Secure Interaction with an Email Client

Bernhard Beckert   Gerd Beuster

Universität Koblenz-Landau

June 15th, 2006

Verisoft Consortium

AbsInt
Angewandte Informatik

BMW Group

Infineon
technologies

INFORMATIK

OFFIS

OneSpin
SOLUTIONS

T··Systems·

TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITÄT KOBLENZ · LANDAU

UNIVERSITÄT DES SAARLANDES

AGKI
artificial intelligence research koblenz

Verisoft Consortium

AbsInt Angewandte Informatik

BMW Group

Infineon technologies

Goal: Pervasively Verified Systems

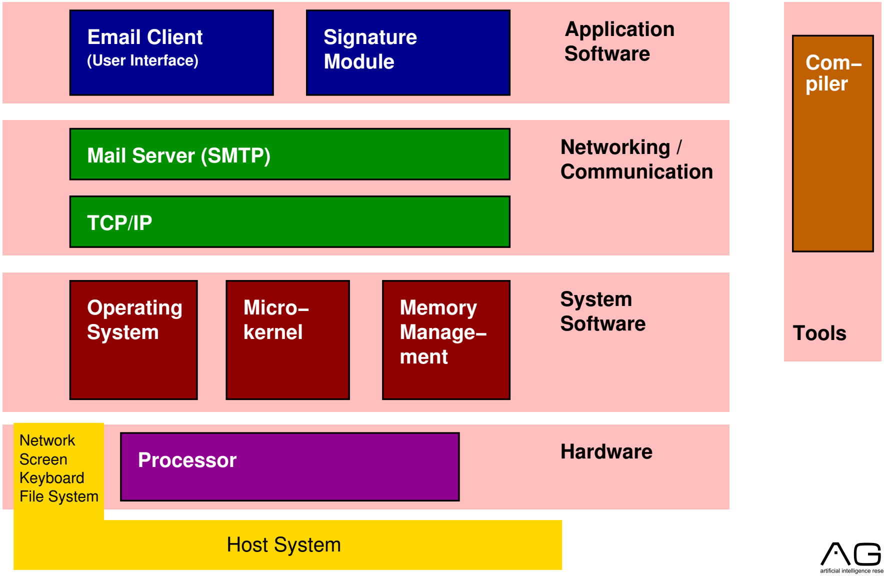
TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITÄT KOBLENZ · LANDAU

UNIVERSITÄT DES SAARLANDES

AGKI artificial intelligence research koblenz

Academic System Overview



Our Involvement & This Talk

Our Involvement

- Development of a Secure Email Client
- PhD thesis about secure user interfaces

This talk

- Demo of the Email System
- Formalizing UI security feature

Verisoft Email System Demo

Demo

Formalizing HCI Security

The user should know what the system is doing.

Consistency

Correspondence between system state and user's opinion about system state

Not guaranteed even if. . .

- the system does show the relevant information (state display conformance).
- the user interprets the information correctly.

Asynchronous updates can lead to inconsistencies!

Formalizing HCI Security

The user should know what the system is doing.

Consistency

Correspondence between system state and user's opinion about system state

Not guaranteed even if. . .

- the system does show the relevant information (state display conformance).
- the user interprets the information correctly.

Asynchronous updates can lead to inconsistencies!

Formalizing HCI Security

The user should know what the system is doing.

Consistency

Correspondence between system state and user's opinion about system state

Not guaranteed even if. . .

- the system does show the relevant information (state display conformance).
- the user interprets the information correctly.

Asynchronous updates can lead to inconsistencies!

Example: Email Client

```
Result of latest command:  No new email arrived
------------------------------------------------










------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
Result of latest command:  No new email arrived
--------------------------------------------------------------










--------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

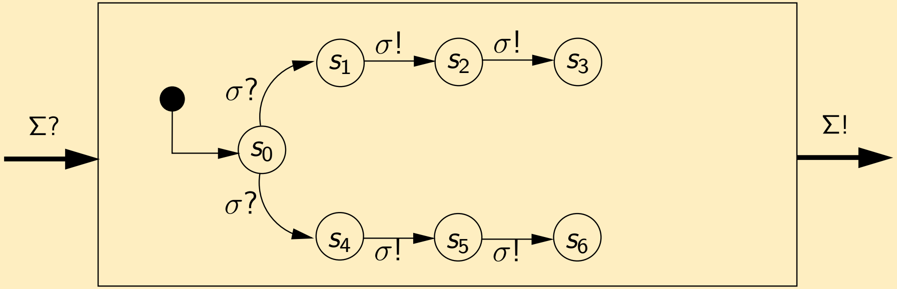
How to Guarantee Consistency?

We need. . .

- a formal method to describe HCI
- a formal definition of consistency
- formal model(s) of interaction(s)

Formal Method

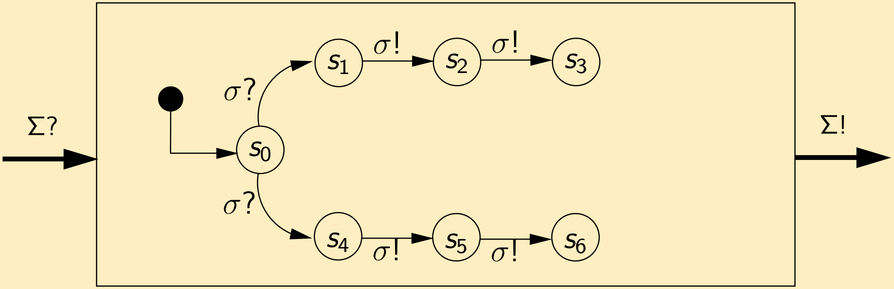
Describe user and application by IOLTS



Define security properties by LTL formulae.

Formal Method

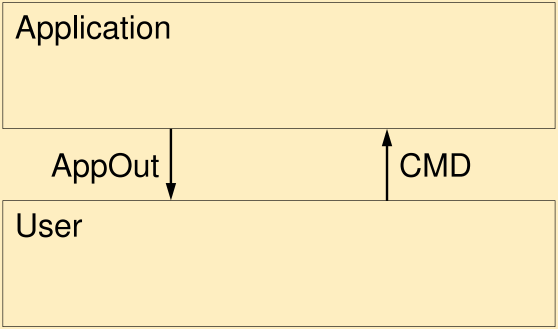
Describe user and application by IOLTS



Define security properties by LTL formulae.

(see also B. Beckert and G. Beuster, “A Method for Formalizing, Analyzing, and Verifying Secure User Interfaces” at ICFEM 2006)

Basic Model



Formal definition of consistency

The user knows what the system is doing.

Let...

- *critical* hold in all critical states of the application
- $a_0, \ldots, a_n$ represent critical properties of the application
- $u_0, \ldots, u_n$ representing the user's assumptions about these properties

Then consistency is defined as...

$$\mathbf{G}(critical \rightarrow ((a_0 \leftrightarrow u_0) \land (a_1 \leftrightarrow u_1) \land \cdots \land (a_n \leftrightarrow u_n)))$$

Formal definition of consistency

The user knows what the system is doing.

Let...

- *critical* hold in all critical states of the application
- $a_0, \ldots, a_n$ represent critical properties of the application
- $u_0, \ldots, u_n$ representing the user's assumptions about these properties

Then consistency is defined as...

$$\mathbf{G}(critical \rightarrow ((a_0 \leftrightarrow u_0) \wedge (a_1 \leftrightarrow u_1) \wedge \cdots \wedge (a_n \leftrightarrow u_n)))$$

Formal definition of consistency

The user knows what the system is doing.

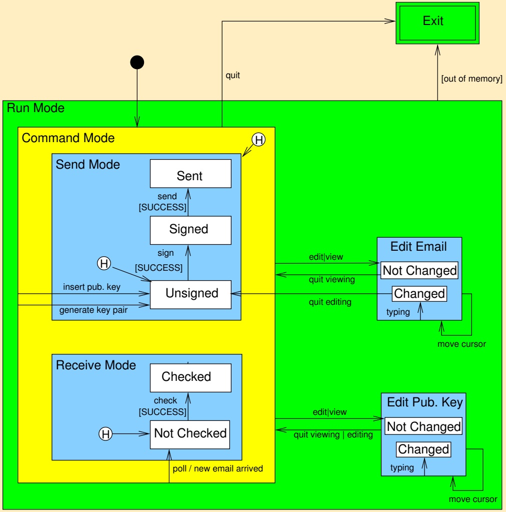
Let. . .

- *critical* hold in all critical states of the application
- $a_0, \ldots, a_n$ represent critical properties of the application
- $u_0, \ldots, u_n$ representing the user's assumptions about these properties

Then consistency is defined as. . .

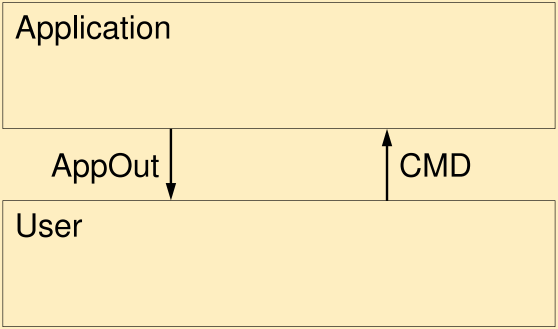
$$\mathbf{G}(\mathit{critical} \longrightarrow ((a_0 \leftrightarrow u_0) \wedge (a_1 \leftrightarrow u_1) \wedge \cdots \wedge (a_n \leftrightarrow u_n)))$$

Application Model

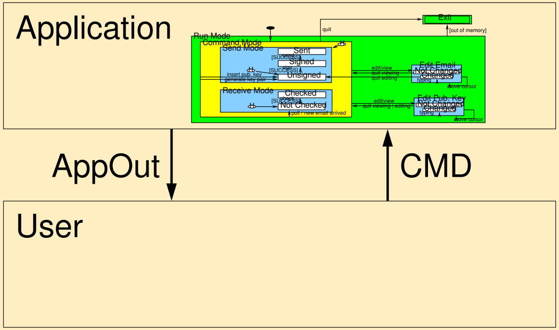


Basic Model

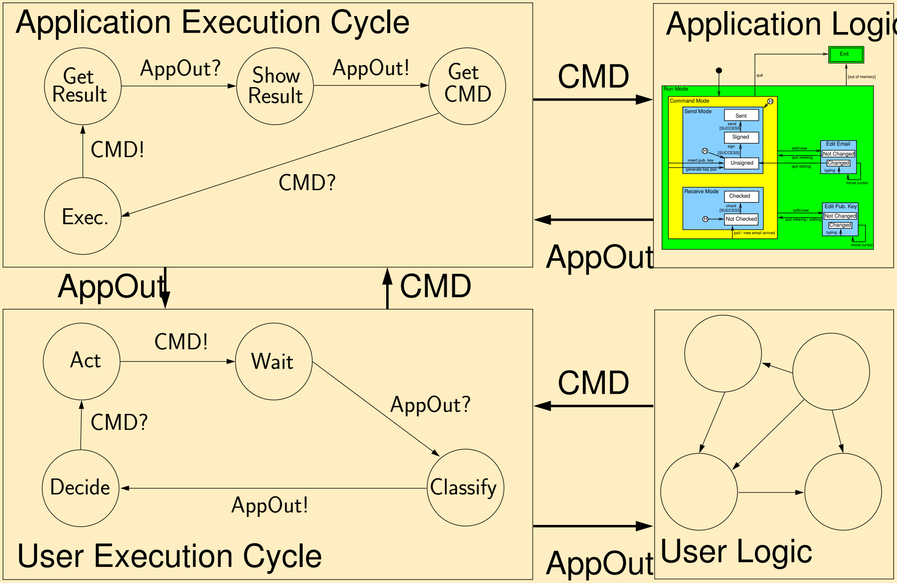
Basic Model



Basic Model with Application



Basic Model with Main Loops



```
Result of latest command:  No new email arrived
--------------------------------------------------------------









--------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
Result of latest command:  No new email arrived
------------------------------------------------------------
```

Problem:
How does the user know if command execution is completed?

```
------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
Result of latest command:  No new email arrived
------------------------------------------------------------




------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Problem:
How does the user know if command execution is completed?

Add a status line!

```
System:  Waiting for input...
Result of latest command:  No new email arrived
---------------------------------------------------------------







---------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
System:  Waiting for input...
Result of latest command:  No new email arrived
-------------------------------------------------------------------









-------------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
System:  Waiting for input...
Result of latest command:  No new email arrived
---------------------------------------------------------------






---------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Problem:
A status line is not enough...

Example: Email Client

```
System:  Waiting for input...
Result of latest command:  No new email arrived
----------------------------------------------------------------



----------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Problem:
A status line is not enough...

Add a “status” key!

Example: Email Client

```
System:  Waiting for input...
Result of latest command:  No new email arrived
-------------------------------------------------------------








-------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

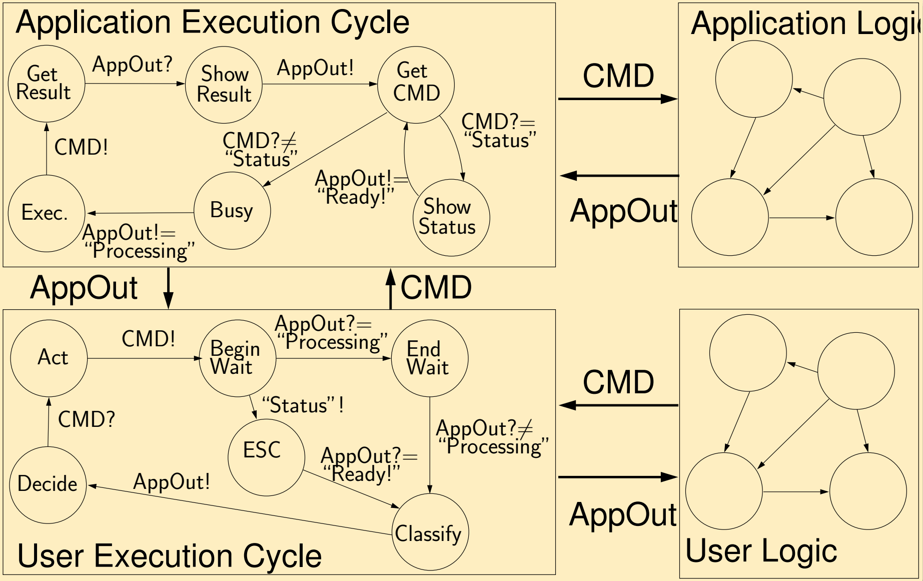
Example: Email Client

```
System:  Waiting for input...
Result of latest command:  No new email arrived
--------------------------------------------------------------






--------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

Example: Email Client

```
System:  Ready!
Result of latest command:  No new email arrived
--------------------------------------------------------------








--------------------------------------------------------------
edit (m)ail | (s)end mail | (f)etch mail
```

A Better Model



Summary

- Developed formal model of HCI describing TTY applications
- Defined consistency
- Showed that common implementations are not consistent
- Proposed a consistent alternative
- Integrated with other methods (not shown in this presentation)