level reflexive adaptation. Similarly, contributions such as [PSBM12] and [BKMS+12] explored possible institutions for autonomic societies, inspired by actual human societies. Our contribution conforms to these theoretic initiatives, while providing more concrete, reusable software engineering support for facilitating their development. Namely, in the proposed approach the abstract concept of "society" is reified into a generic reusable architecture.

In addition, to our knowledge, the organisational patterns presented in the next section have no direct equivalent in the literature. Horling04 surveys organisational paradigms for multi-agent systems in general, at a high abstraction level. The patterns they identify do not particularly tackle management conflicts and cannot be used as integration solutions. Contributions such as [BCD+06] [DWH07] [FMAS+11] propose patterns for message-based communication and spatial interactions among mobile agents, such as gradient-based chemotaxis. Again, these contributions are not directly applicable to the control and conflict resolution issues we consider here.

7.2 Organisational patterns

This section presents four concrete integration patterns - **Hierarchy**, **Stigmergy** and **Collaboration** - that aim to provide reusable software engineering support for setting in place organisations. Each pattern follows a different implementation strategy and presents different characteristics, that we compare. The patterns presented in this section aim to address the generic situation where a set of autonomic systems (the "participants"), with their particular objectives, belong to the scope of a collective global objective (cf. Fig. 7.1). This will be the case for instance for autonomic prosumers in a grid being in the scope of a power management objective for this same grid.

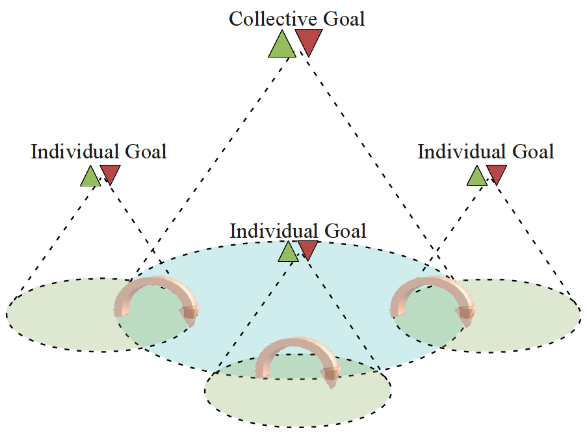


Figure 7.1: Generic conflict situation that the patterns aim to address.

Participants are in the action scope of the global objective, that is, the global objective intersects the scopes of the participants' individual objectives. This is a direct conflict situation between local objectives and the global objective. For instance, a local temperature objective for an autonomic heater conflicts with the global prosumption objective for the house. Local objectives also conflict with each other, at least indirectly, since one participant not helping with the global objective may put additional constraints on other participants. For instance, a prosumer not reducing its consumption during a load peak may indirectly force another participant to do so, and vice-versa.

7.2.1 Hierarchy pattern

Context: There is an acceptable centralised solution to the conflict, where an orchestrator carries the global objective and ensures integration with the participants at the same time.

Description: The integration organisation comprises a single *Hierarch* role and several *Subordinates*. The Subordinates expose their capabilities through a public status to the Hierarch, such as informations

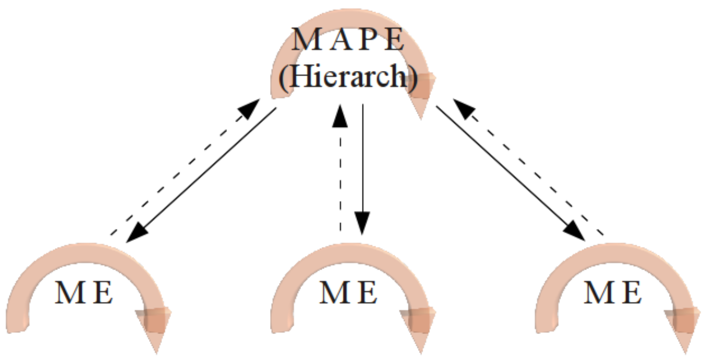


Figure 7.2: The Hierarchy pattern.

on their own objectives, their ability to participate to the global objective, or possibly an evaluation function. The Hierarch in turn computes an optimal solution, including particular behaviours of Subordinates, and sends direct, individual orders to those for orchestrating the solution. Possibly, rules may allow Subordinates to escape orchestration orders when given the appropriate derogatory status by the user.

Advantages and Limitations: The Hierarchy pattern relies on centralised orchestration with most of the management and conflict resolution logic being contained in a central entity. Therefore it is rather intrusive in that it forces Subordinates to publish some of their internal state and rely on a third-party manager to regulate their behaviour. The Hierarch control precision is in general proportional to the amount of information available on Subordinates, and therefore to coupling in the organisation and complexity of the orchestration logic.

Evaluation: Central orchestration is in general both efficient and effective, since handling conflict resolution in a single manager allows for both optimisation and verification of the control logic. It will be easier to reason about the global objective and ensure global properties from the Hierarch point of view, possibly to the detriment of local objectives.

Indeed, the drawbacks of this centralised solution may be perceived first form the Subordinates' point of view: the Hierarch may not be able to take all their specificities into consideration. The more information Subordinates make available, the more complex the orchestration logic becomes, and the latter becomes a bottleneck of the management flow, and possibly a single point of failure of the system. Scalability and capability of the solution to integrate widely heterogeneous Subordinates may be limited for the same reason. Depending on how specifically the Hierarch orchestrates Subordinates, the latter may or may not be able to function properly in case the former fails or disappears from the organisation.

Examples: Hierarchical organisations of autonomic managers are common in the domain, starting with the very first Blueprints for autonomic computing by IBM [IBM06]. Notable examples in frameworks for autonomic systems include AutoHome [BDLM11] which relies on a hierarchical organisation of managers, or Automate [ABL+03] which makes use of central "access control agents" to coordinate concurrent resources. Next section presents a smart-grid specific power management organisation model that also presents hierarchical features, utilised in this specific domain by solutions such as [htth], [BAR+10] or [PKR12].

7.2.2 Stigmergy pattern

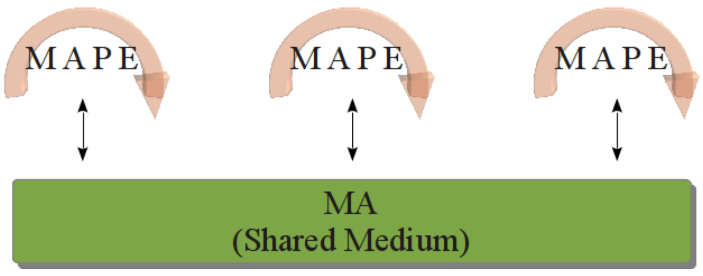


Figure 7.3: The Stigmergy pattern.

Context: There is a decentralised solution, relying on participants handling themselves their integration with the global objective via an organisation-wide shared state, akin to a blackboard pattern [Cor91].

Description: The organisation features a single role type, Peer, that read and write into a global shared state. The shared state may feature additional monitoring and analysis functionalities providing high-level diagnostics on the situation in the organisation, however there is no centralised planning or decision logic in the pattern. Each Peer, in turn, is responsible for evaluating its possible contribution to the global objective and adapt its own behaviour accordingly. In particular, Peers may have to implement collective coordination mechanisms so as to avoid over-reactions or oscillations of the global state.

Advantages and Limitations: When relying on choreography, achieving the global objective successfully entirely depends on the effectiveness of the decentralised control by Peers. In return, Peers are in charge of integrating their specific objectives to the global one. Hence, difficult integration functionalities are implemented on the Peer side, keeping the global monitoring and analysis features specific to the domain of the global objective.

Evaluation: Since it is decentralised, the pattern should scale well: the global shared state is a bottleneck of the organisation, but it does not feature complicated decision logic. Communication solutions such as publish-subscribe based broadcast allow efficient organisation-wide diffusion of the global state. The whole organisation should be robust to individual Peer failures. Relying on Peers to implement their own specific integration logic increases the universality of the global Organisation.

What the pattern provides in terms of scalability and robustness, it lacks in terms of control precision and insurance the global objective will be effectively reached. Proving a decentralised control solution will converge fast enough to an acceptable global situation may be difficult or even impossible. Over-reaction or oscillatory behaviours can appear, that may require additional and possibly costly global coordination mechanisms, which would reduce the scalability and robustness of the solution while increasing its complexity. Such coordination mechanisms include: global synchronisation via a common clock (either centralised or decentralised) and probabilistic behaviours (cf. next section for an example of these).

Examples: Stigmergy is a pattern that is often found in emergent systems, notably those inspired by collective insects communicating through their environment [JMB05] [BCD$^+$06] [BM08] [FMAS$^+$11]. It is particularly relevant for mobile wireless networks where system dynamism and environment unpredictability rule out centralised control solutions. For the specific domain of power management in smart grids, notable solutions such as [BBH12] and [MC12] utilise a stigmergic architecture to address ultra-large scalability concerns.

7.2.3 Collaboration pattern

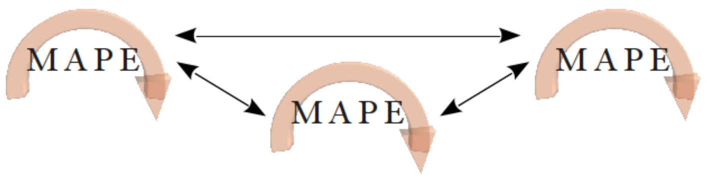


Figure 7.4: The Collaboration pattern.

Context: There is a decentralised solution where participants negotiate their respective behaviours and the achievement of the global objective, in a peer-to-peer fashion.

Description: There is a single Peer role in such an organisation. Each Peer is in charge of its own behaviour as well as of negotiating directly with other Peers to achieve the global goal.

Advantages and Limitations: This pattern is generic as it encompasses a wide range of multi-agent approaches. Peers may implement various dialogue or competition algorithms, using either point-to-point or broadcast communications. The only restriction is that no Peer should play a central role that would be indispensable to the organisation control flow.

Evaluation: Since it is completely decentralised, the Collaboration organisation is free of bottlenecks and single points of failure that would limit its scalability and robustness. Peers are in full control of their behaviour, therefore they are pretty much ensured to reach their individual objectives - to the possible detriment of the global one.

However, decentralisation comes at the cost of increased communications between Peers, which may prove costly in terms of global performance. In addition, research of a consensus for reaching the global goal may require long convergence times, or even not converge at all, which of the main drawback of decentralised solutions. Designing and implementing a correct decentralised algorithm is usually more difficult than in centralised situations.

Examples: The domain of artificial intelligence is rich of multi-agent decentralised organisations such as [JB03] [Nag04] [Ant04] [KST09]. This very generic pattern embraces a number of contributions in various domains. In the particular case of smart grids, as next section discusses, notable contributions such as [MDC$^+$07] [MRWJ$^+$10] [Kim11] [Che12] rely on decentralised power control architectures.

7.2.4 Organisation evaluation and adaptation

In this section we propose a theoretical approach for evaluating organisations, that we present as a strong indication for future work in this direction.

Stand-alone autonomic systems provide built-in evaluation facilities through goal sensors, reporting whether or not the system actually reaches its objectives. Autonomic management organisations provide similar evaluation facilities at a macro, collective level by analysing how conflict resolution affects participants and possibly prevents them from achieving their objectives.

For instance, let us consider an organisation that successfully reaches its macro objective. However, at the same time, the state of its members indicates that many micro objectives were sacrificed for the sake of the collectivity. Further analysis and tuning of the participant's configuration may allow reducing to a minimum the negative side effects of imposing the global goal over the individual ones.

On the other hand, an organisation that fails to reach its macro objective may reveal a variety of reasons for that unsatisfactory result:

- ◇ participants statuses shows that micro objectives are globally favoured over the macro objective; this may be the user's choice, and in case it is not, the solution lies in lowering the micro-objectives' priorities.
- ◇ participant statuses show they are benevolent towards the macro goal, but they are not sufficiently numerous of powerful to achieve it; in this case the solution is to increase the number and/or capacities of participants.
- ◇ participant statuses show they are benevolent towards the macro goal, and their number and capabilities should be sufficient to reach the goal; in this case, faulty participant behaviour is probably behinf the unsatisfactory organisation behaviour.

Fig. 7.5 proposes a graphical interpretation of trade-offs in organisations, inspired by the concept of Pareto efficiency [Osb94]. The graphs aim to represent the state space of an organisation goal achievement, depending on its participants respective goal achievements. They were drawn here by hand with arbitrary examples for illustration purposes. In a real situation, such graphs would be obtained at runtime via collecting evaluations from each participant and the organisation assessing their respective goal's achievements. Such information would then be used by system administrators (or possibly a meta-manager) to reconfigure the organisation's rules and/or the participants behaviours.

Let us consider a simple organisation composed of two participants, each one with its goal and respective viability domain. The organisation that these two participants belong to has a global objective defining its own viability domain. In the three graphs A, B and C, each axis represents the state space of each participant, with two alternative policies: an "unflexible" policy - targeting a reduced, supposedly optimal viability domain; and, a "flexible" policy - allowing a larger, less constraining viability domain. For both policies, desirable states are represented in green on the axis, red denoting undesirable states. The graph then represents, for a given state of each participant, whether the resulting global state is acceptable (green point) or not (red point) from the organisation's perspective.

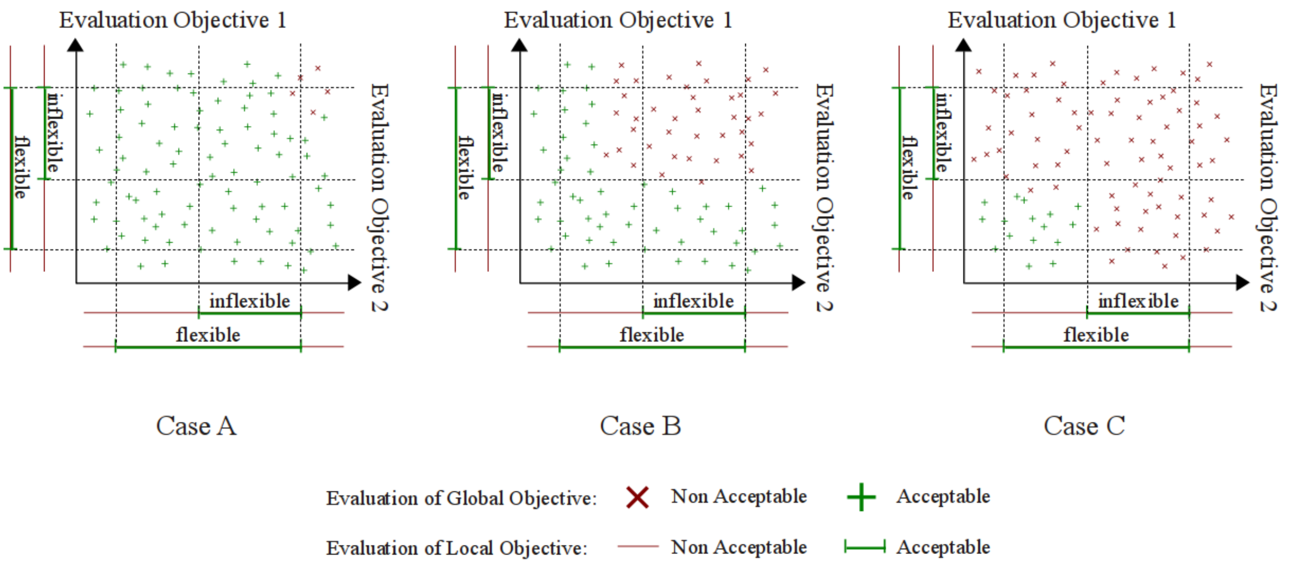


Figure 7.5: Representation of trade-offs in an organisation.

In graph A, conditions are quite favourable, since both participants can be "inflexible" and hence achieve maximum quality of service while still allowing acceptable results at the global level. In graph B, the situation is less favourable, since the global objective would not be fulfilled unless one of the participant is in "flexible" mode. Or, conversely, if both participants are "inflexible" the global goal will not be achieved. In graph C, the situation is even worse, since both participants must be "flexible" in order to attain the global objective.

This illustrates a strength of the proposed approach: no assumption is made on the actual behaviour of the system, in particular, on whether or not individual policies affect the achievement of the global objective. Instead, each participant formulates its preferences, and chooses whether or not to participate to achieving the global objective, which is also evaluated dynamically. Therefore, the organisation and the participants are able to explore their unpredictable state spaces and adapt to the situation accordingly. In case trade-offs must be made, statuses provide an intuitive way to evaluate possible outcomes and choose which objectives or participants should be privileged over the others.

7.3 Application: power management organisations

This section presents a sample application of the proposed patterns in the form of power management organisations for the smart grid. They solve the issue of integrating autonomic prosumers, with their particular usages, objectives and behaviours, with a power management system carrying energy objectives. We first expose a generic, abstract organisation, and then several variants, each derived from a particular pattern. We discuss their respective strengths and weaknesses, and study their applicability to the micro-grid use case.

In section 7.3.1 we present a specification of generic power management organisations. In section 7.3.2 we show how integration in such power management organisations is equivalent to solving a scheduling problem. In section 7.3.3 we discuss how to integrate several such organisations in a holonic fashion, namely integrating several house organisations with a district organisation. Finally, in section 7.3.4, we present concrete designs of power management organisations following the different patterns we proposed before.

7.3.1 Generic organisation specification

Let us consider a simple grid containing a set of prosumers and a single aggregator. The aggregator's "prosumption" P_{agg} is equal to the sum of prosumer prosumptions:

$$P_{agg} = \sum_{prosumer \in grid} p_{prosumer}$$

We suppose the power objective for this grid defines a viability domain $[P_{agg}^{min}, P_{agg}^{max}]$ with arbitrary boundaries, depending on runtime user preferences. The actual decision process leading to a target prosumption domain will be discussed in section 7.3.3.

Roles & Rules

A power management organisation defines two roles:

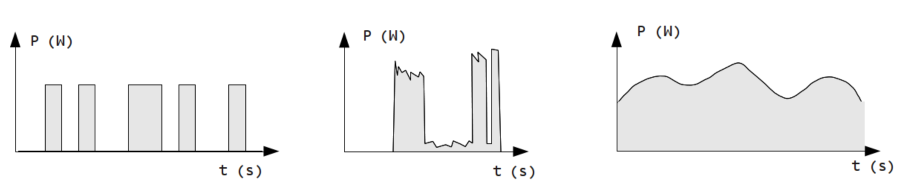
- ◇ a unique **Power Manager** role, implemented by the manager carrying the power management goal. This role is indeed optional, as some variants of the power management organisation, presented below, do not feature a Power Manager, or a non-decisive, reduced version of it.
- ◇ a variable number of **Prosumer** roles, implemented by prosumer managers. A Prosumer is notably characterised by its **status**, either *Flexible* or *Non-Flexible*, advertising the prosumers availability to the power management objective.

Prosumers advertise their current state and capabilities via the following:

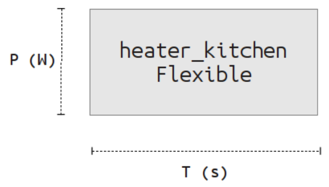
- ◇ **Prosumption packets** modelling prosumer prosumption, either present, past or future in case estimation are available. Prosumption packets may contain the following information:
  - time markers, such as packet start and end dates, earliest and latest deadlines in case packet boundaries are variable, packet length (possibly minimum or maximum).
  - prosumption measures for the packet duration, such as mean, minimum, maximum prosumption or total energy consumed.
  - possible time dependencies between packets, in case prosumer usage (e.g., a washing machine with several successive cycles) follows several successive profiles.

  In a real-world scenario, a proper, detailed taxonomy of prosumers prosumption profiles would be necessary, which is not the purpose of the simple prosumption packet model presented here.
- ◇ a **status**, either *Flexible* or *Non-Flexible*, advertising the prosumer's availability to the power management objective. Status can be attached to a prosumption packet and precise which parameters of the packet (time boundaries, prosumption levels) are susceptible to be adapted. Advanced status may propose possible reconfiguration alternatives in terms of acceptable power objectives, advertising a range of possible deadlines or the different consumption levels an appliance can adopt.

The Power Manager can monitor the state of the grid through its Prosumers' Prosumption Packets (either past, present or predicted future) and determine whether or not the target prosumption domain will be attained or not. In case the goal is not fulfilled, the Power Manager may issue power management objectives to Prosumers, redefining their target prosumption domain. Such objectives are similar to scheduling **orders**, shifting packets' start date or prosumption levels, or choosing amongst alternative configurations. The Power Manager may also advertise possible goal violations in advance, for instance warning Prosumers not to consume between 6:00 and 9:00 pm.



(a) Sample appliance consumption profiles.



(b) Prosumption packet.

Figure 7.6: Prosumption packets model variable prosumption profiles.

The key integration point lies in that orders may only target Flexible packets, since Non-Flexible ones are not supposed to be reconfigured. Therefore, in case a Prosumer favours its own particular objectives over power management, it advertises a Non-Flexible prosumption packet. On the other hand, Prosumers with “eco” profiles would show Flexible profiles and propose reconfiguration opportunities for serving the power goal.

7.3.2 Conflict resolution via scheduling in power management organisations

Power management in the grid is equivalent to a scheduling problem, namely fitting prosumption packets in the grid’s target prosumption domain at any given time. Whether the scheduling problem has a solution or not depends on how collaborative Prosumers are towards the power goal, which in turn depend on user preferences setting the respective priorities of objectives.

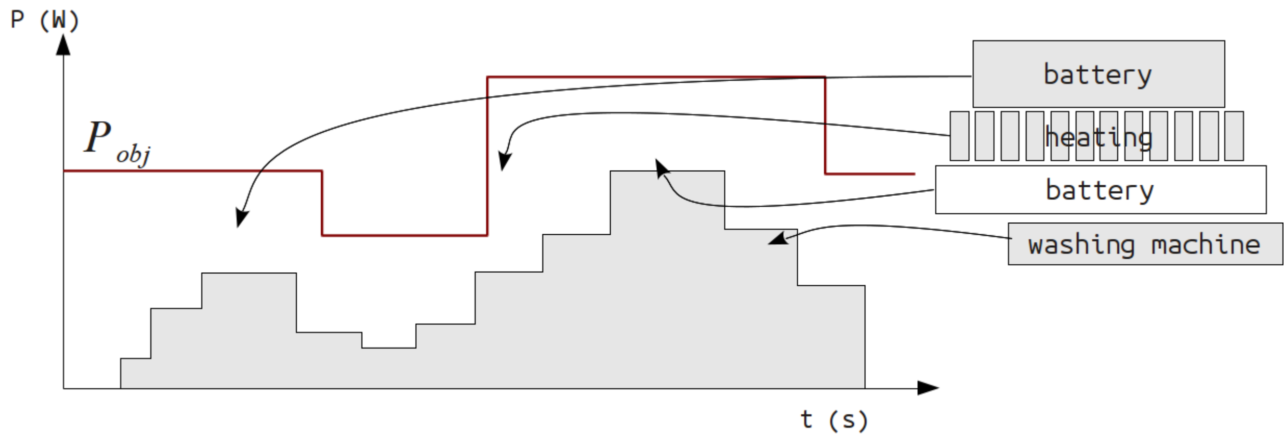


Figure 7.7: Example representation of a power scheduling problem.

In cases where Prosumers all follow “comfort” profiles, meaning they favour quality of service over energy savings, poor flexibility will be available from prosumption packets, and the Power Manager may not have the possibility to fulfil its power goal. On the other hand, “eco” Prosumers should offer opportunities to the Power Manager to find appropriate power schedule for the grid. Whether or not it manages to achieve the global power goal, the organisation ensures that a **best effort** approach is taken, allowing every possible Prosumer contribution to the power goal to be considered.

At the house level, where the same administrative authority sets all management objectives, the trade off between power goal and other goals, such as comfort, can be explicitly formulated. However, at the district level where Prosumers are private houses and the power goal is set by a grid administrator, the situation is much more competitive and the outcome of the scheduling process may not be easily predictable. Chapter 8 proposes several simulated scenarios depicting possible outcomes of the scheduling under variable conditions.

7.3.3 Integration of organisations at different holonic levels

House-level Power Managers are a special case, since they play both a Power Manager role in a house power management organisation, and a Prosumer role at the district level. Since they are common to house and district organisations, they provide a bridge between the two and their behaviour reflects the holonic nature of the grid (cf. chapter 5). This section details this behaviour and the implications at both grid levels.

Aside from issuing power management orders to local Prosumers, the key functionality of the house Power Manager is translation: translating the state of the local house grid into a Prosumer-like representation for the district grid, and translating power management orders received from the district grid into orders for local Prosumer appliances. This translation is related to the holonic nature of the micro-grid: the house grid with all its appliances and its own power management control system and objectives is seen as a single black-box Prosumer from the district point of view.

When translating the local grid state for the district power management organisation, the house Power Manager aggregates local Prosumption Packets into house-level Packets. These Packets represent the Flexible and Non-Flexible parts of the house prosumption, as well as alternative house profiles, aggregated the individual alternative appliance profiles. The granularity of prosumption packets, both in terms of time and prosumption levels, should be coarser at the district level than at the house level. This reflects the different orders of magnitude in prosumptions, number of prosumers and reaction time between the two scales, in addition to accumulated variability and possible errors due to the aggregation mechanism.

Conversely, when receiving a power management order from the district, the order defines or redefines a target prosumption domain for the house. Hence, when accepted, this new viability domain will automatically be enforced by the local scheduling mechanism that will turn it into new appliance-level target prosumption domains, according to Prosumers' capabilities at the moment. This is the reverse side of the black-box nature of the house: from the district point of view, it does not matter which local Prosumer will actually modify its behaviour, as long as the house-level goal is reached. In case in-house conditions are such that the house prosumption target cannot be reached, the district will see one of its Prosumers not achieving its goal, which requires another Prosumer to compensate for it.

7.3.4 Possible organisation variants

The actual implementation of scheduling in the power management organisation may follow any of the three patterns presented previously. This section presents and compares these variants, and discusses their applicability to the micro-grid.

Centralised scheduling: the Hierarchy pattern

Hierarchical Power Management organisations implement scheduling in a classical fashion, with a centralised scheduler, the Power Manager, issuing scheduling orders to Prosumers so as to attain its prosumption target for the grid. This kind of organisation presents the qualities and limitations of centralised architectures:

- ◇ the Power Manager can ensure fast, precise control of Flexible Prosumers and enforce optimal scheduling solutions.
- ◇ the Power Manager represents a potential bottleneck and single point of failure of the organisation, which may limit the scalability of the solution; the organisation can not function in case the Power Manager fails.
- ◇ the more sophisticated Prosumer profiles are (with respect to prosumption models, advertised scheduling alternatives) the more precise their scheduling can be, and the more complex and costly the Power Manager's scheduling algorithm becomes.
- ◇ in case emergency procedures should be executed so as to avoid black-outs, the Power Manager can use special orders with exceptional priority to rapidly enforce Prosumer shut-down.
- ◇ the scheduling logic being centralised, it is easier to monitor and modify than in decentralised solutions, shown below. In particular, a central scheduler may enforce fairness amongst Prosumers in the scheduling process (for instance, considering commercial agreements at the district level) that may not be easy to implement in decentralised, competitive organisations.

Examples: NiceGrid project controls residential storages and heat pumps in a district, allowing it to go into "island mode" for a limited period of time each day, cutting consumption completely from the parent grid [htth]. This kind of controller provides good control of the district prosumption on demand of the parent grid, yet its influence is limited to specific devices in limited numbers. Furthermore, data privacy is a pending concern each time appliance control is deported out of its owner's reach. [BAR+10]

[PKR12] try to bring Hierarch-like solutions into houses, centralising control of smart appliances into a house-level loop. Appliances provide profiles and user preferences to the hierarch which in turn computes optimal prosumption scheduling and executes it. Since appliance profiles are hard to establish it is unsure how such central hierarchs would apply to very different house configurations, and how tolerant they would be to configuration variations (appliance churn).

Decentralised scheduling: the Stigmergy and Collaboration patterns

Decentralised Power Management organisations reduce the role of the Power Manager to monitoring and global analysis of the organisation's situation. Instead of a centralised scheduling algorithm, as in Hierarchical organisations, Prosumers adapt to the situation in the grid, according to the following variants:

- ◇ Stigmergic organisations rely on shared global state, namely a global schedule where participants publish information accessible to the whole organisation. In this schedule, Prosumers advertise their prosumption packets; the Power Manager publishes organisation-wide analysis information such as the grid's target prosumption domain, aggregated prosumption levels, overshoot with regard to the objective and possibly coordination data (discussed below). Then, it is up to each Prosumer to monitor the schedule and adapt its prosumption behaviour to it. A priori there is no participant-to-participant communication in a stigmergic organisation.
- ◇ Collaborative organisations may use any kind of communication, be it broadcast or participant-to-participant communication. Since the collaboration pattern is extremely generic, many decentralised scheduling solutions may fall in this category, based on well-known multi-agent approaches such as market places [Che12], Contract Net Protocol [Kim11] or game theory [MRWJ+10].

In both cases, decentralisation raises possible coordination issues to avoid oscillations (for instance, all Flexible Prosumers reacting to a high-load situation all at the same time, resulting in an excessive load drop). Several algorithmic solutions can be applied, such as a centralised clock or token ordering Prosumer reactions one after the other. Stigmergic organisations may also rely on probabilistic behaviour: in case of goal overshoot, the Power Manager computes a global estimate of the necessary economies, associated with a probability for Prosumers to adapt to this event. Each Flexible Prosumer then adapts its prosumption with the specified probability, so that the expected economy is realised at the organisation level. As a simple example, with 100 Prosumers consuming each 1W, economising globally 10W will be achieved by asking Prosumers to shut down with probability 1/10. Such probabilistic methods are designed to cope with ultra-large scales, working with millions of Prosumers, and may not be efficient at small scales [BBH12] [MC12].

In comparison with hierarchical solutions, decentralised organisations exhibit the following characteristics:

- ◇ since it is decentralised, the scheduling decision process may converge slowly and sub-optimally.
- ◇ since it is free of bottleneck and single point of failure for their decision logic, the organisation can scale more easily, since it can support significant Prosumer churn and resist to individual failures. Stigmergic solutions, in particular, are known for scaling up to millions of devices.
- ◇ since each Prosumer is in charge of scheduling itself with respect to a global state, Prosumer-specific decision logic is implemented by the Prosumers' manager itself. Therefore organisation designers are partially relieved from taking all possible Prosumer specificities into account.

Examples: Recently, a new class of grid control architecture have emerged, that fit into the stigmergy pattern. These proposals aim to provide ultra-large scale control of individual appliances via the use of global shared state and probabilistic control. [BBH12] relies on users setting flexibility preferences for their appliances via a color code. [MC12] relies on the ability of thermal storages (heaters, fridges, air conditioners...) to switch on or off rapidly at certain points of their thermodynamic cycle without comfort degradation. Both of these solutions scale up nicely while allowing massive appliance variety and runtime variability, be it the result of changing user preferences, device churn or partial failures in the grid. A significant drawback of such extreme scalability is the incompatibility of these solutions with small scales. For instance, allowing a single house to go into "island mode" and act as if disconnected from the district grid is not a possibility these solutions address.

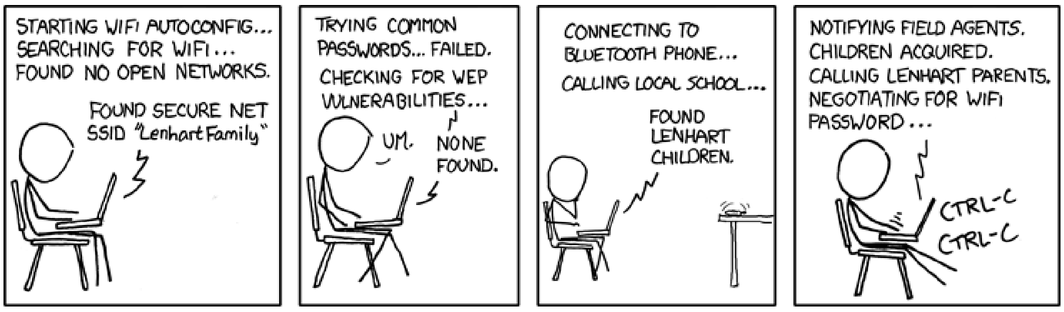
Chapter conclusion

In this chapter, we presented a generic approach for the design of open organisations of heterogeneous, third-party autonomic systems. Organisations rely on roles and rules that specify possible profiles of participants and enforce conflict resolution among participants. We identified several generic organisational patterns that we presented and compared with respect to several metrics including scalability, design complexity and robustness.

To illustrate our approach, we proposed a generic power organisation template for micro grids. Such organisations are intended to integrate smart appliances at the house scale, and smart houses at the district scale. The organisation resolves the conflicts between a global power objective and several prosumer objectives, according to several possible resolution semantics. In this case, conflict resolution is equivalent to solving a scheduling problem with power prosumptions, according to user preferences in terms of power and comfort objectives. Therefore, the outcome of conflict resolution depends on whether the scheduling problem will have a solution or not. Proof-of-concept implementations of power management organisations are presented in next chapter, along with promising experimental results.

Chapter 8

Proof-of-concept implementation and experiments


STARTING WIFI AUTOCONFIG... SEARCHING FOR WIFI... FOUND NO OPEN NETWORKS.
FOUND SECURE NET SSID "Lenhart Family"
TRYING COMMON PASSWORDS... FAILED. CHECKING FOR WEP VULNERABILITIES...
UM.
NONE FOUND.
CONNECTING TO BLUETOOTH PHONE... CALLING LOCAL SCHOOL...
FOUND LENHART CHILDREN.
NOTIFYING FIELD AGENTS. CHILDREN ACQUIRED. CALLING LENHART PARENTS. NEGOTIATING FOR WIFI PASSWORD...
CTRL-C
CTRL-C

http://xkcd.com/416

Chapter introduction

This chapter presents MisTiGriD, the platform we developed as a validation support for our approach. MisTiGriD features a simple grid simulation, modelling the behaviour of several houses in a district, each house containing various appliances, either producers or consumers. MisTiGriD also features a temperature simulation, models heat transfers, so as to create dynamic environmental conditions for a smart heater use case. Several variants of autonomic managers and management organisations have been implemented and tested on the simulator, as a proof-of-concept of the generic contributions we proposed.

MisTiGriD is **not** intended to simulate realistic grid models, environmental conditions or user behaviours. Indeed, many physical values and user inputs used in our scenarios are arbitrary and rather unrealistic at times. For this reason, MisTiGriD is not a candidate support for practical grid management algorithms. Its actual purpose is to provide a dynamic, extensible, interactive simulation environment for multi-objective autonomic managers and organisations of such managers. The presented implementations, experimentation and results are intended to:

◇ exemplify, via simulated artefacts, the goal-driven analysis framework we presented in chapter 5.
◇ show how multi-objective managers developed based on the proposed integration design patterns can function in dynamic, unpredictable environments, and be able to adapt to variable user objectives and conflict resolution semantics.
◇ show how several autonomic managers designed in this manner can be integrated via the various organisation patterns we presented, and how the resulting organisations will behave under variable user objectives and environmental conditions.

This chapter first focuses on the most important features of the simulation platform, and details relevant examples of manager implementations in MisTiGriD. A physical extension of the simulator, consisting of a smart house model in miniature is also presented. Then, we describe the prototype implementation of autonomic managers and power management organisations, at the appliance, house and district level. Finally, we discuss several simulation scenarios at the appliance, house and district levels, as they illustrate our approach with concrete autonomic behaviours.

8.1 Simulation features

This section presents the main characteristics of MisTiGriD's simulation of temperature, electrical network and appliances. A MisTiGriD instance typically simulates a house with its rooms and appliances. Therefore, to simulate a whole district with several houses, several MisTiGriD instances are deployed in parallel, possibly on different distributed machines, which are then interconnected via a network.

8.1.1 Temperature

MisTiGriD features several types of *thermic objects* that model systems with a temperature: rooms, heaters (in rooms) and the outside atmosphere. Adjacent thermic objects exchange heat through their common interface, be it a house wall (between rooms and atmosphere), a room wall (between rooms) or an appliance's physical envelope (between a heater and a room). The interface can contain an opening, such as a door or a window, that can be opened and closed at runtime, modifying the conducting properties of the interface.

The atmosphere is considered a "thermal reservoir": its temperature is not influenced by other thermic objects, and the user can set it to arbitrary values at run-time to simulate variable weather conditions. For non-atmosphere thermic objects, MisTiGriD models temperatures via Fourier's law of heat transfer [httb], implemented as follows.

Let us consider τ a thermic object of temperature T (in K) and heat capacity C (in $J.K^{-1}$), inversely proportional to its volume. Let N be the set of neighbour thermic object in contact with τ: for a given neighbour n, k_n is the heat transfer coefficient between τ and n (in $W.m^{-2}.K^{-1}$) which depends on the nature of the surface (house wall, room wall, heater convector, etc.) and state (presence of door or window, either open or closed). A_n is the contact surface between τ and n (in m^2).

Then, evolution of T is determined by τ's heat transfer with N, noted dQ and defined as follows (illustration on Fig. 8.1):

$$T(t+dt) = T(t) + C.dQ(t,dt)$$

$$dQ(t,dt) = -\sum_{n\in N} k_n A_n [T(t) - T_n(t)].dt$$

Given that expected temperatures T evolve around $20°C$, with wall surfaces A of a few square meters, the actual values of C and k have been empirically determined so as to yield visible results for a few seconds of simulation. Realistic values taken from existing thermal diagnostic tools would not change the simulator's behaviour qualitatively, and would cost additional tuning so as to provide visible results at run-time, which are MisTiGriD's primary purpose.

Given this static temperature model, the layout of rooms is entirely configurable by the experimenter, who can deploy any number of room in a house with arbitrary wall topology and connections between thermic objects. Fig. 8.1 shows a possible room layout and the resulting simulated heat transfers.

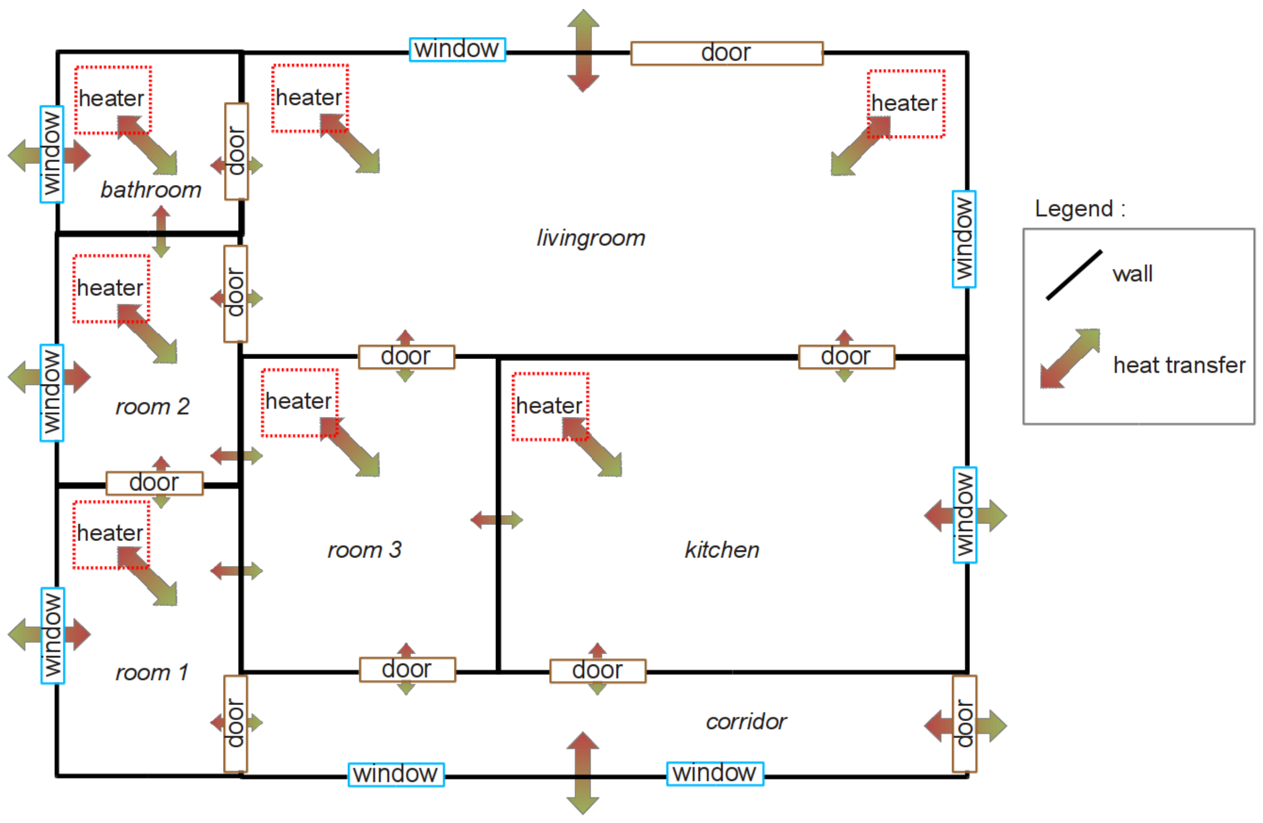


Figure 8.1: Sample house layout, showing rooms, heating appliances and simulated heat transfers.

8.1.2 Electric network

The electrical network in MisTiGriD is a tree of depth 2: leaves are domestic appliances, their prosumptions are aggregated by their father at the house level, house aggregators are themselves the children of the root of the tree, i.e. the district aggregator. Each aggregator Agg (either house or district) is equivalent to a prosumer with a prosumption p_{Agg} equal to the sum of prosumptions of its children C (either appliances or houses, respectively):

$$p_{Agg} = \sum_{child \in C} p_{child}$$

Each time a house appliance prosumption changes - lamp being switched on or off by the user, solar panel production, etc. - its prosumed power value is updated for its (house) aggregator, which in turn propagates its new aggregated prosumption to its own (district) aggregator. All prosumption data are made available at run-time, notably to the experimenter, to a persistence service, and to autonomic managers in the house, emulating a complete (and perfect) instrumentation of the grid.

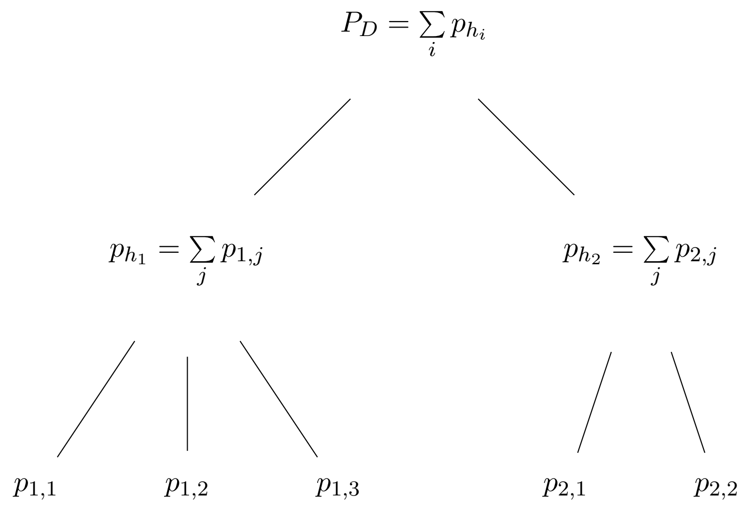


Figure 8.2: Sample district grid: district aggregator D and 2 houses h_1 and h_2 with 3 and 2 appliances respectively.

House aggregator and appliances normally run in the same house simulation instance, whereas the district aggregator is run separately. Hence a typical district simulation is composed of a number of house simulators, possibly run on several physical machines, and remotely connected to a separate district aggregator instance. Only aggregator prosumption data is available remotely (i.e. outside of a house), emulating probable data privacy restriction in future micro-grids.

8.1.3 Appliances

MisTiGriD proposes a several appliance models that are all customisable by configuration. The experimenter can deploy any number of appliance instance in the simulation, possibly at runtime, and control their behaviour as a real user would do. This section presents some of the available appliance models proposed that belong to an open, extensible library.

Lamp

A “lamp” models a consumer with a constant consumption and an on/off behaviour. A wide range of appliances falls in this category, not restricted to lightings: TV set, hoover, micro-wave, kettle, computer may indeed be modelled as “lamps”, each with their specific consumption level and typical period of use.

A lamp is normally controlled by a switch the user can use directly or hand over as a touch-point to an autonomic manager. The switch may propose several consumption levels, such as lightings proposing full-power or dimmed light levels.

Heater

A heater is a thermic object and a consumer which converts electrical power into heat. When switched on, the heater consumes a constant amount of power and raises its internal temperature to a high level (typically, $200W$ for $50°C$ in the simulation), heating the room around it according to the law of heat transfers described previously. When switched off, the heater's temperature progressively converges to its room's temperature. The cycle of heating/idle cycle is normally determined by the heater's autonomic manager.

Producers

In residential contexts, producers often depend on external energy sources such as sun light or wind. MisTiGriD provides generic prosumer components that can be easily customised to produce arbitrary production profiles, depending on time or manual input by the experimenter. For instance, a solar panel is a producer with a typical bell shape daily production curve, altered with significant noise due to high sensibility to local weather conditions. Extremely irregular producers such as windmills are not provided by the library, however, once a temporal model of those is known, integrating such prosumer to the existing collection is straightforward.

Storage

MisTiGriD provides a generic storage component, that can be customised with particular maximal capacity (in W.h) and input and output debit (in W). A battery provides a simple control touch-point, allowing the experimenter or an autonomic manager to put it into load, unload or standby mode, with respect to its current charge and total capacity.

8.2 Simulator Implementation

8.2.1 Main software technologies

MisTiGriD relies on Java component technologies for modularity: it is based on the OSGi [httd] dynamic service platform and iPOJO [httc] component framework. These technologies allow to design and develop loosely-coupled components (thermic objects, appliances, management tasks) and to insert them dynamically, at run-time, on the simulation platform. A dedicated Scala DSL (Domain Specific Language) allows experimenters to specify complete or partial house configurations, such as room layouts, appliance parameters and the relations between them. The simulator is easily extensible, since new appliances or house configurations can be easily integrated into an existing setup. An open API (Application Programming Interface) provides programmers with the necessary interfaces to develop their own simulated components and integrate them into the existing library.

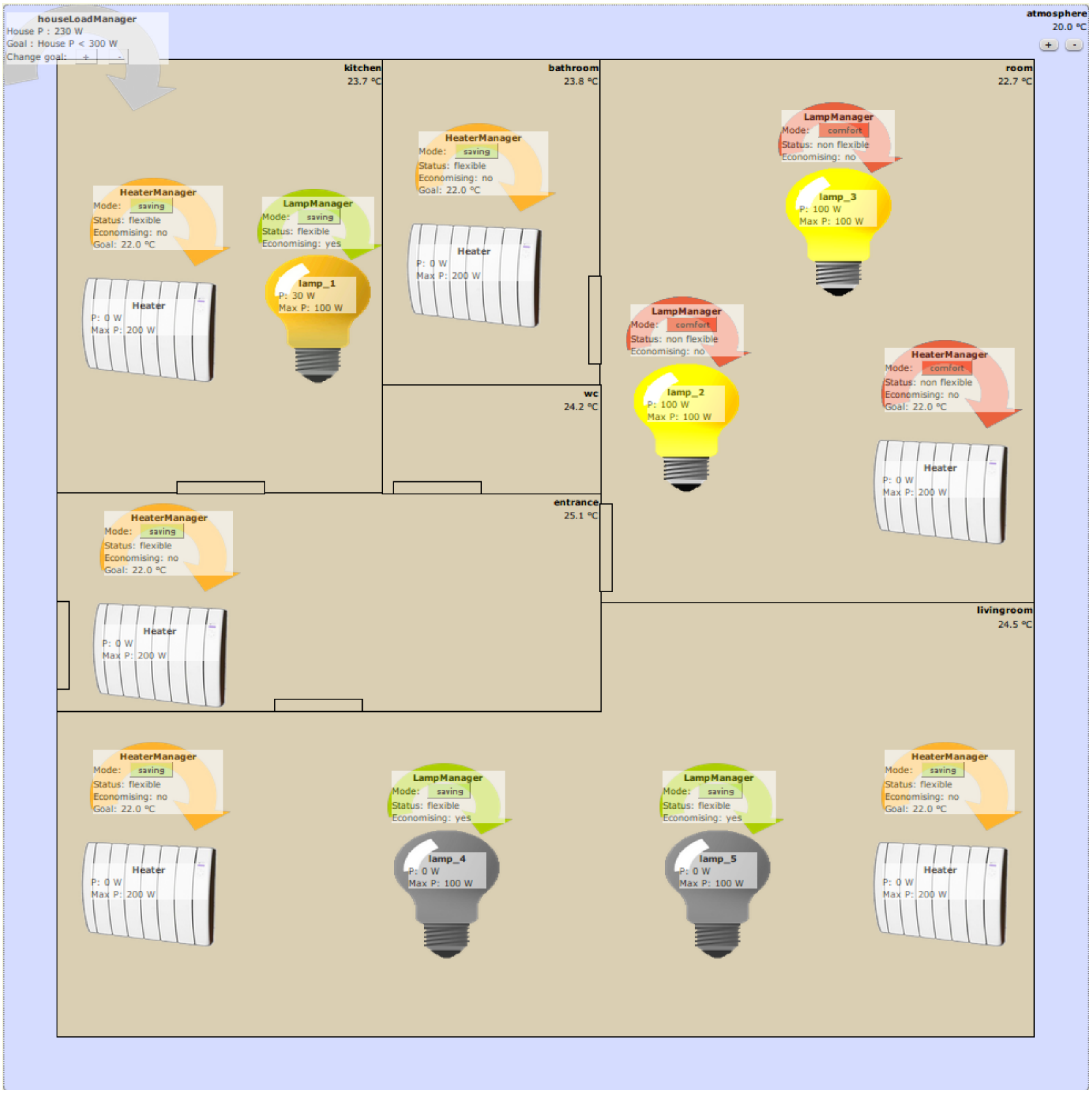
User interaction is performed via a web Graphic User Interface (see Fig. 8.3 for a sample screen capture), implemented in HTML / CSS / JavaScript and provided by the OSGi embedded web server. This allows experimenters to scale the simulation up to dozens of houses deployed on remote machines, observing and controlling running instances via a browser.

Management resources are dynamic components with significant amounts of (possibly remote) communications and synchronisation issues. The Akka framework [htta] implements the message-based Actor model [HBS73] which alleviates these difficulties and facilitates developing concurrent systems such as autonomic managers.

8.2.2 Smarduino House

In parallel with the simulator, we developed a miniature house model, made out of wood, polystyrene and glass. Small resistors provide rooms with heating capabilities, while colour LEDs allow to light different parts of the house and simulate usages. Temperature, light and electrical consumption sensors instrument the model along with a set of switches (manual and electronic, either on/off or potentiometer-like) controlling the LEDs and resistors.

The electric and electronic parts are implemented with the Arduino technology [httf]. Its simplicity of use and easy interfacing with the JVM allowed us to run instances of the MisTiGriD simulator that


houseLoadManager
House P : 230 W
Goal : House P < 300 W
Change goal: + -

atmosphere
20.0 °C

kitchen
23.7 °C

HeaterManager
Mode: saving
Status: flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

LampManager
Mode: saving
Status: flexible
Economising: yes

lamp_1
P: 30 W
Max P: 100 W

bathroom
23.8 °C

HeaterManager
Mode: saving
Status: flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

wc
24.2 °C

room
22.7 °C

LampManager
Mode: comfort
Status: non flexible
Economising: no

lamp_3
P: 100 W
Max P: 100 W

LampManager
Mode: comfort
Status: non flexible
Economising: no

lamp_2
P: 100 W
Max P: 100 W

HeaterManager
Mode: comfort
Status: non flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

entrance
25.1 °C

HeaterManager
Mode: saving
Status: flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

livingroom
24.5 °C

HeaterManager
Mode: saving
Status: flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

LampManager
Mode: saving
Status: flexible
Economising: yes

lamp_4
P: 0 W
Max P: 100 W

LampManager
Mode: saving
Status: flexible
Economising: yes

lamp_5
P: 0 W
Max P: 100 W

HeaterManager
Mode: saving
Status: flexible
Economising: no
Goal: 22.0 °C

Heater
P: 0 W
Max P: 200 W

Figure 8.3: Screen capture of MisTiGriD's main Graphic User Interface.

replace simulated components (heaters, lights, solar panel) with their physical counterparts measured on the model. For instance we could check that the hysteresis temperature controllers shown in chapter 6 would indeed effectively achieve temperature goals in "actual" rooms. The house simulation scenarios presented in the next sections can also be executed on the house model. However, experiments would be run on considerable larger time scales due to the slow evolution rate of the real world, compared to MisTiGriD's simulated time. We also produced a short video clip presenting the house and illustrating our approach to smart grids [hoch].

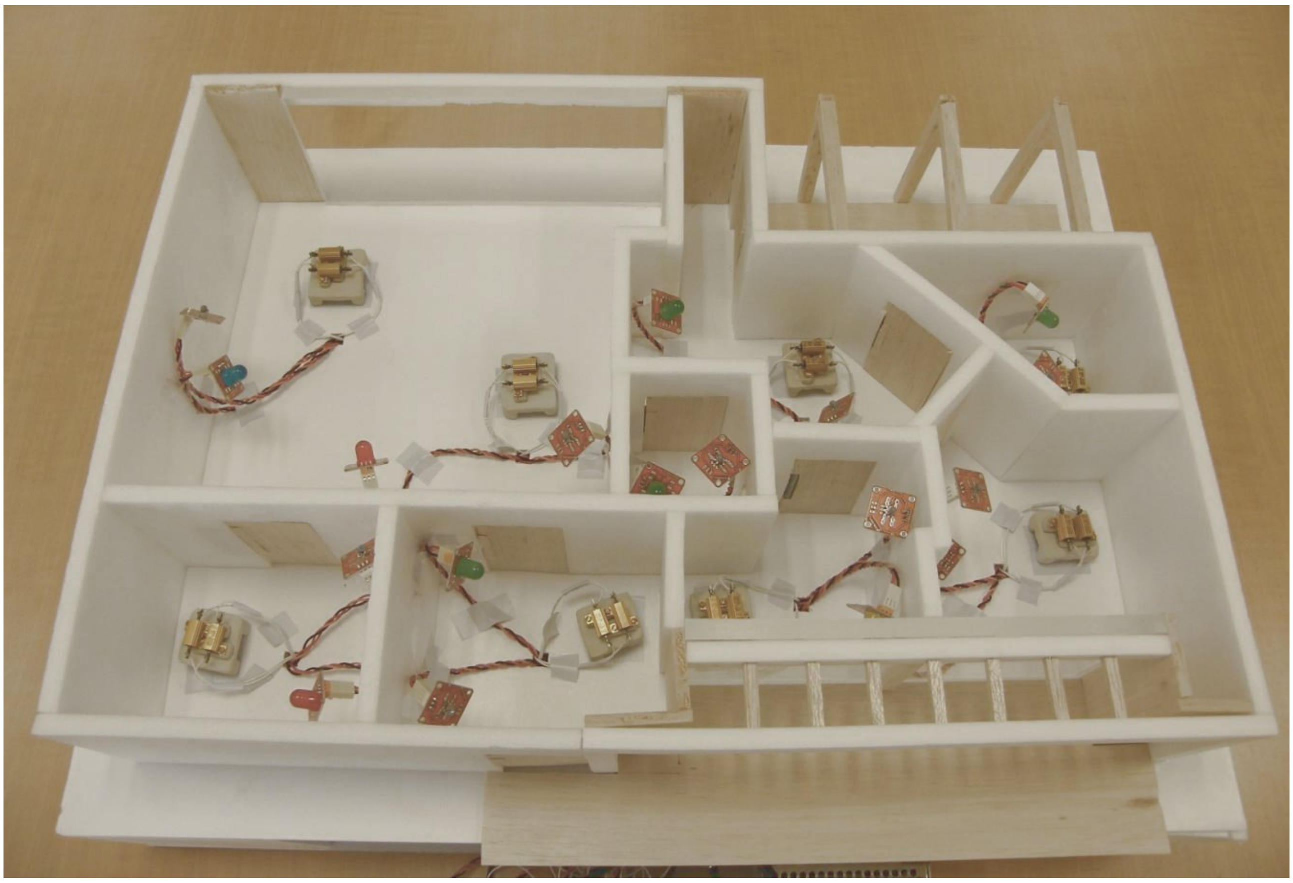


Figure 8.4: Global view of the Smarduino house.

8.3 Implementation of Managers and Power Management Organisations

For the sake of simplicity, the scenarios we present here use a reduced experimental setup, considering only consuming appliances and maximal consumption levels as power management objectives. Production and storage facilities are thus not utilised, which simplifies the expression of power objectives (a simple maximal consumption limit P_{house}^{max} instead of a prosumption domain) and the scheduling algorithm (only consumption packets and consumption reduction orders are to be considered). However restrictive, this simplification brings a limited loss of generality: due to the simple prosumption model we adopted, a production is no more than a negative consumption, and a target interval corresponds to a combination of two thresholds (minimum and maximum).

The scenarios use the two following types of appliances:

- ◇ heaters with 200W consumption in their heating phase.
- ◇ lamps with 100W full-power consumption, and a "dimmed" mode consuming 30W.

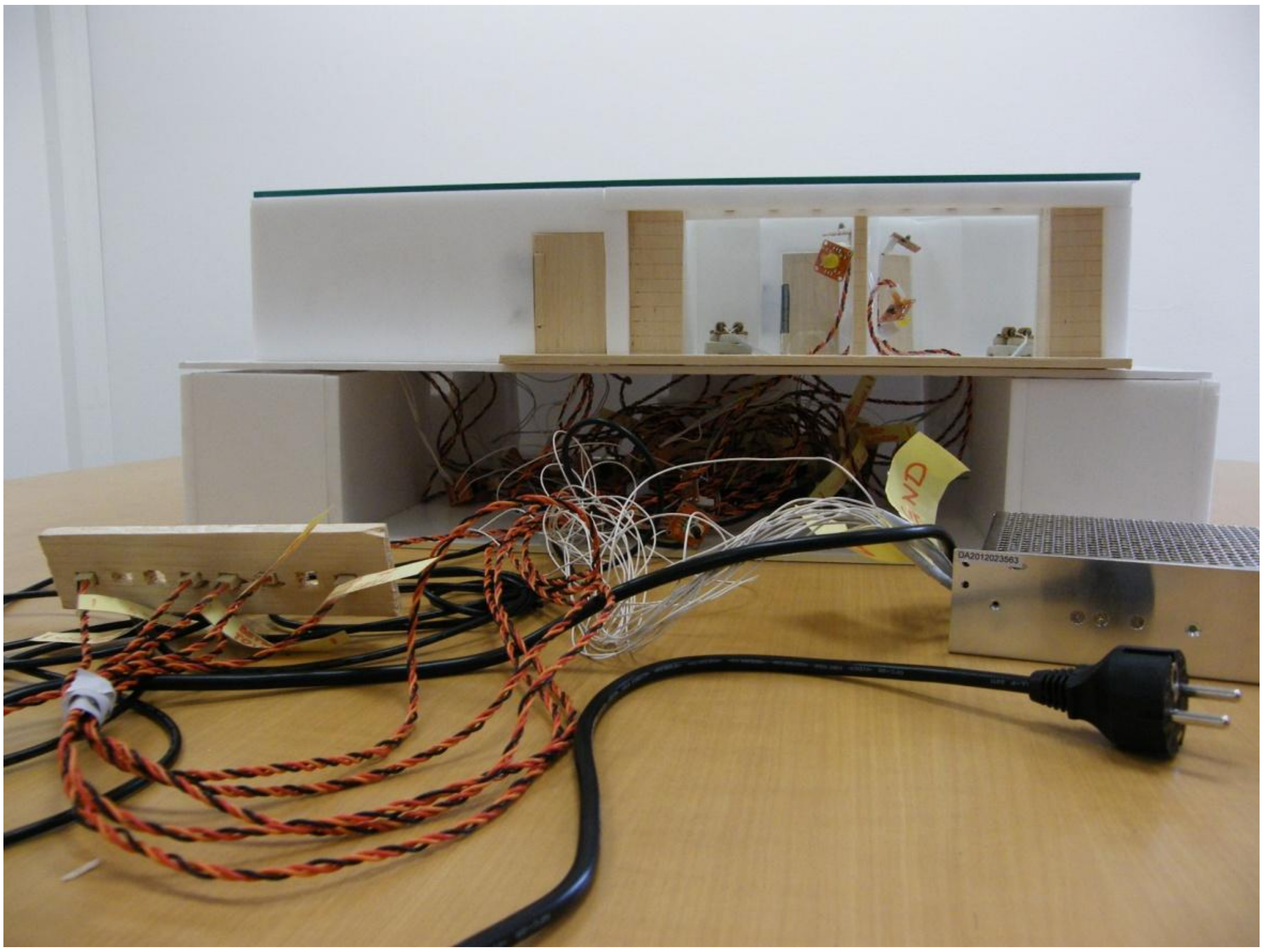


Figure 8.5: Focus on the underlying electronics of the Smarduino house.

The managers for theses appliances are the following:

- ◇ heater managers implement the behaviour presented as an example in chapter 6, according to a management mode (either "eco" or "comfort") dynamically set by the user/experimenter. In "comfort" mode, the manager maintains a $[22°C, 25°C]$ temperature interval and ignores any power management orders, appearing "Non Flexible" in the house organisation. In "eco" mode, the target temperature interval is shifted to $[20°C, 23°C]$. In addition, when on a heating cycle and not in a refractory phase, the manager appears "Flexible" and accepts to shut down its 200W consumption if necessary (cf. the description of the power management organisation below).
- ◇ lamp managers also provide "eco" or "comfort" management modes. When in "comfort" mode, the lamp consumes full power (100W) when switched up and appears "Non Flexible". When in "eco" mode, the lamp appears "Flexible" and reduces its consumption to the dimmed level (30W) if necessary (cf. the description of the power management organisation below).

At the house level, the power management organisation follows a stigmergic pattern: consuming appliances publish their consumption levels as consumption packets in a shared schedule. The house Power Manager publishes, for each time slot of the schedule, whether or not total house consumption crosses its maximum consumption threshold P^{max}_{house}. This global state of the organisation (either "high load" or "normal load" is in turn monitored by "Flexible" appliances, which reduce their consumption in case of high load, as described before.

At the district level, the power management organisation follows a hierarchical pattern: house power managers publish two consumption packets each, one for the total "Flexible" consumption of their appliances, and one for the total "Non Flexible" part. The district power manager compares the current district consumption with its own maximum threshold $P^{max}_{district}$, and in case the threshold is crossed, sends individual "Reduce Consumption" orders to house power managers, prioritising houses with high Flexible consumption levels. A house power manager receiving such an order reduces in turn its own maximum consumption objective to $P^{max,reduced}_{house} < P^{max}_{house}$. When district consumption comes back below the limit by a sufficient margin, the district power manager issues "Any Consumption" orders back to houses, returning them to their original P^{max}_{house} objective.

Determining proper numerical values for the thresholds P^{max}_{house}, $P^{max,reduced}_{house}$ and $P^{max}_{district}$ is essential for the functioning of the whole grid. These values can be adapted dynamically by the experimenter via the graphic interface. In practice, the following heuristics have been determined empirically:

- ◇ P^{max}_{house} should allow "normal" consumption levels, e.g. heaters maintaining comfortable temperature levels with limited lamp consumption.
- ◇ $P^{max,reduced}_{house}$ should correspond to minimal comfort conditions, e.g. all lamps shut down and all heaters in "eco" mode.
- ◇ $P^{max}_{district}$ can range from $\sum\limits_{house} P^{max,reduced}_{house}$ to $\sum\limits_{house} P^{max}_{house}$ and above, as the next scenario will show.

In practical applications, these values should be determined and updated via more formal heuristics, and possibly online learning algorithms. Economical or business concerns are also underlying motivations here, hence we do not elaborate further in this direction. The scenarios presented here aim at showing qualitative characteristics of the systems we consider, such as adaptability to varying conditions and objectives, or conflict resolution, to which a precise quantitative evaluation of the systems' behaviour is of little relevance.

The possibility to adapt the target threshold of a power management organisation is key to ensure its openness. Indeed, integrating new appliances in a house, or new houses in a district, only necessitates changing the viability domain of the power goal of the corresponding grid. In turn, the scheduling algorithm ensures that individual prosumer behaviours adapt to the new global objective. Although this aspect is not explicitly shown in the following scenarios, it is an important feature of our contribution that we aim to develop in future research.

8.4 Experimental Scenarios and Results

This section presents and explains several simulation scenarios we run on MisTiGriD [FDMD13a]. The scenarios feature multi-objective appliances (heaters and lamps) participating to house-level power management organisations. In turn, houses organisations are themselves part of a district-level power management organisation. The results we present show:

- ◇ how flexible the behaviour of multi-objective appliances is, adapting to variable environmental conditions and user preferences.
- ◇ how such appliances integrate with each other in organisations, and how the organisation integrates particular appliance objectives with a collective power objective.
- ◇ how the district-level organisation of organisations integrates different house behaviours with each other, and the effects of management decisions at all levels (appliance, house, district) on the other ones.
- ◇ how the balance is achieved between micro (house) and macro (district) goals.

8.4.1 House scenarios and results

These scenarios consider a single house, depicted on Fig. 8.3, featuring 6 rooms with 6 heaters and 5 lamps, as specified above. For such a house, the objective values where chosen to be $P_{house}^{max} = 800W$ and $P_{house}^{max,reduced} = 300W$. As a reminder, each heater consumes 200W when heating and each lamp consumes either 100W or 30W when switched on. At the beginning of the scenarios, outside temperature oscillates around $18.5°C \pm 1°C$. The scenarios divide into three consecutive phases.

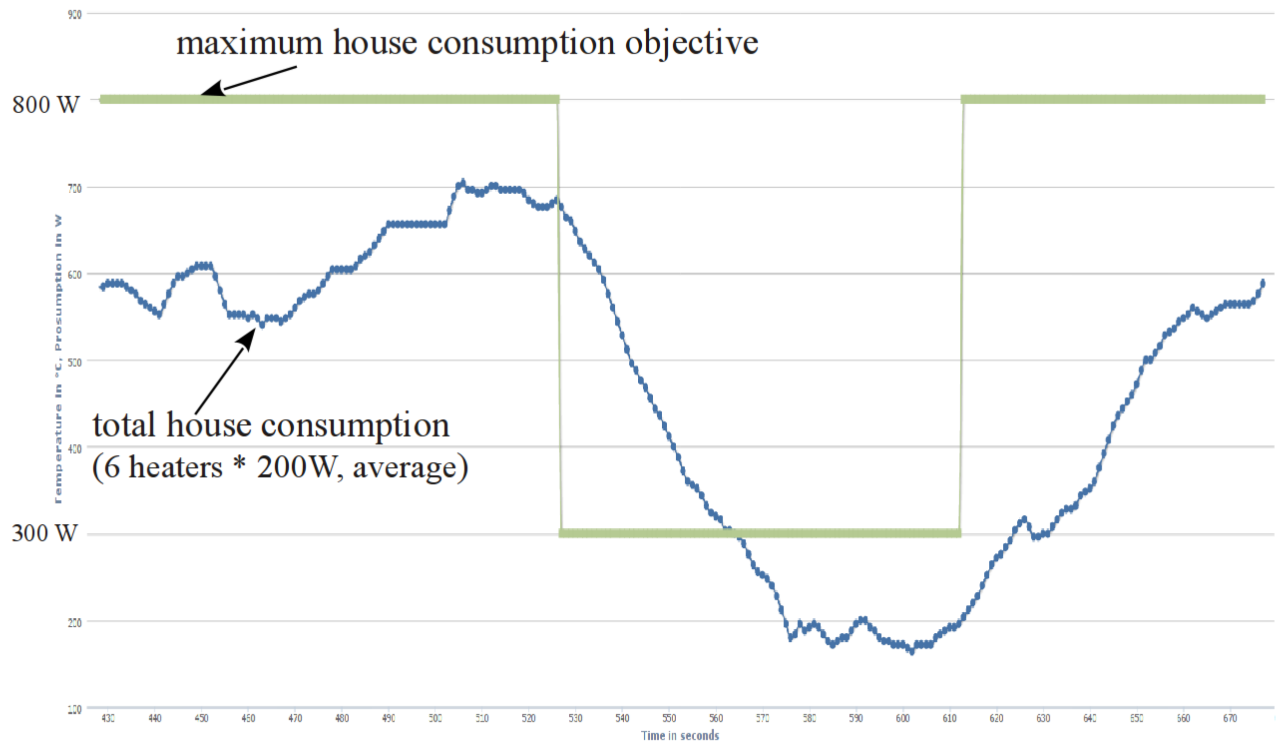


Figure 8.6: House scenario I : flexible appliances adapt to a variable objective.

Scenario I

During scenario I the house power objective is reduced from $P_{house}^{max} = 800W$ to $P_{house}^{max,reduced} = 300W$, due to the user setting the goal manually or to an order from the district power manager. Lamps are inactive for this scenario, and heaters are in "eco" mode. Fig. 8.6 shows the house power objective and average total consumption, i.e. the heaters' consumption. This value is averaged over a short window (50 samples) to facilitate reading the graph, since the aggregated on/off behaviours of the 6 heaters render a highly irregular raw curve.

The heaters dispose of a thin 300W allocation for the 6 of them. This shows a limit case of the scheduling algorithm, where only one heater should be allowed to be consuming at any given moment. The heaters indeed manage to do so - the curve on the figure evolves slowly due to the averaging window. The heaters return to standard behaviour when the house consumption goal is set back to $P_{house}^{max} = 800W$.

This phase of the scenario shows how to pull the organisation to one of its limits, where individuals sacrifice their particular objectives for the profit of the common one. This case shows a user management choice prioritising energy savings over comfort, and the resulting appropriate response from the system. Scenario III will show next the opposite extreme, where individual objectives prevent from achieving the global goal. This phase of the scenario also shows how the house can reduce its consumption in answer to district-level power management order, shown in next section.

Scenario II

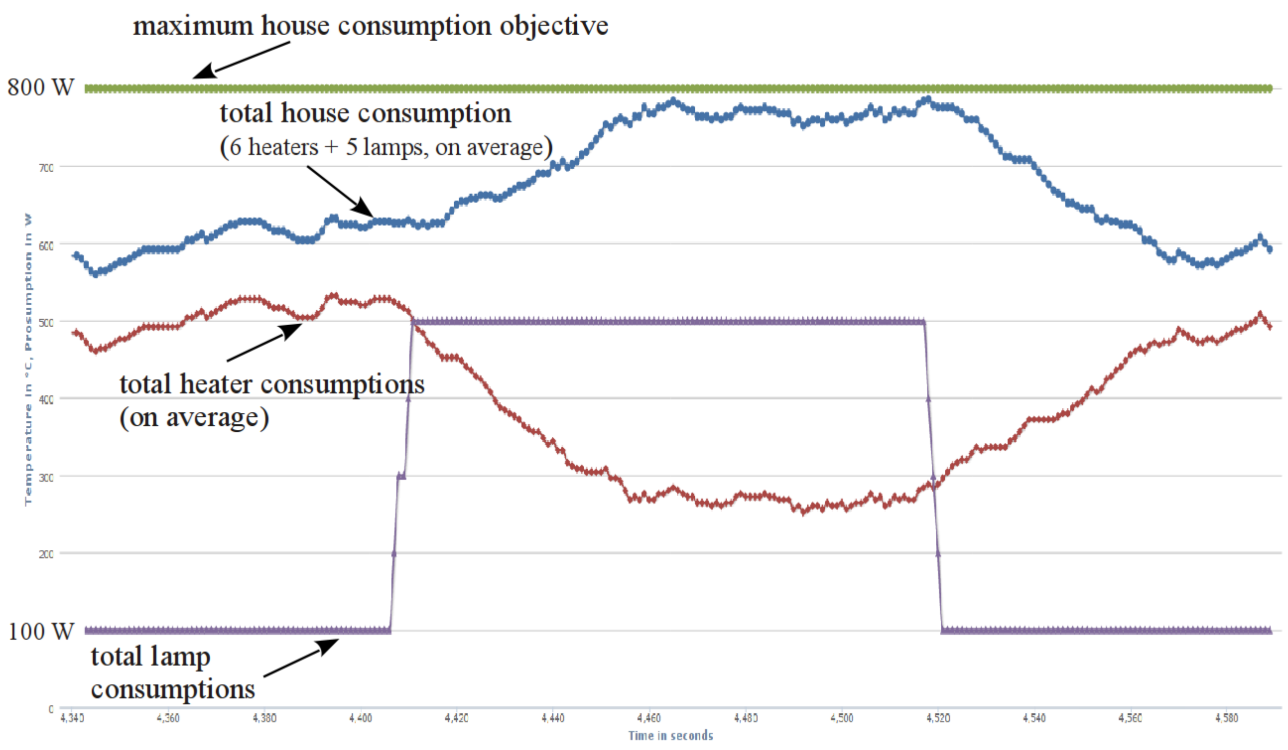


Figure 8.7: House scenario II : flexible appliances (heaters) compensate for inflexible appliances (lamps).

During scenario II the house consumption objective remains fixed to $P_{house}^{max} = 800W$. All heaters are in "eco" mode, lamps are in "comfort" mode and initially turned off except for one of them. Fig. 8.7 shows the house power objective, total lamp consumption, average total heater consumption and average total house consumption (i.e. heaters + lamps). The last two curves are averaged over 50 samples for the same readability reasons.

At the beginning, heaters do follow their cycle without much constraints, since the house total consumption evolves around $600W < P_{house}^{max} = 800W$. Then the user turns the 4 remaining lamps on successively, adding a total 400W non-flexible consumption to the house consumption. The 6 heaters therefore only dispose of a 300W "passing band" for their heating objective with lesser priority. They shift their heating cycles and possibly tolerate limited temperature drops, as described previously, for the duration of the inflexible lamp consumption, and the house power goal is respected. Heater behaviour returns to normal when lamps are switched off.

This scenario shows how “eco” appliances, here the heaters, can compensate for the “comfortable” ones - the lamps. Similar scenarios can be run, with for instance “eco” lamps reducing their consumption to compensate for “comfort” heaters enduring cold external conditions, or any mixed appliance mode setup. Here, sufficient flexibility is found in certain prosumers to adapt to the behaviours of inflexible ones, but this situation does not happen all the time, as shown in next scenario.

Scenario III

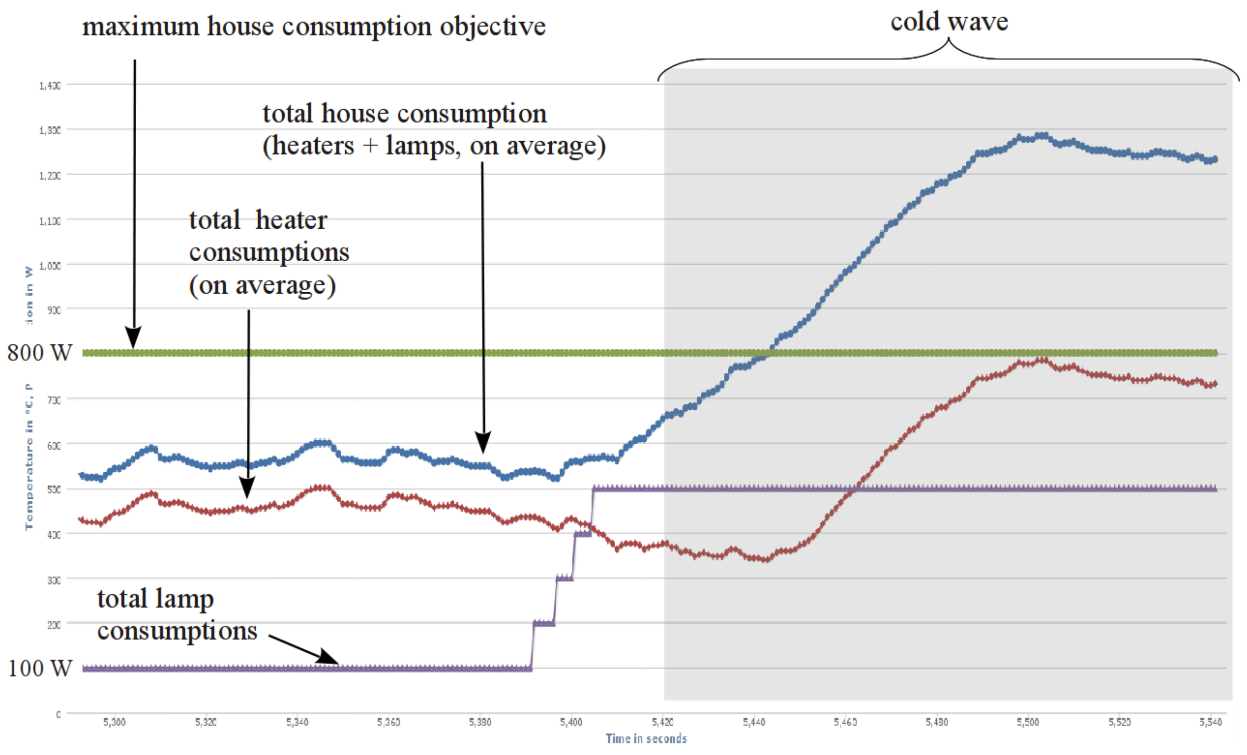


Figure 8.8: House scenario III : particular (appliance) objectives get prioritized over the collective (house) objective.

During scenario III the house power objective is constant: $P^{max}_{house} = 800W$. As in scenario I, the user switches on the 5 “comfort” lamps during the scenario, consuming 100W each. However, this time, the heaters are also in “comfort” mode, that is, they will not participate to the power management objective. This can correspond to a user requiring maximal comfort in case of a visit or illness for instance. In addition, a cold wave is simulated at the end of the phase, temperature dropping progressively to $10°C$, as shown on Fig. 8.8.

In such circumstances, i.e. the user clearly prioritising comfort over energy consumption, the scheduling algorithm implemented by the power management organisation rapidly runs short of solutions. As a result, the maximum consumption goal is largely overshoot, again, **according to user preference**.

Being able to sacrifice the global objective to the profit of individual ones is an important integration feature of the power management organisation, ensuring that the user can benefit from the whole range of possibilities offered by his appliances.

8.4.2 District scenarios and results

The district scenario features 8 houses similar to the one presented in the previous house scenario. Due to the grid's holonic architecture, power management organisation behave quite similarly at house and district levels, despite following different implementation patterns. Therefore, similar scenarios can be run at the district scale:

- ◇ all houses reducing their consumption to the minimum, limiting total district consumption as much as possible, similar to the house scenario I (cf. Fig. 8.9).
- ◇ houses with high flexibility reducing their consumption, compensating for high-consuming houses, and maintaining the district power under variable conditions, similar to the house scenario II.
- ◇ or conversely, all houses following comfort objectives, sacrificing their energy savings, and not respecting the district power goal, similar to the house scenario III (cf. Fig. 8.10).

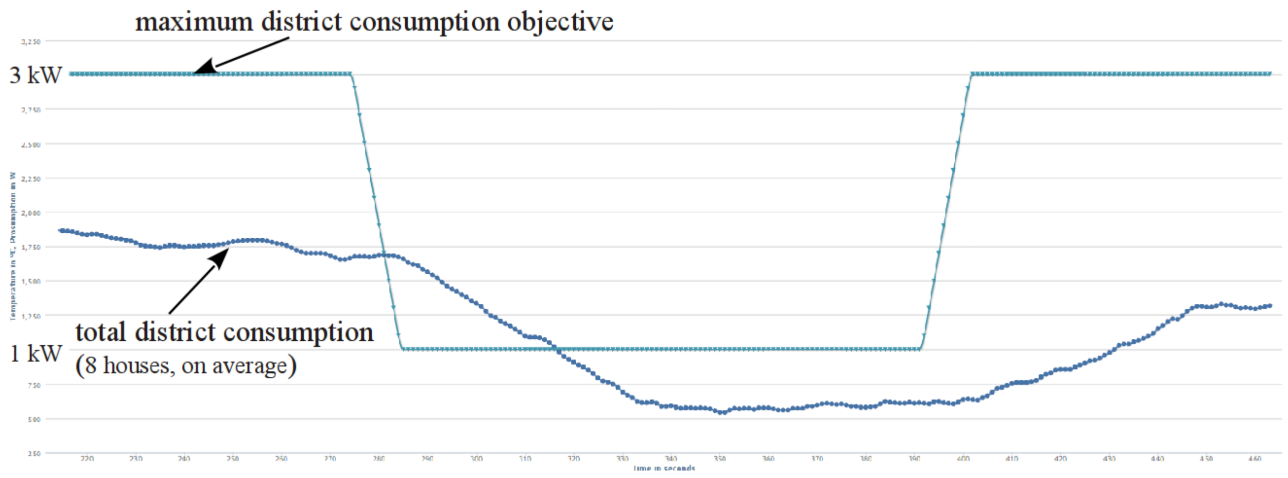


Figure 8.9: District scenario I : flexible houses adapt to a variable objective.

This section also presents another type of long scenario, where the district administrator evaluates the range of possible responses to different consumption thresholds. As shown on Fig. 8.11, the administrator starts with an excessively high maximum consumption allowed for the district, putting therefore no initial constraints on houses, which consume around 5kW total. Then, the threshold is reduced step by step, progressively restraining the allowed consumption band. The district power

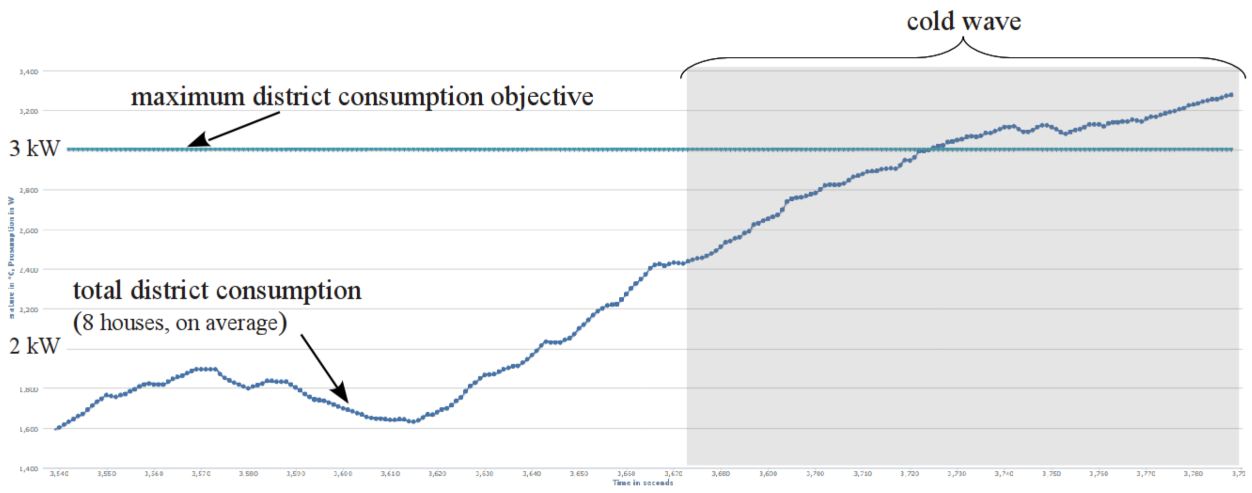


Figure 8.10: District scenario III : particular (house) objectives get prioritized over the collective (district) objective.

managers schedules house consumption reductions more and more frequently, to the point where all houses are asked to limit their consumption to the maximum and no further savings are possible, around 800s on the figure. District consumption has been eventually reduced to approximately 1.5kW.

As Fig. 8.11 shows, the houses' answer to variable district power goals is relatively smooth and close to the allowed maximum, notably between 200s and 800s where the district grid is neither unconstrained nor bound to an unreachable goal. This indicates that:

- ◇ Appliance flexibility can be exploited form the district point of view, even indirectly, by negotiating power management goals with black-box house managers.
- ◇ There are certainly characteristic boundaries of the district's consumption, and flexible intermediates that can be reached with relative precision between these extrema. As we already discussed, determining these characteristics is a crucial matter, to be achieve via formal heuristics and learning.
- ◇ Multi-level integration can be achieved between organisations of individual autonomic appliances, with minimal assumptions on their particular objectives, and while letting room for their own specific behaviours. This was precisely the overall aim of the integration approach proposed in this thesis.

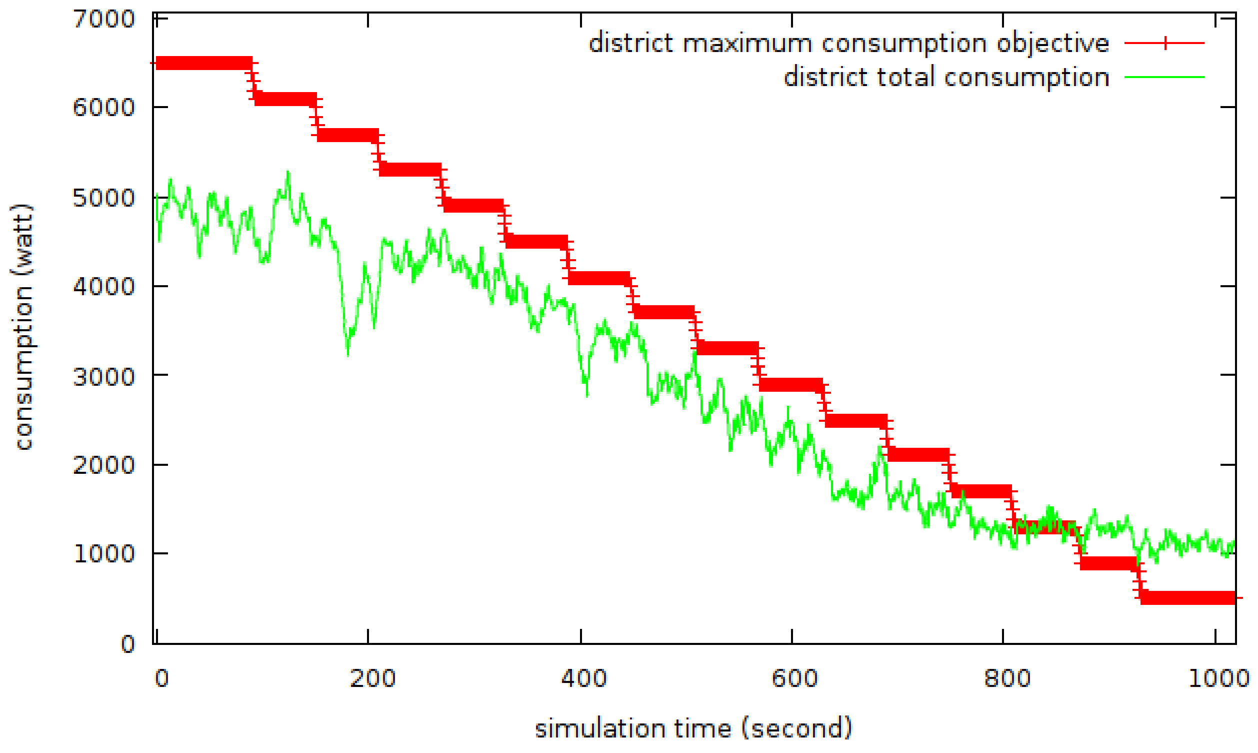


Figure 8.11: District scenario IV : flexible response from houses to a variable district objective.

Chapter conclusion

In this chapter, we presented MisTiGriD, a micro-grid platform developed during this thesis. We detailed MisTiGriD's simulation capabilities, including simple grid and temperature models, and some of the most important elements in its library of simulated house appliances.

We also presented a sample implementation of the multi-objective heater managers, lamp managers, and house and district power management organisations designed in chapters 6 and 7 respectively. This implementation is intended as a proof-of-concept of our approach, validating the multi-objective conflict resolution semantics and patterns we introduced, both at the individual and collective levels.

Finally, we presented and discussed a series of experimental results, obtained from several management scenarios run on the simulator. The scenarios are not necessarily realistic with respect to real-world smart grids. Nonetheless, they showcase the flexibility, manageability and openness of our integration solutions under various environmental conditions and user preferences. These results are promising in that they validate our integration approach and open perspectives on numerous complex management infrastructures for future smart grids, and complex autonomic systems in general.

Chapter 9

Conclusions & Future Work

Cada uno es hijo de sus obras.[1]
Miguel de Cervantes - *Don Quijote de la Mancha*

9.1 Summary of Thesis Contributions

In this thesis, our objective was to provide generic support for facilitating the design of *complex autonomic systems*, that is, systems that are dynamic, large scale, heterogeneous, open and multi-objective (cf. chapter 2). In order to address this complexity, we first provided a generic formalisation of management objectives, management conflicts and possible conflict resolution semantics that are necessary for a multi-objective autonomic system to function. This formalisation also helps in designing the distribution of control resources over a complex system (cf. chapter 5). Second, we provided a generic architecture for building adaptable multi-objective autonomic managers that can cope with the complexity factors we identified. Such managers rely on modularisation of their control logic into loosely coupled control components. These can compose using the integration patterns we identified and compared (chapter 6). Third, we provided an architecture for open organisations able to integrate heterogeneous, third-party autonomic systems. These organisations rely on coarser integration patterns for stand-alone autonomic systems (chapter 7).

Our contributions - conceptual formalisations, architectures and design patterns - are generic and aim to help the design of a wide range of complex autonomic systems. In this thesis, as a motivating use

[1] *Every one is the son of his own works.*

case and for illustration purposes, we relied on the application domain of smart micro-grids. Each of our generic contributions was exemplified via different parts of this broad application domain: our goal formalisation in the micro-grid lead us to a holonic control architecture, also identified in the literature (chapter 5); and, our multi-objective smart heaters and power management organisations proved compliant with state-of-the-art load control techniques (chapters 6 and 7). To illustrate the behaviour of our solutions, we developed the MisTiGriD platform comprising a micro-grid simulator and proof-of-concept controller implementations (chapter 8). Experimental scenarios run on our simulator produced promising results, indicating the viability of our solutions. Even if our contributions are generic we believe that these are relevant to the smart grid domain.

In the remaining of this chapter, we review some limitations of our contribution and identify future research directions.

9.2 Limitations and Future Work

9.2.1 Towards a Generic Framework for Autonomic Systems

Providing a full-fledged software framework for developing autonomic managers is out of the scope of this thesis. However, given the integration approach we presented in this thesis and our experience in implementing the proof-of-concept platform, we can isolate the following requirements for developing such framework:

◇ Modularity-enforcing technologies are indispensable to achieve proper decomposition of autonomic managers into independent control components. In this regard, we utilised component-based frameworks using service-oriented approaches, such as OSGi [httd] and iPOJO [httc], also used in the Ceylon framework [Mau10]. However, unlike Ceylon, we did not formalise our approach as a well-defined component middleware.

◇ The intrinsic dynamism of autonomic systems and their environments, and the complex interaction patterns between autonomic systems requires dedicated support for remote communication and concurrency handling. We found the messaged-oriented Actor Model [HBS73] particularly useful for addressing this concern, and utilised the Akka middleware [htta] for implementation. However, in our case, a reusable integration between Akka and OSGi/iPOJO remains to be provided.

- ◇ An important requirement of autonomic systems is their ability for reflection and runtime evolution. Administrators should be able to assess their system's state and modify it at run-time, and possibly automatise this task via a meta-manager administrating the system's managers. This aspect of autonomic systems is further developed in the following section on "meta-management".
- ◇ A complementary requirement of the former requirement is the need for traceability and persistence of the autonomic system state, so as to facilitate its long-term evaluation and evolution. Implementation of such persistence for the management layers would be tightly related to its meta-modelisation.

To our knowledge, unified technologies answering all of the above requirements remain to be developed. For instance, complex Java EE stacks may answer the modularisation and persistence requirements, yet, since these technologies target primarily application servers, they provide little consideration for the concerns of openness or resource constraints, which characterise autonomic systems.

9.2.2 Support for Time-related Concerns in Autonomic Managers and Organisations

Time-related issues are a major concern in dynamic systems. We have identified several control aspects that would benefit from being captured into a generic framework for autonomic systems:

- ◇ data expiration: monitoring information represents a snapshot of the state of a system under observation, and this snapshot may be reliable for a short period of time only. Consequently, refreshing monitoring information is a critical functionality, be it active (the autonomic manager fetches new data when existing ones expire) or passive (the observed system notifies relevant changes in its state).
- ◇ control delays: an autonomic manager's decision process may take time that is comparable to or longer than the evolution of the systems it controls. Therefore, when the computation of control decisions is longer than the changes of the system under control, these decisions may be outdated by the time they are applied. The autonomic manager may react to a previous, outdated state, which may lead to oscillations or other undesirable states and behaviours.
- ◇ fast unpredictable variations: complementary to the two previous concerns, it may happen that unexpected changes of the system under observation and control require fast responses that

interrupt current computations being carried out by the autonomic manager. Conversely, a manager that receives monitoring data may have to choose between starting an analysis-planning phase or waiting for extra information to arrive.

◇ goal time scope: in relation to the previous concerns, the variability of goals is an important aspect to consider from the manager's point of view. Although in chapter 5 we formalised goal scopes in terms of time and space, the time dimension was never utilised in the management scenarios.

◇ multi-loop coordination: when considering several flows of control in multi-loop organisations, timing correlations are necessary to synchronise autonomic managers. Such correlations would depend, in particular, on the respective reaction times of each member of the organisation.

◇ long-term persistence and predictive analysis: learning a system's behaviour over the long term and predicting its future changes requires expert analysis and knowledge. Support for persistence in the underlying middleware would be essential for addressing this concern.

9.2.3 Meta-Management

An important feature of autonomic systems is that of run-time introspection and evaluation of both the managed system and of its controller. This implies the introduction of an additional management layer - i.e., meta-management - for controlling the underlying autonomic manager. Meta-management is essential for adapting the autonomic management logic to changing requirements and environments, as we described in chapter 6.

Meta-management can be performed manually by the administrator who observes the system's state and controls it via an instrumentation layer. This is the process we followed in chapter 6, yet without a clear formalisation of what a formal manager meta-model would be. [NRS10] [MD03] for instance propose definitions for such a meta-model.

However, as defined by IBM's taxonomy of autonomic systems (cf. chapter 2) instrumentation is only the first step towards full autonomicity. We believe that an interesting long-term goal would be to provide support for full-fledged meta-management, featuring complete meta-managers performing long-term adaptation of the management logic. Indeed, this is an architectural pattern that has been often identified in the literature, for instance [KM07] [MLD11] [MSS11].

An example application of meta-management would be adapting control parameters to particular environments or usages for optimised management performance, which would allow generic controllers to be finely tuned to a variety of contexts. For instance, in the smart heater example we presented in chapter 6, a meta-manager would change the threshold temperatures of the heater manager so as to adapt it to the particular physical characteristics of the heater and the comfort sensibility of the house users. A meta-manager could also deploy additional Monitoring automatically at run-time when new thermometers are detected in the environment, as exemplified in section 6.3.2.

9.2.4 Autonomic Societies

In chapter 7 we proposed an abstract architecture for open organisations of autonomic systems. Integration of third-party autonomic systems is a wide question, in this section we identify several aspects of it that our contribution does not address so far.

First, our architecture allows each participant to adapt its behaviour towards the rest of the organisation. However, the rules of the organisation may also be adapted, which would provide even greater flexibility. Works such as [PSBM12] [BKMS+12] [HMG+11] [Rod05] explore autonomic societies and possible meta-rules that would formalise the long-term adaptation of autonomic organisations. As before, meta-organisation could be performed manually by administrators, or it could be automatised by a dedicated meta-management system. In case the organisation supports "democratic" adaptation processes, the organisation participants could determine and adapt the rules their organisation follows by themselves, at any time.

In section 7.2.4 we proposed an interpretation of trade-offs between objectives in organisations. Formalising these trade-offs via a game theory approach would certainly help to evaluate and possibly predict the collective behaviour of an organisation. Such formalisation would also help organisation meta-management, as described in the last paragraph, for instance with formal metrics of participants and organisation "satisfaction".

9.2.5 Ergonomics of Autonomic Systems

In this thesis, we did not consider the modalities of user and administrator interactions with the autonomic systems. Certainly, non-expert users of domestic appliances would not have the same interaction requirements as expert administrators trying to optimise an industrial system.

An important aspect when dealing with domestic appliances is intrusiveness: users do not like being disrupted in their habits by an appliance, be it “smart” or not, and pay little attention to power management concerns in general. This has been confirmed by recent studies that showed that even “power-aware” users would give up changing their habits for the sake of power management, past a certain period of adaptation [KKP12]. As a consequence, domestic autonomic systems will require perfect transparency and non-intrusiveness in order to be accepted and become successful in households. User commands should be kept extremely simple, such as the “eco” vs. “comfort” objectives we used in our implementations. Derogation on individual appliances (for instance, requiring more comfort on a single heater while the rest of the house remains economical) should not differ from standard usage.

9.2.6 Scheduling Algorithms for Power Management

An important technical challenge of the power management organisations we presented in chapters 7 and 8 is that of designing the scheduling algorithms for organisations. In our implementation, we used rather naive, greedy solutions that would certainly benefit from further investigations. For instance, our implementations only react to the current state of the system and do not anticipate nor pre-emptively re-schedule planned prosumptions. The scenarios we presented feature neither production nor storage capabilities, which would be an important extension. The study of monetary incentives for power scheduling is a hot topic in the smart grid domain (for instance, cf. [Che12] [MRWJ+10] [CH11b] [MC12]) that may represent a possible approach for implementing a realistic scheduling process at the district level.

9.2.7 MisTiGriD 2.0

The MisTiGriD simulator has been designed with extensibility and adaptability in mind. The physical characteristics of the appliances and houses featured in this thesis’ scenarios could be easily modified for increased realism. Developing new house configurations and appliance profiles is made easy by the simulator’s internal APIs. In particular, even if the presented scenarios do not feature production or storage facilities, such appliances do exist in MisTiGriD’s library of prosumers.

The MisTiGriD platform also represents a valuable opportunity to illustrate and teach the principles of autonomic computing. The platform was used in several student computing projects and demonstrated its accessibility and re-usability. With further development, MisTiGriD could provide a base software support for a course on autonomic computing.

9.2.8 Autonomic Management at Every Grid Level

In this thesis, we limited the use-case to a residential district grid. However, the holonic power management architecture we proposed could be applied at coarser granularities, up to entire continental grids. The nature of the considered (virtual) prosumers, the characteristics of the networks, such as its physical and temporal scales, and the administrative entities would be very different from the ones we considered for residential micro-grids. In particular, industrial and national interests would be at stake. The grid behaviour would also be much more complex to model than the simple continuous approximation we considered in this thesis. Yet, we believe that control organisations would still follow the holonic patterns we presented, adapted to the particular granularities under consideration.

9.2.9 Holonic Patterns

In chapter 5 we presented an holonic control architecture for the smart grid. In this case, the holonic nature of the grid justifies such an architecture, also identified in the literature [NB12]. However, holonic systems are present in numerous contributions in the multi-agent domain, which [Rod05] identifies. Indeed, we believe that holons may be characteristic of a wide range of complex systems, not limited to smart grids. The identification of generic holonic patterns will be the topic of future developments of this work.

9.2.10 Application to Other Domains

In this thesis, we focused on the domain of smart grids to illustrate and validate our contribution. Although we identified our architectures and patterns in different domains, applying our design methodology from scratch on very different use-cases would better support the validity of our claims. Contiguous projects such as the Ceylon framework [Mau10] and Cube for data mediation [DDL12] share common inspiration with this work.

Bibliography

[ABL$^+$03] M. Agarwal, V. Bhat, H. Liu, V. Matossian, V. Putty, C. Schmidt, G. Zhang, L. Zhen, M. Parashar, B. Khargharia, and S. Hariri. Automate: Enabling autonomic applications on the grid. In *IN: AUTONOMIC COMPUTING WORKSHOP, THE FIFTH ANNUAL INTERNATIONAL WORKSHOP ON ACTIVE MIDDLEWARE SERVICES (AMS 2003*, pages 48–57. IEEE Computer Society Press, 2003.

[ABS11] Florian Allerding, Birger Becker, and Hartmut Schmeck. Decentralised energy management for smart homes. In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*, pages 605–607. Springer Basel, 2011.

[Ant04] Richard John Anthony. Emergence: A paradigm for robust and scalable distributed applications. In *International Conference on Autonomic Computing (ICAC)*, pages 132–139. IEEE Computer Society, 2004.

[aut] `http://www2.engr.arizona.edu/~hpdc/projects/autonomia_programmable/`.

[BAR$^+$10] B. Becker, F. Allerding, U. Reiner, M. Kahl, U. Richter, D. Pathmaperuma, H. Schmeck, and T. Leibfried. Decentralized energy-management to control smart-home architectures. In *ARCS*, volume 5974 of *Lecture Notes in Computer Science*, pages 150–161, 2010.

[BBH12] J. Beal, J. Berliner, and K. Hunter. Fast precise distributed control for energy demand management. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* [DBL12].

[BCD$^+$06] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni A. Di Caro, Frederick Ducatelle, Luca M. Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, Alberto Montresor, and Tore Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, September 2006.

[BDLM11] Johann Bourcier, Ada Diaconescu, Philippe Lalanda, and Julie A. McCann. Autohome: An autonomic management framework for pervasive home applications. *TAAS*, 6(1):8, 2011.

[BKMS^{+}12] Yvonne Bernard, Lukas Klejnowski, Christian Müller-Schloer, Jeremy Pitt, and Julia Schaumeier. Enduring institutions and self-organising trust-adaptive systems for an open grid computing infrastructure. In *SASO Workshops*, pages 163–168. IEEE Computer Society, 2012.

[BLE10] Jonathan Bardin, Philippe Lalanda, and Clement Escoffier. Towards an automatic integration of heterogeneous services and devices. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 171–178. IEEE, 2010.

[BM08] M. Breza and J.A. McCann. Lessons in implementing bio-inspired algorithms on wireless sensor networks. In *Adaptive Hardware and Systems, 2008. AHS '08. NASA/ESA Conference on*, pages 271–276, 2008.

[CH11a] Duncan Callaway and Ian A. Hiskens. Achieving controllability of electric loads. *Proceedings of the IEEE*, 99(1):184–199, 2011.

[CH11b] Duncan Callaway and Ian A. Hiskens. Achieving controllability of electric loads. *Proceedings of the IEEE*, 99(1):184–199, 2011.

[Che08] Shang-Wen Cheng. *Rainbow: cost-effective software architecture-based self-adaptation.* PhD thesis, Pittsburgh, PA, USA, 2008. AAI3305807.

[Che12] Y. Cheng. Architecture and principles of smart grids for distributed power generation and demand side management. In *SMARTGREENS, 1st International Conference on Smart Grids and Green IT Systems, Porto, Portugal*, 2012.

[Cor91] Daniel Corkill. Blackboard Systems. *AI Expert*, 6(9), January 1991.

[CSWW04] D.M. Chess, A. Segal, I. Whalley, and S.R. White. Unity: experiences with a prototype autonomic computing system. In *Autonomic Computing, 2004. Proceedings. International Conference on*, pages 140 – 147, may 2004.

[DBL12] *Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2012, Lyon, France, September 10-14, 2012*. IEEE Computer Society, 2012.

[DDL12] Bassem Debbabi, Ada Diaconescu, and Philippe Lalanda. Controlling self-organising software applications with archetypes. In *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, pages 69–78, 2012.

[DDLS01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *LECTURE NOTES IN COMPUTER SCIENCE*, pages 18–38. Springer-Verlag, 2001.

[DHX$^+$03] Xiangdong Dong, S. Hariri, Lizhi Xue, Huoping Chen, Ming Zhang, S. Pavuluri, and S. Rao. Autonomia: an autonomic computing environment. In *Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International*, pages 61 – 68, april 2003.

[DSNH10] Simon Dobson, Roy Sterritt, Paddy Nixon, and Mike Hinchey. Fulfilling the vision of autonomic computing. *Computer*, 43:35–41, 2010.

[DWH07] Tom De Wolf and Tom Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. In *Proceedings of the 4th international conference on Engineering self-organising systems*, ESOA'06, pages 28–49, Berlin, Heidelberg, 2007. Springer-Verlag.

[FDD12] S. Frey, A. Diaconescu, and I. Demeure. Architectural integration patterns for autonomic management systems. *9th IEEE International Conference and Workshops on the Engineering of Autonomic and Autonomous Systems (EASe)*, 2012.

[FDMD13a] S. Frey, A. Diaconescu, D. Menga, and I. Demeure. Exemplifying conflict resolution in multi-objective smart micro-grids. *Demo Session of Seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2013.

[FDMD13b] S. Frey, A. Diaconescu, D. Menga, and I. Demeure. A holonic control architecture for a heterogeneous multi-objective micro smart grid. *Seventh IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2013.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, pages 21–35, London, UK, 1997. Springer-Verlag.

[FHM$^+$12] S. Frey, F. Huguet, C. Mivielle, D. Menga, A. Diaconescu, and I. Demeure. Scenarios for an autonomic micro smart grid. *1st International Conference on Smart Grids and Green IT Systems (SmartGreens 2012)*, 2012.

[FMAS$^+$11] Jose Luis Fernandez-Marquez, Josep Lluis Arcos, Giovanna Di Marzo Serugendo, Mirko Viroli, and Sara Montagna. Description and composition of bio-inspired design patterns: the gradient case. In *Proceedings of the 3rd workshop on Biologically inspired algorithms for distributed systems*, BADS '11, pages 25–32, New York, NY, USA, 2011. ACM.

[Gas01] Les Gasser. Mutli-agents systems and applications. chapter Perspectives on organizations in multi-agent systems, pages 1–16. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[GCH$^+$04] D. Garlan, Shang-Wen Cheng, An-Cheng Huang, B. Schmerl, and P. Steenkiste. Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence, 1973.

[HL04] Bryan Horling and Victor Lesser. A survey of multi-agent organizational paradigms. *Knowl. Eng. Rev.*, 19:281–316, December 2004.

[HMG$^+$11] M. Huhn, J.P. Muller, J. Gormer, G. Homoceanu, Nguyen-Thinh Le, L. Martin, C. Mumme, C. Schulz, N. Pinkwart, and C. Muller-Schloer. Autonomous agents in organized localities regulated by institutions. In *Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on*, pages 54–61, 2011.

[hoch] http://www.youtube.com/watch?v=tI6z9LGaFTY or cf. http://perso.enst.fr/sfrey.

[htta] http://akka.io/.

[httb] http://en.wikipedia.org/wiki/Newton's_law_of_cooling.

[httc] http://ipojo.org/.

[httd] http://osgi.org/.

[htte] http://wikiadele.imag.fr/index.php/Cilia.

[httf] http://www.arduino.cc/.

[httg] http://www.fipa.org/.

[htth] http://www.nicegrid.fr/.

[IBM01] IBM. Autonomic computing manifesto, 2001.

[IBM06] IBM. An Architectural Blueprint for Autonomic Computing. 2006.

[JB03] Nicholas R. Jennings and Stefan Bussmann. Agent-based control systems. *IEEE Control Systems Magazine*, 23:61–74, 2003.

[JMB05] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, August 2005.

[KC03] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36:41–50, January 2003.

[KCD+07] Jeffrey O. Kephart, Hoi Chan, Rajarshi Das, David W. Levine, Gerald Tesauro, Freeman Rawson, and Charles Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In *Proceedings of the Fourth International Conference on Autonomic Computing*, pages 24–, Washington, DC, USA, 2007. IEEE Computer Society.

[Kim11] W.; Kinoshita T. Kim, H.-M.; Wei. A new modified cnp for autonomous microgrid operation based on multiagent system. *Journal of Electrical Engineering and Techonology; 6, 1; 139-146*, pages 139–146, 2011.

[KKP12] Andreas Kamilaris, Giannis Kitromilides, and Andreas Pitsillides. Energy conservation through social competitions in blocks of flats. In *SMARTGREENS*, pages 167–174, 2012.

[KLM+97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *ECOOP*, pages 220–242, 1997.

[KM07] Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. In *2007 Future of Software Engineering*, FOSE '07, pages 259–268, Washington, DC, USA, 2007. IEEE Computer Society.

[Koe48] Arthur Koestler. The ghost in the machine. 1948.

[Kra97] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94:79–97, 1997.

[KST09] Serge Kernbach, Thomas Schmickl, and Jon Timmis. Collective Adaptive Systems: Challenges Beyond Evolvability, 2009.

[KW04] Jeffrey O. Kephart and William E. Walsh. An artificial intelligence perspective on autonomic computing policies. In *POLICY*, pages 3–12. IEEE Computer Society, 2004.

[KWZ11] M. Köksalan, J. Wallenius, and S. Zionts. *Multiple Criteria Decision Making: From Early History to the 21st Century.* World Scientific Publishing Company, Incorporated, 2011.

[Lan11] Christopher Landauer. Problem posing as a system engineering paradigm. In Henry Selvaraj and Dawid Zydek, editors, *ICSEng*, pages 346–351. IEEE, 2011.

[Lev96] William S. Levine. *The control handbook.* The electrical engineering handbook series. CRC Press New York, Boca Raton (Fl.), 1996.

[Mau10] Y. Maurel. *CEYLAN: un canevas pour la création de gestionnaires autonomiques extensibles et dynamiques.* PhD thesis, Université Joseph Fourier, Grenoble, 2010.

[MC12] Johanna L. Mathieu and Duncan Callaway. State estimation and control of heterogeneous thermostatically controlled loads for load following. In *HICSS*, pages 2002–2011, 2012.

[MD03] Jacques Malenfant and Simon Denier. Arm : un modèle réflexif asynchrone pour les objets répartis et réactifs. *L'OBJET*, 9(1-2):91–103, 2003.

[MDC$^+$07] S.D.J. McArthur, E.M. Davidson, V.M. Catterson, A.L. Dimeas, N.D. Hatziargyriou, F. Ponci, and T. Funabashi. Multi-agent systems for power engineering applications 2014;part i: Concepts, approaches, and technical challenges. *Power Systems, IEEE Transactions on*, 22(4):1743–1752, 2007.

[MDL10] Yoann Maurel, Ada Diaconescu, and Philippe Lalanda. Ceylon: A service-oriented framework for building autonomic managers. *Engineering of Autonomic and Autonomous Systems, IEEE International Workshop on*, 0:3–11, 2010.

[MHS$^+$11] Martina Maggio, Henry Hoffmann, Marco D. Santambrogio, Anant Agarwal, and Alberto Leva. Decision making in autonomic computing systems: comparison of approaches and techniques. In *8th International Conference on Autonomic Computing (ICAC)*, pages 201–204, 2011.

[MLD11] Yoann Maurel, Philippe Lalanda, and Ada Diaconescu. Towards a service-oriented component model for autonomic management. In Hans-Arno Jacobsen, Yang Wang, and Patrick Hung, editors, *IEEE SCC*, pages 544–551. IEEE, 2011.

[MRWJ$^+$10] A.-H. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid, IEEE Transactions on*, 1(3):320–331, 2010.

[MSS11] Christian Müller-Schloer and Hartmut Schmeck. Organic computing: Quo vadis? In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*, pages 615–627. Springer Basel, 2011.

[MV06] C. Marnay and G. Venkataramanan. Microgrids in the evolving electricity generation and delivery infrastructure. In *Power Engineering Society General Meeting, 2006. IEEE*, pages 5 pp.–, 2006.

[Nag04] Radhika Nagpal. A catalog of biologically-inspired primitives for engineering self-organization. In Giovanna Marzo Serugendo, Anthony Karageorgos, OmerF. Rana, and Franco Zambonelli, editors, *Engineering Self-Organising Systems*, volume 2977 of *Lecture Notes in Computer Science*, pages 53–62. Springer Berlin Heidelberg, 2004.

[NB12] E. Negeri and N. Baken. Architecting the smart grid as a holarchy. In *SMARTGREENS, 1st International Conference on Smart Grids and Green IT Systems, Porto, Portugal*, 2012.

[NPM11] M. Negrete-Pincetic and S. Meyn. Intelligence by design for the entropic grid. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–8. IEEE, 2011.

[NRS10] Russel Nzekwa, Romain Rouvoy, and Lionel Seinturier. Modelling Feedback Control Loops for Self-Adaptive Systems. In *Third International DisCoTec Workshop on Context-Aware Adaptation Mechanisms for Pervasive and Ubiquitous Services*, 2010.

[Osb94] Martin J. Osborne. *A Course in Game Theory.* MIT Press, USA, 1994.

[Osb03] Martin J. Osborne. *An Introduction to Game Theory.* Oxford University Press, USA, 2003.

[PBC12] Glenn Platt, Adam Berry, and David Cornforth. Chapter 8 - what role for microgrids? In Fereidoon P. Sioshansi, editor, *Smart Grid*, pages 185 – 207. Academic Press, Boston, 2012.

[PKR12] M. Pipattanasomporn, M. Kuzlu, and S. Rahman. An algorithm for intelligent home energy management and demand response analysis. *IEEE Trans. Smart Grid*, 3(4):2166–2173, 2012.

[PSBM12] Jeremy Pitt, Julia Schaumeier, Dídac Busquets, and Sam Macbeth. Self-organising common-pool resource allocation and canons of distributive justice. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)* [DBL12], pages 119–128.

[Ren] Terry Renner. *Quantities, Units and Symbols in Physical Chemistry.* The Royal Society of Chemistry.

[RHT$^+$13] Daniel Romero, Gabriel Hermosillo, Amirhosein Taherkordi, Russel Nzekwa, Romain Rouvoy, and Frank Eliassen. The DigiHome Service-Oriented Platform. *Software: Practice and Experience*, 43(10):1143–1239, 2013.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education, 2010.

[Rod05] S. Rodriguez. *From analysis to design of Holonic Multi-Agent Systems: A Framework, Methodological Guidelines and Applications.* PhD thesis, Université de Franche-Comté, 2005.

[Sch05] Hartmut Schmeck. Organic computing - a new vision for distributed embedded systems. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, ISORC '05, pages 201–203, Washington, DC, USA, 2005. IEEE Computer Society.

[SD06] John W. Sweitzer and T. Christine Draper. Architecture overview for autonomic computing, in autonomic computing: Concepts, infrastructure, and applications (book), 2006.

[SFLD$^+$07] A. Schaeffer-Filho, E. Lupu, N. Dulay, Sye Loong Keoh, K. Twidle, M. Sloman, S. Heeps, S. Strowes, and J. Sventek. Towards supporting interactions between self-managed cells. In *Self-Adaptive and Self-Organizing Systems, 2007. SASO '07. First International Conference on*, pages 224 –236, july 2007.

[SH10] R. Sterritt and M. Hinchey. Spaace iv: Self-properties for an autonomous and autonomic computing environment, part iv a newish hope. In *Engineering of Autonomic and Autonomous Systems (EASe), 2010 Seventh IEEE International Conference and Workshops on*, pages 119 –125, march 2010.

[Sim69] Herbert A. Simon. The sciences of the artificial. MIT Press, Cambridge, Mass, 1st edition, 1969.

[TDLS09] K. Twidle, N. Dulay, E. Lupu, and M. Sloman. Ponder2: A policy system for autonomous pervasive environments. In *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, pages 330 –335, april 2009.

[UD11] Mihaela Ulieru and René Doursat. Emergent engineering: a radical paradigm shift. *IJAACS*, 4(1):39–60, 2011.

[WHH09] Danny Weyns, Alexander Helleboogh, and Tom Holvoet. How to get multi-agent systems accepted in industry? *Int. J. Agent-Oriented Softw. Eng.*, 3:383–390, May 2009.

[WHH$^+$10] Danny Weyns, Robrecht Haesevoets, Alexander Helleboogh, Tom Holvoet, and Wouter Joosen. The macodo middleware for context-driven dynamic agent organizations. *ACM Trans. Auton. Adapt. Syst.*, 5:3:1–3:28, February 2010.

[WJ95] Michael Wooldridge and Nicholas Jennings. Agent theories, architectures, and languages: A survey. In Michael Wooldridge and Nicholas Jennings, editors, *Intelligent Agents*, volume 890 of *Lecture Notes in Computer Science*, pages 1–39. Springer Berlin / Heidelberg, 1995. 10.1007/3-540-58855-8_1.

Architectures Génériques pour des Systèmes Autonomiques Multi-Objectifs Ouverts - Application aux Micro-Grilles Intelligentes

Sylvain FREY

RESUME : L'autonomicité - la capacité des systèmes à se gérer eux-mêmes - est une qualité nécessaire pour parvenir à contrôler des systèmes complexes, c'est à dire des systèmes ouverts, à grande échelle, dynamiques, composés de sous-systèmes tiers hétérogènes et suivant de multiples objectifs, éventuellement en conflit. Dans cette thèse, nous cherchons à fournir des supports génériques et réutilisables pour la conception de tels systèmes autonomiques complexes. Nous proposons une formalisation des objectifs de gestion, une architecture générique pour la conception de systèmes autonomiques multi-objectifs et adaptables, et des organisations génériques pour l'intégration de tels systèmes autonomiques. Nous appliquons nôtre approche au cas d'utilisation des réseaux électriques intelligents, qui sont un parfait exemple de complexité. Nous présentons une plateforme de simulation que nous avons développée et via laquelle nous illustrons nôtre approche, au travers de plusieurs scénarios de simulation.

MOTS-CLEFS : informatique autonomique, systèmes complexes, architecture, patrons de conception, grilles électriques intelligentes, maison intelligente

ABSTRACT : Autonomic features, i.e. the capability of systems to manage themselves, are necessary to control complex systems, i.e. systems that are open, large scale, dynamic, comprise heterogeneous third-party sub-systems and follow multiple, sometimes conflicting objectives. In this thesis, we aim to provide generic reusable supports for designing complex autonomic systems. We propose a formalisation of management objectives, a generic architecture for designing adaptable multi-objective autonomic systems, and generic organisations integrating such autonomic systems. We apply our approach to the concrete case of smart micro-grids which is a relevant example of such complexity. We present a simulation platform we developped and illustrate our approach via several simulation scenarios.

KEYWORDS : autonomic computing, complex systems, architecture, design patterns, smart grid, smart home