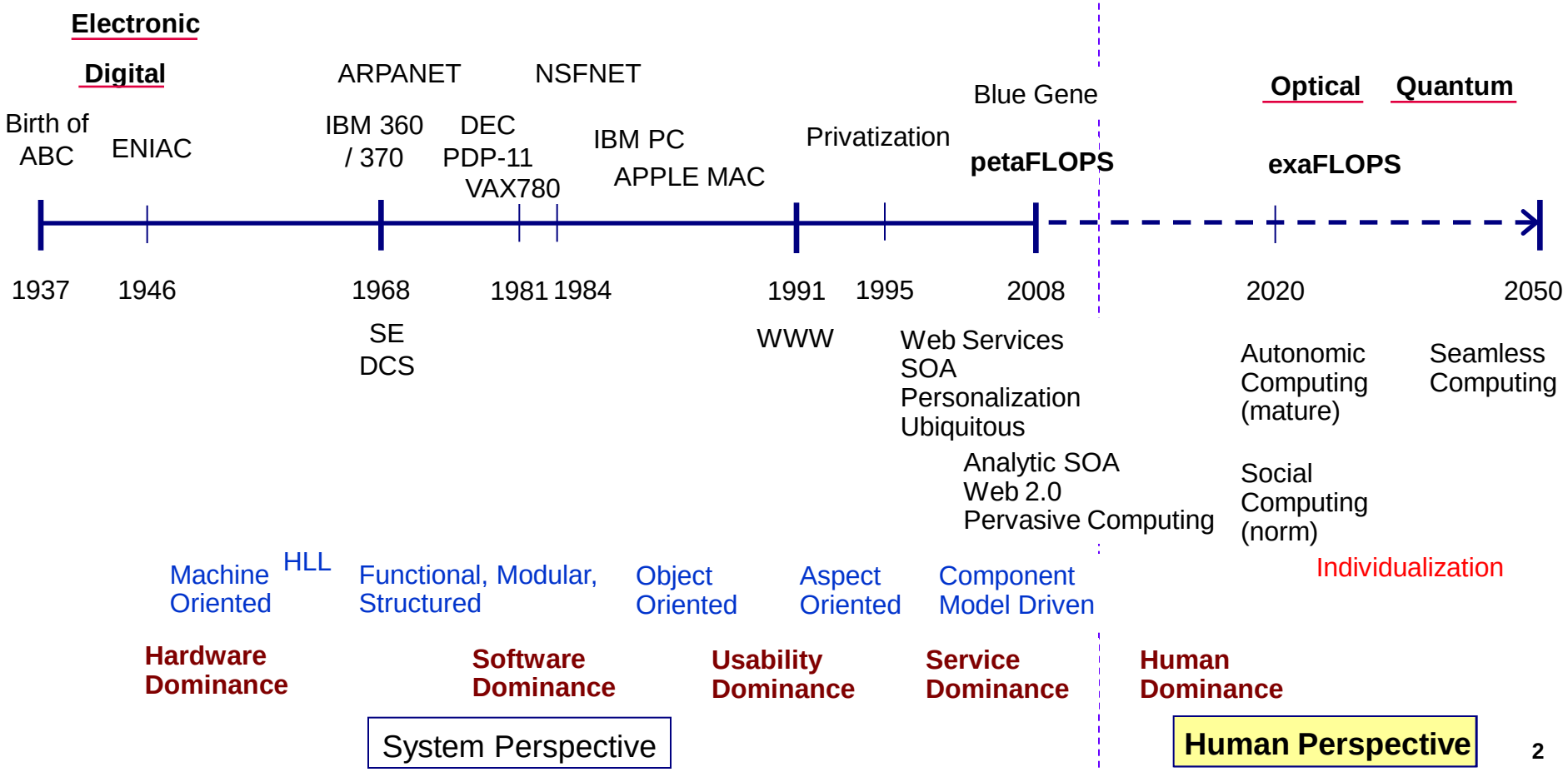
Situational Software Engineering

October 29, 2013

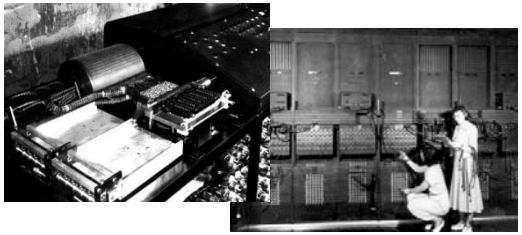
Carl K. Chang
Iowa State University

Trend Analysis of Computing

Past, Present, Future



Trend Analysis of Computing



Distant Past

Recent Past

Electronic

Digital

Birth of ABC

ENIAC

IBM 360 / 370

ARPANET

DEC PDP-11

NSFNET

VAX780

IBM PC

APPLE MAC

1937 1946 1968 1981 1984 1991

SE

DCS

WWW

HLL

Machine Oriented

Functional, Modular Structured

Object Oriented

Aspect Oriented

Hardware Dominance

Software Dominance

Usability Dominance

System Perspective

Trend Analysis of Computing

Present

Recent Future

Distant Future

Privatization

Blue Gene

petaFLOPS

Optical

exaFLOPS

Quantum

1995 2008 **2013** 2020 2050

Web Services
SOA
Personalization
Ubiquitous

Analytic SOA
Web 2.0
Pervasive Computing

Autonomic Computing (mature)

Social Computing (norm)

Seamless Computing

Aspect Oriented

Component Model Driven

Service Dominance

Personalization

Human Dominance

Individualization

Human Perspective

Six Disruptive Drivers*
shaping the future

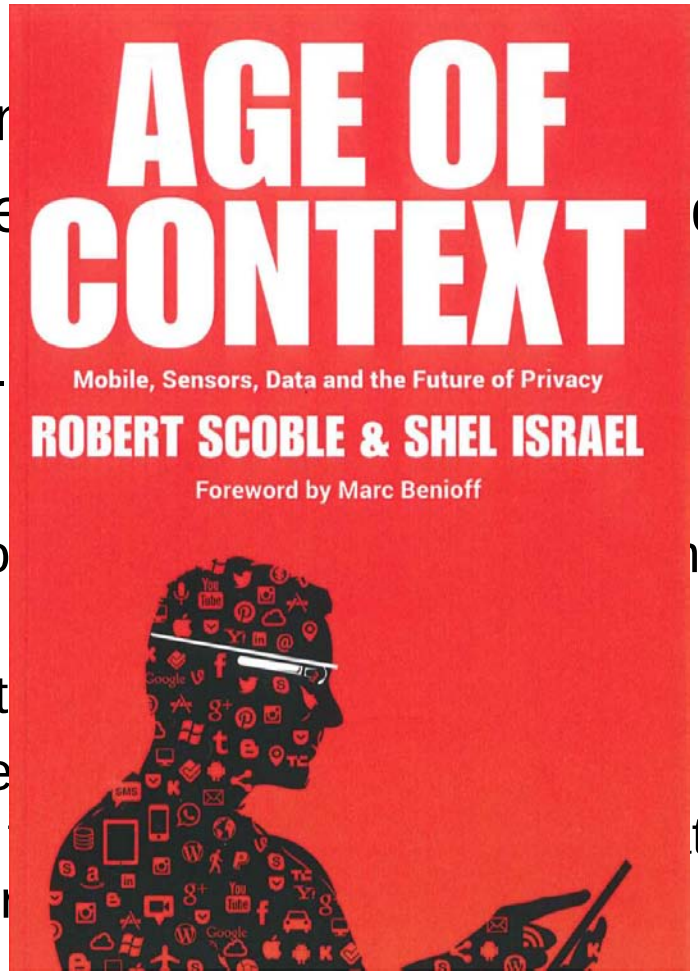
- Extreme Longevity
- Rise of a Smart Planet
- Computational World
- New Media Ecology [Tim O'Reilly]
- Super-structured Organizations
- Globally Connected World

Context

** "Workforce Preparedness": Future Work Skills 2020 Workshop. Apollo Research Institute for the Future, 2011.*

The Five Forces of Context

- *Age of Context*. Robert Scoble & Shel Israel. 2014. Patrick Brewster Press.
- Mobile, Social Media, Data, Ser
- Software engineering researche de on these five forces.
- SE succeeded to some extent -
- Falling short of:
  - Recognizing the importance of "co ent for human-in-the-loop
  - Recognizing the dominance of dat
  - Recognizing the needs to satisfy e desire/appetite of individual users tion" through parameterization can offe



One Recent Example: Google Glass

- Thad Starner, technical lead/manager and Georgia Tech for Google Glass Development
- Weight only 49 grams with computing power > 1976 Cray-1 Supercomputer cost $8.8 million
- In 1991, Starner's doctoral thesis mentioned "that on-body systems can sense the user's context."
- Today, many wearable technologies further boost the contextual technologies.
- The volatile nature of context-aware computing introduces new challenges to software engineering.

Forward of the Book: Age of Context

- Forward by Marc Benioff, Founder/Chairman/CEO of Salesforce.com:
- “In the connected world, customers are no longer just a number or account; they are unique human beings with a distinct set of needs. They have a powerful voice that they know how to use. They want a relationship on equal terms, and they expect to be at the center of your world. Companies must listen and engage and earn their trust every day.”
- SE has not been able to really treat each human being as a unique individual.

Now - Computing with a Lot of DATA

- Data warehousing, data mining, text mining, image mining, audio mining, video mining, spatial mining, relation mining, crowd mining -> When there is data, there is mining.
- The legendary “diapers and beer” connection (e.g. Wal-Mart) started the Media Gold Rush!
- As we are marching into the Web 3.0 era, we need software engineers, data engineers, knowledge engineers and ontologists.

So much data – how to cope with it?

- Examples? Twitter: Number of tweets follows the Moore's Law !
- The DIKW structure of information
- What is missing in SE research is an effective mechanism to organize contextual data for enhanced computational intelligence in software development and maintenance.
- That missing link, IMHO, is:

Situation

Context vs. Situation

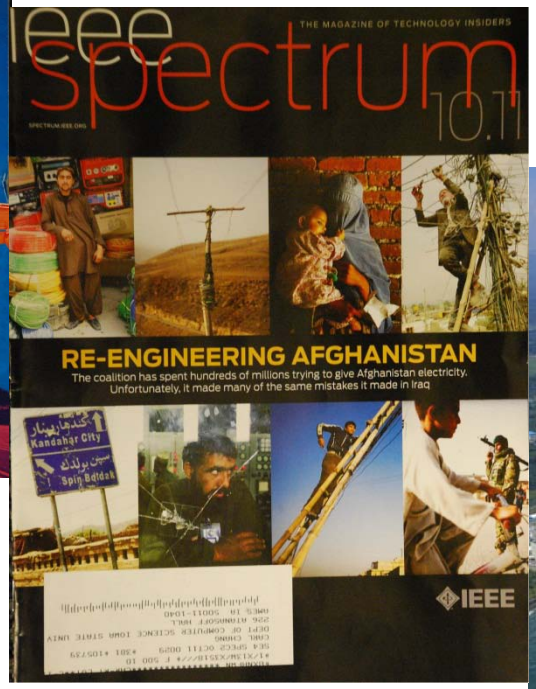
- Context-awareness: fitting humans into data
  - Data is the first-class citizen
- Situation awareness: fitting data into humans
  - Human is the first-class citizen
- Paradox: Is usability data-dominance or human-dominance?
  - Requirements elicitation
  - Usability study
  - Metrics collected
  - Data talks – Human listens -> This is not real human-centric
- Humans deal with situations not contexts.
- Our lives are primarily driven by situations, not contexts.
- Situations are -

COMPLEX

Situational Adaptability

David Autor, MIT

Sudden [Rocky]



Surprise [Sandy]

Perpetual [Muddy]

In the Era of Services/Cloud Computing

- Computing becomes more pervasive, mobile, embedded and often invisible
- Computing with massive data
- Vastly diverse and versatile software features arise – software services
- On-demand and *in situ* services are expected
- Upon the arrival of Services & Cloud Computing armed with millions of servers Software Crisis persists!
- Why so?

The Puzzled Productivity Study

- Moore's Law: "Density of transistors on integrated circuits doubles every two years"
- Brooks's Law (twisted): "Programmer's productivity is 10 NCSL per day"
- Fernando J. Corbató Law: "The number of lines of code a programmer can write in a fixed period of time is the same independent of the language used."
- Question: If software (programming) productivity follows the Moore's Law, how many NCSL a software engineer (programmer) should produce per day?

A Hypothetical Calculation

- If a programmer used HLL (e.g. FORTRAN in 1954) to produce 50,000 FLOPs of FORTRAN code per day: following the Moore's Law -> 50,000*2**(60/2)=53,687,091,200,000 FLOPs by 2014
- A software organization with 1,000 programmers will be able to produce 53,687,091,200,000,000 FLOPs = 53.687 Petaflops that can be executed by the latest China's Milkyway-2 in one second.

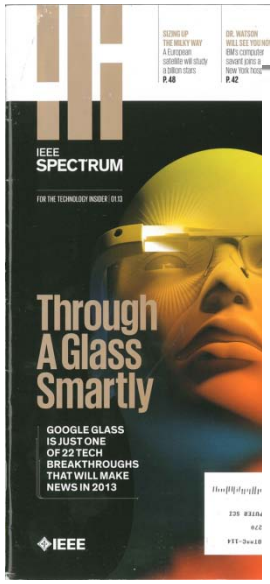
Make no mistake…

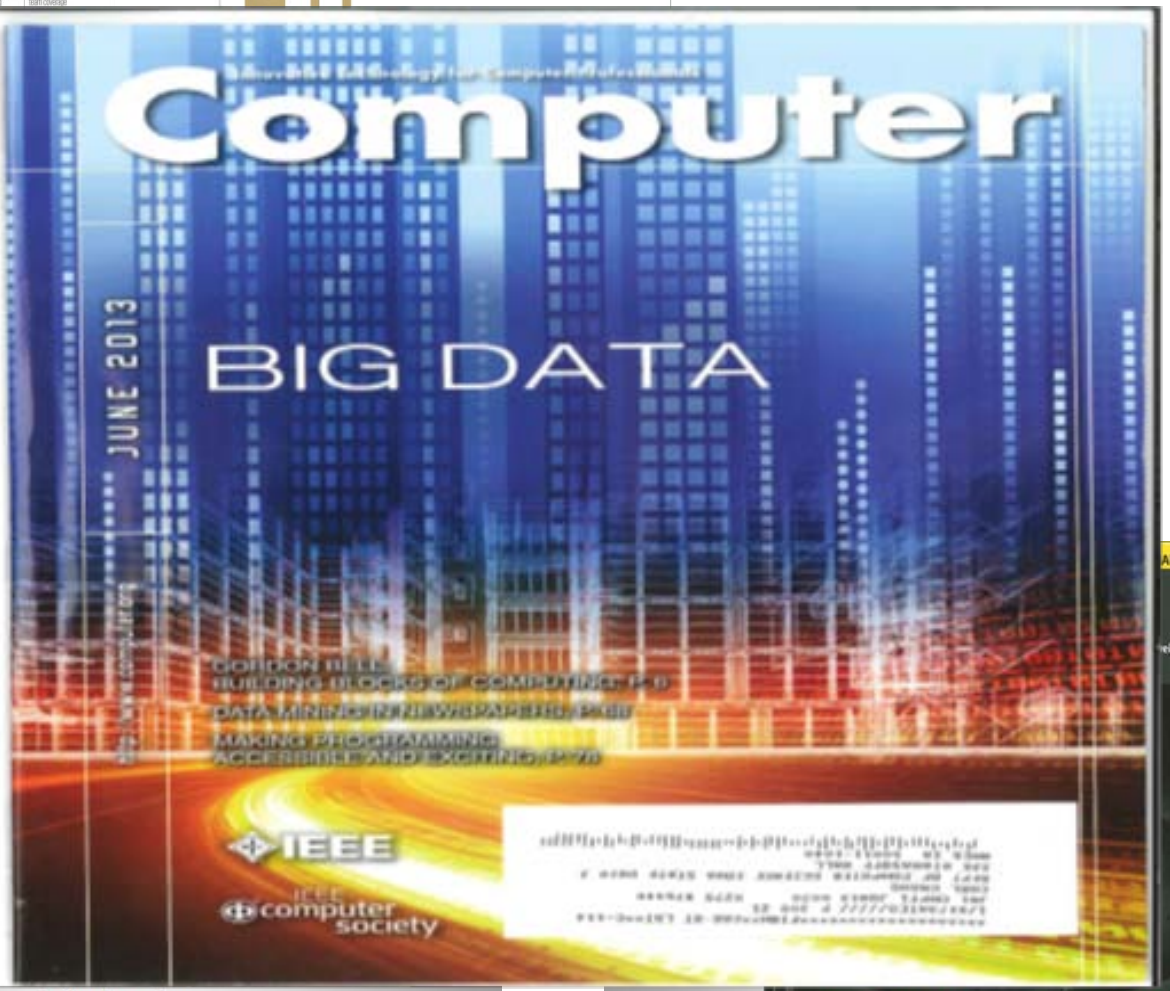
- **Software Engineering was coined in 1968 – only 45 years of history**
- **Software Engineering also has great principles – modularity and information hiding (Parnas)**

1. Manage using a phased life-cycle plan.
2. Perform continuous validation.
3. Maintain disciplined product control.
4. Use modern programming practices.
5. Maintain clear accountability for results.
6. Use better and fewer people.
7. Maintain a commitment to improve the process. [Boehm 1983]

Software Engineering had done great service to mankind, such as:

Pervasive Software Services





Driverless Car
Source: IEEE Spectrum
http://spectrum.ieee.org/podcast/at-work/innovation/selfdriving-cars-heading-our-way

Now, Answer the question again…

- Upon the arrival of Services & Cloud Computing armed with millions of servers Software Crisis persists!

WHY?

Because of False Assumptions in SE

- **We often assume that**
  - Infrastructure and resources are plenty and forever
  - Devices are unlikely to break down
  - Stable requirements are to be base-lined for builds
  - Usability Study is sufficient to capture relative human intentions
- **However, we often forget that**
  - Computing is now more volatile
  - Humans do evolve (~~and often lie~~…)
  - Usability study actually insulates developers … builds often done in an isolated cubicle (developer's comfort zone)
  - And, Cloud is already here and will stay – new technical challenges
- Thus, the product cannot meet challenges of the evolving environments and human expectations
- What can we do about it?

A Word on the Complex Human

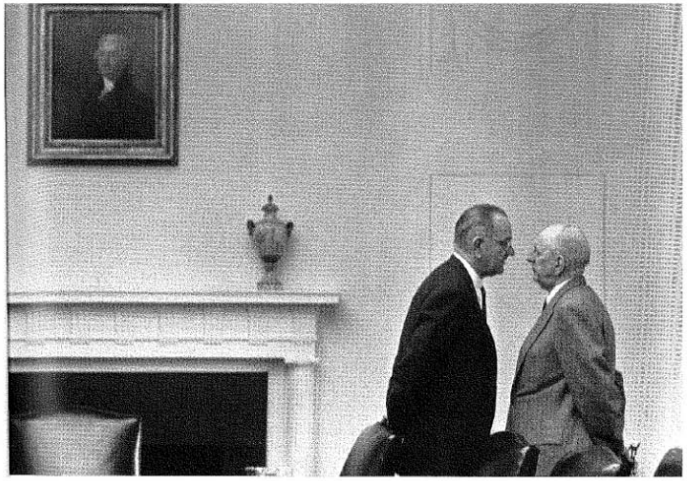
- Mentally, psychologically, physically and physiologically complex
- Humans are never perfect – not the system engineers nor the users
- Humans evolve as “situations” arise – human sensory adaptability to contextual cues from the environment is transformed into mind adaptability to perceived situations
- Extensive studies on human states: mental, emotional motivational, intentional, etc. by psychologists, cognitive scientists, physiologists, neuroscientists, sociologists, logicians, linguists, anthropologists, etc., and computer scientists (“things” must be computational!)

Human Beings Are Complex

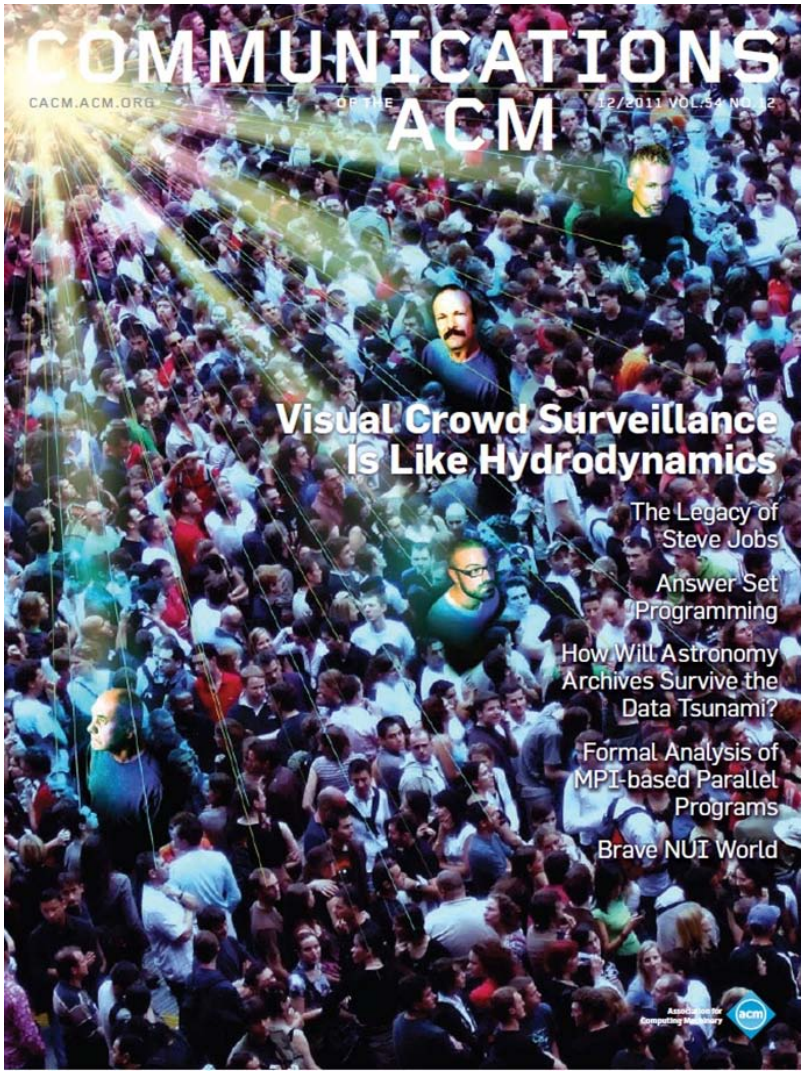
To Signal Is Human

Real-time data mining unmasks the power of imitation, kith and charisma in our face-to-face social networks

Alex (Sandy) Pentland



President Johnson vs. Senator Richard Russell (D, GA).
Professor Alex Pentland, MIT.
American Scientist, Vol. 98, Sigma Xi, 2010



Human Demands Software Evolution

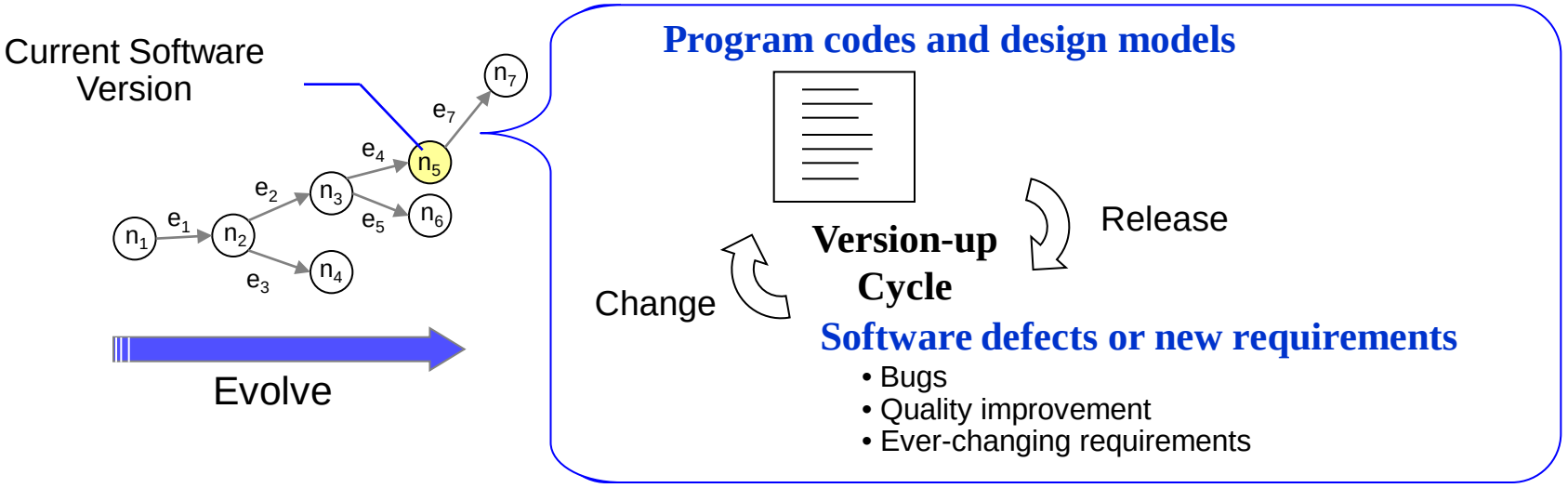
- As humans do perceive situational changes they would expect Situational Adaptability of the system
- They also often desire "*in situ*" changes
- However, software systems must be designed with the possibility of "in situ" changes or it can never be done
- The reality is that the technical challenge is huge so we must make progress a step at a time
- The first step is to gain a fundamental understanding about a feasible computational model of situations and intentions, as well as the basic set of mechanisms to enable changes

Breaking Down Tasks to Services

- There are often several ways to achieve a task
- For example: GUI Designer must decide on a preferred event sequence
- It may not best fit a particular user's comfort zone – habit, background, health condition, etc.
- Ordering the events to complete a service goal
- Listing of the parameter values (data range)
- Classifying the intended users
- Tasks are meant to meet human intentions; services are meant to satisfy human desires
- Common Practice: Altering the properties of tasks or rearranging task steps results in services to satisfy new/changing desires

Basic Concepts on Software Evolution

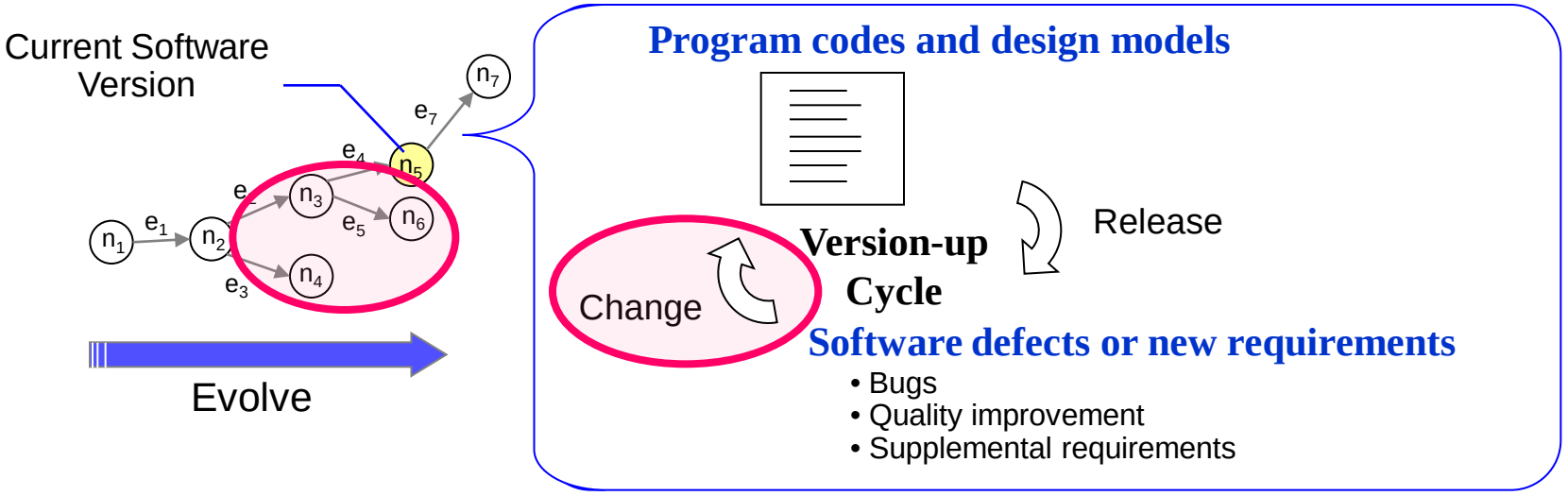
- **Software Evolution as a Version-up Cycle**
  - Most researchers have used the term as "**various changes in software systems** (e.g., software release and update)."
  - Software Evolution has a **version-up cycle** to adapt the software to user requests and environments by producing new versions.



Basic Concepts on Software Evolution (cont'd)

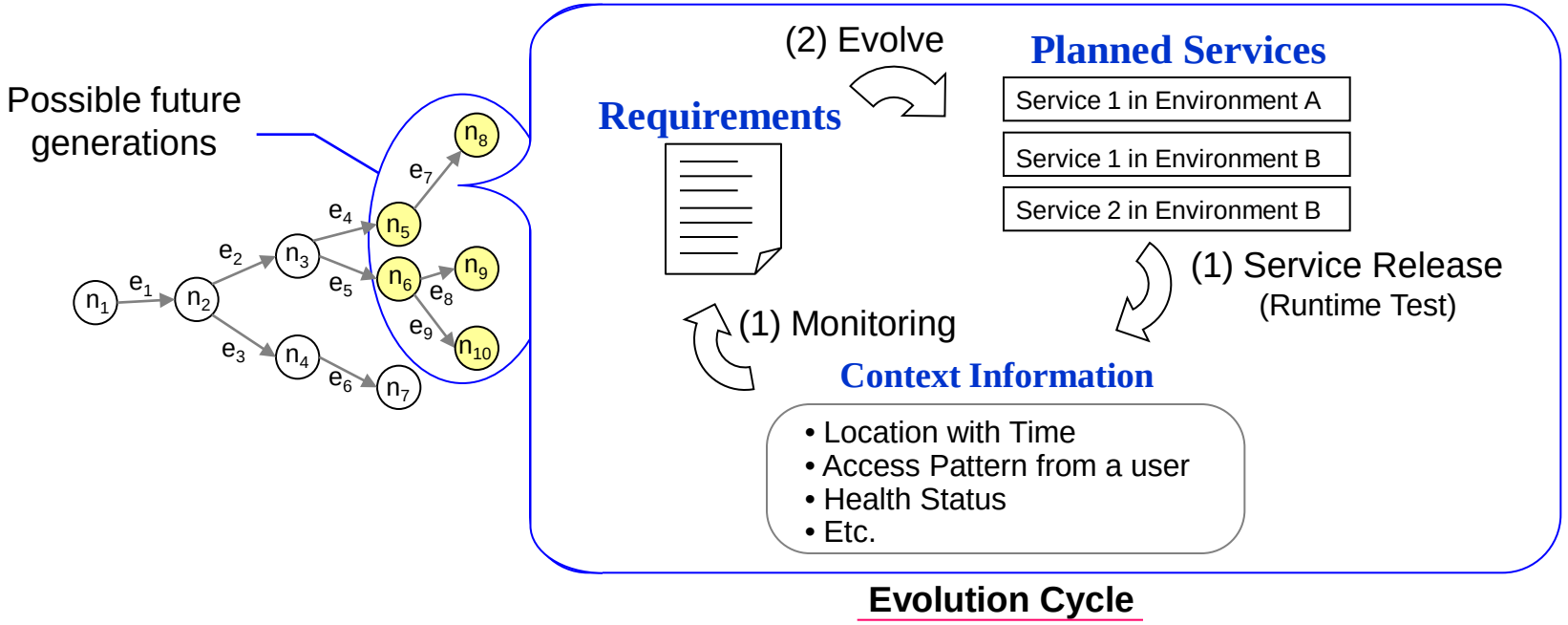
- **Software Evolution as a Version-up Cycle**
  - However this version-up cycle typically ignore…
    - **Personalization** (privileged group accommodations)
    - **Adaptation to rapid changes**
      - Individual users' situation
      - Next-generation software

  ⇨ Modern software evolution should adapt to the followings:
    - **Timely provision** of possible future generation of services
    - **Direct feedback from user contexts** during run-time



Recent Concepts on Software Evolution

- **Software Evolution @ Runtime**
  - We are seeking a new paradigm for software evolution to use **runtime feedback** **of context information**.
  - It should support instant speculation of requirement changes and propose possible future generation of services on the fly.



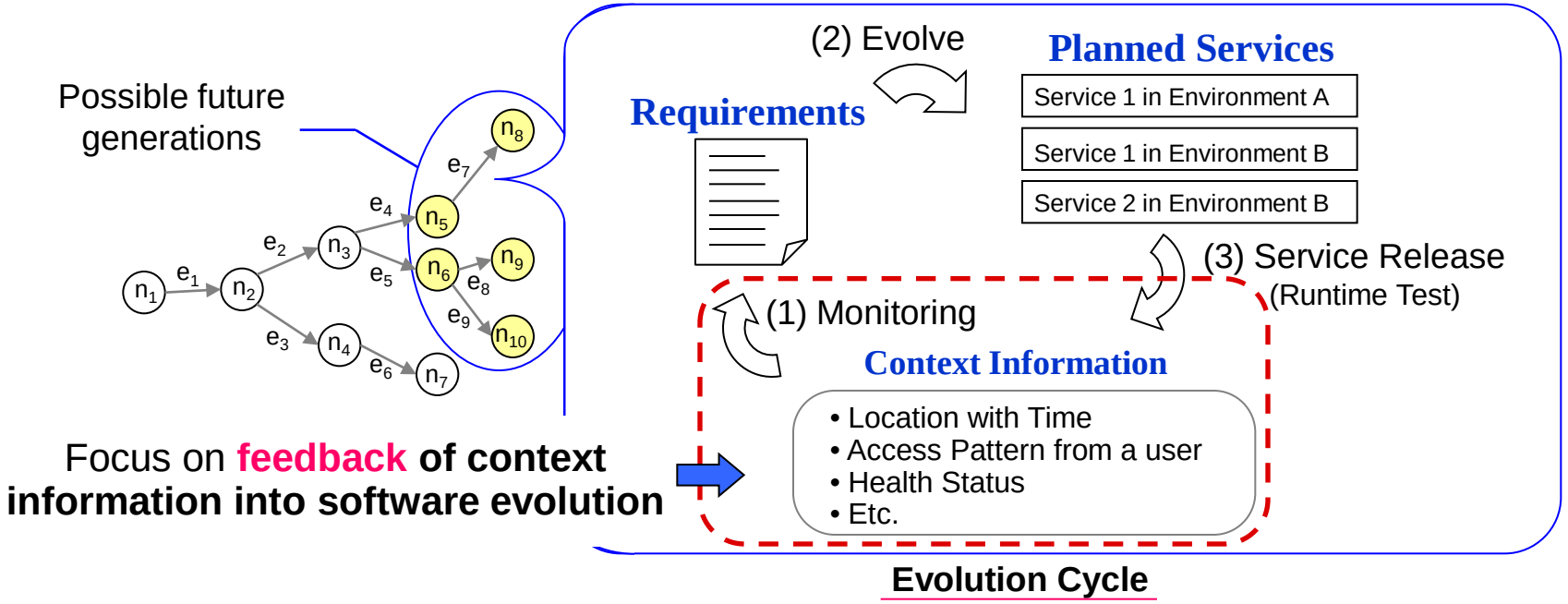
Recent Concepts on Software Evolution

- **Software Evolution @ Runtime**
  - Evolution cycle includes **runtime feedback** through three phases.

  1) **Monitoring** to observe changes of individual users' situation and intention,

  2) **Evolve** to modify business process and corresponding services,

  3) **Service Release** with runtime test.



A Paradigm Shift in Software Engineering

- My thesis: Evolution can be “rapid” if we can capture human intention change from a context-aware environment where situations and intentions are the first-class citizens in the requirements engineering and design process, and can be captured and reasoned in run-time
- As a result, a paradigm shift from the conventional software engineering to the ***Situational Software Engineering*** is necessary and perhaps imminent in order to support the fast emerging ***Situational Computing***

Exactly what is a Situation?

- **Logician's definition:**

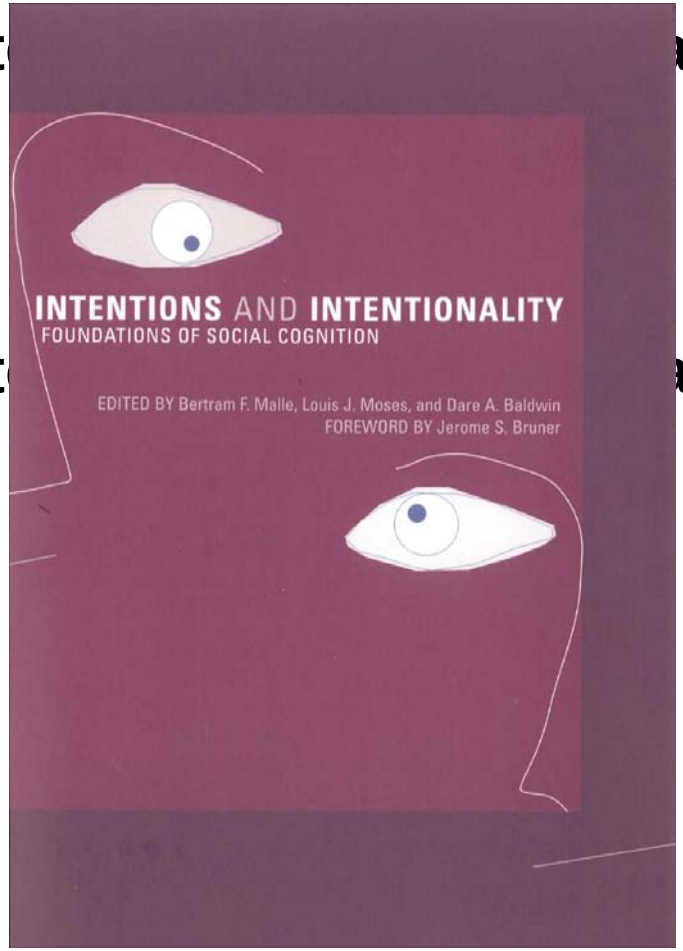
The world consists of *objects*, *properties* of objects and *relations among objects*. And there are parts of the ***world***, clearly or vaguely *recognized* in *common sense* and *human* language. These parts of the world are called *situations*. Events and episodes are situations in *time*, scenes are visually *perceived* situations, etc. [Barwise at el., 1980]

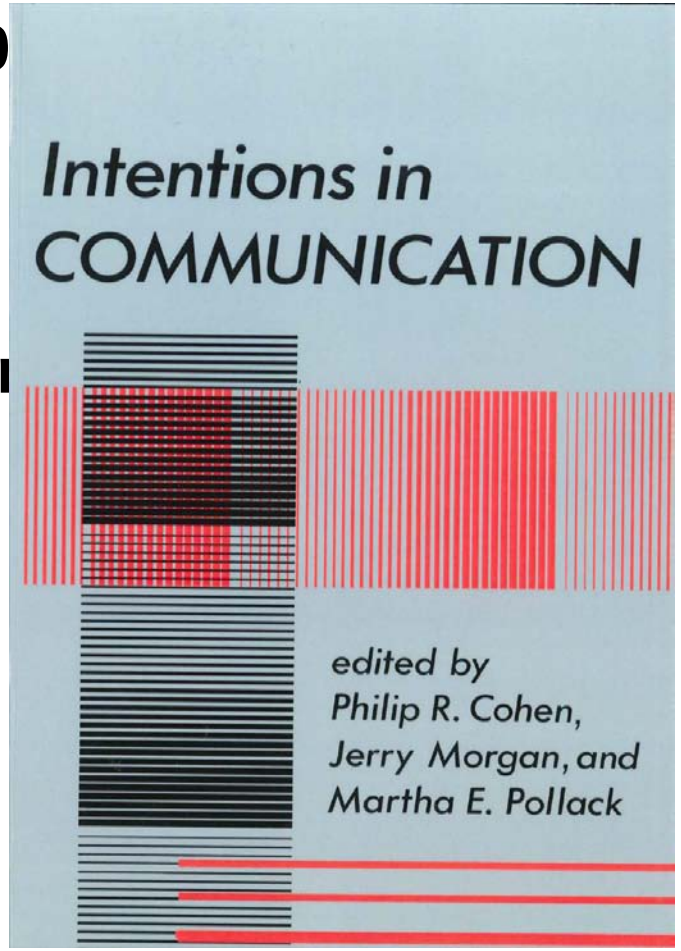
Studies on Situations

- **Philosophy [McCarthy et al., 1969] [Barwise, 1989]**
- **Mathematical logic [Barwise et al., 1989]**
- **Cognitive and psychological sciences [McCarthy, 1968] [Barwise et al., 1983]**
- **Computational linguistic [Devlin, 1991] [Devlin et al., 1996]**
- **Business communication [Devlin et al., 1996] [Devlin, 2001]**
- **Artificial intelligence [Reiter 1991] [Pinto, 1994] [McCarthy, 1995]**
- **Software engineering [Yau et al., 2008] [Mastrogiovanni et al., 2008]**

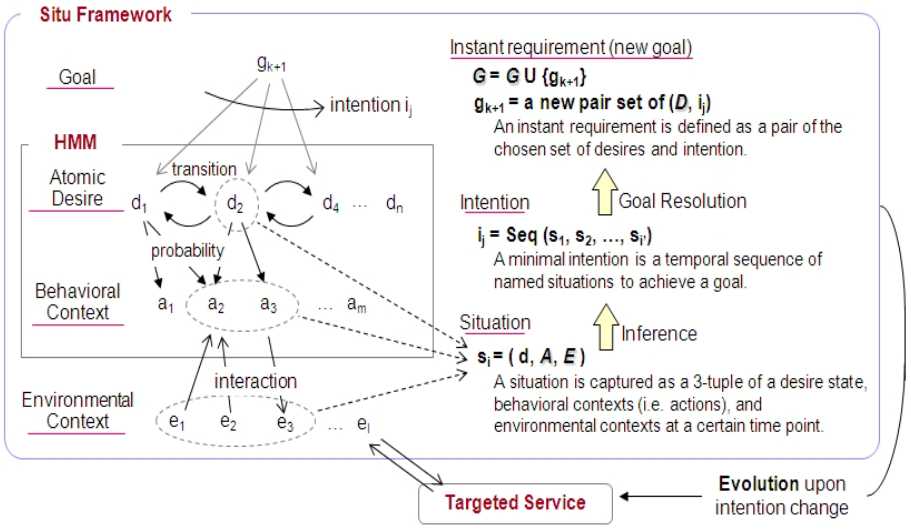
What is Intention?

- **Int** **at [Malle 20**
  - –
  - –
  - –
- **Int** **asoning [Br**
  - –
  - –
  - –





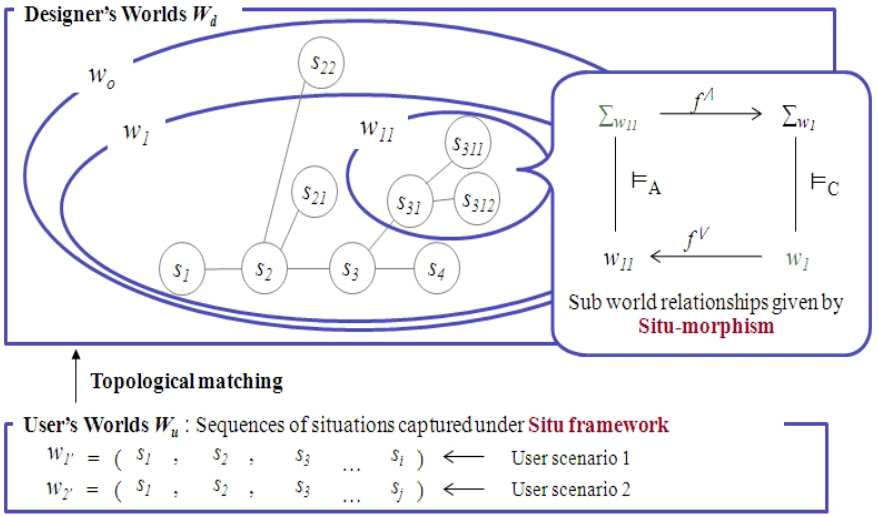
A Situation-Theoretic Human-intention Driven Approach to Runtime Software Service Evolution



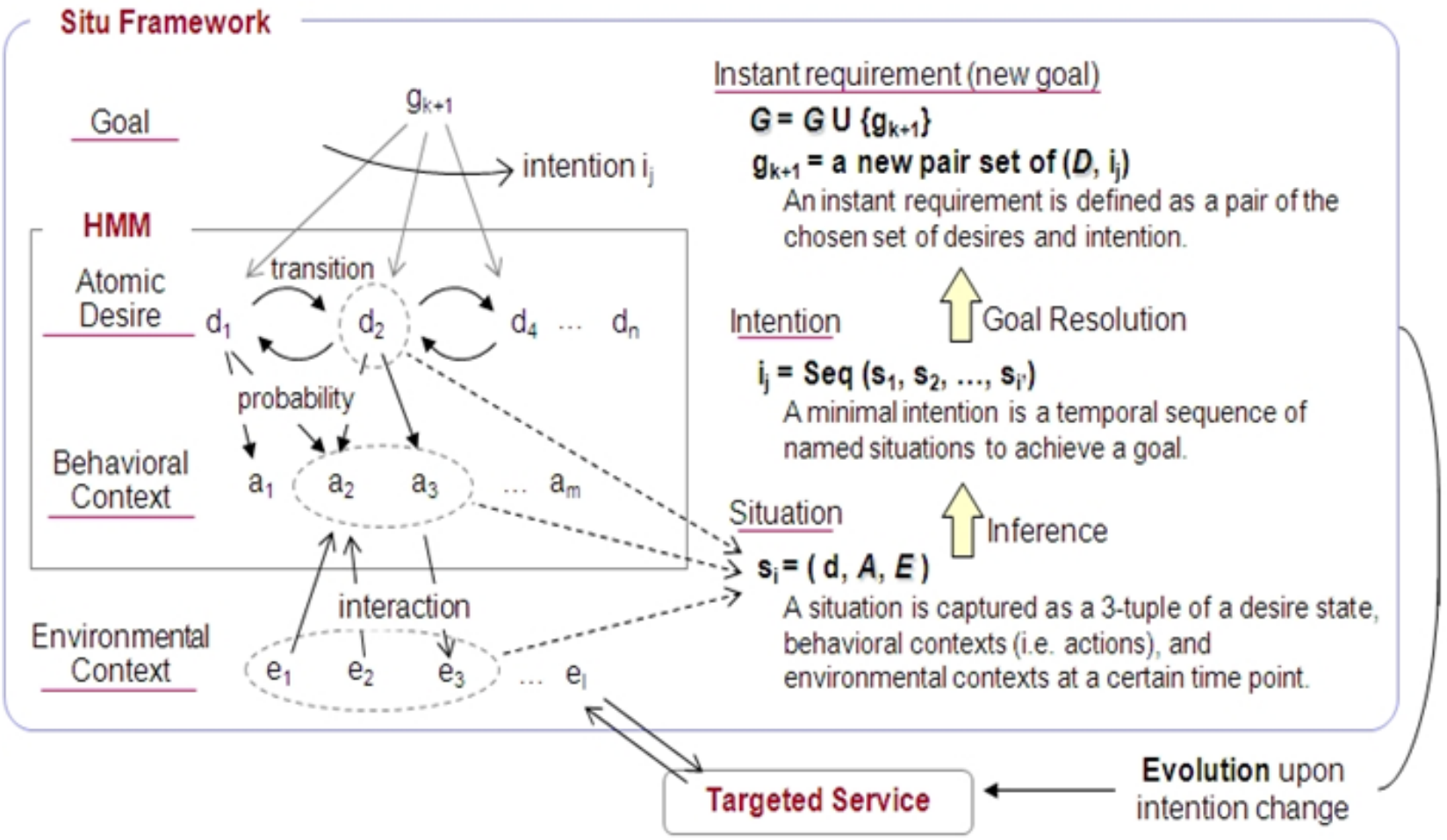
Situ Framework – A situation-theoretic human-intention driven framework in support of context-aware service evolution has been developed to realize this new paradigm [1]. Rooted in **situation theory**, *Situ* enhances upon the original propositions of situation with human internal mental states and environmental parameters as well as observable user actions to form semantically richer definitions of **situations** and **intentions** that are computationally feasible.

[1] C. K. Chang, H. Jiang, H. Ming and K. Oyama, "Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution," IEEE Transactions on Services Computing, col. 2, no. 3, 261-275, 2009.

Reasoning about Human Intention Changes – The research project aims to develop a formal computational model to monitor and reason about human intention changes under the Situ framework. Combined with the notion of **possible worlds** in Kripke semantics, which allows formal description of the relations between users' needs (user's worlds) and designers' understandings (designer's worlds) of user requirements, **Situ-morphism** provides the rules to determines whether the current implementation no longer satisfies a user's intentions (i.e., sequence of situations) and locate the part of the system that requires evolution or replacement, in order to provide a new release to satisfy the user. The traditional "retrofit" policy and practice for such feature replacement takes too long by the emerging (tomorrow's) standard. A rapid "runtime retrofit" of modified or new features that directly enhances user's experience in the field seems to be overdue.



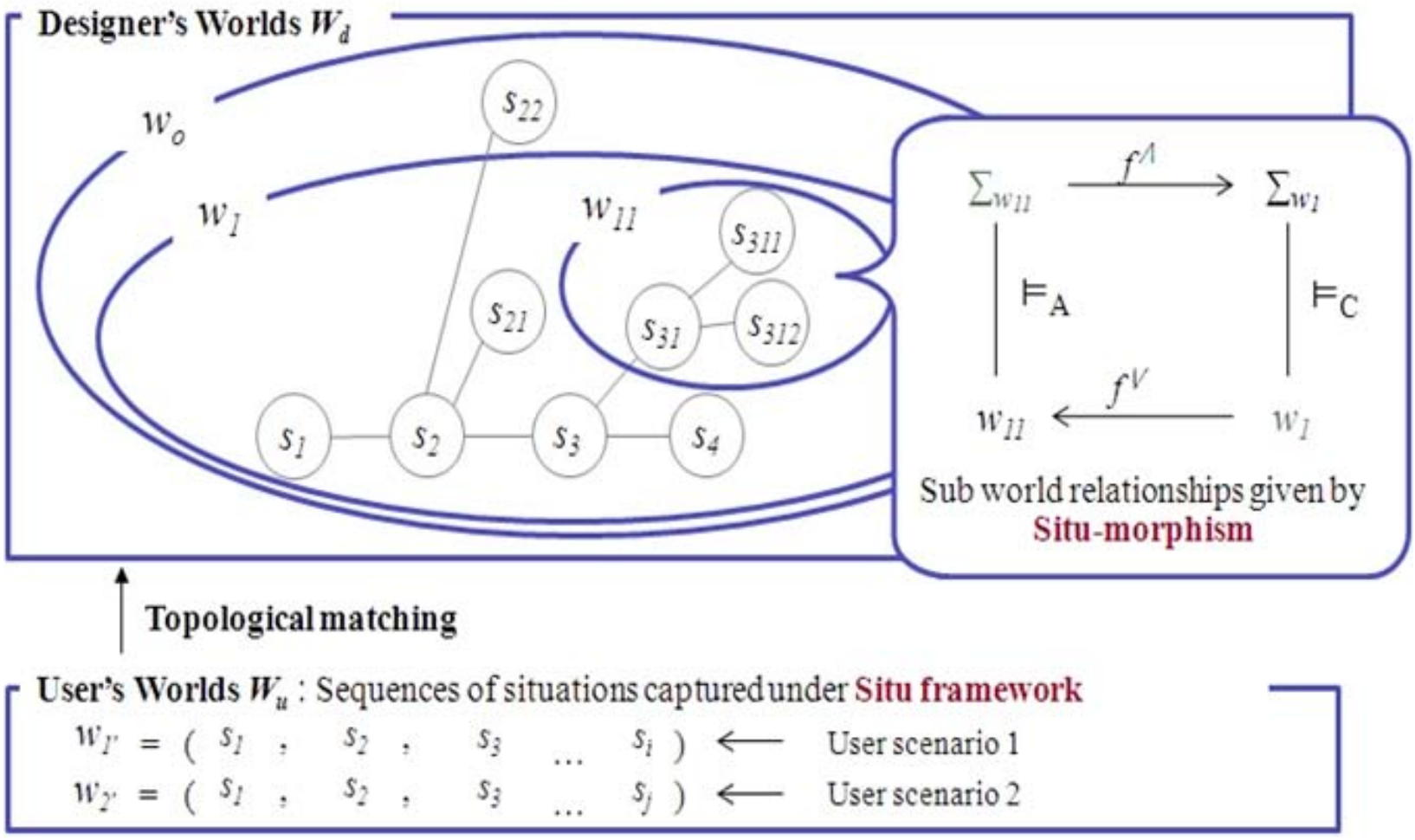
The Situ Framework



Situation-Centric Software Methodology

- Situation Engineering: A domain expert who can elicit and specify application requirements to capture situations and human intentions.
- Software Engineering: A software engineer who can build a service environment that can serve and evolve according to situation specification and is tailored to an individual (supported by “situation programming”.)
- Test Engineering: A test engineer who can ensure system integrity for initial deployment and later evolution cycles.
- Human Engineering: A human engineer who can model, capture and/or infer human mental states to support human-intention driven service environments.

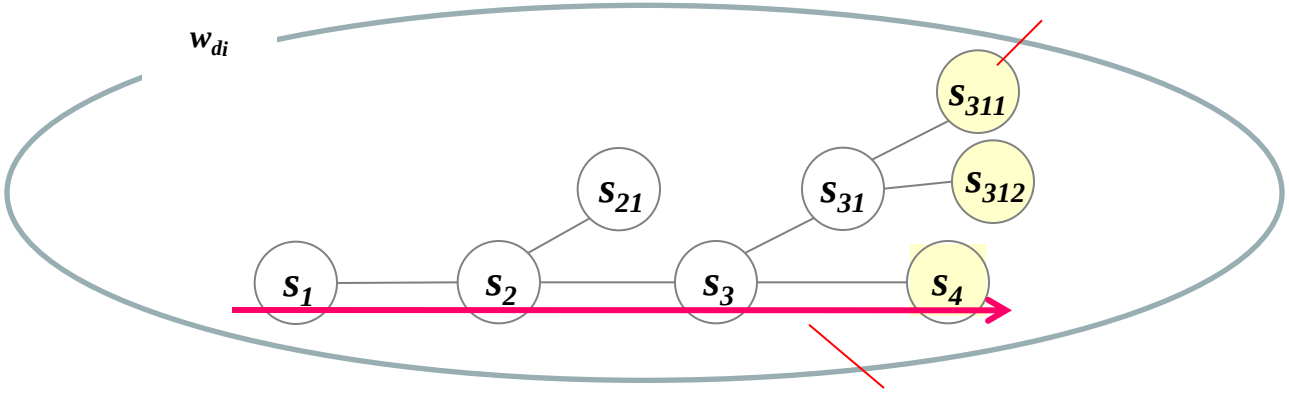
Situ-morphism



Services are to Achieve System Goals

- **There is a sub world $w_{di} \subseteq \Sigma_i$ expected by a designer *d*.**
- **$<i, \Sigma_i, \models_i>$ is a Situ-module where Σ_i is a requirement and *i* is an implementation.**
- **A⇆C is Situ-morphism representing whole and part relationship between requirements to implement a whole software service.**

A leaf nodes in w_{di} are goal for each sequence of the situations addressed in Situ framework.



Situation S_1 to S_4 should be achievable by the user who takes the actions to use a target software service.

Software Evolution @ Runtime

How to Capture or Predict Human Intention Change?

- **One possible solution: To apply DBBN (e.g. Hidden Markov Model) or CRF to infer intention change.**
  - Intentions => hidden variables (i.e. mental states are not observable)
  - Contexts => observable variables (i.e. environment)
  - Intention Change ⇔ State Change or State Emergence

How to Effect Software Evolution?

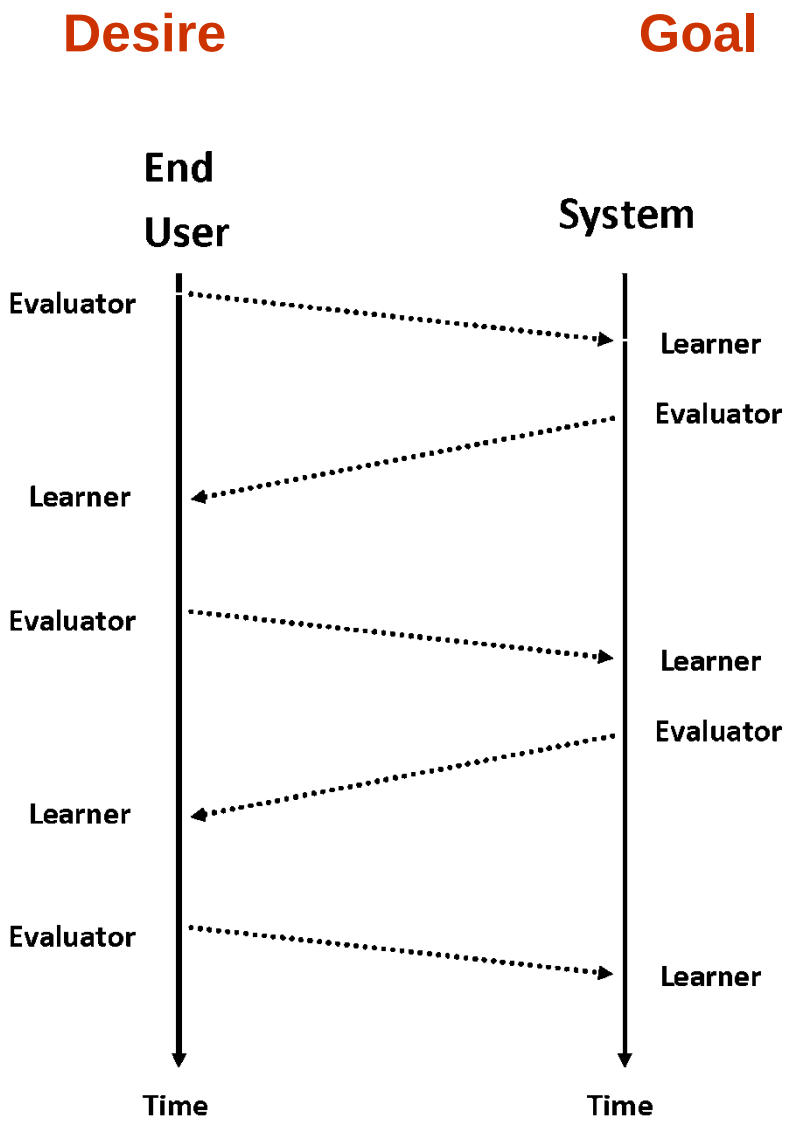
- **One possible solution: To apply Genetic Programming to evolve the system**
  - Suggested revised code segments with fitness values

In Sum: Prevailing approaches to Software Evolution

- Brute Force
- Version/Configuration Control
- Parameterization
- Reconfigurable System
- Product Line Engineering (commonality/variability)
- Restructuring / Refactoring
- Evolutionary Paradigm
- My main thesis: *We need a paradigm shift in order to have a breakthrough.*

Human/System Co-Evolution

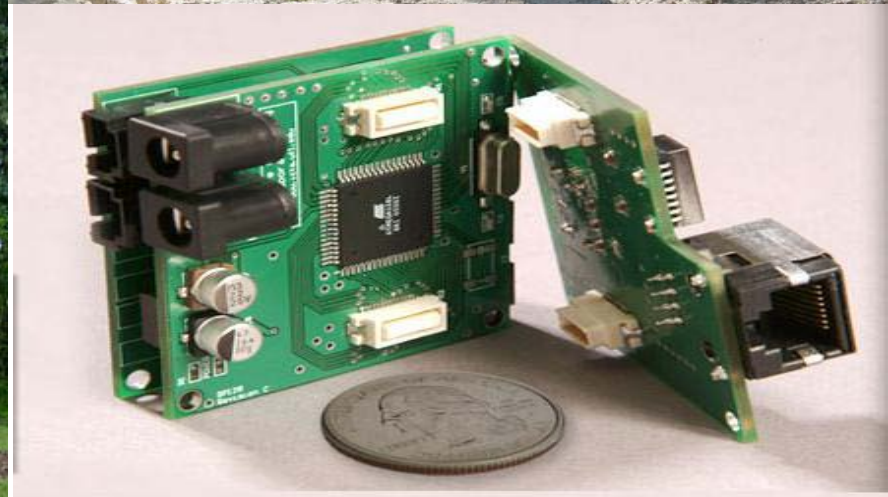
- **Concept of Coevolution**
  - Based on De Jong's definition of coevolution, one of the two populations is a learner and another one is evaluator.
  - End user can be the evaluator since s/he can decide whether the system evolution is successful or not.
  - End user can become a learner because s/he has to learn the evolved system (then system becomes the evaluator.)



Put Into Practice – Smart Homes

- While there are many practical examples where context-aware and situation-aware systems may be considered we will focus on a peculiar example that are emerging on the technology horizon and you and me will be affected in our life time – the Smart Home.
- Worldwide the population has been rapidly graying – we need to afford “Age in Place”

University of Florida Smart House



Smart Home – What in the Kitchen?

- **Use your imagination…**
- **…to make it smarter!**

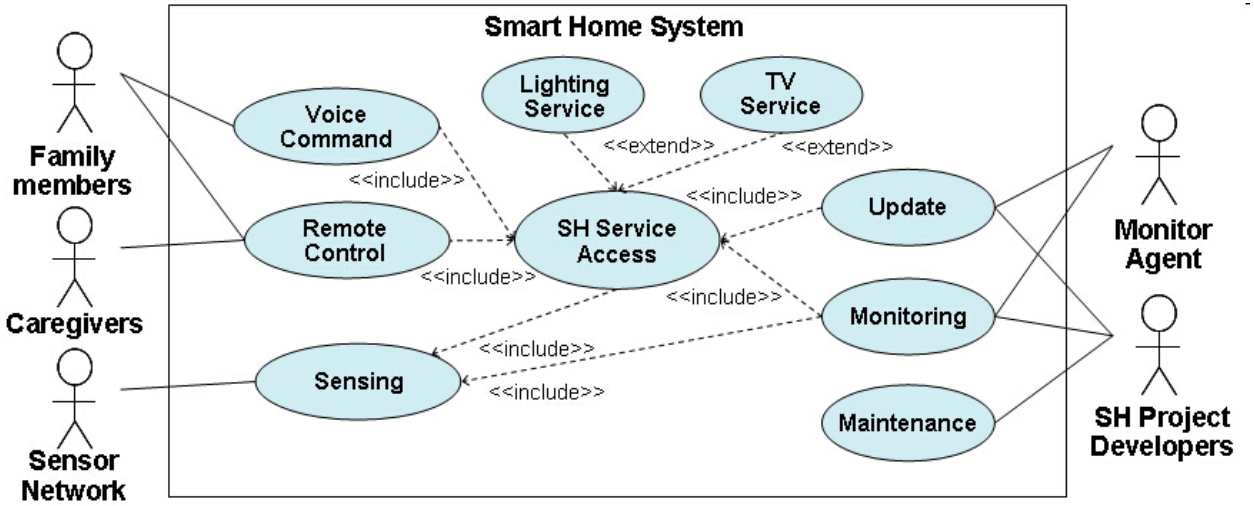
Smart Home Example

Scenario in Smart Home

We use an example to illustrate what kind of intention changes in the Smart Home should be monitored and handled. Consider a **90-year-old woman** who lives alone and has difficulty moving around and turning on/off switches in each room.

Suppose that there is an intelligent smart home. The system's use cases are:

1) An automatic lighting service,
2) Voice command service to control appliances such as TV.



Use Cases of the Smart Home

Smart Home Example (cont'd)

- **Scenario in Smart Home – Case 1**

  Intention Change 1: This elderly resident **wants to keep the light on even if she already left the room** due to sudden changes in her vision range in the past 24 hours.

  – Unfortunately, this new intention is against the predefined requirements and therefore, the system would not respond as she intends. She has to struggle to turn the light back on each time (supposing the system allows manual overriding).

  – With a voice command system, she can override the system without difficulty, and the room light works as she desires.

  - Statistically, from observing a large population of the user community and the frequent occurring of the same/similar episode, we may be able to identify the intention change.
  - The SH system then notifies remote SH project developers of this episode, as well as the in situ provision of alternative lighting service. The service can be either automatically customized with her action or updated by the SH project developer's patch.

Smart Home Example (cont'd)

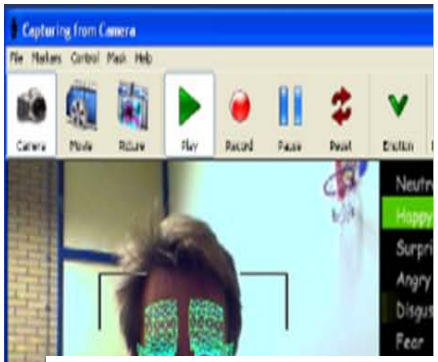
Scenario in Smart Home – Case 2

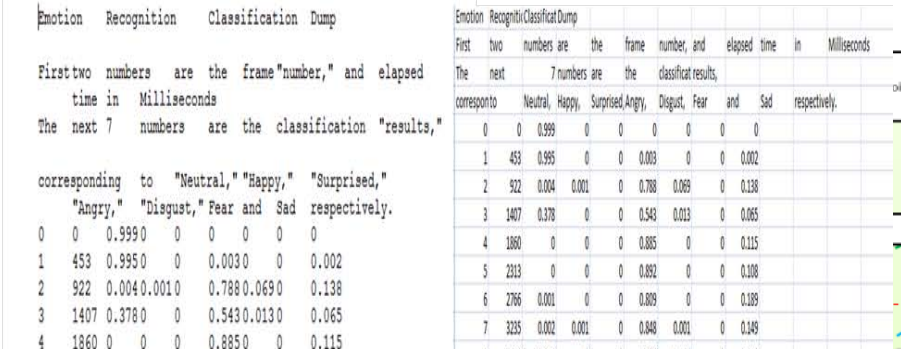
Intention Change 2: One day, she finds it difficult to walk. The caregiver helps her to move around to accomplish certain tasks, and sometimes tries to use the voice command feature of the SH remotely. Since the voice command system is customized for the resident and not the caregiver, **the caregiver's voice is not accepted**.

- The SH system should have the ability to identify residents with **special access privileges**.
- The system should be able to detect the caregiver's need to access the voice command system, and notify SH project developers. In this case, the caregiver should be granted access privileges by request of the resident and through confirmation of the "authority". Security policies in the SH system should be modified on the fly.

Smart Home Example (cont'd)

- Smart Home Lab (Desire Inference [Dong 2003])
- 24 subjects participated; 21 usable sets of data - 10 testing/11 training
- Facial Recognition & EEG observations were used





Emotion Recognition Classification Dump

First two numbers are the frame "number," and elapsed time in Milliseconds
The next 7 numbers are the classification "results,"

corresponding to "Neutral," "Happy," "Surprised," "Angry," "Disgust," Fear and Sad respectively.

0	0	0.999	0	0	0	0	0	0
1	453	0.995	0	0	0.003	0	0	0.002
2	922	0.004	0.001	0	0.788	0.069	0	0.138
3	1407	0.378	0	0	0.543	0.013	0	0.065
4	1860	0	0	0	0.885	0	0	0.115

Emotion Recognition Classification Dump

First two numbers are the frame number, and elapsed time in Milliseconds
The next 7 numbers are the classification results,
corresponding to Neutral, Happy, Surprised, Angry, Disgust, Fear and Sad respectively.

0	0	0.999	0	0	0	0	0	0
1	453	0.995	0	0	0.003	0	0	0.002
2	922	0.004	0.001	0	0.788	0.069	0	0.138
3	1407	0.378	0	0	0.543	0.013	0	0.065
4	1860	0	0	0	0.885	0	0	0.115
5	2313	0	0	0	0.892	0	0	0.108
6	2766	0.001	0	0	0.809	0	0	0.189
7	3225	0.002	0.001	0	0.848	0.001	0	0.149

Table 1. Each Desire-Emotion Correlation

Emotion \ Desire	Watch Movie	Eat a Cookie	Play the Game	Use Computer	View a photo	Listen to music	Read Comics	Wild Card
Neutral	0.269	0.004	0.420	**0.655**	0.273	**0.870**	**0.939**	**0.825**
Happy	**0.567**	**0.530**	**0.535**	0.001	0.004	0.002	0.004	0.002
Surprised	0.115	0.40	0.024	0.287	**0.618**	0.062	0.042	0.134
Angry	0.012	0.028	0.001	0.001	0.002	0.004	0	0
Disgust	0.001	0	0	0.012	0.015	0.017	0.003	0.009
Fear	0.034	0	0.008	0.002	0.022	0.001	0.009	0.002
Sad	0.004	0.002	0.005	0.032	0.043	0.037	0	0.022

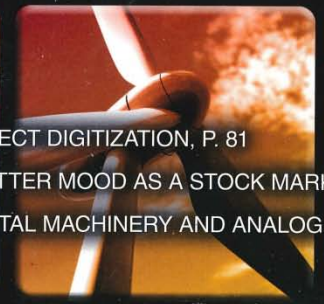
Innovative Technology for Computer Professionals

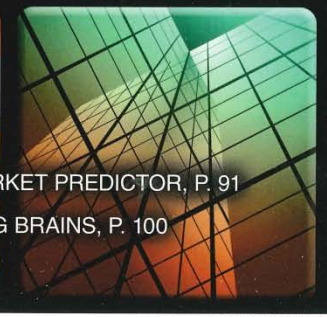
Computer

OCTOBER 2011

http://www.computer.org

SOFTWARE ENGINEERING MEETS ...





IEEE

IEEE computer society

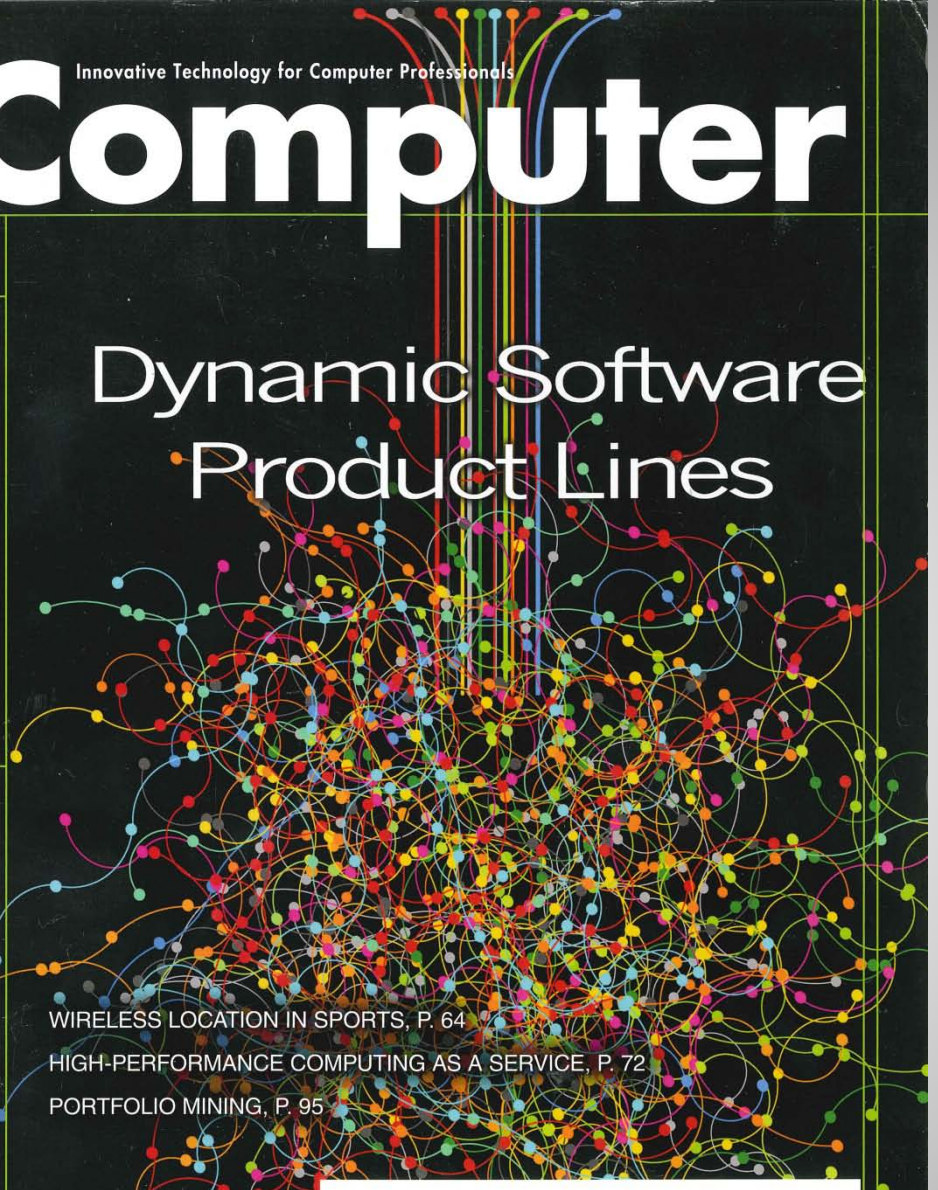
Innovative Technology for Computer Professionals

Computer

OCTOBER 2012

http://www.computer.org

Dynamic Software Product Lines



IEEE

IEEE computer society

Conclusions

- A more powerful situation-driven software evolution approach is envisioned to support individualized services that can evolve at runtime.
- Human intention changes underscore the reasons for service evolution; situational software engineering ushers in a novel software evolution process.
- Situations may become better, or worse -> The technology must simultaneously adapt to both a growth model and a degradation model
- *Situ* research requires interdisciplinary collaboration – high risk, yet high payoff

Some Speculations

- A Hypothetical Software Law: The rate of software productivity increase will eventually parallel that of computer (hardware) engineering such as the Moore's Law once software engineering researchers figure out a way to allow software to evolve like humans
- For example, advances in brain informatics could facilitate SE researchers to investigate the possibility to provide individualized services to end-users

More Speculations

- Who owns contextual information?
- Can Google freely exploit contextual information?
- Is a connected human or connected society healthy?
- Will Misinformation Superhighway repeat in the Age of Context Era, e.g., False-Contextual (Deceptive) Marketing?
- Can crowd sourcing support software services evolution?
- Intention inference may not always successful; then?

A Wild

PERSPECTIVES

NEUROSCIENCE

What Are Mini-Brains?

Human cerebral organoids grown in the lab may quickly advance our understanding of brain development and disease.

Byoung-il Bae and Christopher A. Walsh

christopher.walsh@childrens.harvard.edu

oid show the normal process of neocortical

brain parts and functional cortical neurons

References

- [Chang 2009]: Carl K. Chang, et al. Situ: A Situation-Theoretic Approach to Context-Aware Services Evolution. IEEE Trans. on Services Computing, 2:3, pp. 261-275.
- [Barr, 1979]: Barr, M. *Autonomous Categories*, Lecture Notes in Mathematics, 752. Springer-Verlag, Heidelberg
- [Barr, 1991]: Barr, M. *Autonomous categories and linear logic*, Mathematical Structures in Computer Science, 1. 159-178
- [Devlin et al., 1996]: Keith Devlin, Duska Rosenberg, *Language At Work-Analyzing Communication Breakdown in the Workplace to Inform Systems Design*, CSLI Lectures Notes Number 66, Stanford University, 1996

References

- [McCarthy et al., 1969]: J. McCarthy and P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence. In: B. Meltzer and D. Michie, Editors, Machine Intelligence, vol. 4, Edinburgh University Press (1969), pp. 463–502 http://www-formal.stanford.edu/jmc/mcchay69.html
- [McCarthy, 1995]: J. McCarthy, Situation calculus with concurrent events and narrative http://www-formal.stanford.edu/jmc/narrative.html (1995) web only
- [Pinto, 1994]: J.A. Pinto, Temporal Reasoning in the Situation Calculus (2nd edition), *Ph.D. Thesis*, Department of Computer Science, University of Toronto, Toronto, ON (Feb. 1994) Available as Technical Report KRR-TR-94-1

References

- [Barwise et al., 1983]: Barwise, J. and Perry, J., Situation and Attitudes, Bradford Books, MIT Press (1983)
- [Barwise at el., 1980]: Barwise, J. and Perry, J. *The Situation Underground*, in Stanford Working Papers in Semantics, Vol. 1, eds. J. Barwise and I. Sag, Stanford Cognitive Science Group 1980, Section D, pp.1–55
- [Devlin, 1991]: Keith Devlin, *Logic and information*, Cambridge University Press, 1991
- [Devlin, 2001]: Keith Devlin, *INFOSENSE: Turing Information into Knowledge*, W.H. Freeman and Company (2001)
- [Bratman 1989]: Michael Bratman, “Chapter 2 What is Intention?”, in **Intentions in COMMUNICATION**, Ed. Philip Cohen, Jerry Morgan and Martha Pollack, pp. 15-31, The MIT Press,1989.

References

- [Reiter, 1991]: Raymond Reiter, The Frame Problem in Situation Calculus: A Simple Solution (Sometimes) and a Complete Result for Goal Regression, in Vladimir Lifschitz, editor, Artificial Intelligence and Mathematical Theory of Computation: Paper in Honor of John McCarthy, pages 359-380. Academic Press, San Diego, CA, 1991
- [Poole, 1998]: David Poole, Decision Theory, the Situation Calculus, and Conditional Plans. http://www.cs.ubc.ca/spider/poole
- [Malle 2001]: Bertram Malle and Joshua Knobe, “The Distinction between Desire and Intention: A Folk-Conceptual Analysis, pp. 45-67, The MIT Press, 2001.

References

- [Yau et al., 2008]: S. Yau, H. Gong, D. Huang, W. Gao, and L. Zhu, *Specification, decomposition and agent synthesis for situation-aware service-based systems*, Journal of Systems and Software, Vol. 81, No. 10, pp. 1663-1680, 2008
- [Mastrogiovanni et al., 2008]: F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, Understanding events relationally and temporally related: Context assessment strategies for a smart home, in Second International Symposium on Universal Communication (ISUC' 08), 2008, pp. 217-224

References

- [Kalfoglou et al., 2003]: YANNIS KALFOGLOU and MARCO SCHORLEMMER, *Ontology mapping: the state of the art,* The Knowledge Engineering Review (2003), 18:1:1-31 Cambridge University Press 2003
- [De Jong 2004]: De Jong, E.D. and J. B. Pollack. Ideal Evaluation of Coevolution. Evolutionary Computation, Vol. 12, Issue 2, pp. 159-192.
- [Town, 2004]: Christopher Town, *Ontology-Driven Baysian Networks for Dynamic Scene Understanding*, 2004 conference on Computer Vision and Pattern Recognition Workshop (CVPRW ‘04) Volume 7, pp. 116,, 2004

Innovative Technology for Computer Professionals

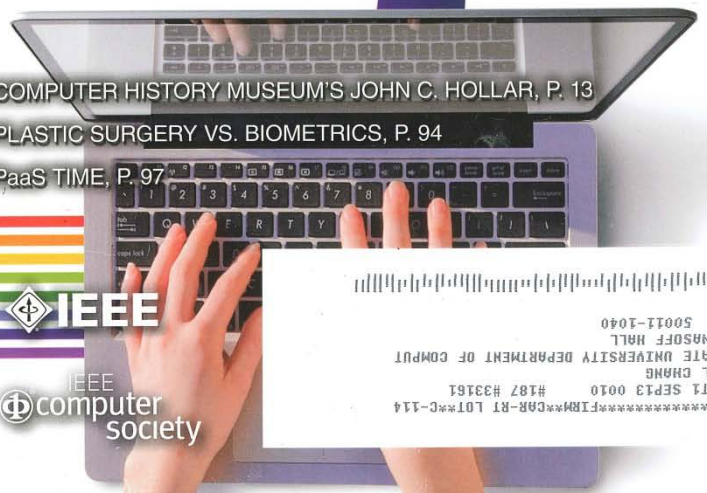
Computer

SEPTEMBER 2013

Q?

Computing Education

http://www.computer.org



IEEE
IEEE computer society