Elements of Foundations for Ubiquitous Computing

the beautiful, the useful and the rest

Vladimiro Sassone

University of Southampton
Ubiquitous Computing: what’s that?

Ubiquitous Computing:
  computation over a global network of mobile, bounded resources shared among mobile entities which move between highly dynamic, largely unknown, untrusted networks.

Difficulties:
  Extreme dynamic reconfigurability; lack of coordination and trust; limited capabilities; partial knowledge . . .

Issues:
  Protection and management of resources; privacy and confidentiality of data; . . .
Ubiquitous Computing: what’s that?

Ubiquitous Computing:
computation over a global network of mobile, bounded resources shared among mobile entities which move between highly dynamic, largely unknown, untrusted networks.

Difficulties:
Extreme dynamic reconfigurability; lack of coordination and trust; limited capabilities; partial knowledge . . .

Issues:
Protection and management of resources; privacy and confidentiality of data; . . .

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."
Mark Weiser, "The Computer for the Twenty-First Century,” Scientific American, pp. 94-10, September 1991
Ubiquitous Computing: what’s that?

From computers to ubiquitous computing, by 2020

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

invisible computing
Ubiquitous Computing: what’s that?

invisible computing
sentient computing

From computers to ubiquitous computing, by 2020

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”
Ubiquitous Computing: what’s that?

invisible computing
sentient computing
mobile computing

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

From computers to ubiquitous computing, by 2020
Ubiquitous Computing: what’s that?

invisible computing
sentient computing
mobile computing
autonomic computing

“The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”
From computers to ubiquitous computing, by 2020

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."


Ubiquitous Computing: what’s that?

- invisible computing
- sentient computing
- mobile computing
- autonomic computing
- networks architectures
- embedded systems
- sensor networks
- sensor networks
- networks architectures
Ubiquitous Computing: what’s that?

invisible computing
sentient computing
mobile computing
autonomic computing

networks architectures
embedded systems
sensor networks
progr. languages

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

From computers to ubiquitous computing, by 2020
From computers to ubiquitous computing, by 2020

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

From computers to ubiquitous computing, by 2020

Ubiquitous Computing: what’s that?

- invisible computing
- sentient computing
- mobile computing
- autonomic computing
- networks architectures
- embedded systems
- medical computing
- sensor networks
- progr. languages
- power awareness
- legal, social, ethical issues

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

From computers to ubiquitous computing, by 2020

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Ubiquitous Computing: what’s that?

invisible computing
sentient computing
mobile computing
autonomic computing
networks architectures
embedded systems
medical computing
power awareness
privacy
crypto & security

Professor Marta Kwiatkowska
Professor Tom Rodden
Professor Vladimiro Sassone

Location:
The Royal Society
6-9 Carlton House Terrace
London SW1Y 5AG

17-18 March 2008
A lot of embedded devices and smart space
A lot of embedded devices and smart space
Ubiquitous Computing: my perspective

- Models for Concurrency
- Semantic Theories
- Spatial Logics
- Programming Languages
- Resource Control
Ubiquitous Computing: my perspective

Models for Concurrency

Petri Nets Based Models and Calculi
A distributed timed-arc Petri net is a Petri net together with
- a interval time constraint on transitions, either discrete or continuous;
- a clock synchronisation relation $\Sigma$ on places.

Tokens age, transitions are enabled accordingly. Time elapses at the same speed at $p$ and $p'$ if $p \Sigma p'$.

Spatial Logics

Globally Asynchronous, Locally Synchronous
Global Time: $\Sigma = P \times P$  Local Time: $\Sigma = \Delta_P$

Programming Languages

A Separation Result: Reachability for safe LT nets is decidable, but undecidable for safe GT nets.
Ubiquitous Computing: my perspective

Models for Concurrency

Semantic Theories

Spatial Logics

Programming Languages

Resource Control

Labels from Reductions

- A categorical machinery which allows the derivation of LTSs from reduction systems.
- Bisimulation on such LTSs is a congruence, provided a general condition is met.

Coinduction Principle Desiderata:

- Correspondence: \( p \Downarrow q \iff p \xrightarrow{\tau} q \)
- Correctness: \( p \approx q \) implies \( p \approx q \)
- Completeness: \( p \approx q \) implies \( p \approx q \)

The intuition: \( a \xrightarrow{e} b \iff e[a] \Downarrow b \)

Eg: \( a \xrightarrow{\neg a} 0, \quad M \xrightarrow{(\lambda x. \cdot)^N} M[N/x], \quad KM \xrightarrow{N} M \)
Two related continuations:

(1) What “barbs” i.e. observations are required to give rise to an observation theory corresponding to the contexts as labels?

(2) How to generate transition systems out of from SOS specification systems in the case of stochastic transition systems?
Ubiquitous Computing: my perspective

Models for Concurrency

Semantic Theories

Spatial Logics

Programming Languages

Resource Control

Structural Bilogics

- Spatial logics: Separation in space
  \[ \ell_1[ a@\ell.P ] \parallel \ell_2[ \bar{a}@\ell.Q ] \]

- Separation logics: Separation of resources
  \[ \ell[ a.nil | b.nil ] \]

A more expressiveness and unified approach: Eg,

\[ PC_a(\text{in}_c \otimes T) \otimes PC_b(\text{out}_c \otimes T) \]

describes two PCs linked to the network by “separated” \( a \) and \( b \), and to each other by “shared” \( c \).

Results: Proof and model theory for BiLog, encoding of previous logics, decidability issues.
Ubiquitous Computing: my perspective

Models for Concurrency
- Java (no synchronized(), wait(), notify(), notifyAll()) for business code;
- Linear Time Temporal Logic for synchronization code (method guards).

```
public class MyClass {
    sync { m : φ; ... }

    ... // Standard Java class def
}
```

where \( m \) is a method identifier and \( φ \) is an LTL formula. When \( m \) is invoked, the thread is holds unless \( φ \). When the condition is true, all waiting threads are awaken. \( m \) is implicitly synchronised.
Ubiquitous Computing: my perspective

- Models for Concurrency
  - Resources: Models, Types, Logics, Languages
    - Access Control
    - Access Authorisation
  - Secrecy for Mobile Agents
  - Trust Management
  - Bounds Control
Ubiquitous Computing: my perspective

- Models for Concurrency
  - Resources: Models, Types, Logics, Languages
    - Access Control
    - Access Authorisation
    - Secrecy for Mobile Agents
    - Trust Management
- Semantic Theories
- Spatial Logics
- Programming Languages
- Resource Control
Trust in UbiCom

Features of Ubiquitous Computing like scalability, mobility, and incomplete information deeply affect security requirements.

One of the proposed approaches is to use a notion of computational trust, resembling the concept of trust among human beings.
Approaches to Trust

 Credential-based Models

- trust predicated on possession of predefined credential
- eg, password, RSA key, certificate, role, history, provenance, ...

 Predictive Models (“observe & learn”)

- a probabilistic model assigns a degree of confidence to a principal’s ability to predict another principal’s behaviour.
- eg, the behaviour of a principal $A$ may be defined as the probability that interaction with $A$ yields a certain outcome.

Overarching notion: **Trust Policy** express complex conditions based on elementary trust values.
(Meta)data is almost entirely neglected in the process calculi.

Track data provenance both for its important applications and as an challenging exercise in modelling (meta)data. Aim at simplicity:

- data annotations representing provenance
- structure, interpretation and management of provenance information
- provenance tracking

Provenance-based security (*trust* + *data confidentiality*)

- Example: conference submission

The overall ambition is to underpin and develop practical stuff, like trust-policy languages and protocols, and provenance-middleware.
Provenance model

\[ v : K \]
Provenance model

Annotated value

\[ v : K \]
Provenance model

Annotated value

Value

Actual data

$v : K$
Provenance model

- Annotated value
- Value
- Provenance

Value \( v : K \)

Actual data

Meta information describing the origin of the value
Provenance model
Structure and interpretation of provenance

\[ v : \varepsilon ; a! \kappa_1 ; b ? \kappa_2 ; b!(\varepsilon ; c! \kappa_3 , b ? \kappa_4 ) ; ... \]
Provenance model
Structure and interpretation of provenance

\[ v : \varepsilon ; a!\kappa_1 ; b?\kappa_2 ; b!(\varepsilon;c!\kappa_3 , b?\kappa_4 ) ; \ldots \]
Provenance model
Structure and interpretation of provenance

Value

Provenance

\[ v : \varepsilon ; a!\kappa_1 ; b?\kappa_2 ; b!(\varepsilon; c!\kappa_3, b?\kappa_4) ; \ldots \]

“Operations” that were performed on the value. They record the principals that “influenced” the value and how.
Provenance model
Structure and interpretation of provenance

$\varepsilon$ (empty provenance) denotes value $\nu$ originated here

$\nu : \varepsilon$
Provenance model
Structure and interpretation of provenance

$v : \varepsilon ; a!\kappa_1$

$\varepsilon$ (empty provenance) denotes value $v$ originated here

It was sent by $a$ on a channel with provenance $\kappa_1$
Provenance model
Structure and interpretation of provenance

\( \varepsilon \) (empty provenance) denotes value \( v \) originated here:

\[ v : \varepsilon ; a!\kappa_1 ; b?\kappa_2 \]

It was sent by \( a \) on a channel with provenance \( \kappa_1 \):

Was then received by \( b \) on a channel with provenance \( \kappa_2 \).
Provenance model
Structure and interpretation of provenance

\[ v : \varepsilon ; a!\kappa_1 ; b?\kappa_2 ; b!(\varepsilon;c!\kappa_3,b?\kappa_4) ; \ldots \]

- \( \varepsilon \) (empty provenance) denotes value \( v \) originated here.
- It was sent by \( a \) on a channel with provenance \( \kappa_1 \).
- And then sent by \( b \) on a channel that \( b \) received from \( c \)...
- Was then received by \( b \) on a channel with provenance \( \kappa_2 \).
- Provenance model
Confidentiality in provenance systems

- Data may be public, yet its provenance confidential, or vice versa
- Principals who may access data are not necessarily the same as those who may access its provenance
- Fine grained access control over provenance “histories” is needed as different parts of it have different sensitivity

Security requirements of data ≠ Security requirements of its provenance
Hiding provenance trees
Example: conference submissions

c:  Author
a:  PC Chair
j:  Referee
Hiding provenance trees
Example: conference submissions

c: Author
a: PC Chair
j: Referee

entry: ε; c!κs
Hiding provenance trees
Example: conference submissions

c: Author
a: PC Chair
j: Referee

c: entry : ε;c!κ
j: entry : ε;c!κ; a?κ'; dκ'

a
Hiding provenance trees
Example: conference submissions

- Entry: ε; c!K_s
- Entry: ε; c!K_s; a?K'_s; a!K'_s
- Entry: ε; c!K_s; a?K'_s; a!K'_s; j?K'_s; j!K'_s
- Score: ε; j!K''_s

- c: Author
- a: PC Chair
- j: Referee
Hiding provenance trees
Example: conference submissions

c: Author
a: PC Chair
j: Referee
Hiding provenance trees
Example: conference submissions
Hiding provenance trees
Example: conference submissions

Hiding provenance trees
Example: conference submissions

Hiding provenance trees
Example: conference submissions
Hiding provenance trees
Example: conference submissions
Multiple provenance views

- **One value, multiple views**

  Different principals have different views of the same provenance list based on their privileges

  \[ \text{entry : } \varepsilon; c!\kappa_s; a?\kappa'_s; a!\kappa'_r; j?\kappa''_r; j!\kappa''_n; a?\kappa'_n; a!\kappa'_m \]
One value, multiple views

Different principals have different views of the same provenance list based on their privileges

entry: $\varepsilon; c! \kappa_s; a? \kappa'_s; a! \kappa'_r; j? \kappa''_r; j! \kappa''_n; a? \kappa'_n; a! \kappa'_m$
Multiple provenance views

- One value, multiple views

Different principals have different views of the same provenance list based on their privileges

entry: $\varepsilon; c! \delta_s; a? \delta_s'; a! \delta_r'; j? \delta_r''; j! \delta_r'''; a? \delta_n'; a! \delta_m'$
Multiple provenance views

One value, multiple views

Different principals have different views of the same provenance list based on their privileges

\[ entry : \varepsilon, c!\kappa_s; a?\kappa'_s; a!\kappa'_r; j?\kappa''_r; j!\kappa''_n; a?\kappa'_n; a!\kappa'_m \]
Multiple provenance views

*One value, multiple views*

Different principals have different views of the same provenance list based on their privileges

entry: $\varepsilon; c!\kappa_s; a?\kappa'_s; a!\kappa'_f; j?\kappa''_f; j!\kappa''_n; a?\kappa'_n; a!\kappa'_m$

\[a\quad c\quad j\]
Inferring probability distributions

- Examples of applications in trust & security
  - Estimate trust in an individual or set of individuals
  - Estimate input distribution of a noisy channel to compute the Bayes risk
  - Apply the Bayesian approach to hypothesis testing (anonymity, information flow)
  - ...

The outcome of an interaction between a principal $a$ and a partner $b$ is either successful or unsuccessful:

$$o \in \{\text{Succ, Fail}\}$$

The probability that a partner $b$ interacts successfully with $a$ is governed by the parameter $\theta$ where:

$$\theta = \Pr(o = \text{Succ})$$

Goal: infer (an approximation of) the probability of success

Means: Observe sequence of trials (observations)
Note that: the behaviour of the partner $b$ represented by $\theta$ is assumed to be fixed over time.

The estimated probability of success, $B(Succ \mid o)$, at time $t$ is the expected value of $\theta$ given the sequence of outcomes $o = \{o_0, o_1, \ldots, o_t\}$

$$B(Succ \mid o) = E[\theta \mid o]$$
Using evidence to infer $\theta$

**The “Frequentist” method:**

$$F(n, s) = \frac{s}{n}$$

**The “Bayesian” method:**

Assume an *a priori* probability distribution for $\theta$ (representing your partial knowledge about $\theta$, whatever the source may be) and combine it with the *evidence*, using Bayes’ theorem, to obtain the *a posteriori* distribution.
A Bayesian approach

- **Assumption**: \( \theta \) is the generic value of a continuous random variable \( \Theta \) whose probability density is a *Beta distribution* with (unknown) parameters \( \sigma, \varphi \)

\[
B(\sigma, \varphi)(\theta) = \frac{\Gamma(\sigma+\varphi)}{\Gamma(\sigma)\Gamma(\varphi)} \theta^{\sigma-1}(1 - \theta)^{\varphi-1}
\]

where \( \Gamma \) is the extension of the factorial function i.e. \( \Gamma(n) = (n - 1)! \) for \( n \) natural number

- The uniform distribution is a particular case of Beta, for \( \sigma = 1, \varphi = 1 \)

- \( B(\sigma, \varphi) \) can be seen as the a posteriori probability density of \( \Theta \) given by a uniform a priori (principle of *maximum entropy*) and a trial sequence resulting in \( \sigma - 1 \) successes and \( \varphi - 1 \) failures.
The Bayesian Approach

Following the approach, we have three probability density functions for $\Theta$:

- $B(\sigma, \varphi)$: the “real” distribution of $\Theta$
- $B(\alpha, \beta)$: the a priori
  (our estimate of the distribution of $\Theta$)
- $B(s + \alpha, f + \beta)$: the a posteriori
  (the distribution of $\Theta$ after the trials)

The result of the mean-based algorithm is:

$$A_{\alpha,\beta}(n, s) = E_{B(s+\alpha,f+\beta)}(\Theta) = \frac{s + \alpha}{s + f + \alpha + \beta} = \frac{s + \alpha}{n + \alpha + \beta}$$
Trust Inference Process

\[ O_1 = S \]
\[ O_2 = S \]
\[ O_3 = F \]
\[ O_4 = F \]
\[ O_5 = S \]
Trust Inference Process

The distribution of $\theta$ after 40 interactions
25 Successful and 15 Failed
The Frequentist approach can be worse than the Bayesian approach even when the trials give a “good” result, or when we consider the average difference (from the “true” $\theta$) wrt all possible results.

Example: “true $\theta$” = 1/2, $n = 1$

$$F(n, s) = \frac{s}{n} = \begin{cases} 0 & s = 0 \\ 1 & s = 1 \end{cases}$$

The difference from the true distribution is 1/2.
Bayesian vs Frequentist

The Frequentist approach can be worse than the Bayesian approach even when the trials give a “good” result, or when we consider the average difference (from the “true” $\theta$) wrt all possible results.

Example: “true $\theta$” = 1/2, $n = 1$

$$F(n, s) = \frac{s}{n} = \begin{cases} 0 & s = 0 \\ 1 & s = 1 \end{cases}$$

The difference from the true distribution is 1/2

A better function would be

$$F_c(n, s) = \frac{s + 1}{n + 2} = \begin{cases} \frac{1}{3} & s = 0 \\ \frac{2}{3} & s = 1 \end{cases}$$

The difference from the true distribution is 1/6
Bayesian vs Frequentist

The Frequentist approach can be worse than the Bayesian approach even when the trials give a “good” result, or when we consider the average difference (from the “true” $\theta$) wrt all possible results.

Example: “true $\theta$” = 1/2, $n$ = 2

$$F(n, s) = \frac{s}{n} = \begin{cases} 0 & s = 0 \\ \frac{1}{2} & s = 1 \\ 1 & s = 2 \end{cases}$$

The average difference from the true distribution is 1/4.
Bayesian vs Frequentist

The Frequentist approach can be worse than the Bayesian approach even when the trials give a “good” result, or when we consider the average difference (from the “true” $\theta$) wrt all possible results.

Example: “true $\theta$” = 1/2, $n = 2$

$$F(n, s) = \frac{s}{n} = \begin{cases} 
0 & s = 0 \\ 
\frac{1}{2} & s = 1 \\ 
1 & s = 2 
\end{cases}$$

The average distance from the true distribution is 1/4

Again, a better function would be

$$F_c(n, s) = \frac{s + 1}{n + 2} = \begin{cases} 
\frac{1}{4} & s = 0 \\ 
\frac{1}{2} & s = 1 \\ 
\frac{3}{4} & s = 2 
\end{cases}$$

The average distance from the true distribution is 1/8
Measuring the precision of Bayesian algorithms

- Define a “difference” $D(A(n,s), \theta)$ (not necessarily a distance)
  - non-negative
  - zero iff $A(n,s) = \theta$

- Consider the expected value $D_E(A,n,\theta)$ of $D(A(n,s), \theta)$ with respect to the likelihood (the conditional probability of $s \mid \theta$)

- **Risk of $A$** : the expected value $R(A,n)$ of $D_E(A,n,\theta)$ with respect to the “true” distribution of $\Theta$

$$D_E(A,n,\theta) = \sum_{s=0}^{n} Pr(s \mid \theta) D(A(n,s), \theta)$$

$$R(A,n) = \int_{0}^{1} Pd(\theta) \ D_E(A,n,\theta) \ d\theta$$
We have considered the following candidates for $D(x,y)$ (all of which can be extended to the n-ary case):

- The norms:
  - $|x - y|$
  - $|x - y|^2$
  - ...
  - $|x - y|^k$
  - ...
- The Kullback-Leibler divergence

$$D_{KL}((y, 1-y) \parallel (x, 1-x)) = y \log_2 \frac{y}{x} + (1 - y) \log_2 \frac{1 - y}{1 - x}$$
"Theorem. For the mean-based Bayesian algorithms, with a priori $B(\alpha, \beta)$, we have that the condition is satisfied (i.e. the Risk is minimum when $\alpha, \beta$ coincide with the parameters $\sigma, \varphi$ of the “true” distribution), by the following functions:

- The 2nd norm $(x - y)^2$
- The Kullback-Leibler divergence

Surprising that the condition is satisfied by these two very different functions, and not by any of the other norms $|x - y|^k$ for $k \neq 2$.

It leaves the search open for a measure for assessment and comparison of trust algorithm.
Potential applications

- We can use $D_E$ to compare two different estimation algorithms; develop a measure of quality for "decision-making" algorithms.

- Mean-based vs other ways of selecting a $\theta$.

- Bayesian vs non-Bayesian.

- In more complicated scenarios there may be different Bayesian mean-based algorithms; eg.: noisy channels.
Potential applications (ctd)

- $D_E$ induces a metric on distributions. Bayes’ equations define transformations on this metric space from the a priori to the a posteriori.

- Study the properties of such transformations to reveal interesting properties of the corresponding Bayesian methods, independent of the a priori.

- Hypothesis testing (privacy, anonymity, confidentiality, information flow analysis, input distribution analysis, ...) :

  - determine (probabilistic) bounds as to what probability-distribution inference algorithm may determine about you, your online activity, your software.
Limitation of the Beta model

- The assumption that a principal behaviour is fixed is not always realistic:

- The behaviour of a principal may depend on its internal state which may change over time.
Modelling Dynamic Behaviour

- Modelling static behaviour as a probability distribution over outcomes leads to modelling the dynamic behaviour by a *Hidden Markov Model (HMM)*.

- A single state in an HMM models the system behaviour at a particular time.
Hidden Markov Model:

\[ S = \{1, 2\} \]
\[ V = \{X, Y\} \]

\[ A = \begin{bmatrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{bmatrix} \]
\[ B = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix} \]

\[ \text{Pr}(X) = 0.1 \]
\[ \text{Pr}(Y) = 0.9 \]

\[ \text{Pr}(X) = 0.9 \]
\[ \text{Pr}(Y) = 0.1 \]
A simpler model: Beta with Decay

- The probability distribution over outcomes changes over time.
- Old observations are given less weight (decayed) than more recent observations.
- Weights of observations are controlled by the decay factor $r$. 
Given a decay factor $0 \leq r < 1$ and an observation sequence $o = \{o_0, ..., o_L\}$ then

$$B_r(Succ \mid o) = \frac{m_r(o) + 1}{m_r(o) + n_r(o) + 2}$$

$$B_r(Fail \mid o) = \frac{m_r(o) + 1}{m_r(o) + n_r(o) + 2}$$

where

$$m_r(o) = \sum_{i=0}^{L} r^{L-i} \cdot \delta_{Succ}(o_i)$$

$$n_r(o) = \sum_{i=0}^{L} r^{L-i} \cdot \delta_{Fail}(o_i)$$

and

$$\delta_x(o) = \begin{cases} 
1 & \text{if } x = o \\
0 & \text{otherwise}
\end{cases}$$
How good is the model?

Given a dynamic system modelled by an HMM $\lambda$ we define Beta estimation error as follows:

$$\text{Error}(\lambda, r) = E \left[ (B(\text{Succ} \mid o) - \alpha)^2 \right]$$

where $r$ is the decay factor, and $\alpha$ is the real probability that next outcome is Success.
System stability

- **System stability** is the expected probability of the HMM remaining in the same state.

- Consider the system modelled by HMM:

\[
A_\lambda = \begin{bmatrix}
    s & \frac{1-s}{3} & \frac{1-s}{3} & \frac{1-s}{3} \\
    \frac{1-s}{3} & s & \frac{1-s}{3} & \frac{1-s}{3} \\
    \frac{1-s}{3} & \frac{1-s}{3} & s & \frac{1-s}{3} \\
    \frac{1-s}{3} & \frac{1-s}{3} & \frac{1-s}{3} & s
\end{bmatrix}
\]

\[
\Theta_\lambda = \begin{bmatrix}
    1.0 \\
    0.7 \\
    0.3 \\
    0.0
\end{bmatrix}
\]
Unstable system

Graphs showing expected error versus decay factor for different stability values:
- Stability = 0.0
- Stability = 0.1
- Stability = 0.25
- Stability = 0.4
Stable system

- Stability = 0.6
- Stability = 0.7
- Stability = 0.8
- Stability = 0.9

Expected Error vs. Decay Factor
Very stable system

Stability = 0.99

Stability = 0.995

Stability = 0.999

Stability = 1
A whole wholly-different conception of computing to be developed: hard to talk of “further” work in general.

Chiefly, nowhere like here apps w/out sound models are dangerous, and theory without practice is pointless.
A whole wholly-different conception of computing to be developed: hard to talk of “further” work in general

The gap between Theory and Practice matters in practice (although it may not matter in theory)

are dangerous, and theory without practice is pointless
Conclusion (in general)

A whole wholly-different conception of computing to be developed: hard to talk of “further” work in general

The gap between Theory and Practice matters in practice (although it may not matter in theory)
are dangerous, and theory without practice is pointless

One thing I know: as one cannot “model-check” UbiNet, security & privacy in UbiCom must be coupled with trust
Conclusion (personal take)

in the short term:
- hiding and multiview in provenance trees
- measures suitable to compare trust-algorithms
- reputation in HMMs
- integration of anonymity protocols and trust

in the longer term:
- programming language bindings
- data confidentiality and then privacy
- ...
- ...