

# Situation Analytics: A Foundation for a New Software Engineering Paradigm

**Carl K. Chang**, Iowa State University

*Advances in cognitive science along with modern-day smart technologies and software services that take into account our mental state will enable a software industry that is poised to meet customers' needs on the fly in new and truly individualized ways.*

**T**he more powerful, intelligent, and complex software becomes, the better the software engineering (SE) methods must be to effectively handle software system development, operation, and maintenance. In modern-day computing, software services must also be able to adapt to dynamic environments that are increasingly context aware. Context- and situation-aware computing are among the latest technological advances to allow software engineers to envision cutting-edge software services. Yet, even context-aware, service-centric development models lack the capability to continuously explore users' mental states (whether they are content or frustrated with deployed software, for example) because such states are largely hidden and continuously evolving.

One reason for this failure is the monolithic, system-dominant view of software development. Traditional views about how to structure software mostly incorporate concepts such as functional modules or objects, instead of more human-centric, life-specific abstractions that would include situations.

From a developer's perspective, a situation can be defined as having not only externally observable contexts, such as environments and behaviors, but also hidden contexts that include information gathered from users' desires. Thus, a situation presents itself as a more coherent holistic concept and helps advance the state of software development. As part of a situation-centric framework, the timely definition of individualized service requirements via runtime monitoring of users' mental states could become a plausible proposition.

This new method might also significantly shorten the service evolution cycle and help create a more responsive software industry. Service individualization is an attractive idea, but to be effective, it requires a fundamentally different approach to software development.

Moving from the old-fashioned waterfall model, to the spiral model emphasizing prototyping and risk assessment, to the most recent and widespread agile model, the development methods software engineers use reflect their aptitude for adaptation in the face of rapid technological advances and shifting marketplace realities. Increasingly, SE researchers are looking beyond traditional horizons to identify and incorporate techniques from other fields. For example, SE and knowledge engineering (SEKE) have long been studied together,<sup>1</sup> and in recent years, the search-based SE (SBSE) community flourished when evolutionary computation found its way into mainstream SE research.<sup>2</sup> As a continuation of this line of reasoning, this article explores a new situation analytics-based SE paradigm that considers situations as a computational unit in the software system.

## WORKING WITH USERS

Most software systems provide services to human users, and to do this, software engineers must first understand which services users desire. Typically, this is achieved through requirements engineering (RE) processes. In a nutshell, requirements engineers work with users to determine their needs and wants, which are eventually morphed into system goals during the RE process.<sup>3</sup> Yet, users have a tendency—and the right—to change their minds, reprioritizing their various wants and

needs throughout a system's life cycle. Given the variety of computing devices and access modalities available in our daily lives, eliciting clear user requirements is ever more challenging for software requirements engineers.

In this article, *desire* refers to the end user's motivation. The RE research community has long recognized that requirements elicited from users via interviews, questionnaires, and other traditional methods can be unreliable and volatile. Recent requirements elicitation technique research has moved beyond merely engaging human users, and instead contextual clues are collected from the operating environments that have become socially and culturally complex and largely sensor laden. Requirements captured from the resulting contextual awareness are termed *contextual requirements*, and they are only valid in certain contexts.<sup>4</sup> Even with the best

set of biological cues, such as facial expressions, gestures, gazes, and brain waves. The difficulty is in how requirements engineers can tackle the challenge of having to always keep a human in the loop, not only during the initial RE sessions, but also after system deployment. As a result, software engineers, hampered by such inadequacies, can overlook the possibility of continuously probing clients' intentions while the deployed system is being used. Yet, this is necessary after system deployment because the user's ever-changing mental states and occasionally changing physical conditions can make software features obsolete or useless. This is particularly true for software developers working in the e-health industry, for example, when designing a system to support elderly patients with disabilities such as limited mobility or dementia. In some cases, software

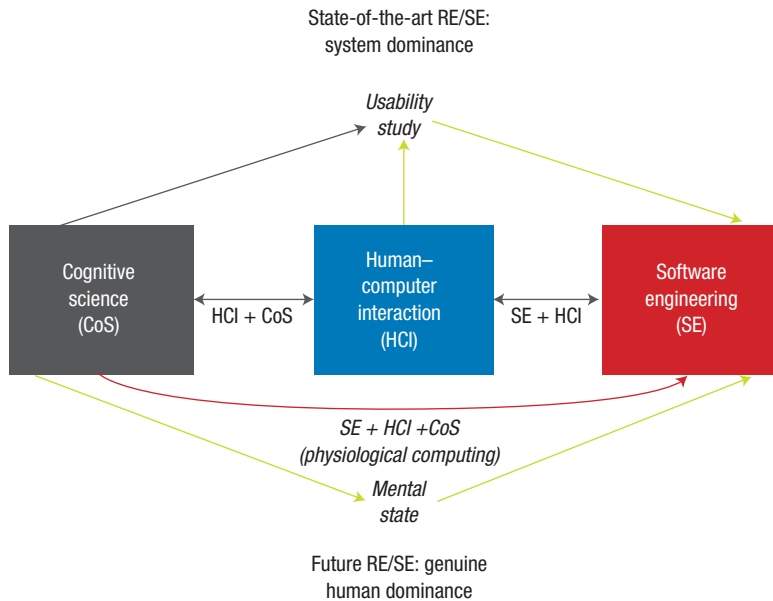
## SERVICE INDIVIDUALIZATION REQUIRES A FUNDAMENTALLY DIFFERENT APPROACH TO SOFTWARE DEVELOPMENT.

efforts of requirements engineers, it is likely that ambiguous and uncertain requirements creep into the process given the highly complex situations and environments facing users.

RE professionals have attempted but have not yet been able to effectively acquire implicit or unspoken requirements from users, although users' true desires can be obtained or inferred using cutting-edge tools and techniques that explore a rich

services could require constant adaptation to meet an individual's changing needs, such as unpredictable health conditions.

Thus, user needs and requirements must be carefully and correctly determined as smart devices and mobile apps move closer and closer to our minds and bodies. Human-computer interaction (HCI) researchers have conducted extensive studies on human factors to ensure that computer features



**FIGURE 1.** Software engineering (SE) in the human dominance era. SE professionals can employ cognitive sciences and neuroscience to fully and continuously explore the hidden mental states of their customers, even after system deployment. RE is requirements engineering.

can truly satisfy human needs and are effective for their intended purposes. Although a user's external behavior can be indicative of whether a service is useful or satisfying, a truer measure might be hidden in a user's mind.

As computing is increasingly integrated into our daily activities, modeling and understanding human mental states, including emotions, has emerged as a new research area. For example, affective computing extends HCI studies deeper into human affective processes and has attracted a unique community of researchers.<sup>5</sup> Usability studies were once somewhat biased toward a system perspective, although by definition they involve human subjects. As affective computing took hold, HCI researchers had a new tool to facilitate working with genuine human perspectives; they can factor in users' emotional states on the fly as they provide services to the users. To date, software engineers' use of usability studies to develop computer services falls short of a genuine human perspective.

With the emergence of ubiquitous and pervasive computing technologies and the abundant mobile and wearable

devices, one thing is certain: HCI-centric mental-state studies will be increasingly influential for SE researchers. State-of-the-art SE techniques fall short of effectively and continuously exploring the hidden or untold needs in a user's mind. Smart technologies and tools that have already been applied to or experimented with in our homes, medicine, and cars only allow software engineers to investigate ways to provide individualized services, but thus far personalized services have been largely achieved via parameterization. As of today, popular parameterized services are simply based on a collective profile of a class of users; they are not genuinely and dynamically tailored to specific individuals.

Humans are complex, with ever-changing mental states, so how can we find out what is really hidden in a person's mind? Modern technologies provide many potential answers. Adventurous computer scientists have already delved into several notable interdisciplinary studies to bridge the gap between computer science and other physiological or psychological research fields. For example, recent work by

physiological computing researchers explored the use of brain waves in assessing cognitive workload.<sup>6</sup> However, even as they are exposed to such technological advances, SE researchers have yet to explore the possible uses of multidisciplinary cutting-edge technologies to advance their own field. Perhaps if SE researchers look to cognitive sciences and neuroscience researchers to fully and continuously explore the hidden mental states of their customers, they will have a better chance to deliver software services that, to a large extent, satisfy the users' needs.

The possibility of runtime software adaptation based on real-time observations of human users' internal states will be a genuine game changer in SE research (see Figure 1). Yet the latest dynamic adaptation techniques are aimed at supporting system developers, not end users. Traditional methods, such as examining error logs or requirements monitoring to detect changes in contexts or users' intentions, will not afford the new generation of SE professionals an opportunity to offer truly individualized software service.

## AWARE COMPUTING

Bill Schilit and Marvin Theimer first proposed the concept of context-aware computing in 1994.<sup>7</sup> Since then, numerous studies on context-aware computing have gradually expanded to cover situation awareness. The area has been further enriched by studies exploring more human-centric dimensions such as preferences, attitudes, and emotions. Context-aware applications are popping up in an ever-expanding array of human endeavors such as disaster management, education in social skills, and tracking human mental well-being.<sup>8</sup>

Today, people constantly cope with everyday life situations with the support of software services. Thus, software engineers working with customers to capture requirements in a development process designed to ultimately satisfy situational needs must place an emphasis on situation awareness. Of course, situations have been studied for decades in areas such as logic, philosophy, psychology, business, artificial intelligence, and computer science.<sup>9</sup> What we require now, however, is an effective computational model for situations and intentions.

In the computer literature, situation awareness is defined on top of context awareness. For example, Stephen Yau and his colleagues defined situation in a declarative manner: “a situation can be an atomic situation, a logical composite situation or a temporal situation.”<sup>10</sup> They gave each situation a Boolean value and considered each situation a precondition to trigger a service. Duckki Lee and Sumi Helal defined a situation as “a triplet of a user’s activities, a set of devices’ actions and contexts over a period of time,” with the goal of assessing user behavior response (compliance) in persuasive systems.<sup>11</sup> Earlier, Mica Endsley tied situation awareness to a dynamic decision-making process and defined it as “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the future.”<sup>12</sup>

Clearly, none of these models comprise all the necessary elements to completely characterize a situation so that we can accurately compute or infer it. Even when all contexts present coherent profiles or identical values, human beings might still perceive the situations differently because of differing

emotional or empathic natures or their changing health states.

In an earlier work,<sup>9</sup> my colleagues and I briefly explained that situations consist of the following three contexts: environmental, behavioral, and hidden.

### Environmental contexts

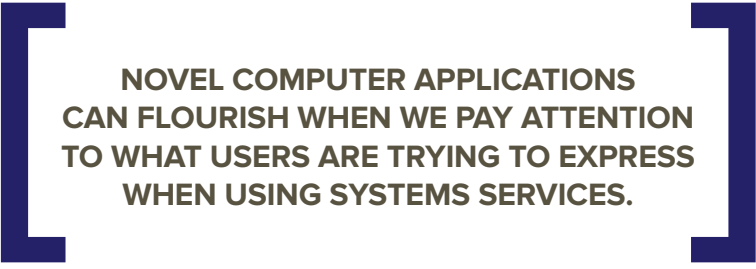
Through much improved sensing and actuating technologies, software engineers can now capture a great deal of information from or apply alterations to the user’s operational environment, such as location, direction, and velocity; sound and noise level; and temperature. Computer services based on location, for example, can be either provided or prohibited based on the availability of such environmental contexts or privacy concerns. Most early research literature focused on this class of contexts.

### Behavioral contexts

Human behavior is either a natural response to events or an effort at influencing the surrounding world. With the advent of smart technologies,

they are pleased, confused, frustrated, or irritated—we can determine whether we have achieved the goal of providing a service. After all, we can typically capture behaviors by observing a user’s actions.

Most researchers working in aware computing appear to be satisfied with this view. They argue that a great deal of understanding about a person’s situation can be determined by tracking environmental and behavioral contexts. However, even if environmental and behavioral contexts are identical, individual human beings may still perceive situations differently. Psychologists present evidence that “factors other than perceived reinforcers in the situation influence how situations are perceived and how people behave in them.”<sup>13</sup> This is because perceived situations are subjective and largely dependent on the perceiver’s mental state—a missing dimension in the prevalent definition of contexts. I believe that novel computer applications can flourish when we pay attention to both what users are trying to



**NOVEL COMPUTER APPLICATIONS  
CAN FLOURISH WHEN WE PAY ATTENTION  
TO WHAT USERS ARE TRYING TO EXPRESS  
WHEN USING SYSTEMS SERVICES.**

in personal or social-computing settings, group dynamics can be observed from afar; thus, researchers have been drawn into this fertile field of study to develop computational methods for modeling and analyzing human behaviors. The hope is that through observing users’ behaviors—whether

express when using systems services and to users’ hidden mental states—what they *really* feel.

### Hidden contexts

Recently some researchers have delved into the rich sensing domain in which hidden mental states are

further explored in addition to tracking environmental and behavioral contexts. For example, in addition to tracking physical activity and social engagement, Mark Matthews and his colleagues investigated patients' sleep patterns to better understand mental health.<sup>14</sup> Information about users' mental states can provide software developers and software service providers with rich information during the design time, runtime, and feedback period after a software service has been deployed. Unfortunately, most studies on context- and situation-aware computing have overlooked this dimension, and SE researchers have yet to realize its significance.

### FROM SITUATION TO INTENTION

Unlike logicians, philosophers, psychologists, and others, computer scientists should pursue a situation/intention model that is *computable*. Furthermore, software engineers should be able to put this computational model to use when they develop software services for human users.

Taking into consideration the three contexts I just described, we can define a situation as a three-tuple:  $\langle M, B, E \rangle$ , where  $M$  represents the user's mental state that is hidden,  $B$  represents the behavior context, and  $E$  represents the environmental context.

After we establish  $\langle M, B, E \rangle$  as the "complete" representation of situation, we must take into account another factor: time. Over time, any situational realization is only a snapshot of a person's mental state. Thus, it is more significant if we can represent a situation as  $Sit = \langle M, B, E \rangle_t$ —that is, a user perceives that a situation emerged at time  $t$ .

A computer system often needs to

observe its user over a time period to determine whether it has correctly provided a service. In SE terms, when a service feature is being executed and monitored following a scenario, its completion can be punctuated at the time when the scenario reaches its end state, or, in RE terms, when a system goal is met.

As such, we then define *human intention* as a path from the service's initiation, while annotating the path with the situations perceived by the user or reasoned by the system throughout the course of a service scenario's execution, until the desired service goal is fulfilled. This definition assumes the earlier argument that human desire is the origin of service creation by software engineers. Desire triggers the initiation of a service scenario. Situations guide the scenario to proceed in a particular order. Intention paves the scenario path toward goal completion. Thus, we can define *intention* as a temporal sequence of situations from an initial state to a completed goal state. This definition is compatible with the original concept of "future-directed intention" proposed by Michael Bratman.<sup>15</sup> (See the "From Intention to Physiological Computing" sidebar for more details.)

Mathematically, we can formulate intention as an action-laden sequence:  $\text{Intention} = \langle \text{Sit}_1, \text{Sit}_2, \dots, \text{Sit}_n \rangle$ , where  $\text{Sit}_1$  is the goal-triggering situation and  $\text{Sit}_n$  is the goal-satisfying situation.

### SITUATION ANALYTICS FOR COMPUTING

Therefore, a new computer science study, termed *situation analytics*, can help advance SE technologies. This new area of study will expand current research efforts to exploit mental states to achieve imminent objectives, such as reducing a user's frustration

with using software services. It will help make mental-state exploration an indispensable element of software-defined situations, wherein we can build software services to address user needs as situations arise and evolve at runtime when those needs aren't satisfied. Situation analytics helps ensure a human in the loop not only at design time, but also during runtime.

With such a genuine human-in-the-loop notion, we can challenge system developers to create software services that can adapt at runtime should they fail to satisfy users. Before we reach that stage, however, we need to address three key research questions:

- ▶ How do we discern human desire?
- ▶ How do we infer human intention?
- ▶ How does a service system evolve during runtime given the user's changing mental state?

### Discerning human desire

RE researchers need to expand requirements-elicitation approaches to include information obtained by directly discovering or inferring a user's hidden mental states. Obviously, human desires differ from system goals, so cutting off the human dimension after we have derived the system goals doesn't support a genuinely inclusive human-centric perspective in software development or represent the software service system's entire life cycle. Some readers might argue that it is unreasonable to ask software professionals to move into an unfamiliar, largely unexplored field of study requiring the marriage of SE, HCI, and cognitive science (see Figure 1). Nonetheless, interdisciplinary research has become the norm



## FROM INTENTION TO PHYSIOLOGICAL COMPUTING

for producing high-impact work, and many academic researchers welcome such challenges. I believe it is important that SE researchers have the conviction to progress beyond making the convenient assumptions. Real-world situations frequently deviate from our assumptions.

Are there any existing ways to determine what is truly hidden in a person's mind?

They do exist. Cutting-edge methods and tools can be potentially useful for SE research, including low-cost, non-invasive electroencephalogram (EEG) devices that are useful in studying affective states as a direct means for exploiting human minds. Many research projects are based on experiments done with such devices, although most are not (yet) related to SE.

### Inferring human intention

Internet-based companies have at least partially addressed how to infer human intention, which is a critical part of their business capabilities. E-commerce businesses make use of recommender systems, for example, in efforts to improve their bottom lines. Conceivably, we can use mathematical models such as mathematical logic, hidden Markov models (HMMs), and conditional random fields (CRFs) for intention inference. Although a lot of data has been collected, current work on inferring human intention is still limited to analyzing keystroke sequences.

Neuroscientists are trying to reliably decode the neural signals that initiate intention by capturing brain activities using implanted silicon chips in two locations of the posterior parietal cortex.<sup>16</sup> For example, such decoded "intention signals" can be used to generate motion signals

to help patients move their arms and legs. As they make even more breakthroughs in deciphering our brain signals and relating human intention to externally observable behavior, neuroscientists might be able to support SE research. I can foresee the popularity of this research direction in the emerging big data era, the US BRAIN Initiative (<http://braininitiative.nih.gov>) and the Smart and Connected

Intention, or intentionality, as I frequently mention in this article, has been extensively discussed in the philosophy, psychology, and artificial intelligence literature. For example, Bertram Malle and Joshua Knobe described a three-part model of desires and intentions.<sup>1</sup> Their work agrees with Michael Bratman's proposition on "future-directed intention,"<sup>2</sup> as both suggest that intentions require commitment, actions, and rationality in practical reasoning, whereas desire mostly reflects an attitude. (For more discussions on beliefs, desires, intentions, and plans, see the other derivative essays in Bratman's 1999 book.<sup>2</sup>)

One key concern in reviewing pertinent literature is that most requirements engineering and software engineering researchers indulge in a system-centric approach and leave the human domain too soon. Understandably, because they lack effective tools to fully exploit human intentions, such professionals oftentimes comfortably settle into the familiar system dimension without continuing to explore users' mental states during a software service system's entire life cycle. I believe now is the time to embrace research advances in human-centric disciplines such as computational psychophysiology.<sup>3</sup> To be practical, an extensive study on economics to support such a clinical approach for individualized service is necessary.<sup>4</sup>

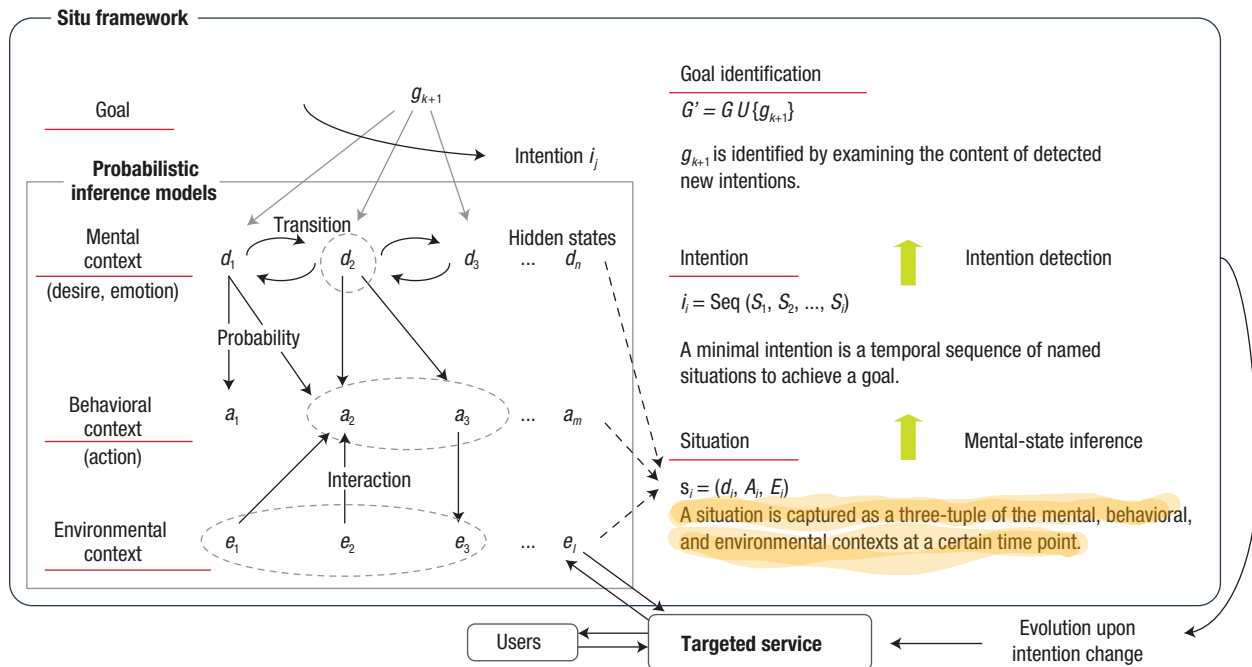
### References

1. B.F. Malle and J. Knobe, "The Distinction between Desire and Intention: A Folk-Conceptual Analysis," *Intention and Intentionality: Foundations of Social Cognition*, B.F. Malle, L.J. Moses, and D.A. Baldwin, eds., MIT Press, 2004, pp. 60–62.
2. M.E. Bratman, *Intention, Plans, and Practical Reason*, CSLI Publications, 1999.
3. M. Posner, "Introducing Computational Psychophysiology," *Advances in Computational Psychophysiology* (Science/AAAS), 2015, p. 4, supplemental material.
4. S. Fickas, "Clinical Requirements Engineering," *Proc. ACM/IEEE Int'l Conf. Software Eng.* (ICSE 05), 2005, pp. 28–34.

Health Program supported by multiple US federal agencies including the National Science Foundation and the National Institutes of Health ([www.nsf.gov/funding/pgm\\_summ.jsp?pims\\_id=504739](http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=504739)).

### System evolution at runtime

Assuming we will one day be able to effectively monitor and determine a user's mental state during runtime,



**FIGURE 2.** Situ framework. A system must be able to adapt itself during runtime upon identifying the user's emerging or changing needs with minimal human intervention.

how can software engineers make the most of such information? Ideally, we should equip a system to modify its own behavior or change its service upon seeing the user's emerging or changing needs without resorting to human intervention. With the current state of the art, this is difficult to accomplish in practice. The SE community has had limited success using evolutionary computation techniques, such as genetic programming (GP), to perform runtime bug fixes.<sup>17</sup>

In the future, however, automated service evolution triggered by changing human intention could be possible by using GP. Consider this scenario: when the system detects a user's mental state that indicates a service it renders does not satisfy a need, causes a certain degree of confusion or frustration, or simply becomes counterproductive, it will explore a few different ways to provide that service and consider alternative services. For example, it may use GP to generate several versions of the modified code in order to satisfy the use objectives, improve its service, or modify its behavior. Most

runtime adaptation techniques adjust system or environmental parameters,<sup>18</sup> but this approach will not exhaustively explore all possible combinations of real-life situations, especially in diverse populations.

Therefore, we must develop a method using new techniques to help SE professionals. This new method should at the very least be able to take the following steps:

1. Identify, via situation traceability, pertinent requirements related to the feature causing an unfavorable mental state and alter these requirements.<sup>18</sup>
2. Modify the system code on the fly<sup>17</sup> to change the system behavior in order to render modified or new services upon detection of an unfavorable user mental state.
3. Verify, by dynamic testing, if any alternative services can meet the new requirements by continuously monitoring the user's mental state.
4. If the analysis reveals a

favorable mental state, adopt the new code; otherwise, if mental state remains unfavorable, continue to explore new alternatives, and return to step 1.

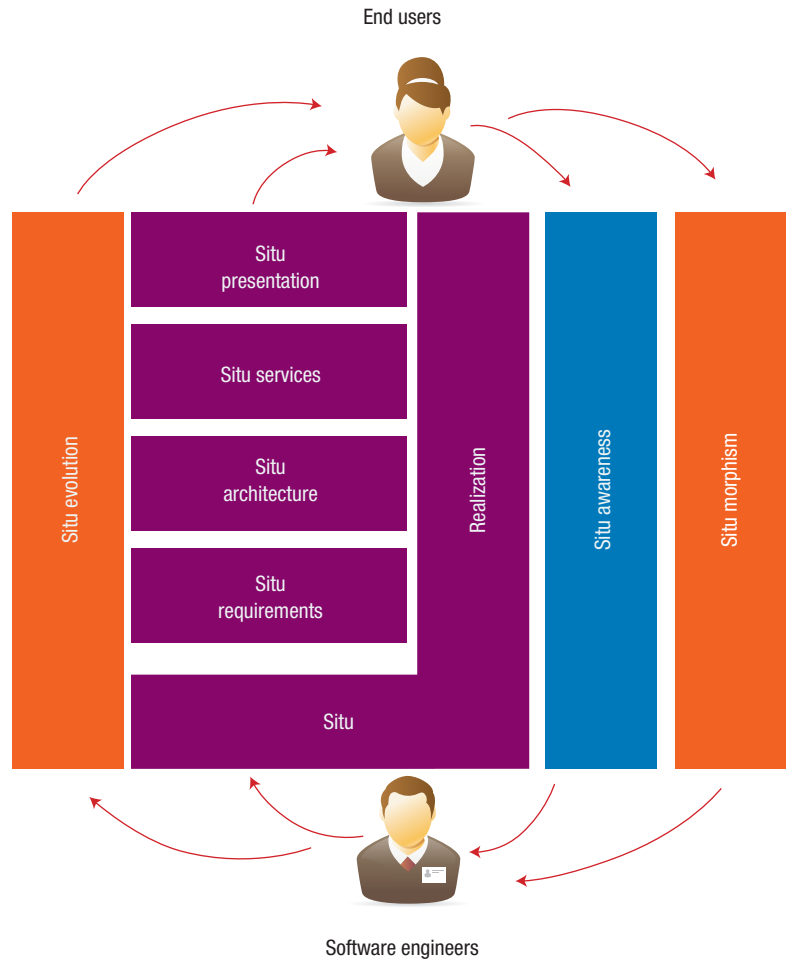
The proposed method raised further complications. Here are a few examples: Are we able to identify a complete set of requirements that need to be altered? Will inserting new code at runtime possibly introduce undesirable interference? How does continuous monitoring of a user's mental state synchronize or coincide with the GP process? How do we configure mental state as a part of the objective/fitness function? Would it be possible to feed actionable intelligence<sup>19</sup> derived from the continuous mental-state data stream to the four-step process? How do we account for the peculiarities of mission- or life-critical systems? What if runtime adaption cannot be satisfactorily achieved? Lastly, how do we address other critical factors in the human domain such as fatigue, privacy, and apathy? Clearly, individualized software

evolution introduces many significant research challenges.

## PUTTING IT TOGETHER: SITUATIONAL SOFTWARE ENGINEERING

To support runtime monitoring and service evolution, my colleagues and I presented the Situ architectural framework in an earlier work (see Figure 2).<sup>9</sup> Based on this situation-centric model, we further investigated situation programming, which helps us produce an executable situation specification running in parallel with the system as a situation-reference model;<sup>20</sup> ways to infer desire via affective computing techniques such as emotion detection;<sup>21</sup> and ways to detect user intention in support of service evolution.<sup>22</sup> The primary concern is how situations can be modeled and captured from an application domain, how they can be monitored and validated during execution, and how they might evolve after deployment. Figure 3 depicts a Situ system architecture showing how such Situ-centric components might fit together. In this context, we precede a concept or artifact with *Situ* to refer to our peculiar definition of situation as the three-tuple  $\langle M, B, E \rangle_t$ .

Because we can never enumerate all human mental states, and each application domain will have its own unique set of situations, we need domain experts to analyze each domain and collaborate with requirements engineers to develop specifications to document situations that can arise from a particular application. Software engineers can then base their work on such situation specifications to further develop the intention specifications on a scenario-by-scenario basis. Thus, a *Situ\_requirements* system document consists of both situation and intention specifications.



**FIGURE 3.** Possible system architecture based on the Situ framework. A concept or artifact preceded by *Situ* refers to definition of situation as the three-tuple  $\langle M, B, E \rangle_t$ , where *M* represents the human user's mental state, *B* represents the behavioral context, and *E* represents the environmental context.

Architectural components are then devised to support situation capturing and monitoring per *Situ\_requirements*.

Figure 3 should be interpreted as a meta-SE model that embeds a peculiar situation abstraction—the three-tuple  $\langle M, B, E \rangle_t$ —in all development and operational activities. It is a metalevel description because we do not necessarily tie it to any specific development methods, such as the agile method. Situation as an abstraction is a modular concept, and it can be implemented as functional modules, objects, or both.<sup>23</sup> Using an agile method such as Scrum, we can write user stories to capture situational scenarios. Figure 3 thus endorses many possible software development


methodologies when the entire process subscribes to a situation-centric view. Clearly, this discussion leaves out other critical crosscutting issues such as security, privacy, human dignity, and safety, as these concerns are outside the scope of this article.

*Situ\_realization* includes implementation of the situation-centric system and the verification and validation (*Situ\_V&V*) processes that include *Situ\_testing*. In such an approach, a system will undergo a series of tests to ensure that situations are constantly monitored and validated according to situation and intention specifications. Developers can employ any traditional testing techniques that support *Situ\_testing*. For example, situation



## ABOUT THE AUTHOR

**CARL K. CHANG** is a professor of computer science and human-computer interaction and the director of the Software Engineering Laboratory at Iowa State University. He also has a courtesy appointment as endowed chair professor at National Central University, Taiwan. His research interests include requirements engineering, software engineering, and smart and connected health. Chang received a PhD in computer science from Northwestern University. He is a Fellow of IEEE and the American Association for the Advancement of Science, a former president of the IEEE Computer Society, and a former editor in chief of *Computer*. Contact him at [chang@iastate.edu](mailto:chang@iastate.edu).

roadmap, based on the notion of situation analytics, to lead us into a new SE paradigm. Admittedly, many more questions need to be answered. We are at the starting line of a long race. 

transition graphs, a possible byproduct of the *Situ\_requirements* process, can support a peculiar branch-testing technique, and the intention specifications can serve as the test driver.

Moreover, any specific service presented will need to be customized to fit the user profile. For example, the system can characterize a user's facial expression at runtime, and if he or she fits the profile of an older adult, it can automatically enlarge the font size when the user retrieves an article from a digital library. After a service system has been deployed, awareness of application-specific situations will need to be enabled to reconcile the system's runtime behavior with situation and intention specifications.

During the entire course of service development, deployment, rendering, and monitoring, there will be situations found that are semantically identical or similar in terms of application-domain knowledge. The *Situ* framework regards such situations as *Situ\_morphic*. Thus, *Situ\_morphism* will serve as the guide to regulate system runtime behaviors and strategic decision making. Establishing a precise definition of *Situ\_morphism* is still largely an open research topic. Nevertheless, when a development organization has accumulated sufficient domain knowledge and artifacts, patterns (*Situ\_patterns*) will emerge as building blocks for building a family of similar service systems.

Finally, when the *Situ\_components*

bind together, service evolution will take place as needed, and situations will be captured, monitored, and analyzed. During this process, the list of situations pertinent to a given application may extend (when new intentions are detected) or collapse (when they are identified as *Situ\_morphic*). In this new situational SE paradigm, the situation becomes a first-class object in all SE tasks performed according to the situations encountered by the target users.

In this era of the Internet of Things, cutting-edge SE practices are needed—namely those that are interdisciplinary. Clearly, SE research alone will not be able to meet the steep technical challenges we discuss here. SE researchers must collaborate with experts in cognitive sciences, AI, data science, evolutionary computation, and ontology research. Continuing research in the current paradigm, while valuable, can only make small, incremental improvements. It is imperative that SE researchers work harder to provide users with genuinely useful, individualized services.

In the past, sporadic initiatives have taken small steps in this direction, and SE researchers have already begun some experimental work in this new territory. What we need now, however, are great strides and leaps. Yet we lack a unified view on this emerging horizon. This article suggests one possible

## REFERENCES

1. H. Xu, "Future Research Directions of Software Engineering and Knowledge Engineering," *Int'l J. Software Eng. and Knowledge Eng.*, vol. 25, no. 2, 2015, p. 415.
2. M. Harman, "Software Engineering Meets Evolutionary Computation," *Computer*, vol. 44, no. 10, Oct. 2011, pp. 31–39.
3. A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-Directed Requirements Acquisition," *Science of Computer Programming*, vol. 20, nos. 1–2, 1993, pp. 3–50.
4. A. Sutcliffe, S. Fickas, and M.M. Sohlberg, "Personal and Contextual Requirements Engineering," *Proc. 13th IEEE Int'l Requirements Eng. Conf. (RE 05)*, 2005, pp. 19–28.
5. R. Picard, "Affective Computing for HCI," *Proc. 8th Int'l Conf. Human-Computer Interaction: Ergonomics and User Interfaces*, 1999, pp. 829–833.
6. D. Rozado and A. Dunser, "Combining EEG with Pupillometry to Improve Cognitive Workload Detection," *Computer*, vol. 48, no. 10, 2015, pp. 18–25.
7. B.N. Schilit and M.M. Theimer, "Disseminating Active Map Information to Mobile Hosts," *IEEE Network*, vol. 8, no. 5, 1994, pp. 22–32.
8. C.K. Chang and B. Schilit, "Aware Computing," *Computer*, vol. 47, no. 4, 2014, pp. 20–21.
9. C.K. Chang et al., "Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution," *IEEE*

- Trans. Services Computing, vol. 2, no. 3, 2009, pp. 261–275.
10. S.S. Yau et al., “Specification, Decomposition and Agent Synthesis for Situation-Aware Service-Based Systems,” *J. Systems and Software*, vol. 81, no. 10, 2008, pp. 1663–1680.
  11. D. Lee and S. Helal, “From Activity Recognition to Situation Recognition,” *Inclusive Society: Health and Wellbeing in the Community, and Care at Home*, J. Biswas et al., eds., LNCS 7910, Springer, 2013, pp. 245–251.
  12. M.R. Endsley, “Toward a Theory of Situation Awareness in Dynamic Systems,” *Human Factors*, vol. 37, no. 1, 1995, pp. 32–64.
  13. B.M. Champagne and L.A. Pervin, “The Relation of Perceived Situation Similarity to Perceived Behavior Similarity: Implications for Social Learning Theory,” *European J. Personality*, vol. 1, no. 2, 1987, pp. 79–91.
  14. M. Matthews, S. Abdullah, and G. Gay, “Tracking Mental Well-Being: Balancing Rich Sensing and Patient Needs,” *Computer*, vol. 47, no. 4, 2014, pp. 36–43.
  15. M.E. Bratman, *Intention, Plans, and Practical Reason*, CSLI Publications, 1999.
  16. J.A. Pruszyński and J. Diedrichsen, “Reading the Mind to Move the Body,” *Science*, vol. 348, no. 6237, May 2015, pp. 860–861.
  17. C. Le Goues et al., “GenProg: A Generic Method for Automatic Software Repair,” *IEEE Trans. Software Eng.*, vol. 38, no. 1, 2012, pp. 54–72.
  18. J. Whittle et al., “Relax: Incorporating Uncertainty into the Specification of Self-Adaptive System,” *Proc. 17th IEEE Int’l Requirements Eng. Conf. (RE 09)*, 2009, pp. 79–88.
  19. C.E. Otero and A. Peter, “Research Directions for Engineering Big Data Analytics Software,” *IEEE Intelligent Systems*, vol. 30, no. 1, 2015, pp. 13–19.
  20. H. Ming, “Situf: A Domain Specific Language and a First Step Towards the Realization of Situ Framework,” PhD dissertation, Dept. Computer Science, Iowa State Univ., 2012.
  21. J. Dong, C.K. Chang, and H.-I. Yang, “Identifying Factors for Human Desire Inference in Smart Home Environments,” *Proc. Int’l Conf. Smart Homes and Health Telematics (ICOST 13)*, 2013, pp. 230–237.
  22. H. Xie and C.K. Chang, “Detection of New Intentions from Users Using the CRF Method for Software Service Evolution in Context-Aware Environments,” *Proc. IEEE 39th Ann. Int’l Computers, Software & Applications Conf. (COMPSAC 15)*, 2015, pp. 71–76.
  23. C.K. Chang et al., “Function-Class Decomposition: A Hybrid Software Engineering Method,” *Computer*, vol. 34, no. 12, 2001, pp. 87–93.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

## Showcase Your Multimedia Content on Computing Now!

IEEE Computer Graphics and Applications seeks computer graphics-related multimedia content (videos, animations, simulations, podcasts, and so on) to feature on its homepage, [www.computer.org/cga](http://www.computer.org/cga).

If you're interested, contact us at [cga@computer.org](mailto:cga@computer.org). All content will be reviewed for relevance and quality.

**IEEE**  
**Computer Graphics**  
AND APPLICATIONS