The ASCENS Case Studies: Results and Common Aspects

Nikola Šerbedžija

Fraunhofer FOKUS

Abstract. This chapter focuses on pragmatic aspects of ASCENS project illustrating the role and significance of the three major application domains (swarm robotics, cloud computing and e-mobility) that motivate and pragmatically justify the approach to construct autonomous systems. A special insight is given into similarities and differences of the ASCENS case studies and their common abstract characteristics that led to a general-purpose methodology for expressing, evaluating and deploying knowledge-based, self-aware and adaptive behaviors. From this perspective selected ASCENS tools and methods to support the system development lifecycle are further discussed and illustrated on concrete examples. Finally future plans are given pointing out to the use and further evolvement of the ASCENS technology.

Keywords: application of collective adaptive systems, service component ensembles, software development life cycle, real-life systems

1 Introduction

The application domain, represented by three major case studies, namely swarm robotics, science cloud and e-mobilioty, played a central role in the ASCENS project[1]. They provide a source of motivation for the ASCENS technology and a treasury of trial examples upon which ASCENS solutions could be tested in practice. Case studies also served as a gravity for joint work among different partners and work packages as the whole spectrum of results had to be put together and applied on the case studies scenarios. This constant interaction between theory and practice made the ASCENS highly thepretical approach unified, pragmatic and well suited for a range of the application domains, far beyond the specific areas of the ASCENS case studies.

The ASCENS project deals with the development and deployment of autonomous systems with a special attention paid on technical awareness and adaptive behavior of the underlying systems at one side and rigorous and formal reasoning about the correct system functioning at another. In the early project phase the development lifecycle for autonomous systems has been proposed (see

[1] ASCENS website: http://www.ascens-ist.eu/

the Chapter 1.4.1 of this book) tracing the methodology and the roadmap for system design and development. A number of distinct phases of the development process have been identified and many tools have been developed to support the modelling and development in each of the lifecycle stages. Due to a highly non-deterministic character of the autonomous systems, whose behavior is dynamic and sensitive to unpredicted situation, system validation and verification plays an important role in the project.

Contrary to majority of computing systems now in use, autonomous systems' behavior is highly dynamic and reactive to unexpected situation. That non-determinism makes the system verification process extremely difficult as the system alters its behavior in run-time replying to the state of surrounding and the knowledge about its own state. Those circumstances cannot be predicted in advance and system cannot be fully tested and de-bugged before it comes to exploitation. Furthermore, when autonomous system is deployed, its variable behavior is a run-time response to live situation and it is hard to differ correct from mal functioning. The ASCENS response to such difficulties is to verify and validate the system in all of its development and deployment phases applying rigorous methodologies and formal methods, from requirement analyses and modelling up to the run-time monitoring.

Having all these challenges in mind, ASCENS strategy was to prove its methodology throughout the development process with the concrete and non-trivial applications. That makes the role of ASCENS case studies manifold:

– Inspirational
– Experimental
– Verifiable
– Pragmatic

From the very project beginning initial concepts for requirement specification, awareness, adaptation and overall system modelling have been taken from problem-rich application domains of swarm robotics, cloud computing and e-mobility. Both typical examples from the application domain and concrete trial scenarios were thoroughly studied. Inspired and motivated by a wide problem space of ASCENS case studies, a number of new methods have been developed, almost from scratch, and a number of existing methods were modified to reply to these challenges. Out of thorough problem specification, a structured knowledge representation in form of KnowLang [22] approach has been designed allowing for a sound (self-) awareness definition based on knowledge. Further system modelling could use this knowledge to exercise awareness rich behavior, making system aware of its functional and non-functional requirements. It furthermore led to development of a unique adaptation model called SOTA [1] that defines adaptation as a system journey in a multidimensional space where the coordinates are awareness aspects of the system. By deploying SOTA on case studies a general-purpose catalog of adaptation patterns have been defined that help designer express and exercise with adaptive behavior. Further adaptive system requirements were supported by In order to guide the design of an ensemble-based system from high-level strategic goals, requirements and patterns to their low-level realization in terms of system architecture (components and ensembles) we can use the Invariant Refinement Method (IRM) [13]

SCEL process algebra [7] is another ASCENS pillar that allows for system modelling and reasoning on the system behavior. It offers means for defining a system as a set of service components and their ensembles extended with local knowledge to express awareness and adaptive policies for predicate-based bindings to express autonomous behavior. The *Helena* approach [15] has been further developed for modeling collaborations using a UML-like notation focusing on the description of the behavior at individiual and collective (ensemble) level. Further design steps from high-level strategic goals (requirements, adaptation patterns) to their low-level system architecture realization (components and ensembles) are supported by the Invariant Refinement Method (IRM) [13]

Experimental significance of the case studies could be seen through numerous pragmatic examples which were used to model and verify corresponding system behavior. Each concrete problem from the case studies domain has been modeled, and analyzed by the corresponding ASCENS tool, testing simultaneously the expressive power of the tool itself and the pragmatic significance of the solution. Throughout the project this interaction between theory and practice contributed to achieve (1) sound and usable methodology and (2) useful pragmatic results for the application domain and industrial partners. The ASCENS work has been characterized by this interaction and mutual influence that enrich both the theory and the practice. Two major means developed from the skretch and for the project purposes were used to deploy and test ASCENS case studies in practice: JRESP [12] and JDEECO [14] both based on SCEL abstraction, integrating numerous other ASCENS tools.

Verifiable significance of the case studies is present at all the development phases. The case studies offered realistic, pragmatic and complex examples of use, making the highly theoretical validation/verification means both sound and pragmatic. Each concept developed within ASCENS has been first validated in its generic form and then applied on a concrete example from the application domain for further evaluation. For example, SOTA adaptation patterns allow for high-level reasoning and proofs for adaptive behavior and appropriate selection of the adaptation patterns for each of pragmatic problem. High level modelling led to further reasoning on important system properties like safety (e.g. proving that e-vehicles will never deadlock each other while using common resources e.g. parking lots or charging station) and liveness (e.g. proving that the system will really find the optimal route for a vehicle respecting major constraints e.g. battery level, timing etc.). D-Finder[4] tool has been used for the compositional variation. Further examples of validating coordination and collaboration algorithms and optimizing local vs. global goal strategies are taken from a reach problem space of swarm robotics, cloud computing and e-mobility (e.g. guaranteeing that each e-vehicle obtains a park place nearby its point of interest, taking into account that the garage needs to satisfy needs of hundreds of other e-vehicles. The ASCENS approach also integrates existing verification tools like BIP (Behavior, Interaction, Priority) [2] or SBIP (Stochastic BIP) [3] and used them together with ASCENS novel tools to perform statistical model checking and perform quantitative verification.

The rest of the chapter further elaborates on a mutual influence between praxes and theory by detailing the application challenges (section 2) that are used to motivate and develop a common approach (section 3) to model, develop and deploy autonomous systems. The set of ASCENS generic tools (section 4) re-visits a wide spectrum of developed means to support the use of ASCENS approach in solving concrete pragmatic problems illustrating ASCENS results and solutions in real application deployments (section 5). Finally, the conclusion (section 6) summarizes the results and discusses a wider pragmatic significance and influence that the ASCENS project has in the domain of adaptive and autonomous systems.

2 Application Challenges

A thorough analysis of the application problem space is crucial, both for successful application design and development and for proving expressive power of the ASCENS methods. This dual role of the case studies has been especially important at the beginning of the project, when the ASCENS approach was defined and developed. The approach has been to decompose the application fields to low-level details, provide partial solutions and to compose those solutions into harmonized methodology that defines complete development lifecycle for autonomous systems.

2.1 Application Overview

To explore the system requirements for autonomous systems, three complex application domains have been closely examined: swarm robotics, cloud computing and e-mobility. The overall strategy has been to analyse separate application domains, findout the charateristics that make these system konwladge aware and autonomus, and finaly to generalize these characteristics into a possibly common set of jont features that could be modelled by a general methodology.

Swarm robotics application domain deals with creation of multi-robot systems that through interaction and coordination among participating simple robots and their environment can accomplish a common goal, which would be impossible to achieve by a single robot. The basic idea behind the application scenario is to organize and control a rescue operation in an emergency situation. Figure 1 illustrates a multi-robot system containing two types of robots with circles shawing possible different grouping (ensemble building) among different or same robot type.

Cloud computing is an approach that provides computing resources to users in a service-based manner, over the internet. By sharing computing resources by many users, significant throughput can be achieved leading to energy and costs savings. This kind of computing calls for novel techniques that would allow for highly dynamic and secure construction of virtual resources that would maintain

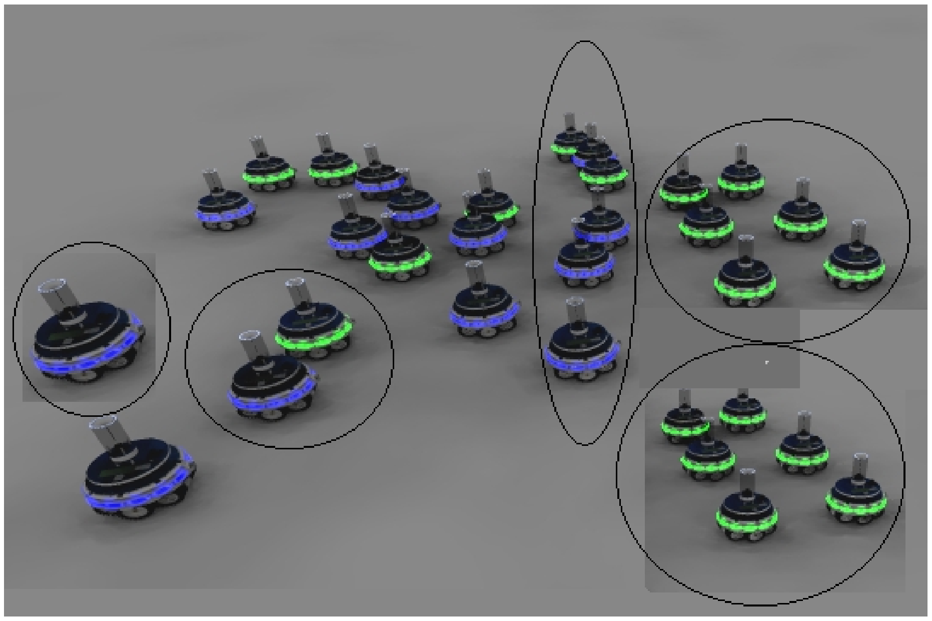

Fig. 1. Swarm robotics

the throughput and efficiency high, wile reducing the number of computer used. This, appreantly contraversial requirements insure enormous reduction in energy use and computing costs, making powerfull computing resources available to everyone. Figure 2 illustrates a collection of computing resources brought together to form a cloud that further offer its services to the users.

E-mobility is a vision of future transportation by means of electric vehicles network allowing people to fulfill their individual mobility needs in an environmental friendly manner (decreasing polution, saving energy, sharing vehicels, etc). Due to limited battery capacity, e-vehicles cannot simply pass long distances, as it is the case with traditional vehicles (and re-filling energy lasts much longer). The ultimate goal of e-mobility is to overcome that problem by offering a range of supporting activities that would allow energy-aware passengers to master distances in required time. Figure 3 illustrates a fleet of e-vehicles with indicated parking lots and charging stations.

2.2 Common Characteristics

In a closer examination the three application areas, though very different in nature, have a number of common characteristics.

Unique simple entities with clearly identified individual goals. In swarm robotics, those are elementary robots with their simple functionality and single role (e.g. a foraging robot moves and explores the area until it finds the target

Fig. 2. Cloud Computing

or come too far away from other robots then it stops). In cloud computing, elements are specific computing resources with their characteristics (e.g. a CPU with its energy consumption, execution speed, throughput etc.). In e-mobility, elements are e-vehicles, parking and/or charging stations and traffic conditions (e.g. a parking lot has its location, price and availability/occupation schedule). Obviously, all three applications can be described by a huge number of (1) single entities with (2) unique individual goals.

Distribution and grouping around global goal In swarm robotics, simple elements are grouped into multi-robot system in order to perform the function that individual robots cannot do alone. In cloud computing, more CPUs could be grouped together to offer more computational power. In e-mobility, multiple resources like charging station and parking lots can be combined to provide better overall service. Further characteristics are existance of (3) global goals, (4) grouping principles to express these global goals and (5) massive interraction that exploit these principles of sharing and collectiveness in order to (6) coordinate and harmonize local and global goals.

Awareness and knowledge are characteristics which are pre-conditions for autonomous behavior. Maintaining the knowledge of own functional and operational capabilities make both single units and their collections self-aware and capable of runtime dynamic responsiveness. Multi-robot system is aware of location and functionality of neighboring robots so that a group of robots can coordinate along the common interest. Cloud computing deals with dynamic (re-)scheduling of available (not fully used) computing resources. Maximal utilization can only be achieved if the cloud is aware of the users processing needs

Fig. 3. E-mobility

and the on-going states of the deployed cloud resources. Only with such knowledge, a cloud can make a good utilization of computers while serving individual users needs. E-mobility can support coordination only if e-vehicles know their own restrictions (battery state), destinations of users, re-charging possibilities, parking availabilities, the state of other e-vehicles nearby. With such knowledge collective behavior may take place, respecting individual goals, energy consumption and environmental requirements. Consequently, (7) self-awareness allows for knowladge-rich (8) adaptation and (9) optimization within the three case studies.

Robustness and continuous operation are crucial features of real-life systems, where an application needs to run non-interrupted, despite the possible mal-function. A multi-robot system does not stop when one robot is down. The cloud computing is per definition a set of boundless resources that can overcome the failure of single component. E-mobility aims at non-stop operation to overcome the restrictions posed by battery life-time making (10) the robustness a major aim of the overall concept.

When taking into account all the mention common characteristic it can be seen that theay all together contribute to make a target system behave (11) autonomously, which is the ultimate goal of the ASCENS approach. All the metioned elevengeneric common features (with their interpretations within all three case studies) are summarized in the table 1.

3 Common Approach

This spectrum of common features serves as a basis for modeling of massively distributed behaviours leading to a generic framework for developing and deploying complex autonomic systems [11]. To behave autonomously, a control system

Common feature	**Swarm Robots**	**Cloud computing**	**E-Mobility**
Single entities	Different types of robots	Computing resources	E-vehicles , parking lot, charging station, infrastructure
Individual goals	Find the victim, carry the obejct, ...	compute, store, ...	reach the destination, charge the battery, ...
Global goals	Build the wall, ...	increase throughput, ...	allocate all parking lots, ...
Grouping principles	”All foraging robots close to the target”, ...	”Connect idle proccessors”, ...	All available parking lots in radius of 500m of the meeting place, ...
Massive interaction	Among robots, ...	Among computing resources, ...	Among vehicles, parking lots, charging stations, ...
Coordination	Coordinate search algorithm, ...	Coordinate free resources, ...	Coordinate park lot allocation, ...
Self-awareness	”About battery state”, ...	”About its usage”, ...	”About own location”, ...
Optimization	Time, energy, performance, ...	Availability, computational task execution, ...	Arriving in time, vehicle/infrastructure usage, ...
Adaptation	To changing plans, single robot malfunction, ...	To resource failure, ...	To traffic situation, battery shortage ...
Robustness	Sensory noise, limited sensory range and battery life, ...	Failing resources, sudden intense computing requirements, ...	Range limitation, battery shortage, infrastructure problems,...
Autonomous behavior	Run-time plan change,	Decentralised decision making, global optimization, ...	Changing the route, re-allocate parking lot, ...

Table 1. Common features of the ASCENS case studies

needs to maintain knowledge about itself (particular objectives, capabilities, execution state and restrictions) and about its environment. Such collection of facts yields awareness of own functionality and effects it has on the environment which further allows for adaptive behaviour. Being capable of operating according to these three principles (knowledge, awareness, adaptation), the system is able to re-configure, re-tune and act appropriately thus behaving in autonomous manner.

The ASCENS approach breaks up a complex control problem into its elementary constituents. It deals with complications at a bottom level, solving issues at a lower scale and then harmonizing these solutions with more global ones. Localization and de-centralization is the fourth major principle of the approach.

Service components with clearly defined elementary objectives are basic system elements. They gather in larger symbiosis called ensembles in order to fulfill collective goals. As the controlled situation changes, i.e. goals are (partially) fulfilled, re-grouping takes place and the symbiosis re-structures. The criteria to construct an ensemble of service-components is some joint interest which can be expressed as a logical sentence, e.g. connect all robots that can carry up to 4kg and are in the radius of 100m with the aim to cooperatively transport 25kg heavy object or select all free parking lots in the radius of 300m that have a charging plug. That makes the communication implicit and predicate-based. The connections are established at run-time, depending on the live situation at particular time. These logical rules for highly dynamic grouping are further used for formal reasoning on optimization and coordination among distributed elements.

The overall system development life cycle consists of the following phases: rigorous design (requirement specification, modelling and validation/verification), deployment (programming) and run-time monitoring (live examination of awareness, adaptation and autonomous behavior). A number of tools have been made that support the development process at each step, thus guiding and facilitating the whole development process. Requirement specification is a phase where the dissection of the problem to be solved takes place (requirement engineering is des cribed in the Chapter 1.4.1 of this book). Each system element is separately defined both functionally (what to do) and non-functionally (how to do) yielding a set of goals that embrace the terms of functioning and description of environment. The knowledge required for system awareness and adaptation is used as a major attribute repository for system construction (formal approach to knowladge. awarenes and adaptation is described in the Chapters 1.3.1, 1.3.2, 1.3.3 and 1.3.4 of this book). The SCEL (service-component ensembles language) [8] has been developed for high-level system modelling with service components and their ensembles. Both service-components and ensembles have local knowledge used to express their goals. Knowledge is represented by ontologies that contain hierarchical and meaningful description of system properties and system goals. The goals are described as rules i.e. logical expressions with system properties (the SCEL language, its design, implementation and verification is described in the Chapter 1.2.1 of this book).

The adaptation phenomenon is formally modeled as a progress in a multi-dimensional space where each axis represents one orthogonal aspect of system awareness (facts about its own functional, operational, or any other necessities defined within requirement specification phase). Adaptation actually happens when the system state moves from one to another position within the space according to the pre and post- condition on each of its awareness- dimensions. Adaptation is a continuous process where a system acts appropriately i.e. in harmony with own capabilities and the observed environment. The SOTA adaptation model is used to extract major application requirements and offer appropriate adaptation patterns that effectively control system dynamics with numerous feedback-loops. In order to guarantee correct and timely behavior in such demanding and highly dynamic circumstances this approach relies on formal methods. The major safety and liveness properties are formally proved using SCEL process algebra (e.g. prove that two e-vehicles will never block each other while competing for a free charging station, or prove that the foraging algorithm of a robot converges in a given time). Further validation and verification of specific optimization algorithms are performed in order to guarantee correct system behavior in early design phase (e.g. prove that the optimization method will deliver the most energy-efficient route for a given multi-routing problem). Once the system is rigorously modelled and validated, the actual deployment may take place sewing the system together. The jRESP and jDEECo deployment tools offer direct Java programming support for the SCEL and SOTA models. Further modlling tool used to specify deep logical and stochastic functioning that describe the system behavior is the POEM language [10].The Iliad implementation of POEM is fully integrated in jRESP and can be used as awareness engine for SCEL programs.

Due to a seamless functioning of autonomous systems, where system changes are means for appropriate behavior, possible malfunctions are difficult to discover. Therefore, a number of tools have been developed for run-time monitoring where internal system knowledge and topology (ensemble construction) as well as awareness and adaptive characteristics are observed. For example, the monitoring tools can visualize how the robots, close to the target and with enough battery-charge are grouped into ensemble to perform joint transport of a heavy object. Once the task is performed, the ensembles are dismantled freeing robots for another assignment. Monitoring inspects and displays major system principles: knowledge, awareness and adaptation, offering a visualization of dynamic ensemble building criteria, thus directly observing autonomous behavior. If some malfunctioning is discovered at run-time, a system modification is considered going back to modelling and design system development phases. The monitoring is done with the following tools: ARGoS [18], AVis Plug-in[2] and POEM, for swarm robotics, the Zimory cloud platform [24] and SCP for science cloud and jDEECo and IRM for e-mobility.

The detailed description of ensemble development life cycle and ASCENS best practice for collective adaptive system is given in the Chapter 1.4.1 of this book.

4 Generic Set of Common Tools

The set of common features, as described in the previous section, served as a basis for further work and experimenting in each of the case studies. At the same time it led to a generic set of common tools that could be used and tested within scenarios from the case studies domains. The figure 4 shows some of generic tools which are available in the rich ASCENS tool repository. Most of the tools are newly developed and/or adjusted for the ASCENS purposes. The cyclic arrows indicates a multi-level feed back loops - present in all development

[2] see the ASCENS User Guide

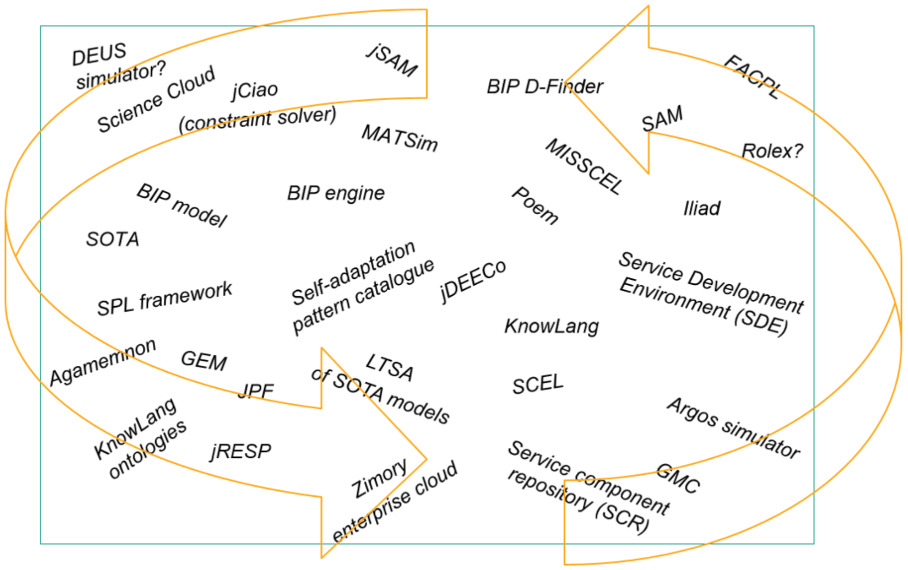

Fig. 4. ASCENS development and deployment tools

phases, as described in the ASCENS development lyfe cycle. At one side, the project creates a comprehensive list of generic tools that could be used in any deployment scenario (fully independent from the ASCENS application domain), at another side, these tools were tested and fine-tuned using complex practical problems with real data. The ASCENS tools were developed within ACENS theoretical work packages, making the tools abstract and general-purpose.

The tool integration has been allocated to a separate work package whose aim was to generate a standard integrated development environment where the modeling and editing tools are placed together with profiles and debuggers, making it possible to ptactically use the whole development life cycle as described in the previous chapter of this book. All of the tools previously described are stored in a common repository making it a common place to apply ASCENS technology and follow ASCENS development life cycle.

All mentioned tools were separately tested in a theoretical context, or using single problems from the case studies domains. Once fully tested, the tools were applied on a large scale practical scenarios from the ASCENS case studies (separate work package). The table 2 list the major ASCENS tools, as they were used within each of the case study and according to the EDLC (Ensemble Development Life Cycle).

The ASCENS tool repository with numerous deploymemnt examples plaid an important and dual role: (1) tools were tested in a real and large scale application domain - proving a wide applicability and a strong practical orientation of the ASCENS approach, (2) the end users and corresponding industrial parties could

EDLC Phase	**Swarm Robots**	**Cloud computing**	**E-Mobility**
Requirements Engineering	SOTA, Gem, POEM	Knowlang, IRM	simSOTA, IRM
Modeling/ Programming	SCEL, jRESP, Poem	SOTA, SCEL, KnowLang	SCEL, SCLP
Verification/ Validation	BIP, jRESP	jRESP	jDEECO
Deployment	ARGoS	SCP SPL, Java, Zimory	jDEECo. Java, MatSim
Monitoring	ARGoS, AVI Plug-In Tool	Zimory, SCP	jDEECO/DiSL/SPL MatSim
Awareness	POEM, ARGoS, AVI	SCP	jDEECo
Self-Adaptation	ARGoS, AVI, POEM	Zimory, SCP	jDEECo, IRM
Feedback	POEM	SPL	MatSim

Table 2. ASCENS tools used for the case studies development

see the benefits (and challenges) of a fully scientific approach to construct and deploy large practical systems, insuring their reliable and correct functioning.

5 Application Deployments

From the very beginning, the project theoretical development has been interleaved with practical exercising, taking various examples from the main ASCENS application areas. Most of these practical results were reported in the theoretical project deliverables. Nevertheless, three major applications served as a pragmatic guideline during the project and they were specified, modeled and developed step-wise during the project (in a separate work package). The task structure of the case study work package is similar to the ASCENS development life cycle and had a following major subtasks:

– Requirements analysis and specification
– Model synthesis
– Integration and simulation
– Implementation and evaluation/validation

In the first project stage (year) a thorough requirement analyses took place, first in an informal way and then using a rich set of ASCENS tools for knowledge expression, self-awareness and adaptive behavior. In the second project stage, major system modelling took place synthetizing most of the modelling techniques developed within project. The third stage has been characterized by numerous integration effort, interfacing different tools and languages as well as undertaking numerous simulations in order to pre-check the system behavior before doing final implementations. In the last project stage, the three ASCENS case studies were deployed, tested, monitored and evaluated.

This sections only gives a short reference to the case studies developments, as each of the case study is fully described in the following chapters (1.5.2, 1.5.3 and 1.5.4) of this book. However, one further special example is described here: a robot race exhibition, presented at the ICT conference in Vilnius in November 2013. The significance of the exhibition was not only to prove pragmatic ASCENS techniology by having real robots performing in real-life settings. It also justified the ASCENS complete ensemble development life cycle as numerous concrete theoretical tools were demonstrated on a concrete example.

5.1 ASCENS Case Studies

The ASCENS project took three major application domain as a major prgmatical inspiration domain: swarm robotics, cloud comput9ng and e-mobility. Each of the area is complex per se, up to date and a subject of many other contemporary research and developments.

Swarrm Robotics The swarm robotics case study deals with a disaster recovery scenario. Numerous separate problems from the scenario were separately specified, modelled and veriifed during the project work. A special attention has been paid to local vs. global behaviors [23] and distributed algorithms which represent typical class of problems within swarm robotics theory. An engineering approach to apply EDLC in designing a multi-robot system is described in [20] and a a separate chapter (Chapter 1.5.2) of this book has been fully dedicated to swarm robotics case study.

Science Cloud The science cloud case study deals with a vision of an autonomic cloud, providing a platform-as-a-service computing infrastructure, which is created and maintained by a free collection of ad hoc connected heterogeneous voluntary computers forming a peer-to-peer network. The science cloud has been developed from scratch fully deploying ASCENS ensemble development lifecycle [16, 19]. A special focus in science cloud case study is on self* features, making the cloud fully aware of its functional and operational state, thus autonomously providing resilience, data redundancy, and failover mechanisms. A separate chapter (Chapter 1.5.3) of this book has been fully dedicated to science cloud.

E-Mobility The e-mobility case study deals with a vision of a future transportation that will include more and more e-vehicles as transportational means, posing a whole range of problems that need to be solved in order to insure the transition and better acceptance of the new generation of e-vehicels. The e-mobility case study was engineered by strictly applying ASCENS methodology [21, 6]. A special attention in the case study has been paid to finding optimal energy routes [9], and overcoming the local vs. global goal optimization, using constraint logic programming techniques [17]. A separate chapter (Chapter 1.5.4) of this book has been fully dedicated to e-mobility.

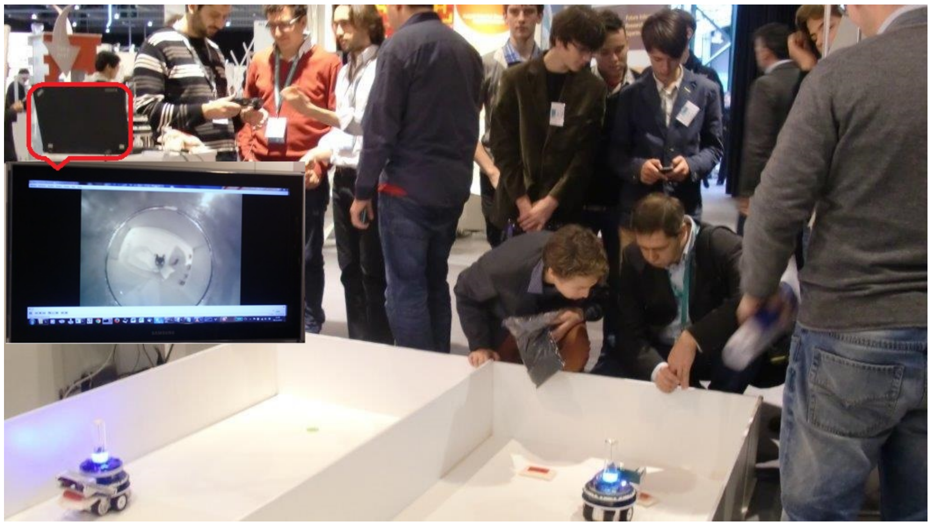

Fig. 5. Robot race

5.2 Robot race

The challenges of controlling the robot behavior in performing certain task can best be understood if seen from the robot perspective. The complexity does not primarily come from the task itself, but rather from the interaction that goes on between the robot sensory system, environment and self-directed robot performance. To illustrate that, an exhibition has been organized at well attended ICT conference (Vilnius, November 2013) where ASCENS autonomous robot competed with a human-controlled robot[3]. The task given to the robots was to find building blocks in a closed area, grab them (one by one), and carry them to the place where a wall should be constructed. The competion arena[4] from ICT Conference is illustared on figure 5.

The ASCENS robot was fully autonomous and a "competitor robot" was operated by a joystick which could move the robot left/right; forward/backwards and instruct it to grab/release the building blocks (the competitor robot had

[3] See the ASCENS blog "Beauty is in the eye of the beholder at: http://blog.ascens-ist.eu/2013/11/beauty-is-in-the-eye-of-the-beholder/

[4] A video clip of the exhibition can be seen at: http://www.aware-project.eu/2013/ascens-ict-2013/

no knowladge on how to find, grab and carry objects and relied on the human operator fully). Both robots belong to the marXbot robot generation [5], a modular and easily re-configurable robots equiped with numerous devices that allow for sensing and acting in the deployed environment. The tasks allocated to the robots seemed trivial to the audience, so that most of the competitors believed that ASCENS autonomous robot does not stand a chance, against the robot controlled by a human. That proved to be wrong. Most people lost, only a couple of young, joystick-virtuous competitors won.

But for those who could outperform the ASCENS autonomous robot, a fair-play rule has been introducd: since the robots sensory system is less sophisticated than ours, the vision of the human competitor has been reduced to the visual system of the robot (a competitor was not supposed to look to the compeeting arena with own eyes, but rather to the screen which mirror "what robot sees" (in the left-upper corner of the Figure 5 a screen shot of the robot vision is illustrated). That gave the competitors equal chances. When both competitors have exactly the same information about environment, ASCENS robot performed much better. That shows how seemingly simple assignment (from human point of view) is actually complex for a fully autonomous robot. Taking into account relatively primitive robot sensory system, the robot performance has been quite good and reliable.

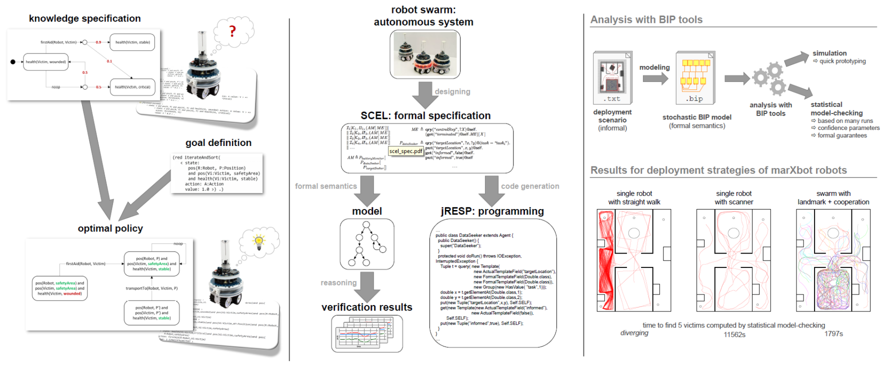
knowledge specification
goal definition
optimal policy
robot swarm: autonomous system
designing
SCEL: formal specification
formal semantics
code generation
model
jRESP: programming
reasoning
verification results
Analysis with BIP tools
deployment scenario (informal)
modeling
stochastic BIP model (formal semantics)
analysis with BIP tools
simulation
quick prototyping
statistical model-checking
based on many runs
confidence parameters
formal guarantees
Results for deployment strategies of marXbot robots
single robot with straight walk
single robot with scanner
swarm with landmark + cooperation
time to find 5 victims computed by statistical model-checking
diverging
11562s
1797s

Fig. 6. Monitoring tool

Showing ASCENS results at well attended congress with several thousand visitors provided a great audience for the ASCENS demo which attracted more than a hundred competitors (people who really competed with the ASCENS robot). The significance of the demonstration at the Vilnius exhibition has been multifold:

1. the ASCENS pragmatic approach has been demonstrated in a vivid and successful error free settings. It has been one of most attended stall at the

congress and the ASCENS robots has been running 3 days non-stop from early morning to late evening

2. ASCENS theoretical work has also been demonstrated through several model descriptions, simulation and verification tools. The Figure 6 contains three posters from the conference illustrating the specification, modelling and verification phase of the robot race demo. It has been a unique situation to discuss the high-level tools in front of the running example, who used those tools.
3. ASCENS evaluation and monitoring approach has been illustrated by the design architecture of the monitoring tool, as shown on Figure 7. The AVI monitoring can show the internal awareness structures of the running example, monitoring and analyzing properties used for ensemble creation.

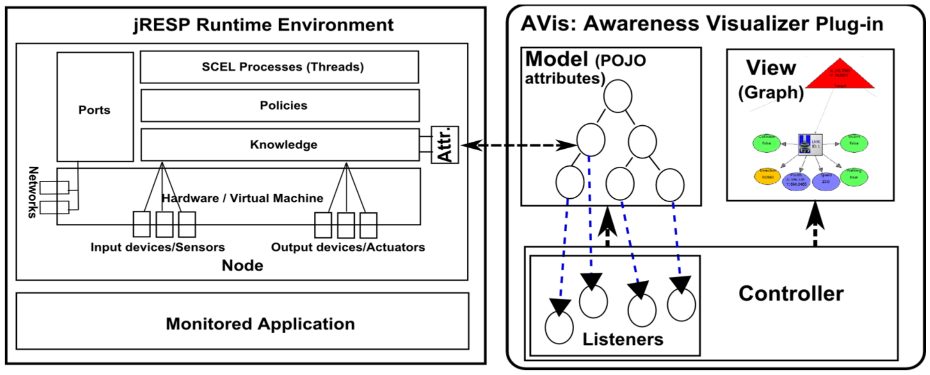

Fig. 7. Monitoring tool

6 Conclusion

This chapter presents the ASCENS achievements as a continuous balancing between the theory and practice. At one side, a number of scientists put their efforts together to make abstract and generic high-level methods and tools to model, analyze, validate and develop autonomous systems. At another side, pragmatic and business driven partners kept the ASCENS achievements applicable and down to deployment terrain . This constant interaction between the theory and practice have been beneficial for both sides: theoreticians got real problems and numerous practical data descriptions that they traditionally do not have, so that their work has been backed with real world problems. Industrial partners, at another side, were in the position to directly influence the theoretical work and tailor its soulution towards own pragmatic goals, which could be used to improve products and achieve results which would not be possible to achieve without such collaboration.

A wider significance and influence of the ASCENS outcomes is expected also in other application domains. Namely, the ASCENS generic results are applicable in any areas where autonomic control is needed. Further exploitation activities like planned summer school and publication of project results at prestigious scientific journals and conferences should re-enforce already well known ASCENS methodology. Pragmatic exploitation of ASCENS results is guaranteed by the project industrial partners and for a wider use, a tool repository (see the Chapter 1.4.3 of this book) and ASCENS user manual are made on-line and available to any interested party. A close collaboration with other EU projects, especially within collective adaptive initiative opens further perspective of continuing and further development and deployment of ASCENS work.

References

1. Abeywickrama, D., Bicocchi, N., Zambonelli, F.: Sota: Towards a general model for self-adaptive systems. In: Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on. pp. 48–53 (June 2012)
2. Basu, A., Bozga, M., Sifakis, J.: Modeling Heterogeneous Real-time Components in BIP. In: SEFM. pp. 3–12. IEEE Computer Society (2006)
3. Bensalem, S., Bozga, M., Delahaye, B., Jégourel, C., Legay, A., Nouri, A.: Statistical Model Checking QoS Properties of Systems with SBIP. In: Margaria, T., Steffen, B. (eds.) ISoLA (1). LNCS, vol. 7609, pp. 327–341. Springer (2012)
4. Bensalem, S., Bozga, M., Sifakis, J., Nguyen, T.H.: Compositional verification for component-based systems and application. In: ATVA (2008)
5. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., Mondada, F.: The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4187–4193. IEEE Press (2010)
6. Bures, T., Nicola, R.D., Gerostathopoulos, I., Hoch, N., Kit, M., Koch, N., Monreale, G.V., Montanari, U., Pugliese, R., Serbedzija, N., Wirsing, M., Zambonelli, F.: A life cycle for the development of autonomic systems: The e-mobility showcase. 2013 IEEE 7th International Conference on Self-Adaptation and Self-Organizing Systems Workshops 0, 71–76 (2013)
7. De Nicola, R., Loreti, M., Pugliese, R., Tiezzi, F.: SCEL: A Language for Autonomic Computing. Tech. rep., IMT Lucca (January 2013)
8. De Nicola, R., Loreti, M., Pugliese, R., Tiezzi, F.: A Formal Approach to Autonomic Systems Programming: The SCEL Language. TAAS 9(2), 7 (2014)
9. Hoch, N., Zemmer, K., Werther, B., Siegwarty, R.Y.: Electric Vehicle Travel Optimization - Customer Satisfaction Despite Resource Constraints. In: Proc. of IEEE IVS. IEEE (2012)
10. Hölzl, M.: The Poem Language (Version 2). Tech. Rep. 7, ASCENS (July 2013), http://www.poem-lang.de/documentation/TR7.pdf
11. Hölzl, M.M., Wirsing, M.: Towards a system model for ensembles. In: Agha, G., Danvy, O., Meseguer, J. (eds.) Formal Modeling: Actors, Open Systems, Biological Systems. LNCS, vol. 7000, pp. 241–261. Springer (2011)
12. jRESP Java Run-time Environment for SCEL Programs (2012)

13. Keznikl, J., Bures, T., Plasil, F., Gerostathopoulos, I., Hnetynka, P., Hoch, N.: Design of Ensemble-Based Component Systems by Invariant Refinement. In: Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering. pp. 91–100. CBSE '13, ACM, New York, NY, USA (2013)
14. Keznikl, J., Bures, T., Plasil, F., Kit, M.: Towards Dependable Emergent Ensembles of Components: The DEECo Component Model. In: WICSA/ECSA. pp. 249–252. IEEE (2012)
15. Klarl, A., Hennicker, R.: Design and Implementation of Dynamically Evolving Ensembles with the HELENA Framework. In: Proceedings of the 23rd Australasian Software Engineering Conference. pp. 15–24. IEEE (2014)
16. Mayer, P., Klarl, A., Hennicker, R., Puviani, M., Tiezzi, F., Pugliese, R., Keznikl, J. Bures, T.: The autonomic cloud: A vision of voluntary, peer-2-peer cloud computing. In: Self-Adaptation and Self-Organizing Systems Workshops (SASOW), 2013 IEEE 7th International Conference on. pp. 89–94 (Sept 2013)
17. Monreale, G.V., Montanari, U., Hoch, N.: Soft Constraint Logic Programming for Electric Vehicle Travel Optimization. CoRR abs/1212.2056 (2012)
18. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Caro, G.D., Ducatelle, F., Stirling, T.S., Gutiérrez, Á., Gambardella, L.M., Dorigo, M.: ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In: IROS. pp. 5027–5034. IEEE (2011)
19. Serbedzija, N., Mayer, P., Klarl, A.: Constructing Autonomous Systems: Major Development Phases. International Journal on Advances in Intelligent Systems 6(4) (December 2013)
20. Serbedzija, N.: Constructing Autonomous Multi-Robot System. In: The Third International Conference on Intelligent Systems and Applications. Sevilla, Spain (June 2013)
21. Serbedzija, N., Bures, T., Keznikl, J.: Engineering Autonomous Systems. In: PCI13 Proceedings of the 17th Panhellenic Conference on Informatics. pp. 128–135. Thesalloniki, Greece (September 2013)
22. Vassev, E., Hinchey, M.: Autonomy Requirements Engineering. IEEE Computer 46(8), 82–84 (August 2013)
23. Yamins, D.: Towards a theory of local to global in distributed multi-agent systems (i). In: Proceedings of the fourth international joint conference on autonomous agents and multiagent systems (AAMAS'04). pp. 183–190. ACM Press, New York, NY (2005)
24. Zimory Software: Zimory Cloud Suite. http://www.zimory.com/ (August 2014)