A Roadmap towards Sustainable Self-aware Service Systems

Schahram Dustdar, Christoph Dorn, Fei Li
Information Systems Institute
Vienna University of Technology
Argentinierstr 8/184-1
1040 Vienna, Austria
lastname@infosys.tuwien.ac.at

Luciano Baresi
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Via Golgi, 40
20133 Milano, Italy
luciano.baresi@polimi.it

Giacomo Cabri
Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905
41125 Modena, Italy
giacomo.cabri@unimore.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Via G. Buffi 13
6904 Lugano, Switzerland
c.pautasso@ieee.org

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
Via G. Amendola 2
42100 Reggio Emilia, Italy
franco.zambonelli@unimore.it

ABSTRACT

Self-awareness and self-adaptation have become primary concerns in large-scale systems as they have become too complex to be managed by human administrators alone, but rather require a new blend of coordination mechanisms between people and software services.

This paper presents a roadmap to effective and efficient system adaptation through coupling self- awareness of global-level goals with sustainability constraints. Sustainability of large-scale systems challenges self-adaptation approaches by its intrinsic characters of global and long-lasting effects. We introduce five levels of awareness: (i) event-awareness, (ii) situation-awareness, (iii) adaptability awareness, (iv) goal-awareness, and (v) future-awareness. Within each level we introduce applicable principles and subsequently outline necessary models, algorithms, and protocols. The approach puts special focus on the interdependencies of human and service elements.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Miscellaneous

Keywords

roadmap, service system, self-awareness, sustainability, context patterns, adaptation coordination

1. INTRODUCTION

Over the past decades we have observed a trend towards large-scale systems [47]. These systems are characterized by decentralized control, continuous change of elements, omnipresence of failures, and conflicting interests. Especially noteworthy is the increasingly blurry boundary between humans and software. Self-adaptation has become a primary concern in those systems as they have become too complex to be managed by human administrators alone [29]. Failures happen too often for humans to follow and react to. Relations between entities have become too impenetrable to extract the relevant dependencies. User requirements shift between different contexts depending on the underlying social structure. Elements belonging to different authorities defy any attempts to exercise any form of centralized, external adaptation on the system.

We focus particularly on service systems where traditional software services reside alongside Human provided Services (HpS) [50]. The concept of Human-provided Services envisions humans to publish their skills and capabilities as Web services. Not only interactions between humans are supported by using HpSs, but also interactions with software services, for example, human interactions in BPEL4People-like processes. Interactions range from ad-hoc to process-centric collaborations. HpS is an integral concept, as it provides the foundation for modeling, measuring, engineering, and controlling a versatile mix of software and Human-provided Services. In the scope of this paper, *service system* always denotes an ensemble comprising Human-provided Services and software-provided services (SpS).

Awareness is a fundamental prerequisite to any self-adaptation endeavor. Extensive and comprehensive information about the configuration of the *self* and the *context* is imperative. The *self* comprises all parts of the system that are monitored, evaluated, and adapted. The *context* describes the system's embedding in its environment. The convergence of technical and social networks enables us to observe the properties that are fundamental to understanding the human interaction structure [36] and how this structure affects the requirements of the technical systems that are engineered to assist such human collaboration. System self-adaptation must consequently regard humans and their relations as integral parts of the overall system, rather than merely a form of context.

Information technology needs to become more sustainable. In service systems, sustainability does not only address energy consumption but any type of limited resource. Adapting the definition from the Brundtland report [9]: "sustainable service systems seek to meet the current goals and requirements of the whole system without compromising the ability to meet those of the future". To this end, awareness has to include resource characteristics to enable sustainable self-adaptation.

In this paper, we outline a roadmap to sustainable self-aware service systems. We define a conceptual framework that specifies levels of awareness. These levels describe the scope of information available for self-adaptation. We further outline the interdependency of principles, models, algorithms, and mechanisms we see suitable and necessary to achieve each level of awareness. Ultimately, each level of awareness permits ever more sophisticated and ever more sustainable self-adaptation.

The following subsection (Sec. 1.1) provides a motivating scenario. We outline the major challenges of awareness and self-adaptation in the subsection thereafter. Section. 2) introduces the building blocks of our roadmap. Section 3 discusses the state of the art in self-adaptive systems and explains how our roadmap extends and improves existing research efforts. Finally, Section 4 concludes this position paper.

1.1 Scenario

Assume a service system comprising Human-provided Services and Software-provided Services (SpS) in the domain of software development. Individual people offer their expertise as requirements engineer, tester, documenter, end user trainer, GUI designer, distributed systems architect, customer key contact, lead developer, team leader. Software-provided Services provide development-centric functionality such as source code storage, communication channels, or document versioning alongside recommend capabilities to support HpS in collaboration, coordination, and communication. The various aspects in this scenario are displayed in Figure 1.

In this example domain we encounter the following sustainability concerns (Figure 1 e): the amount of work time by all current team members is almost stable for every week. Slight deviations above and below are tolerable. Similar, the cost for the current team and potentially adding/removing team members is given for the various project phases. The float time throughout the project phases is likewise modeled as a resource that depletes depending on project delays and is (optionally) refilled at intermediate project phases.

Alice is a software integrator for a large software project where most staff is integrated via Human provided Services. In the first round of integration, a particular component (by Bob) is not delivered on the scheduled date (Figure 1 a). Alice cannot identify the reason behind the pending component without being aware of the overall situation and cannot take mitigating actions without knowing the adaptability of her environment, the high-level goals, and long-term sustainability constraints. It is, for example, difficult to assess whether Bob is overloaded and simply delays the submission of the component, or whether there was a coordination problem with respect to responsibilities, or whether there occurred a communication problem about when and where to provide the requested component (Figure 1 b).

Consequently, Alice cannot safely choose amongst various adaptation actions (Figure 1 c). Is it best to ask for the required input, assign another member with the implementation task, select the input from an already deployed backup worker, escalate the situation to her supervisor, or contact Bob's supervisor? Thus, to take the appropriate actions includes evaluating if the high-level goals (Figure 1 d) are in danger, and whether sustainability constraints (Figure 1 e) have significant impact on available adaptation actions.

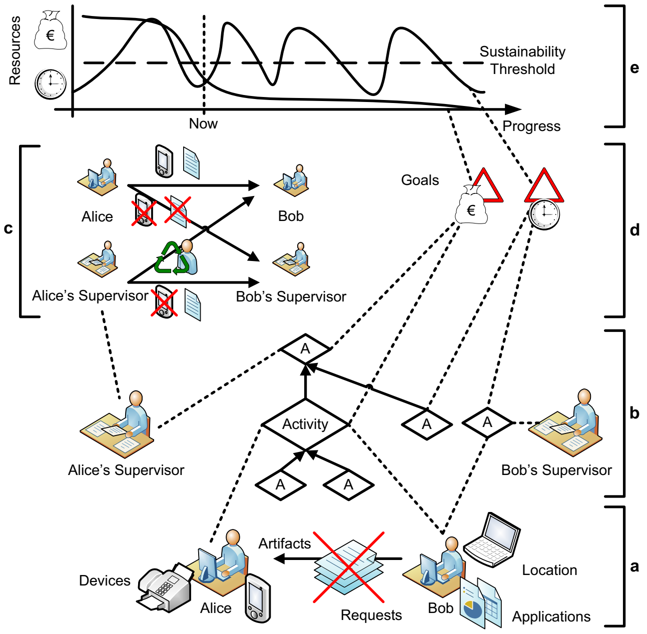

Figure 1: Sustainable Self-adaptation Scenario: letters *a* to *e* highlight the five main aspects of awareness. Dashed lines indicate relations between different aspects.

1.2 Challenges

Self-adaptation in service systems faces the following major challenges:

1. **Limited awareness on resource utilization and long-term effects** Most adaptive systems remain unaware of both the implicit effect of adaptation actions on available resources and the long-term maintainability (i.e., sustainability) of the adaptation actions carried out. Sustainable resource consumption becomes a two-fold issue. On the one hand, a system needs to remain within sustainable resource boundaries under changing environmental and internal conditions. On the other hand, adaptation actions that maintain the short-term functional and non-functional properties of a system have to consider the long term impact

on resource utilization. Such sustainable behavior requires more than extensive knowledge of the current system configuration. Individual services need to become aware of the future effect of their (collective) behavior.

In our scenario, a service giving recommendations to Alice's supervisor needs to be aware of the high-level effects on the project resources for each suggested action (Figure 1 c, d, e).

2. **Lack of understanding the interdependencies between social and technical entities** Most current self-adaptation approaches put little attention on the actual context of the client interacting with the system. With the convergence of social and technical networks [36], dynamic and complex interactions between both humans and services affect what system behavior is considered desirable. Focusing only on individual clients (or even client types) and ignoring the client's interdependencies with other system participants will not yield successful adaptation results. Self-awareness in autonomic computing requires a greater view on relevant entities and their relations. The social structure yields great influence on required service capabilities and on the possible adaptation that can be performed on services. Groups that exhibit great trust amongst members, for example, want to collaborate more freely and unstructured than groups that follow a rigid organizational structure.

   In our example, any adaptation plan to mitigate the missing component has to consider the organizational structure of Alice, Bob, and their respective supervisors (Figure 1 b, c).

3. **Limited, local information view in a heterogeneous, large-scale environment** In large-scale service systems, each entity maintains only connections to a neighboring subset of all entities due to scale. It observes changes merely in its vicinity. Emerging phenomena that arise from the complete set of interactions cannot always be observed by all individuals. Thus, an entity applies mostly limited, local information when deciding what actions to execute next.

   In the scenario, Alice's awareness is limited her supervisor and Bob. She observes only changes concerning common work activities. Without any additional support, she hardly can perceive the global effect of her adaptation actions (Figure 1 a, b, c).

4. **Dynamic, decentralized evolution of requirements, interests, and topology** Service systems grow from entities belonging to multiple organizations. There exists no central authority that controls growth and evolution of the system. It emerges from the common goals its participants share. Changes occur as people shift their interests, as people leave and new ones join in. Technical entities cause changes to equal extent: new services arise, existing services evolve, and some services disappear. These dynamic changes require constant adaptation to keep the service system working.

   In the example, Bob and Alice belong to different organizations, with divergent interests, yet they collaborate based on a common goal. Reaching this goal, however, is subject to different constraints and thus contrasting adaptation actions potentially arise. Bob's delay of his component might be the outcome of an adaptation plan within his organization (Figure 1 a, b, d).

5. **No central authority** As service systems lack central control for adaptation, individual entities need to become aware of the best element to drive adaptation, delegate adaptation control to that element, and monitor progress of the actual distributed execution actions. In doing so, these adaptation actions can proceed in a self-organizing fashion.

   In the scenario, no central authority determines who has to carry out mitigation actions when a component is not delivered on time. Instead, Alice jointly evaluates the situation with her supervisor to determine, that the supervisor is the most suited entity to trigger and control the adaptation actions in the particular situation (Figure 1 b, c).

2. A ROADMAP TO SUSTAINABLE SELF-AWARENESS

2.1 Scientific approach overview

Self-aware systems are evolving towards improving capability of understanding situation. We propose five conceptual level of awareness in Table 1 to characterize the scope of acquiring information and the sophistication of capability to analyze this information. These two factors prescribe the extent and depth a system can adapt to, and subsequently how sustainable the respective adaptation effects are. The basic level is *Event-awareness*, followed by *Situation-awareness*, *Adaptability-awareness*, and *Goal-awareness*, ultimately reaching *Future-awareness*.

The approach to awareness is illustrated in Figure 2. For each level, there are four dimensions to categorize the research focus, namely the essential **principles**, as well as the respective **models**, **algorithms**, and **protocols**. These dimensions and corresponding key research problems are elaborated in the following subsections, highlighted by bold key words.

2.2 Event-awareness

The fundamental principle of the event-awareness level is **self-description** of entities and propagation of events among different entities. This preliminary level of awareness lays the foundation to achieve higher levels by processing the basic elements of shared information. From modeling point of view, this level at first requires a context and self model that is applicable on multiple levels of abstractions to support the use of the same model independent of system hierarchies.

Dynamic context and self modeling techniques must not establish a static, a-priori defined boundary between self and context. The boundary has to emerge dynamically during run-time depending on the scope of the entity accessing context and self data. For an individual service the neighboring entities are considered context, while the self model describes the service's internal properties. Within a composition, however, some of these neighboring services potentially become part of the self.

Level	**Definition**	**1**	**2**	**3**	**4**	**5**
Event	System collects simple events which trigger directly basic Event-Condition-Action rules. The system has no explicit knowledge of the resources needed or whether the adaptation has a long-lasting (positive) effect. In the scenario, the HpS representing Alice is aware of the missing input, Bob's role, Alice's current location, and their current communication means.		++	+	+	
Situation	Ability to perceive the status of an entity by aggregating relevant events. The system understands the implication of individual events in a greater context. In the scenario, the HpS is aware of the overall task the input is required for, the roles of all involved persons (supervisors, monitors, etc.) and the fact that the expected output will not be ready for some time.		++	++	+	+
Adaptability	Awareness of the possible adaptation capability of the observed entity in its environment. At this level, cooperative adaptation can be conducted spontaneously based on the knowledge of adaptability. In the scenario, the HpS becomes aware of the possibilities to control the involved entities (e.g., Bob is an external member and can only be contacted but not forced to deliver output, Alice's supervisor has more adaptation control and greater decision freedom).		++	+		++
Goal	Awareness of the goals of individual entities as well as of the goal of entity groups. In service systems, a goal not only includes the desired functionality of a service (composition), but also its non-functional properties and resource constraints imposed by the environment. In the presence of conflicting goals, this level of awareness also includes the potential trade-offs between individual and group goals. In the scenario, the HpS is aware that for the task at hand the relevant high-level resource constraints are money (e.g., salary of employees and penalties for delays) and time (i.e., deadlines of the overall project).	++		+	++	+
Future	Awareness of a resource's life-cycle describing long-term utilization by the system and resource provisioning by the environment. This requires information on the probably future system state based on scheduled future events and analysis of part system events. Ultimately, this level describes systems that are able to select appropriate short-lived adaptation actions that respect long-term resource constraints and goals. In the scenario, the HpS for Alice's manager is aware that there is still some time for the delivery of the component; and that the following week there members are available that have the required skills to engineer the component. Thus, sustainability constraints on work time and costs are satisfied.	++		+	+	+

Table 1: Levels of Awareness and their (strong ++, medium +, or slight) relation to challenges 1 to 5.

Data filtering and dissemination are provided by this level as infrastructural service to higher-levels. The desirable approach should be independent to the semantics of disseminated data, so that various information from basic events to goal and resource models can be treated in the same manner. A lightweight data plane exploiting self-organized algorithms and mechanisms to data storage and querying will fit this vision. In contrast to Knowledge Networks [7] specific focus needs to be on the management of both models and data.

The presence of Human-provided Services has a significant impact on the event-awareness models, algorithms, and mechanisms. The core concern is the modeling and tracing of the relevant relations and compositions of software and human elements.

2.3 Situation-awareness

The situation-awareness level combines principles and techniques to establish and reason on higher-level and global-level situations. The context and self model is advanced to specify high-level, even global properties of the complete service system, that includes complex relationships between human and service entities. **Activity theory** lays the fundamental principle of this level.

Fuzzy models and granular information are prerequisites to describe the probability of having detected a particular high-level situation. Such situations are not necessarily static but potentially consist of recurring (lower-level) situations that form a **pattern**. Meta-models will be presented for controlling information granularity and information fuzziness. Entities require, furthermore, **information relevance coordination** protocols to detect and characterize higher-level situations.

The most important outcome at this level is abstraction of information in order to provide input to higher-level reasoning and efficient information sharing. The techniques to express levels of detail and fuzzy future situation for the context and self model are keys at this level. Based on the model, relevance-based information retrieval mechanisms will allow entities to selectively access context and self information. The level also provides the model to describe usage patterns of recurring aspects in context, self, and resource information.

Aggregating simple events to form meaningful situations is especially hard for HpS due to their virtually limitless range of activities. A promising approach is focusing on a relevant activity subset within certain contexts.

2.4 Adaptability-awareness

The adaptability-awareness level will support entities to detect the adaptability of other entities as well as open its own interface of adaptation, so that entities will be able to select the best cooperative partners for a certain adaptation target. **Service composition theory and control theory** will guide the technologies to fulfill the goal of this level.

The available **adaptation capabilities** are self-described by explicit models. Suitable adaptation capabilities (interfaces) for all the elements will be identified. Some services may not be self-aware and able to self-adapt, and thus in general we must consider different degrees of openness and controllability.

The coordination models will be defined among the entities identified above that work at the different levels of abstractions in service systems. These coordination capabilities must be distributed and decentralized to support situations of multiple, independent compositions that compete for the same resources. Simultaneously, coordination must provide robust and reliable solutions. Algorithms to **verify entity compositions** will support the desirable adaptive behavior. **Coordination and control protocols** describe suitable mechanisms that fit the entities' willingness and ability to adapt.

Modeling suitable adaptation capabilities for HpSs is not straight forward as humans continuously learn. HpS will exhibit different adaptation possibilities in different contexts. Adaptation control is especially hard to establish between HpS elements. The context at hand determines whether an HpS can be simple replaced or whether this adaptation action requires careful handling by another human.

2.5 Goal-awareness

The goal-awareness level introduces goals which express the overall objective and purpose of the service system. Regular entity actions and adaptation actions are linked to (possibly conflicting) goals and resource utilization.

A suitable **goal model** needs to allow linking high-level goals to low-level **constraints** and adaptation actions. Partial goal satisfaction is an integral aspect since non-functional goals such as reliability or performance are not necessarily achievable in an absolute sense. The goal model needs support objective functions which can then be minimized or maximized during runtime.

In a decentralized environment, services need to monitor each other to build up probabilistic models of goal satisfaction. Basic monitoring enables to **detect conflicting goals** and resource constraints. Probabilistic models help to predict when goals or resource constraints are in danger of being violated.

Runtime goal checking determines continuously which goals are satisfiable given the available resources and sustainability thresholds. To this end, **trade-off analysis** evaluates what effect achieving one goal has on related goals. Particularly, when conflicts arise, reasoning techniques determine whether these can be resolved, respectively which trade-offs are available.

Ultimately, service entities need mechanisms to modify models at runtime and generate dynamic views for efficient update **dissemination** and integration.

In service systems, some distinct goals exist for HpS and SpS, while some goals apply to both type of services. A main challenge lies in comparing and defining trade-offs between former type of goals. Ultimately, goal-awareness can improve cooperation as HpSs can easier perceive the impact of their actions.

2.6 Future-awareness

Future-awareness introduces **sustainability** concepts. Resources are no longer seen as simple, static constraints but require modeling of their **life-cycle**. Detailed knowledge of a resource's status allows us to narrow down the possible future actions, and monitoring activities provides the underlying data to enable prediction of the future resource volume.

Probabilistic reasoning identifies probable state transitions by evaluating past system behavior. Correlation of resource life-cycles with activity patterns detected at the situation-awareness level highlight the actions and conditions that lead to resource drainage and replenishment.

Again, in a decentralized system, these models need to be shared and updated. The dependency of resource, goal, context and self models, however, results in complexity that cannot be addressed by traditional **dissemination mechanisms** alone. Adaptive identification of information significance and service interest — e.g., by self-adapting information diffusion to the specific patterns of querying activities — becomes a key aspect.

Intrinsic to the human nature, predictions of HpS will never be as accurate as those of regular software-provided services. As humans and software become increasingly entangled some sort of prediction of human behavior is essential for effective self-adaptation in service systems.

2.7 SOA for sustainable self-awareness

Ultimately, all models, algorithms, and protocols need grounding in a service-oriented architecture and platform. The architecture has to flexibly integrates the concerns of all five levels.

Orthogonal to the models and mechanisms at each level, the architecture provides different levels of abstractions. Problems must not be addressed at application or infrastructure level in isolation, but have to observe the concerns at the level of infrastructure, applications, and single services to achieve a holistic platform. Multiple platform instances must be able to cooperate properly and must also exchange sufficient data to keep the different views aligned and move sensed data and adaptation directives among the different entities.

A key characteristic is the complete decentralization, that is, the absence of any centralized controller, and the ability to adapt the granularity with which the different parts of the system work with respect to the problem of interest, the actual context, and also the urgency of producing results and corrective actions.

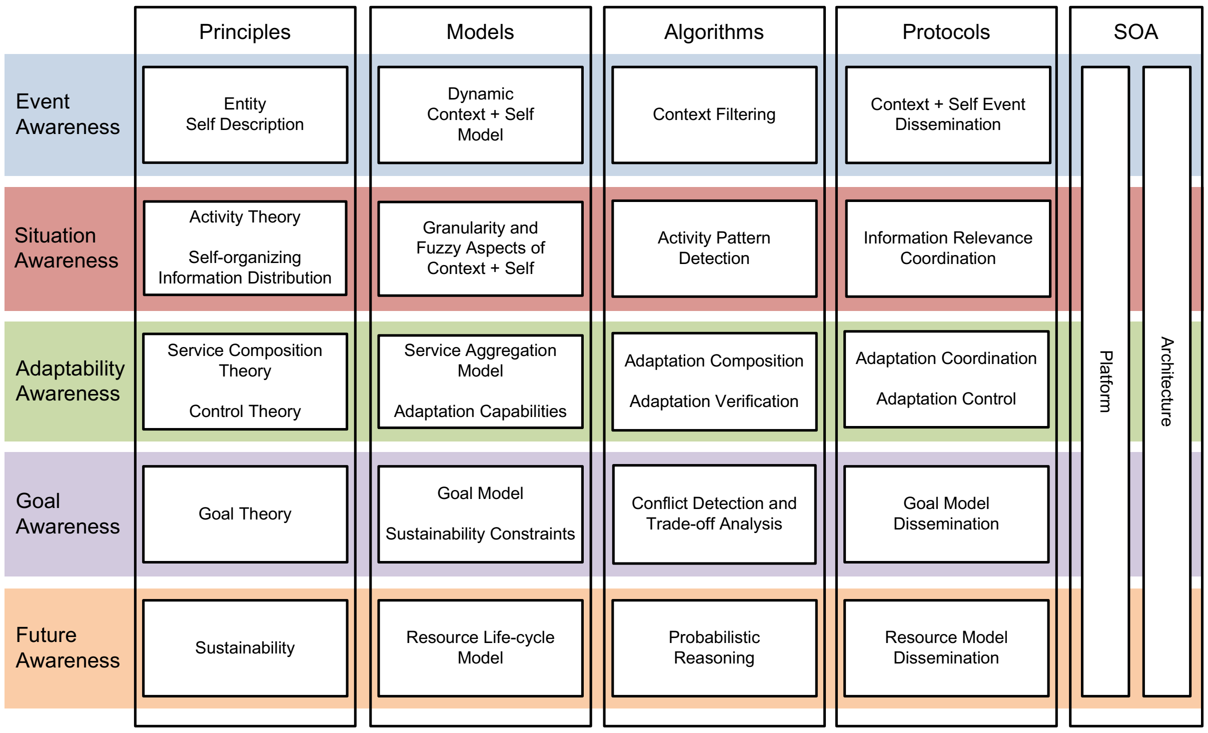

Figure 2: Roadmap towards sustainable self-aware service systems.

2.8 Maturity stages

We are able to build increasingly sustainable service systems as we incrementally materialize the levels from the road map. We can define five stages of maturity of sustainability corresponding to the levels of awareness.

- Stage 1 exhibits simple measurement (if at all) of the resource consumption caused by an entity's action. The amount of consumption is either fixed for that action or has to be explicitly measured for every invocation.
- Stage 2 allows for analysis of resource consumption depending on the underlying context.
- Stage 3 enables entities to describe the variability of resource consumption through adaptation. Also, a group can establish the entities with most efficient resource utilization.
- Stage 4 requires systems to consider trade-offs resource constraints have to be met in the presence of conflicting goals.
- Stage 5 represents the most mature sustainable service systems. Adaptation in these systems takes into account not only goal and resource trade-offs but also applies predictive knowledge to exploit temporal aspects of resource availability and adaptation action effects.

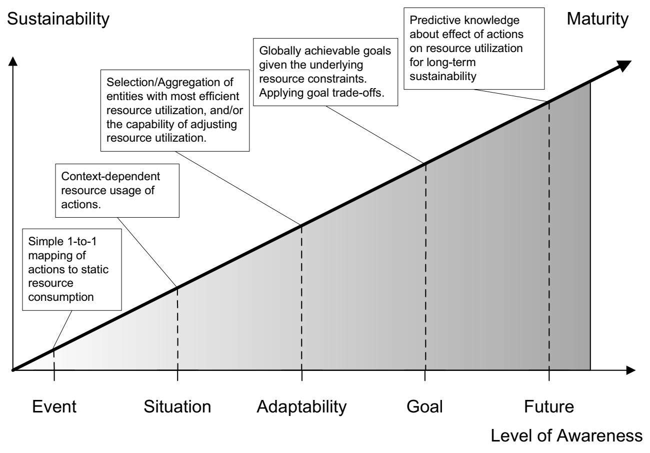

Figure 3: Maturity stages of sustainable, self-aware service systems.

3. RELATED WORK

3.1 Context and Self Modeling

Current State-of-the-Art models of context and self are limited to either side of the service system spectrum - describing either social/human properties or purely technical configurations. Context models describe either social aspects such as personal preferences or focus on service-centric aspects. Traditional models describe human-centric context such as location, devices, presence information, time, and action [16, 5, 24]. Some models include also user preferences specific to services such as cost, speed, QoS, and mobility [55]. In contrast, self-models that describe a service's embedding in its execution environment focus primarily on available service instances, their execution status, and expected termination of service instances [42, 41]. Activity-centric context [20] is essential in our proposed roadmap, but is insufficient by itself for achieving high-level awareness. Various research efforts recognize the importance of activity context for task-awareness [45], self adaptation [21, 52], or resource recommendation [46]. These approaches, however, miss out on the potential of interaction analysis. Relations between activities, resources, and humans are configured during bootstrapping and remain unchanged thereafter. Almost no model or metric describes large-scale system context. Modeling of service system characteristics that emerge at a global level has not received much attention. Analysis of social networks [44, 23, 8, 53] provides insight into the interaction structure of humans but addresses no technical components.

3.2 Goal Modeling

Current approaches to goal modeling typically focus on particular types of goals [38] and work only on a single level of abstraction [3]. To support adaptation in autonomic service systems, goals at different levels of abstraction must be connected so that they can be derived from each other, and must ultimately relate to system metrics and control actions that can influence these metrics. Goal Modeling is a well-researched aspect of requirements engineering. The most advanced approaches that also support reasoning and refinement are KAOS [40], developed at the University of Louvain, and TROPOS [3], developed at the University of Toronto. In the goal-oriented requirements engineering approach of Donzelli [19], high-level goals are refined successively to arrive at goals that can be assigned as responsibilities to specific architectural units of the system to be designed (modules or agents or even people). Goals are refined into AND-subgoals or OR-subgoals. OR-subgoals represent alternative architectural decisions for satisfying the higher-level goals. In requirements engineering, the alternatives may be used in architectural trade-off analysis to arrive at the most appropriate architecture. Both methodologies, KAOS and TROPOS, can express goals and refinements and support probabilistic reasoning for goal satisfaction. As most of the relevant related work in this areas developed during the late 90s, the focus often was on agent-based systems instead of services [19]. These methodologies provide comprehensive frameworks for reasoning about requirements. The ATAM (architectural trade-off analysis model) provides a systematic way of quantifying trade-off analysis among different architectures [34]. Goal models may be used as input to the analysis of the suitability of architectures.

3.3 Resource Modeling and Reasoning

From service provider point of view, many aspects of service management is to address resource provision [17]. Due to the complexity of real-world service systems, autonomic approaches dominate this area [49]. However, in most cases, the capabilities of resources are not explicitly modeled, but rather are implicitly embedded in the frameworks. One exception is the PLASTIC (Providing Lightweight and Adaptable Service Technology for pervasive Information and Communication) [6] platform, which has a resource model built into the conceptual model they use to model mobile service oriented applications.

Reasoning over goal and resource models has two complementary objectives: (1) to determine the best means to achieve goals or find a near-optimal solution for partial goal fulfilment in terms of resource allocation and usage [14, 40], and (2) to map high-level goals to low-level metrics and adaptation actions [11]. There is a large body of work on rule-based systems. Rules implicitly contain the mapping between high-level goals (the intention of the rule creator) and the low-level metric (the rule trigger) and the adaption action (the action of the rule). Goal oriented approaches typically use goals as constraints to evaluated the validity of the current system state and attempt to find actions that move the overall system closer to a desired state.

3.4 Large-scale Data Dissemination

Since the management of information has become a compelling need for large-scale systems, a variety of approaches for managing data and achieving high degree of knowledge awareness and self-adaptability has been proposed. In the Knowledge Plane [12], the idea is to couple the service layer with a heavyweight control plane where tools for the analysis of the knowledge are embedded together with actuating agents that exploit this knowledge. With regard to data storage, several research works [27, 33, 10] get inspiration from tuple space data model to represent contextual information in the form of tuples. Recent research focusing on sensor networks suggests exploiting a tuple-based approach to flexibly access information on sensor networks [13, 32]. Such approaches are of interest in that they consider the possibility of dynamically injecting code into the sensors for aggregating and elaborating the data within the sensor network itself, and eventually enabling services to directly access aggregated data according to their own needs. Another important issue relates to the linking and diffusion of data across networks of nodes. The data plane will embed self-organized algorithms to disseminate information in a lightweight way. Field-based Coordination [43] relies on distributed tuples injected into a network and then propagated to form field-like distributed data structures to be sensed by application components. Other approaches [15, 31, 32] are based on gossip-based algorithms to diffuse information in large-scale networks.

3.5 Autonomic Systems

Computing systems have reached a level of complexity where the human effort required to keep them operational is getting out of hand. Autonomic systems [28] aim to solve this problem and propose to embed the capabilities of the autonomic nervous system in computer applications with the aim of easing their maintenance, improving their robustness, and tackling their complexity. One of the first proposals was by IBM with a generic architecture [2] of an autonomic system with a manager and some managed resources. The manager communicates with the resources through a sensor/actuator mechanism and the decision is elaborated by means of the so-called MAPE cycle in which the manager monitors the sensors, analyzes collected data, plans corrective actions, if needed, and executes them through the actuators [35, 30].

Whereas autonomic computing focuses primarily on local, centralized components, the related domain of *autonomic communications* addresses self-adaptation aspects of distributed systems. Research in autonomic communications, however, focuses mainly on management of communication networks that remain under a single authority [18, 51].

Huebscher and McCann [29] propose an interesting survey of autonomic systems. The term, along with the idea of self-adaptive system, which are used as synonyms here, have been used for a variety of different systems [1]. For example, Garlan [22] proposes a solution at architectural-level, where the actual architecture of the system can evolve autonomously, and a more recent work by Kramer and Magee [37] exploits the analogies between robotic and self-managed systems. Autonomia [25] uses a two-layer approach, where the first contains the execution environment and the other manages the resources, AutoMate [48], which has a multi-layered architecture optimized for scalable environments such as decentralized middleware and peer-to-peer applications, and the CASCADAS approach [26], based on the concept of ACE (Autonomic Computing Element), whose characteristics, behaviour, and interactions can change at runtime based on sensed data. While the approaches above work at application level, we also have other solutions that aim to extend the middleware with autonomic capabilities, e.g., publish and subscribe approaches based on distributed dispatchers, whose reconfiguration is aimed to minimize cost metrics like network traffic [4], or peer-to-peer networks based on unstructured overlays where the goal is to search/share particular information efficiently [54].

4. CONCLUSIONS

We presented a roadmap that outlines the various contributions required to achieve sustainable, self-aware service systems. We put explicit focus on awareness in service systems that comprise human-provided services and software-provided systems. Capturing the various mixtures of humans and services is a fundamental aspect to self-adaptation. Without this knowledge, self-adaptation can never react to nor anticipate the hidden dependencies in the underlying human interaction structure.

Sustainability is the second distinctive feature of our roadmap besides the concept of human-provided services. Modeling the dependencies between resources, goals, and actions increases self-adaptation effectiveness and efficiency as the expected result can be predicted more precisely.

We acknowledge the importance of concerns such as security and privacy but deliberately did not address them in this paper for sake of clarity.

5. REFERENCES

[1] Autonomic computing. www.autonomiccomputing.org.

[2] Autonomic computing toolkit home page. www.ibm.com/developerworks/autonomic/overview.html.

[3] R. J. Anthony. Policy-based autonomic computing with integral support for self-stabilisation. *International Journal of Autonomic Computing*, 1(1):1–33, 2009.

[4] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito. Efficient publish/subscribe through a self-organizing broker overlay and its application to siena. *Comput. J.*, 50(4):444–459, 2007.

[5] R. Belotti, C. Decurtins, M. Grossniklaus, M. C. Norrie, and A. Palinginis. Modelling context for information environments. In *Ubiquitous Mobile Information and Collaboration Systems: Second CAiSE Workshop, UMICS 2004*, pages 43–56, June 2004.

[6] A. Bertolino, W. Emmerich, P. Inverardi, V. Issarny, F. Liotopoulos, and P. Plaza. PLASTIC: Providing lightweight & adaptable service technology for pervasive information & communication. In *23rd IEEE/ACM International Conference on Automated Software Engineering-Workshops, 2008. ASE Workshops 2008*, pages 65–70, 2008.

[7] N. Bicocchi, G. Castelli, M. Mamei, A. Rosi, F. Zambonelli, M. Baumgarten, and M. Mulvenna. Knowledge networks for pervasive services. In *Proceedings of the 2009 international conference on Pervasive services*, pages 103–112. ACM, 2009.

[8] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143, New York, NY, USA, 2006. ACM Press.

[9] G. Brundland et al. Our common future/world commission on environment and development. Technical report, Oxford University Press, 1987.

[10] G. Castelli, M. Mamei, and F. Zambonelli. Engineering contextual knowledge for autonomic pervasive services. *Inf. Softw. Technol.*, 50(1-2):36–50, 2008.

[11] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai. Sla decomposition: Translating service level objectives to system level thresholds. In *ICAC âĂŸ07: Proceedings of the fourth international conference on autonomic computing. IEEE Computer Society, Washington, DC*, 2007.

[12] D. D. Clark, C. Partridge, C. J. Ramming, and J. T. Wroclawski. A knowledge plane for the internet. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–10, New York, NY, USA, 2003. ACM Press.

[13] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco. Programming wireless sensor networks with the teenylimemiddleware. In *Middleware*, pages 429–449, 2007.

[14] A. Dan, C. Dumitrescu, and M. Ripeanu. Connecting client objectives with resource capabilities: an essential component for grid service managent infrastructures. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 57–64. ACM New York, NY, USA, 2004.

[15] A. Datta, S. Quarteroni, and K. Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. In *Proceedings of the International Conference on Semantics of a Networked*, Lecture Notes in Computer Science, pages 126–143. Springer, 2004.

[16] de Freitas and da Graca. Toward a domain-independent semantic model for context-aware computing. pages 10 pp.+, 2005.

[17] Q. Deng, X. Lu, L. Chen, and M. Li. A Resource Model for Large-Scale Non-Hierarchy Grid System.

Lecture Notes in Computer Science, pages 669–676, 2004.

[18] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1(2):223–259, 2006.

[19] P. Donzelli. A goal-driven and agent-based requirements engineering framework. *Requir. Eng.*, 9(1):16–39, 2004.

[20] S. Dustdar. "Caramba Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams". *Distributed Parallel Databases*, 15(1):45–66, 2004.

[21] D. Garlan, V. Poladian, B. R. Schmerl, and J. P. Sousa. Task-based self-adaptation. In *WOSS*, pages 54–57, 2004.

[22] D. Garlan and B. R. Schmerl. Model-based adaptation for self-healing systems. In D. Garlan, J. Kramer, and A. L. Wolf, editors, *WOSS*, pages 27–32. ACM, 2002.

[23] V. Gómez, A. Kaltenbrunner, and V. López. Statistical analysis of the social network and discussion threads in slashdot. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 645–654, New York, NY, USA, 2008. ACM.

[24] T. Gu, H. K. Pung, and D. Q. Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, 2005.

[25] X. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao. Autonomia: an autonomic computing environment. In *IEEE International Performance, Computing, and Communications Conference*, pages 61–68, 2003.

[26] E. Hoefig, B. Wuest, B. Benko, A. Mannella, M. Mamei, , and E. D. Nitto. On concepts for autonomic communication elements. In *International Workshop on Modelling Autonomic Communications*, 2006.

[27] J. I. Hong. The context fabric: an infrastructure for context-aware computing. *CHI '02 extended abstracts on Human factors in computing systems*, pages 554–555, 2002.

[28] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology. Technical report, IBM, 2001.

[29] M. C. Huebscher and J. A. McCann. A survey of autonomic computing - degrees, models, and applications. *ACM Comput. Surv.*, 40(3), 2008.

[30] IBM. An architectural blueprint for autonomic computing, 2005.

[31] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. pages 1–15. 2006.

[32] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[33] C. Julien and G.-C. Roman. Egospaces: facilitating rapid development of context-aware mobile applications. *Software Engineering, IEEE Transactions on*, 32(5):281–298, 2006.

[34] R. Kazman, M. Barbacci, M. Klein, S. J. Carrière, and S. G. Woods. Experience with performing architecture tradeoff analysis. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 54–63, New York, NY, USA, 1999. ACM.

[35] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[36] J. Kleinberg. The convergence of social and technological networks. *Commun. ACM*, 51(11):66–72, 2008.

[37] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In L. C. Briand and A. L. Wolf, editors, *FOSE*, pages 259–268, 2007.

[38] A. v. Lamsweerde. Building formal requirements models for reliable software. In *Ada Europe '01: Proceedings of the 6th Ade-Europe International Conference Leuven on Reliable Software Technologies*, pages 1–20, London, UK, 2001. Springer-Verlag.

[39] E. Letier and A. van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, pages 53–62, New York, NY, USA, 2004. ACM.

[40] E. Letier and A. Van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, pages 53–62. ACM New York, NY, USA, 2004.

[41] Z. Maamar, D. Benslimane, P. Thiran, C. Ghedira, S. Dustdar, and S. Sattanathan. Towards a context-based multi-type policy approach for web services composition. *Data Knowl. Eng.*, 62(2):327–351, 2007.

[42] Z. Maamar, S. Kouadri, and H. Yahyaoui. A web services composition approach based on software agents and context. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1619–1623, New York, NY, USA, 2004. ACM Press.

[43] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications: The tota approach. *ACM Trans. Softw. Eng. Methodol.*, 18(4):1–56, 2009.

[44] J. J. McAuley, L. da Fontoura Costa, and T. S. Caetano. Rich-club phenomenon across complex network hierarchies. *Applied Physics Letters*, 91(8):084103, 2007.

[45] P. Moody, D. Gruen, M. J. Muller, J. Tang, and T. P. Moran. Business Activity Patterns: A New Model for Collaborative Business Applications, 2006.

[46] K. Ning, R. Gong, S. Decker, Y. Chen, and D. O'sullivan. A context-aware resource recommendation system for business collaboration. *Int. Conf. on E-Commerce Technology and the 4th IEEE Int. Conf. on Enterprise Computing (CEC/EEE 2007).*, pages 457–460, 23-26 July 2007.

[47] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau. Ultra-Large-Scale Systems - The Software Challenge of the Future. Technical report, Software Engineering Institute, Carnegie Mellon, June 2006.

[48] M. Parashar, H. Liu, Z. Li, V. Matossian, C. Schmidt,

G. Zhang, and S. Hariri. Automate: Enabling autonomic applications on the grid. *Cluster Computing*, 9(2):161–174, 2006.

[49] C. Pautasso, T. Heinis, and G. Alonso. Autonomic resource provisioning for software business processes. *Information and Software Technology*, 49(1):65–80, 2007.

[50] D. Schall, H.-L. Truong, and S. Dustdar. Unifying human and software services in web-scale collaborations. *IEEE Internet Computing*, 12(3):62–68, May/June 2008.

[51] S. Schmid, M. Sifalakis, and D. Hutchison. Towards autonomic networks. In *Autonomic Networking*, pages 1–11, 2006.

[52] J. P. Sousa, V. Poladian, D. Garlan, and B. R. Schmerl. Capitalizing on awareness of user tasks for guiding self-adaptation. In *CAiSE Workshops (2)*, pages 83–96, 2005.

[53] S. Valverde and R. V. Solé. Self-organization and hierarchy in open source social networks. Technical report, DELIS – Dynamically Evolving, Large-Scale Information Systems, 2006.

[54] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *J. Network Syst. Manage.*, 13(2), 2005.

[55] Y. Yang, F. Mahon, M. H. Williams, and T. Pfeifer. Context-aware dynamic personalised service re-composition in a pervasive service environment. In *UIC*, pages 724–735, 2006.