

Modelling for Context Awareness

Doina Bucur

joint work with Mogens Nielsen

doina@daimi.au.dk

Department of Computer Science
University of Aarhus, Denmark



UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

	distributed system		
	traditional	nomadic	ad hoc
infrastructure	fixed	fixed	?
device	fixed	mobile	mobile
services	in the core	in the core	decentralized
connection	permanent	permanent	intermittent
communication	synchronous	synchronous	asynchronous
context	static	dynamic	dynamic
middleware	transparency	awareness	awareness

UbiComp Applications

UbiComp:

- ★ **Implosion** in size: smaller (embedded) computers
- ★ **Diversification**: dedicated computers
- ★ **Explosion** in user interfaces, interaction, numbers and mobility
- ★ **Decentralization**: distribution of services

Two-tier architecture:

- ★ **Applications**: low local resources, networked, adaptive, cooperative, numerous, mobile
- ★ **Communication** protocols (of context): synchronous or asynchronous

Context-Awareness Challenges

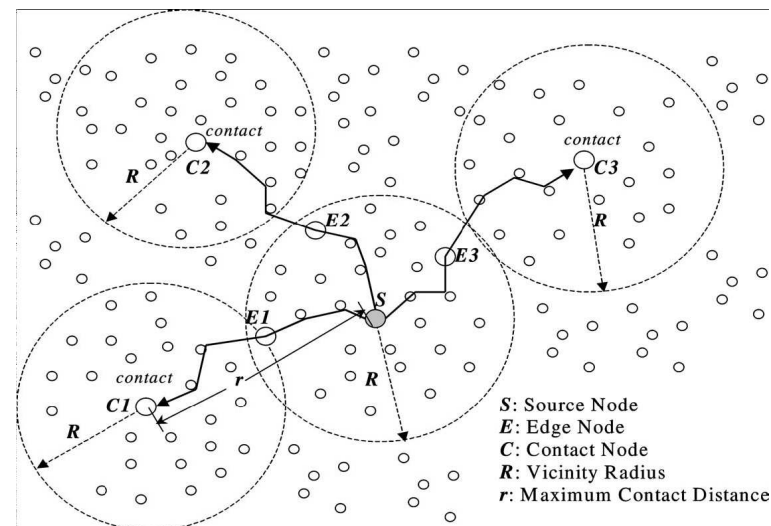
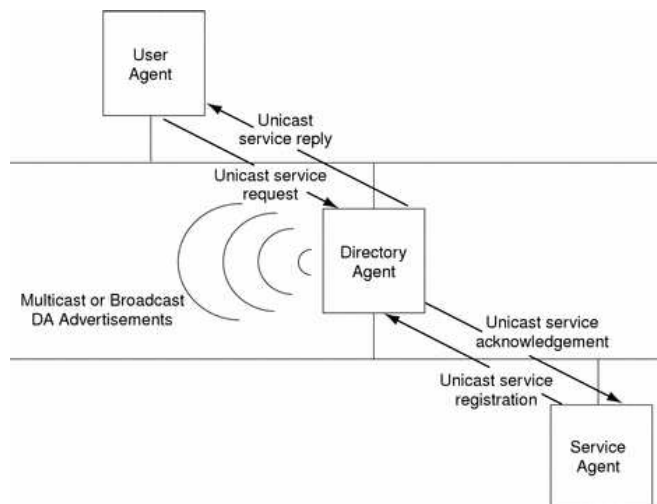
Context awareness:

- ★ Modelling and acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition

Context-Awareness Challenges

Context awareness:

- ★ Modelling and acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification



Context-Awareness Challenges

Context awareness:

- ★ Modelling and acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification
- ★ Dealing with accuracy/uncertainty
 - ✓ sensor fusion

Context-Awareness Challenges

Context awareness:

- ★ Modelling and acquiring contextual information:
 - ✓ [Dey, Abowd] primary, secondary
 - ✓ [Pascoe, Dey] sensing
 - ✓ localization, activity recognition
- ★ Designing infrastructure and protocols
 - ✓ [Pascoe, Dey] provision and discovery
 - ✓ service discovery, protocol verification
- ★ Dealing with accuracy/uncertainty
 - ✓ sensor fusion
- ★ Achieving security/privacy
 - ✓ information flow security, effect-based type systems

Application Features

[Schilit] Awareness:

- ★ passive (manual)
- ★ active (automatic)

Application Features

[Schilit] Awareness:

- ★ passive (manual)
- ★ active (automatic)

Features of context-aware applications:

	information	commands
manual	proximate selection	automatic contextual reconfiguration
automatic	contextual information and commands	context-triggered actions

Process calculi

Universal models of computation. Formally model concurrent systems:

- ★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

Process calculi

Universal models of computation. Formally model concurrent systems:

★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M / x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v / x\}Q$

Process calculi

Universal models of computation. Formally model concurrent systems:

★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M / x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v / x\}Q$

★ Mobility and topology

hierarchical cells	$a[P \mid b[Q]] \mid c[in a.R]$
dynamic process graphs	$\nu c \nu d (\bar{c}d.P \mid c(x).\bar{x}5.Q \mid d(y).R)$

Process calculi

Universal models of computation. Formally model concurrent systems:

- ★ Process terms

process	P
parallel composition	$P \mid Q$
replication	$!P$
name restriction	$(\nu z)P$

- ★ Interaction and synchronization

asynchronous	$\langle M \rangle \mid (x).P \rightarrow \{^M/x\}P$
synchronous	$\bar{c}v.P \mid c(x).Q \rightarrow P \mid \{^v/x\}Q$

- ★ Mobility and topology

hierarchical cells	$a[P \mid b[Q]] \mid c[in\ a.R]$
dynamic process graphs	$\nu c \nu d (\bar{c}d.P \mid c(x).\bar{x}5.Q \mid d(y).R)$

- ★ Algebraic laws to manipulate and analyze processes;
techniques to reason on process equivalence

A Calculus of Context Awareness

Process: first-class object

- ★ returnable from a function
- ★ transmissible among distributed processes
- ★ storable outside running processes

Named macros [Zimmer]:

$$f \triangleright P, \text{ with } f \in \mathcal{F}$$

A Calculus of Context Awareness

Process: first-class object

- ★ returnable from a function
- ★ transmissible among distributed processes
- ★ storable outside running processes

Named macros [Zimmer]:

$$f \triangleright P, \text{ with } f \in \mathcal{F}$$

Macro definition flavours:

- ★ **one-shot**, consumed at call (e.g. network packets)

$$f \triangleright P$$

- ★ **permanent**, infinitely called (e.g. a person's location)

$$!f \triangleright P$$

Infrastructure-based setting

Processes	$P ::= 0$	no process
	$ P \mid P'$	parallel composition
	$ a [P]$	mobile entity, $a \in \mathcal{A}$
	$ \nu z P$	name restriction, $z \in \mathcal{F} \cup \mathcal{A}$
	$ in a.P$	movement in
	$ out.P$	movement out

$$\text{IN} \quad a[P] \mid b[in a.Q \mid R] \longrightarrow a[P \mid b[Q \mid R]]$$

$$\text{OUT} \quad a[P \mid b[out.Q \mid R]] \longrightarrow a[P] \mid b[Q \mid R]$$

$$\text{STRUCT} \quad \frac{P \equiv P' \quad P \longrightarrow Q \quad Q \equiv Q'}{P' \longrightarrow Q'}$$

$$\text{CONTEXT} \quad \frac{P \longrightarrow Q}{\mathbf{C}[P] \longrightarrow \mathbf{C}[Q]}$$

Infrastructure-based setting

Services: naturally layered.

Processes	P	$::=$	f^a	macro call, $f \in \mathcal{F}$
			$def^a D \text{ in } P$	macro definition
			$E P$	public definitions
Definitions	E	$::=$	$(D)^a$	floating definition
	D	$::=$	$F \mid !F$	
	F	$::=$	$f \triangleright P$	macro definition

Infrastructure-based setting

Services: naturally layered.

Processes	P	$::=$	f^a	macro call, $f \in \mathcal{F}$
			$def^a D \text{ in } P$	macro definition
			$E P$	public definitions
Definitions	E	$::=$	$(D)^a$	floating definition
	D	$::=$	$F \mid !F$	
	F	$::=$	$f \triangleright P$	macro definition

DEF $def^a f \triangleright Q \text{ in } P \longrightarrow (f \triangleright Q)^a P$

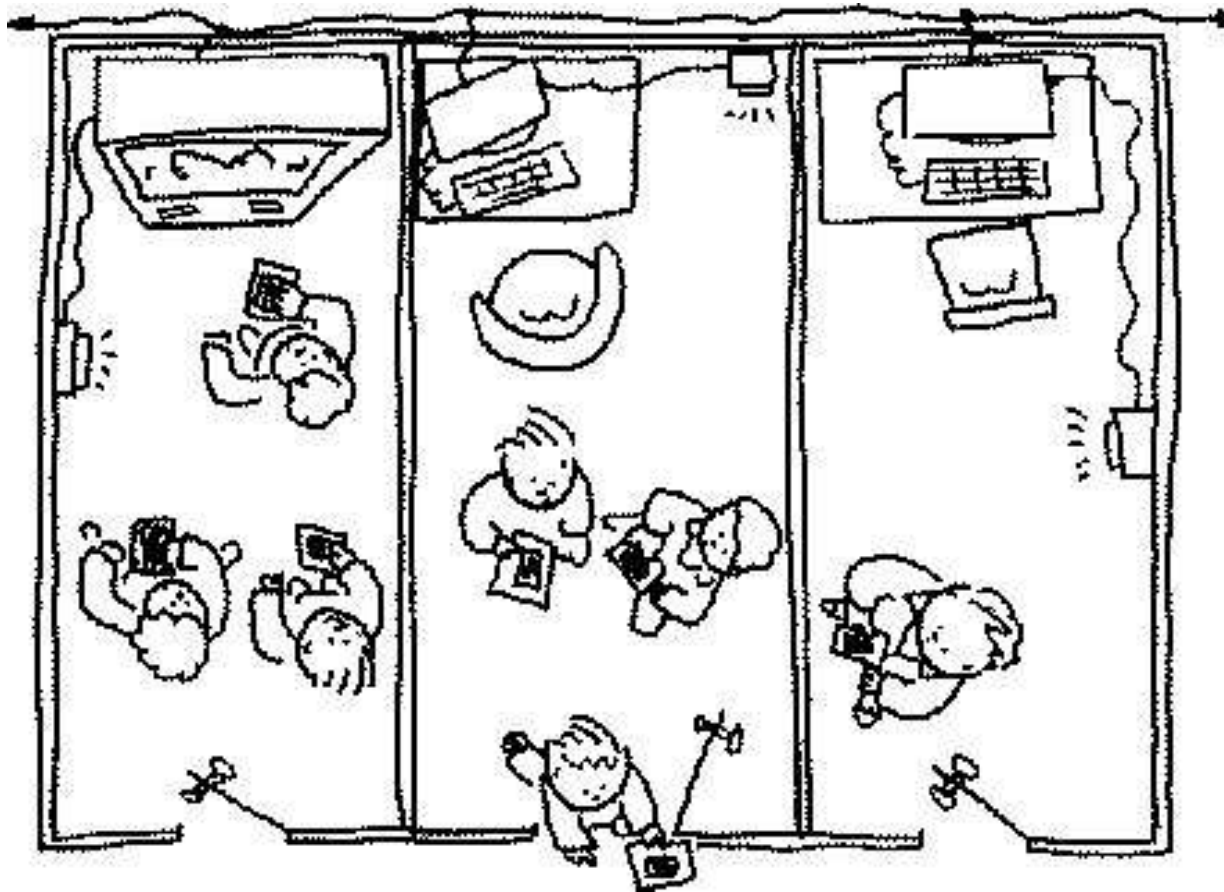
CALL $(f \triangleright Q)^a f^a \longrightarrow Q$

UP, DOWN $(D)^b P \mid Q \equiv (D)^b (P \mid Q) \quad a[(D)^b P] \equiv (D)^b a[P]$

A model of **contextual commands**, **context-triggered actions**.

Example

ParcTab, Xerox Parc, [Want et al., 95]. First context-aware computer.



Ad-hoc setting

Networks	$N ::= 0$	no network
	$ N \mid N'$	parallel composition
	$ a_l[P]$	node at l
Processes	$P ::= 0$	no process
	$ P \mid P'$	parallel composition
	$ \text{move } l.P$	movement

$$\text{MOVE} \quad \frac{(l, l') \in G_{mob}^a}{a_l[\text{move } l'.P \mid Q] \longrightarrow a_{l'}[P \mid Q]}$$

Ad-hoc setting: proactive

Communication paradigm: **broadcast**.

Processes	$P ::= f$	local macro call
	$\text{def}^{\odot n} D \text{ in } P$	proactive macro definition
	$E P$	public definitions
Definitions	$E ::= 0$	no definition
	$(D)^{\odot n}$	floating definition, $(D)^{\odot 0} \equiv 0$

Ad-hoc setting: proactive

Communication paradigm: **broadcast**.

Processes	P	$::=$	f	local macro call
			$\text{def}^{\odot n} D \text{ in } P$	proactive macro definition
			$E P$	public definitions
Definitions	E	$::=$	0	no definition
			$(D)^{\odot n}$	floating definition, $(D)^{\odot 0} \equiv 0$

DEF $\text{def}^{\odot n} D \text{ in } P \longrightarrow (D)^{\odot n} P$

DEFBCAST
$$\frac{n > 0 \quad (l, l') \in G_{com}^a}{a_l[(D)^{\odot n} P] \mid \prod_{b, l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b, l'} b_{l'}[(D)(D)^{\odot n-1} Q]}$$

CALL $(f \triangleright Q)f \longrightarrow Q$

Ad-hoc setting: reactive

Processes	$P ::= f^{\odot n}$	reactive macro call, $n \in \mathbb{N}$
	$\text{def } D \text{ in } P$	local macro definition
	$E P$	public definitions
Definitions	$E ::= 0$	no definition
	(D)	locally floating definition

Ad-hoc setting: reactive

Processes $P ::= f^{\odot n}$ reactive macro call, $n \in \mathbb{N}$
 $| \text{def } D \text{ in } P$ local macro definition
 $| E P$ public definitions

Definitions $E ::= 0$ no definition
 $| (D)$ locally floating definition

DEF $\text{def } D \text{ in } P \longrightarrow (D)P$

CALLBCAST

$$\frac{n > 0 \quad (l, l') \in G_{com}^a}{a_l[f^{\odot n} \mid P] \mid \prod_{b, l'} b_{l'}[Q] \longrightarrow a_l[P] \mid \prod_{b, l'} b_{l'}[f^{\odot n-1} \mid Q]}$$

$$(f \triangleright Q)f^{\odot n} \longrightarrow (f \triangleright Q)^{\odot d}$$

CALL $(f \triangleright Q)f \longrightarrow Q$

Example

An integrated tourist application.



Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	G	\subseteq	\mathcal{A}
Effects	\mathcal{E}	$=$	$\bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	τ	$=$	$\mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	$G \subseteq \mathcal{A}$
Effects	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Policy composition:

$$G \cap H \quad (\tau \circ \sigma)(b)(c) = \tau(b)(c) \cap \sigma(b)(c).$$

Verification: distributed firewalls

Effects and policies, $a_G^\tau[P]$:

Entry policy	$G \subseteq \mathcal{A}$
Effects	$\mathcal{E} = \bigcup_{f \in \mathcal{F}} \{def(f), call(f)\}$
Security policy	$\tau = \mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{P}(\mathcal{E}))$

Policy composition:

$$G \cap H \quad (\tau \circ \sigma)(b)(c) = \tau(b)(c) \cap \sigma(b)(c).$$

Well-typedness:

$$a_G^\tau[P] \text{ well-typed if } a_G^\tau \vdash P \text{ and } \Omega, a_G^\tau \vdash \circ P$$

Errors and type soundness.

And now for something completely similar

Language-based information flow security [Sabelfeld, Boudol]:

- ★ end-to-end confidentiality policy
- ★ access control insufficient

Covert channels:

- ★ implicit flows
- ★ termination channels
- ★ timing channels
- ★ ...

And now for something completely similar

Language-based information flow security [Sabelfeld, Boudol]:

- ★ end-to-end confidentiality policy
- ★ access control insufficient

Covert channels:

- ★ implicit flows
- ★ termination channels
- ★ timing channels
- ★ ...

Awareness:

- ★ incoming context $::= high$, application lines $::= low$
- ★ allow interference, observe changes $low \rightarrow high$

The End