Fortunettes: Une Fonction d'Utilisabilité de Comportement pour les Systèmes Interactifs

David Navarre[1], Philippe Palanque[1-2], Célia Martinie[1] and Kris Luyten[3]

1 Université Toulouse III Paul Sabatier, iCS

2 FlexTech CentraleSupélec-ESTIA Chair

3 Flanders Make, UHasselt EDM

In cooperation with Airbus

Usability Function

Similar to the pending concept of a **security function** (Yoon et al. 2015) or a **safety function** (Lee & Yamada 2010), we argue that feedforward is a **usability function**. While a safety function can be defined as a function added to a system to prevent undesired safety problems (for instance a safety belt in a car does not impact driving capabilities of the driver but only improves safety), we would define a usability function as a function added to an interactive system to prevent undesired usability problems and to globally improve usability (without altering the functionalities offered by the system).

David Navarre, Philippe A. Palanque, Sven Coppers, Kris Luyten, Davy Vanacken. **Model-based Engineering of Feedforward Usability Function for GUI Widgets**. Interacting with Computers 33(1): 73-91 (2021)

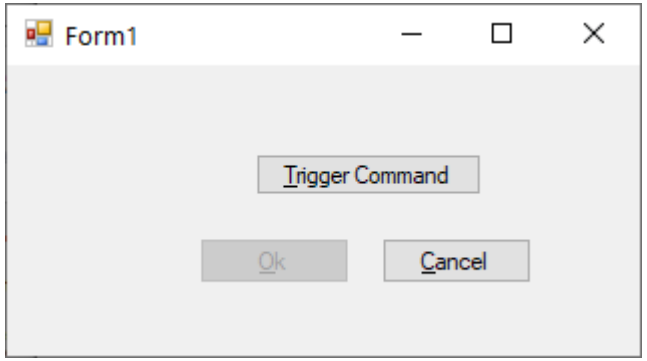
Feedforward

*Feedforward **informs** the user about the result of an action, **before the action** becomes final.*

Djajadiningrat et al. (2002)

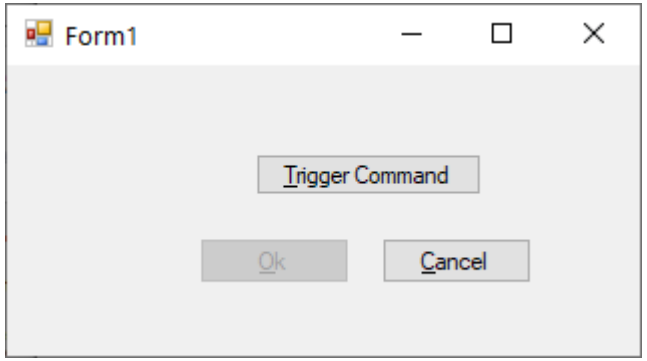
Past, Present and Future within User Interfaces

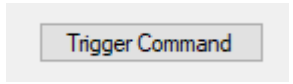
- A User Interface presents
  - The information that is useful for the user is presented (text, labels, graphics)
  - The actions that are available are presented (enabled interactors)
  - The actions that are not available are presented (greyed-out interactors)
- A User Interface does not present
  - Not relevant information to the task is not presented (tab)
  - The actions that will be available after next action
  - The information that will be displayed after next action

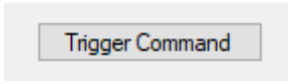


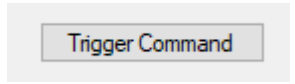
Past, Present and Future within User Interfaces

- A User Interface presents
  - The information that is useful for the user is presented (text, labels, graphics)
  - The actions that are available are presented (enabled interactors)
  - The actions that are not available are presented (greyed-out interactors)
- A User Interface does not present
  - Not relevant information to the task is not presented (tab)
  - The actions that will be available after next action
  - The information that will be displayed after next action
- E.g.
  - This button is available
  - This button is always available (AG)
  - Whatever action you perform the button will not be available next (O)

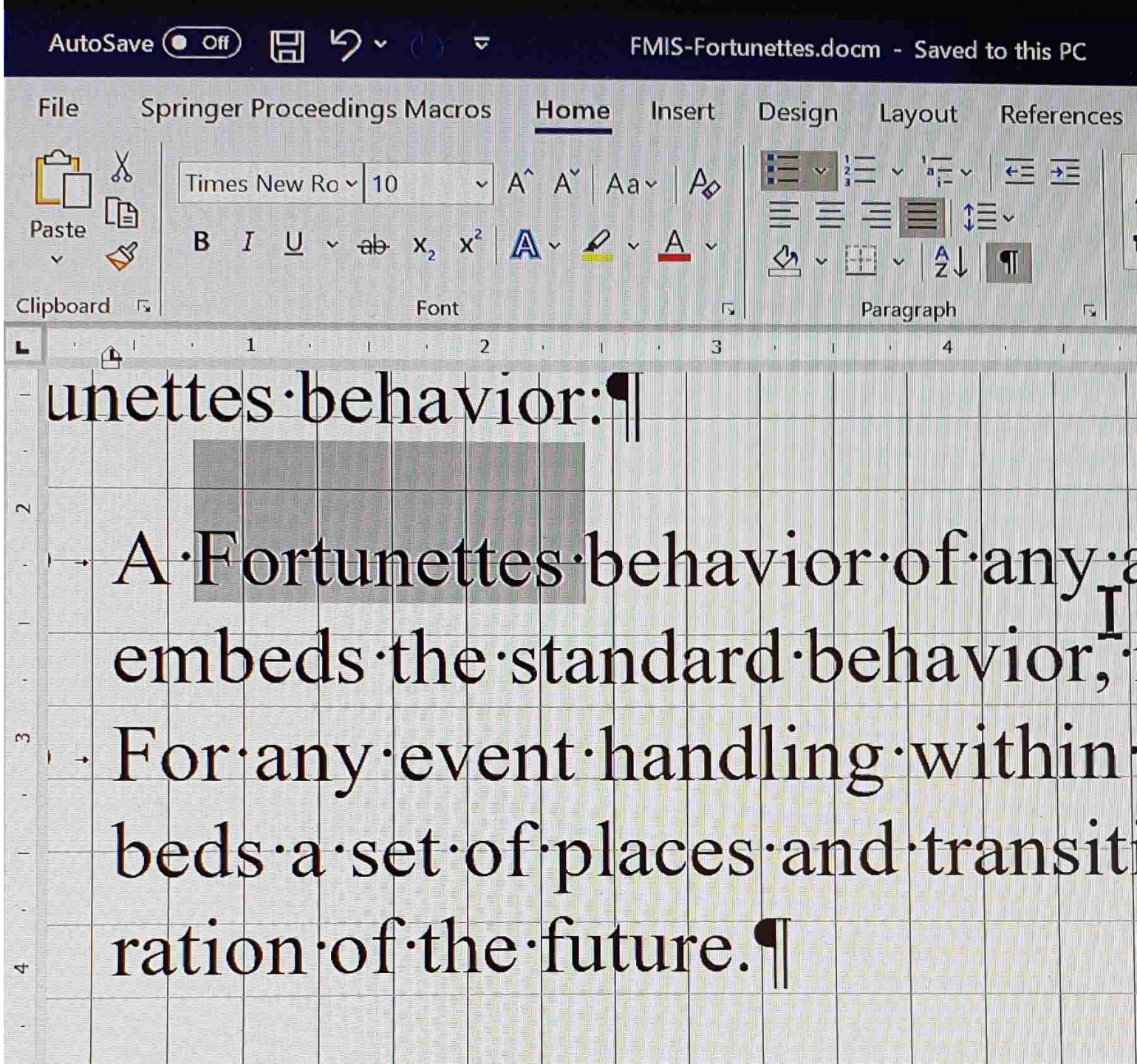




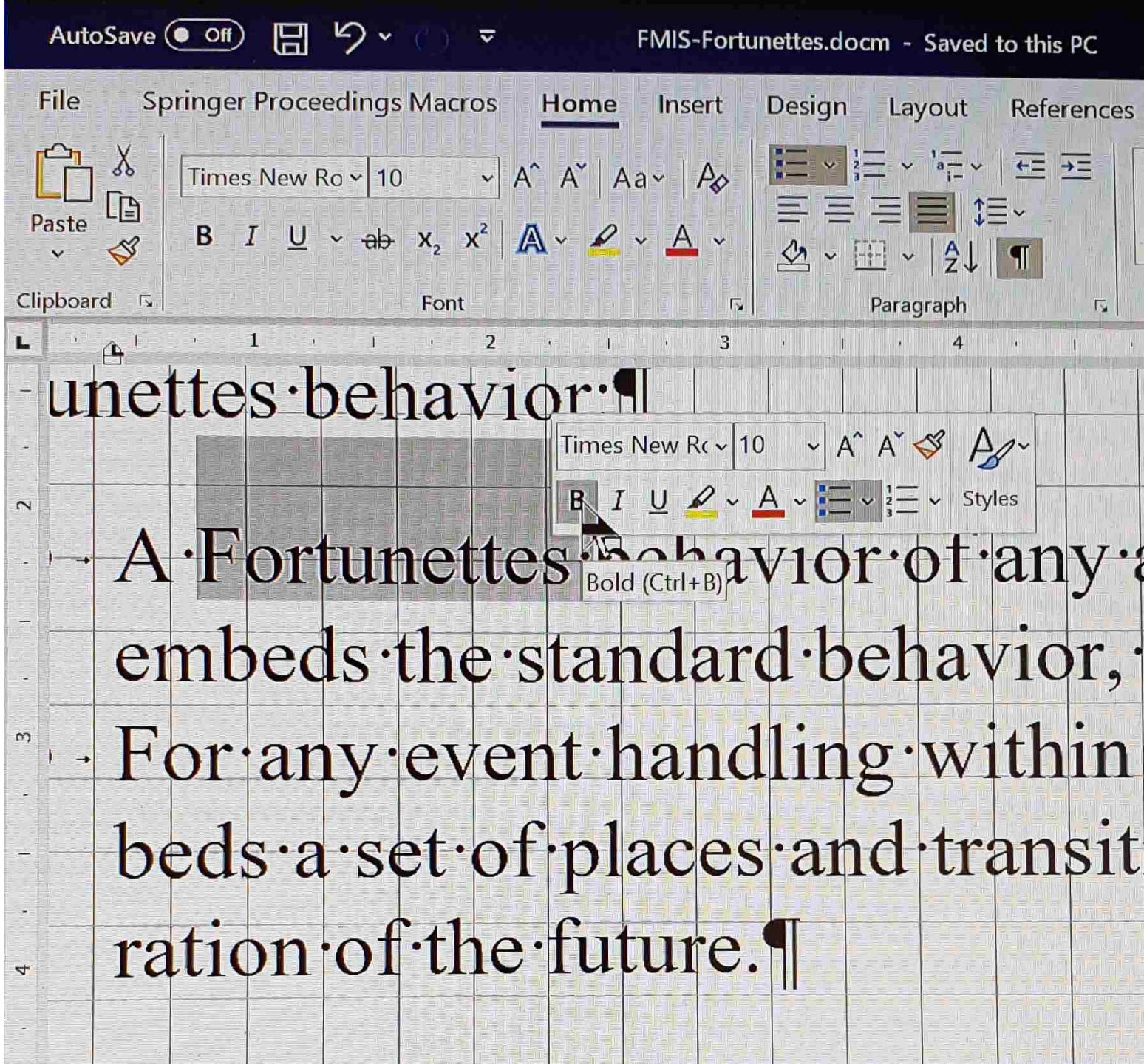




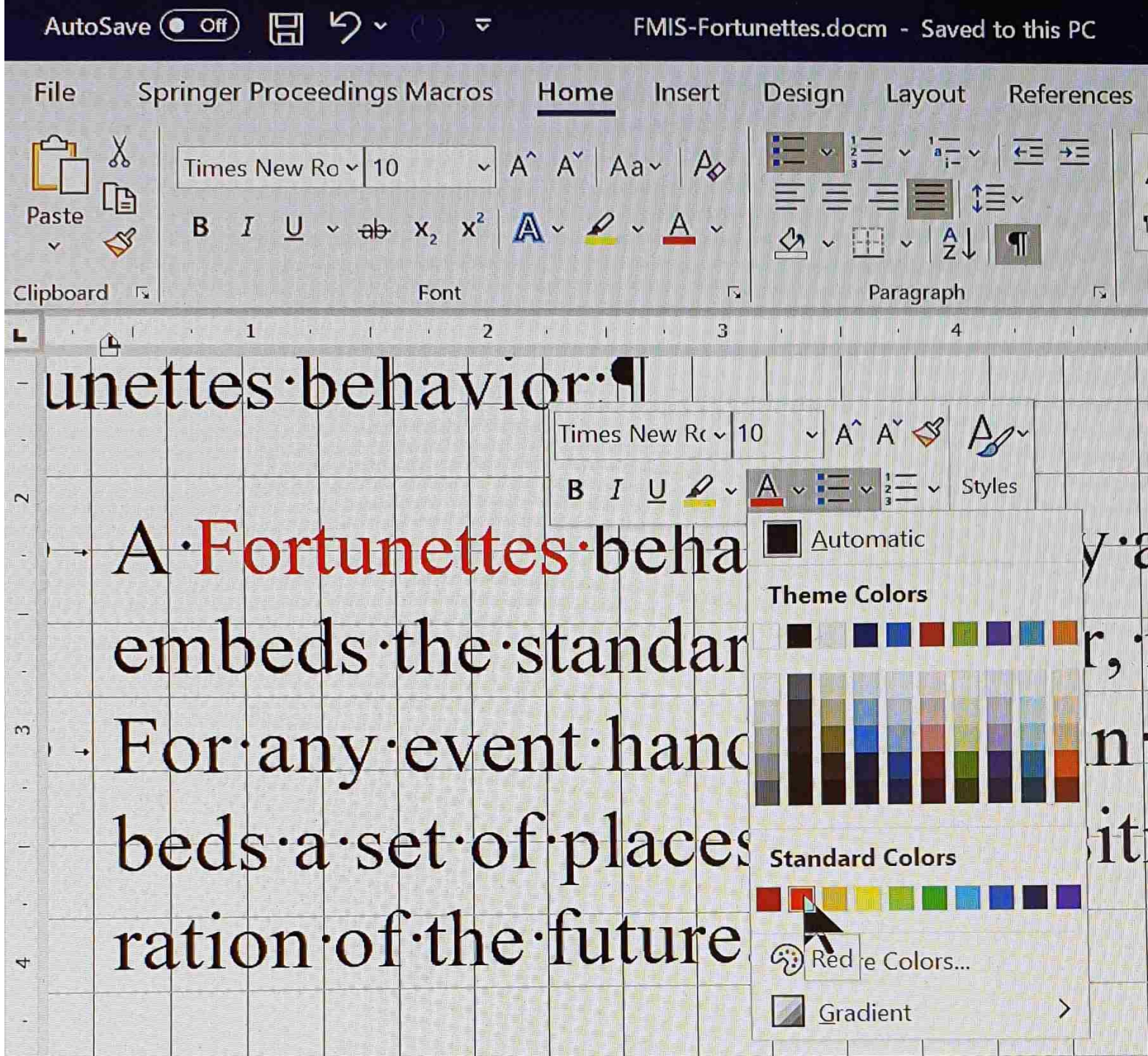
Counter Example



Counter Example

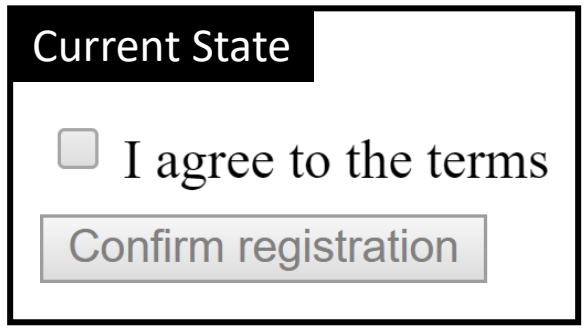


Example


AutoSave Off
FMIS-Fortunettes.docm - Saved to this PC
File
Springer Proceedings Macros
Home
Insert
Design
Layout
References
Paste
Clipboard
Font
Paragraph
unettes·behavior:¶
A·Fortunettes·beha
embeds·the·standar
For·any·event·hanc
beds·a·set·of·places
ration·of·the·future
Styles
Automatic
Theme Colors
Standard Colors
Red
e Colors...
Gradient

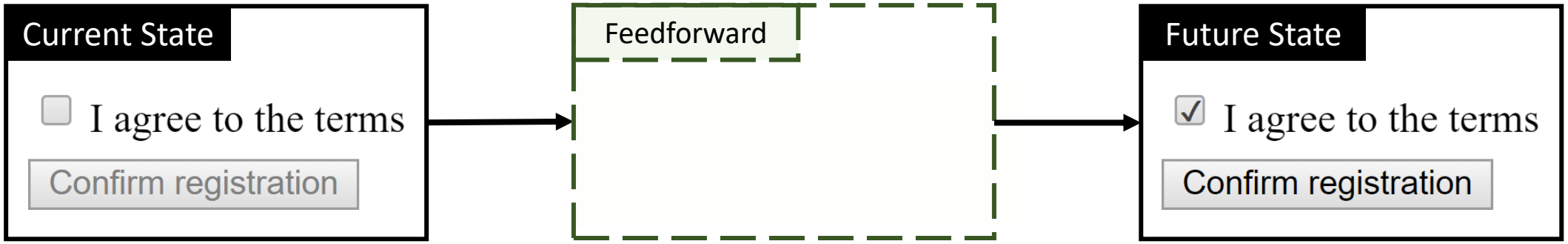
Fortunettes

Feedforward about the Future State of GUI Widgets



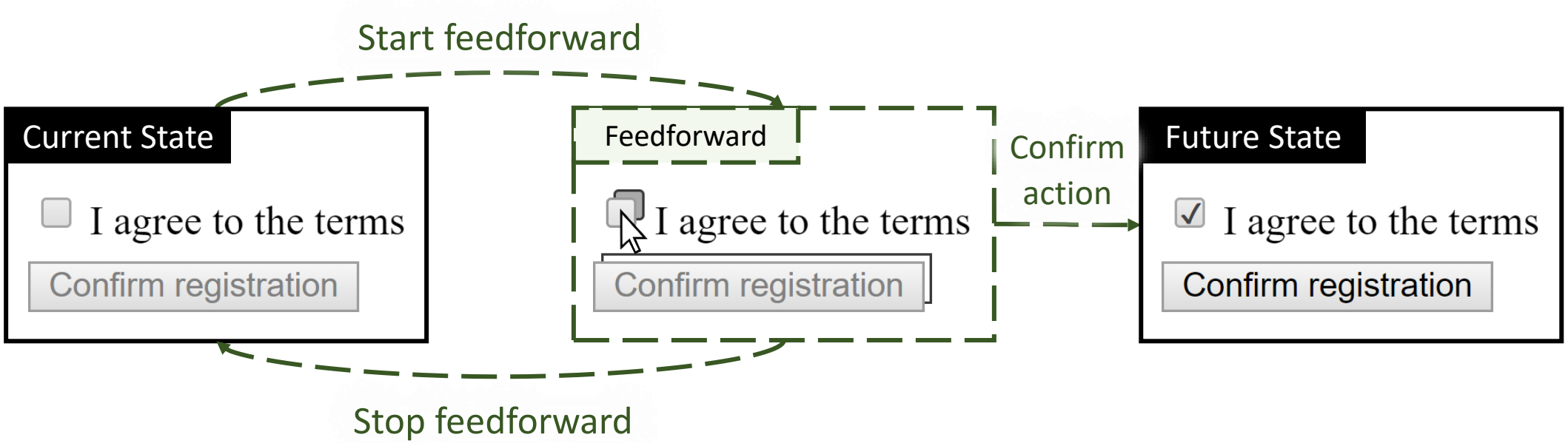
Fortunettes

Feedforward about the Future State of GUI Widgets



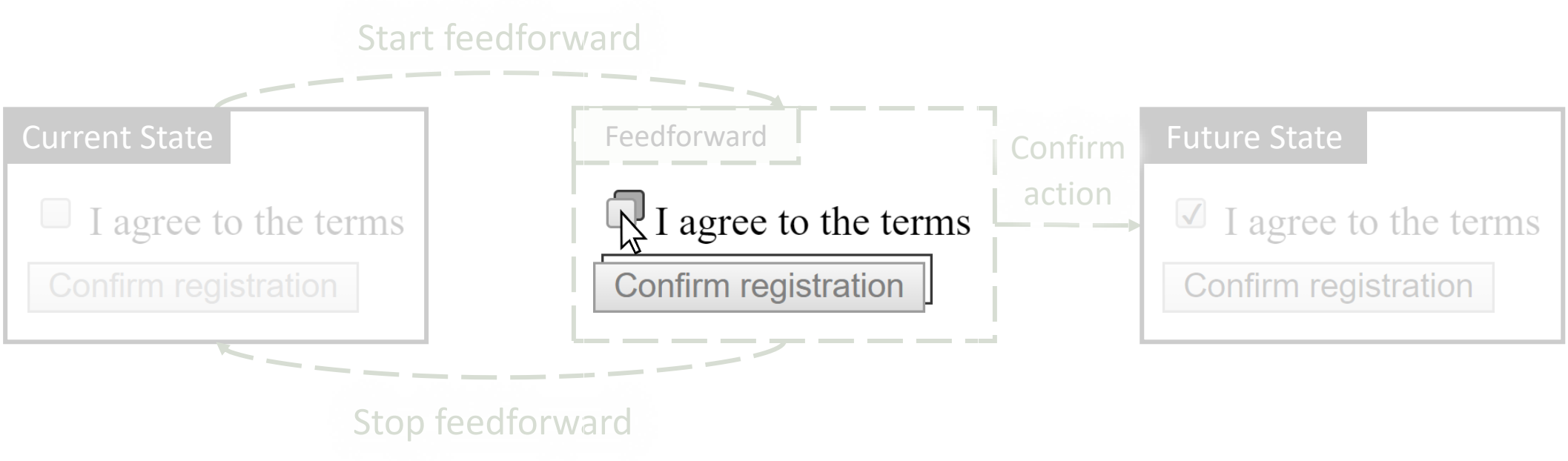
Fortunettes

Feedforward about the Future State of GUI Widgets

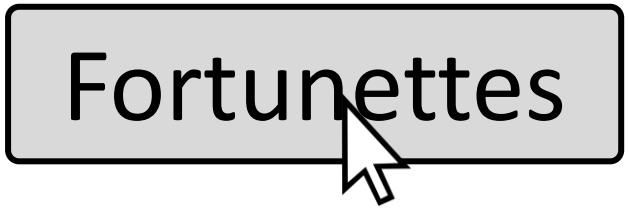


Fortunettes

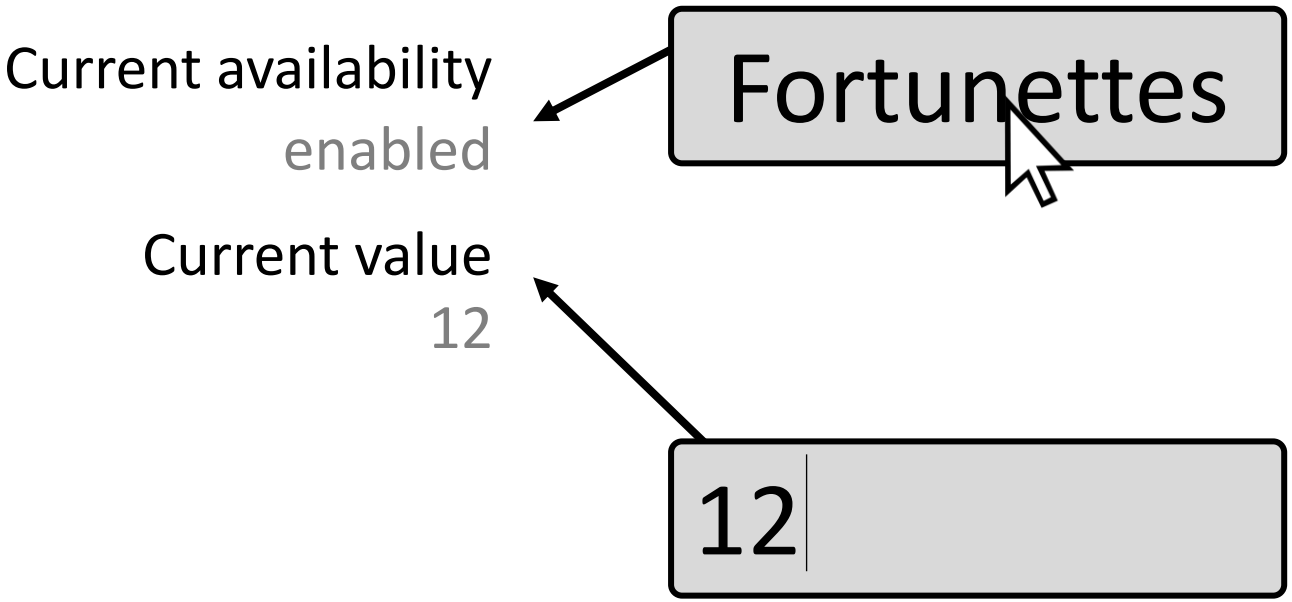
Feedforward about the Future State of GUI Widgets



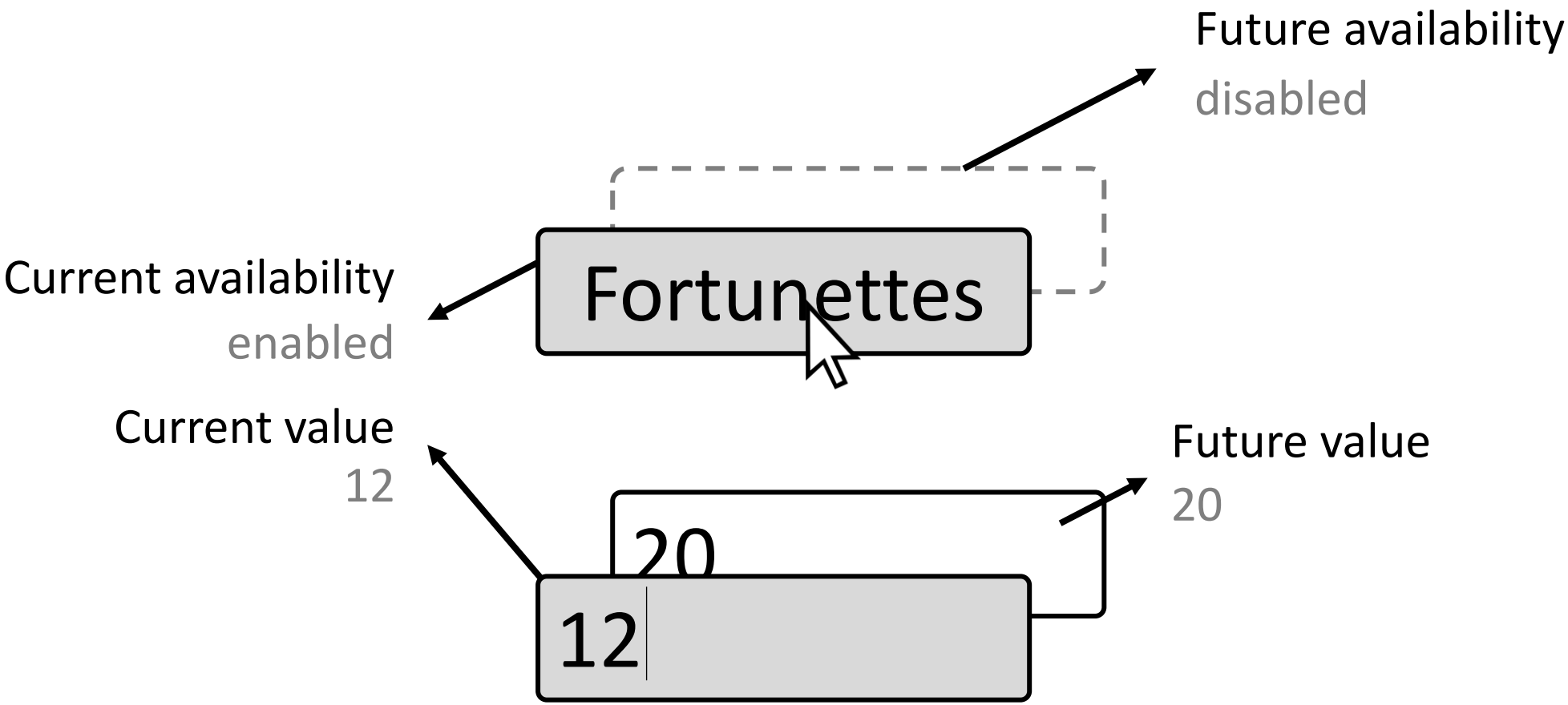
The anatomy of Fortunettes



The anatomy of Fortunettes

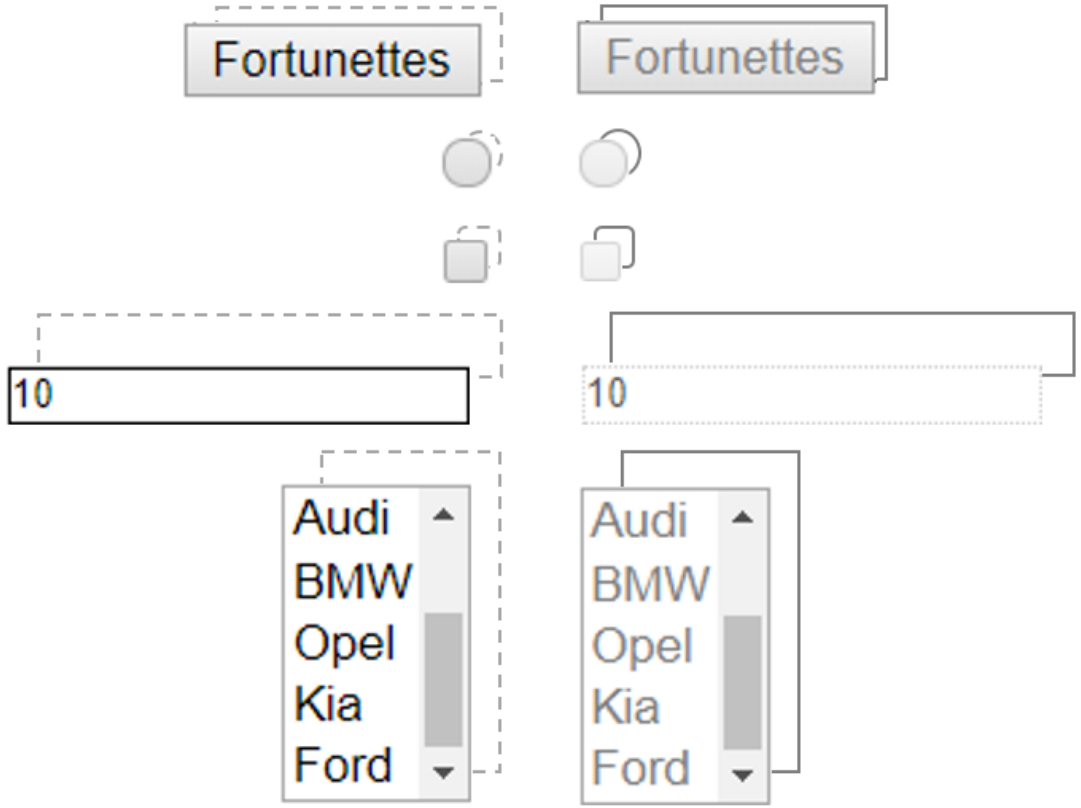


The anatomy of Fortunettes

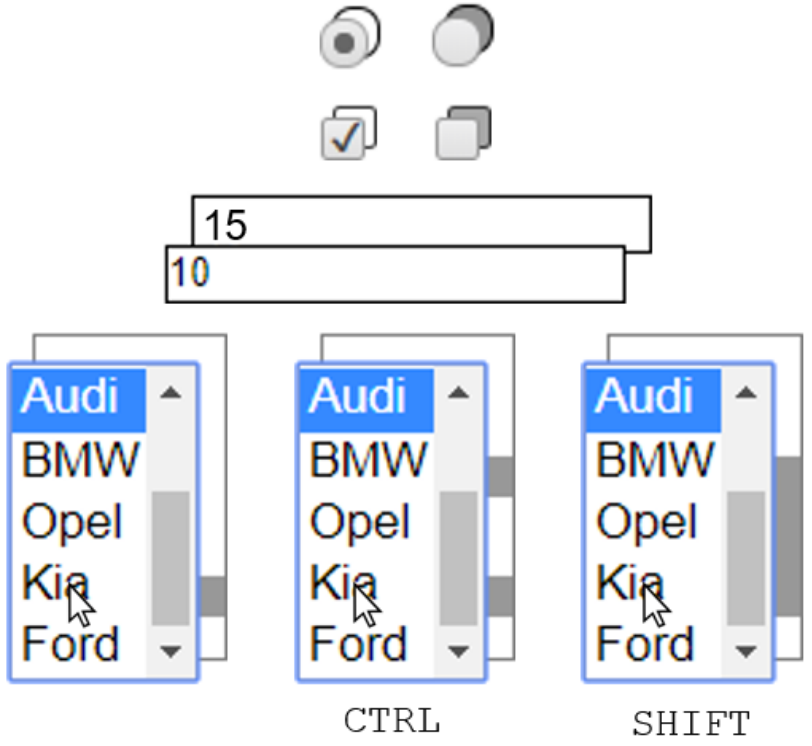


The anatomy of Fortunettes

Coppers et al. 2019. **Fortunettes: Feedforward about the Future State of GUI Widgets**. *Proc. ACM Hum.-Comput. Interact.* 3, **EICS**, Article 20 (June 2019), 20 pages.



Availability

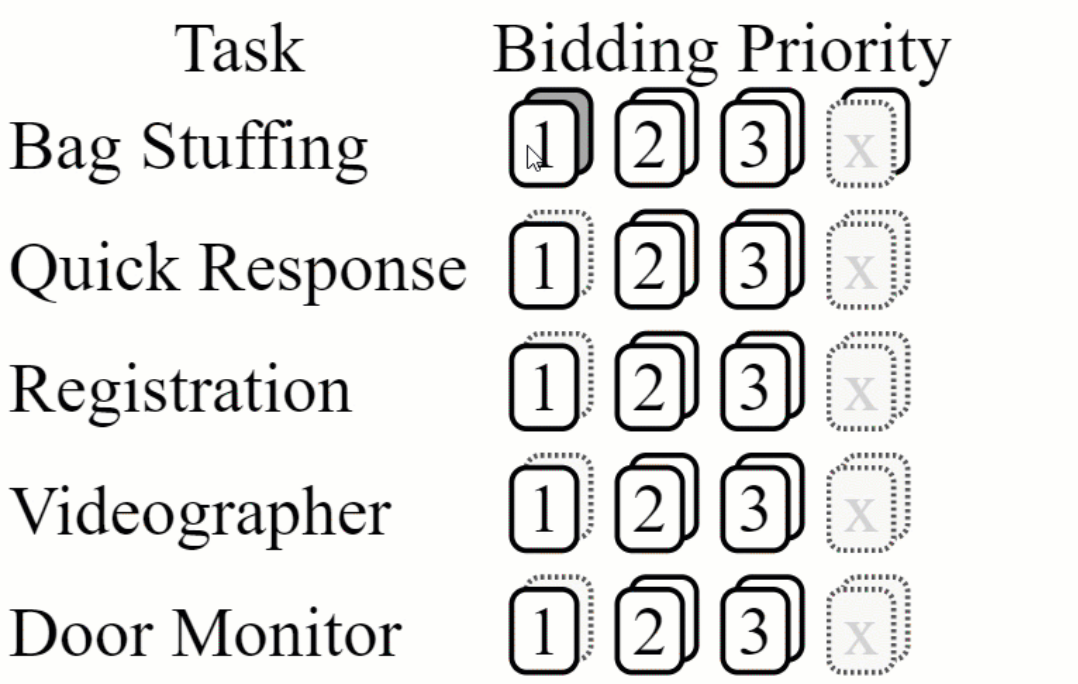


Value

Parsimony Design Principle in Fortunettes

Only show feedforward for changing widgets

An action might affect multiple widgets



Proposed Contribution: Formal Engineering of Fortunettes

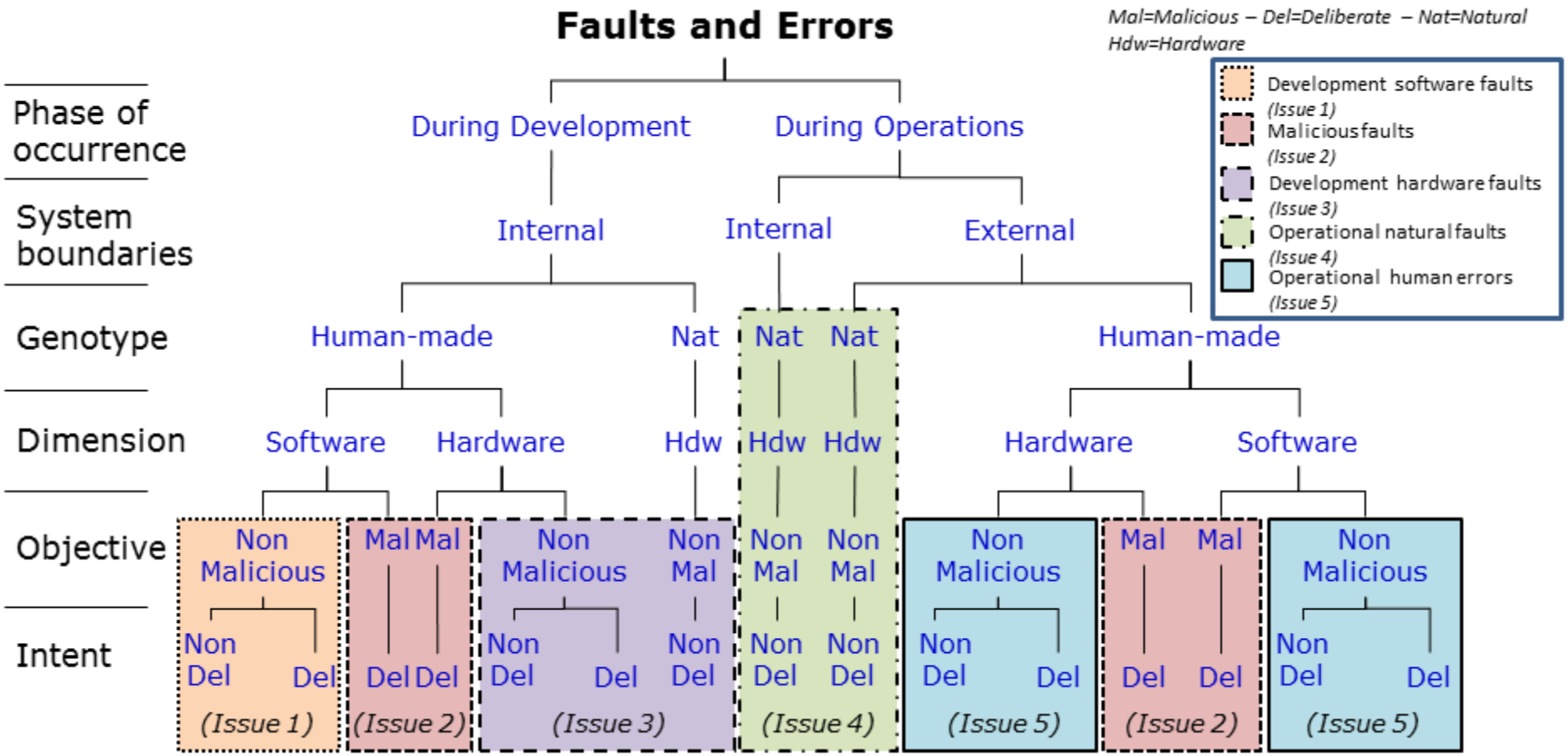
Rationale

A Petri nets-based solution (modeling, verification, simulation)

Future work

Adapted from: Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C. **Basic concepts and taxonomy of dependable and secure computing**. In IEEE Trans. on Dependable and Secure Computing, vol.1, no.1, pp. 11- 33, Jan.-March 2004

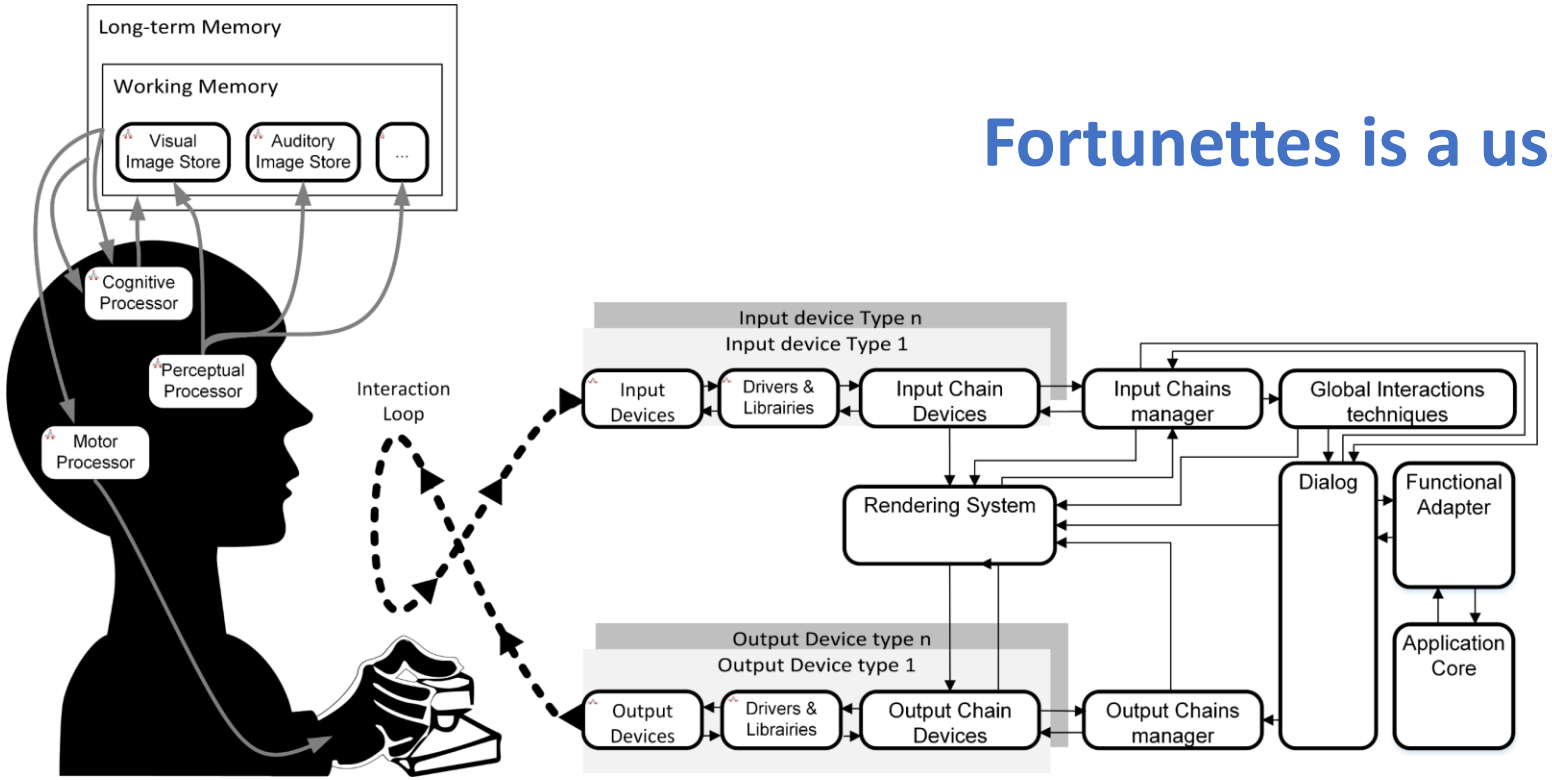
Rationale: Fault Prevention



Fayollas C. et al. (2017) Dealing with Faults During Operations: Beyond Classical Use of Formal Methods. **The Handbook of Formal Methods in Human-Computer Interaction**. Human–Computer Interaction Series. Springer,

Fault Model in a Nutshell

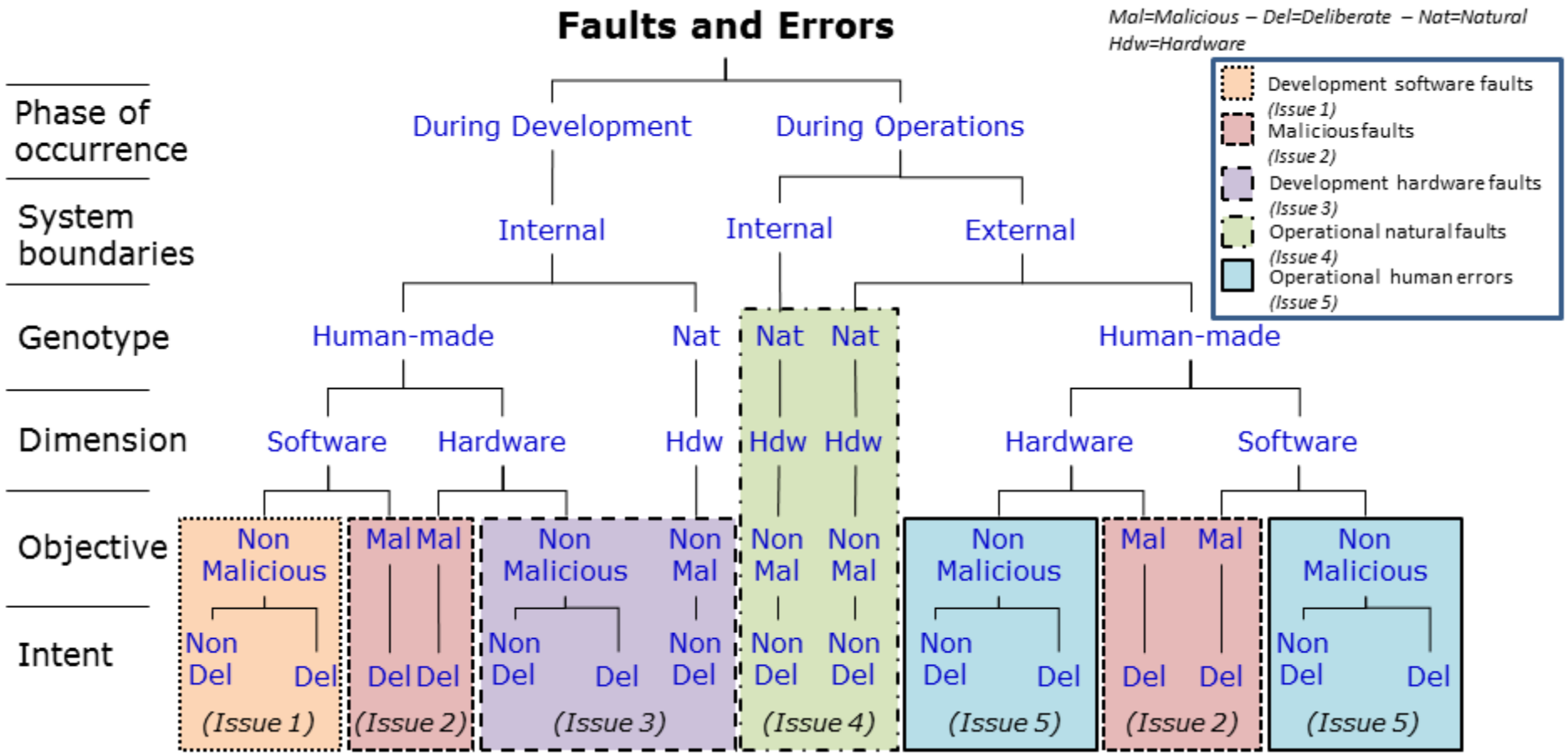
- Insufficient guidance induces faults in the human (who enters an error mode)
- When interaction is needed with the interactive system, failure (called human errors) may occur
- Fortunettes aims at preventing the occurrence of failures triggered by these faults



Fortunettes is a usability function

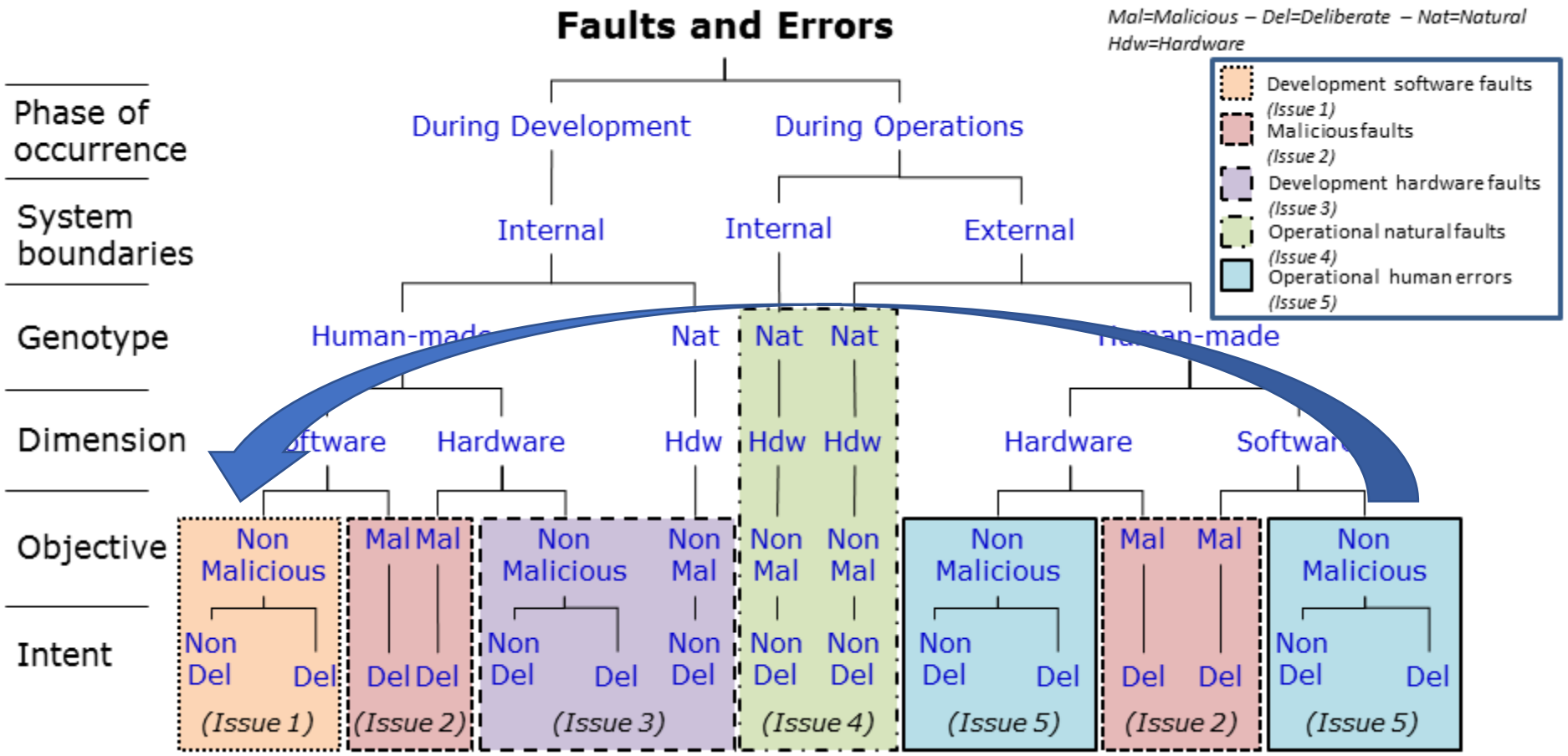
Martin Cronel, Bruno Dumas, Philippe A. Palanque, Alexandre Canny. **MIODMIT: A Generic Architecture for Dynamic Multimodal Interactive Systems.** IFIP WG 13.2 Human Centered Software Engineering 2018: 109-129

Rationale: Fault Prevention



C. Fayollas, et al. **An Approach for Assessing the Impact of Dependability on Usability: Application to Interactive Cockpits**, *2014 Tenth European Dependable Computing Conference*, Newcastle, 2014, pp. 198-209.

Rationale: **Induced** Fault Prevention



Avoiding:
- **operation-time**
- **human made**
- **non malicious**
- **non deliberate**

errors might induce higher number of development faults

C. Fayollas et al. **An Approach for Assessing the Impact of Dependability on Usability: Application to Interactive Cockpits**, *2014 Tenth European Dependable Computing Conference*, Newcastle, 2014, pp. 198-209.

A Formal Approach to Support “Fortunetting” User Interfaces

- Similar to **undo**: crosscutting concerns throughout the interactive application
- A significant quantity of code has to be added
- The code to be added requires a deep understanding of the application state space
  - Event-handlers programming does not support that
  - Programming approaches such as Agile methods
    - Prevent from using models
    - Favor quick and dirty code delivery
    - Increase Tech Debt (Technology debt) pushing developers to choose simple and minimal solution – unlikely they will integrate Fortunettes in their backlog
- The interaction to be added increases complexity of event handlers

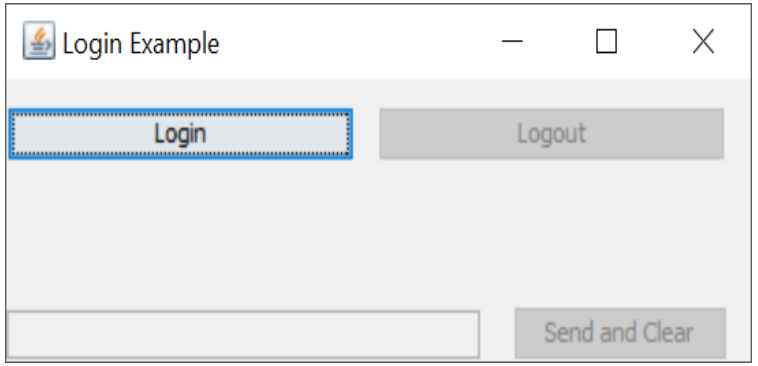
Fortune Nets: using Petri nets to support “Fortunetting” User Interfaces

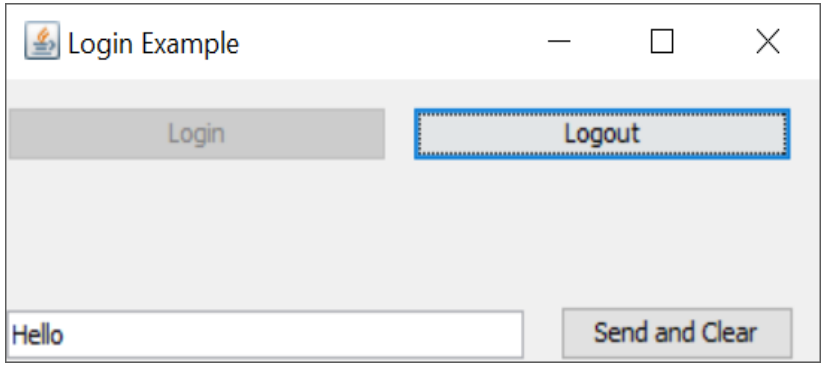
- User Interaction is based on extending widgets behavior
  - To trigger the fortunettes events
  - To render the future states
- ICO Petri nets based modelling
  - Object Oriented structuring of interactive applications
  - Applicable at widget level, application level, server levels
  - An Object Petri net describing the behavior of each class
  - UI event trigger Petri nets’ transitions – **activation function**
  - Tokens (add, remove, test) trigger rendering on the UI – **rendering function**
  - Meta-level events for **activation rendering**
    - None of the transitions associated to an event are available – UI objects disabling
    - At least one of the transitions associated to an event is available - UI objects enabling

Engineering Approach for Fortunettes

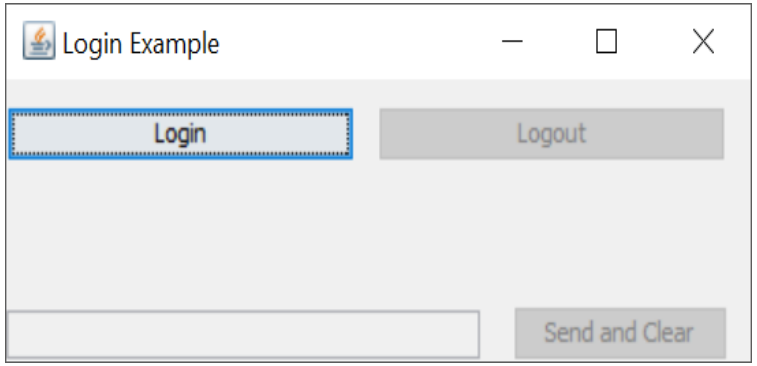
- The behavioral model of the application remains untouched (ICO model)
- An additional model is produced to describe the Fortunettes behavior ICO Fortunettes model)
  - Duplicates the application behavior
  - Does not incorporate Functionnal core method calls
  - Adds to that behavior the Fortunettes pattern for each event
  - The content of the model is generated automatically (not the appearance that has to be tuned by the engineer)

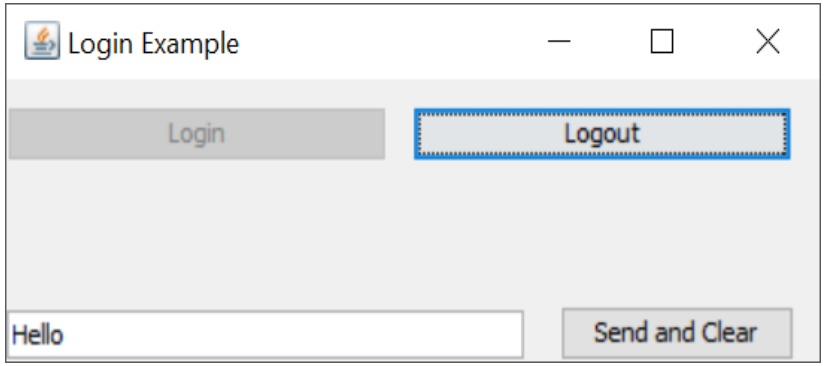
Login – Logout: an Illustrative Example

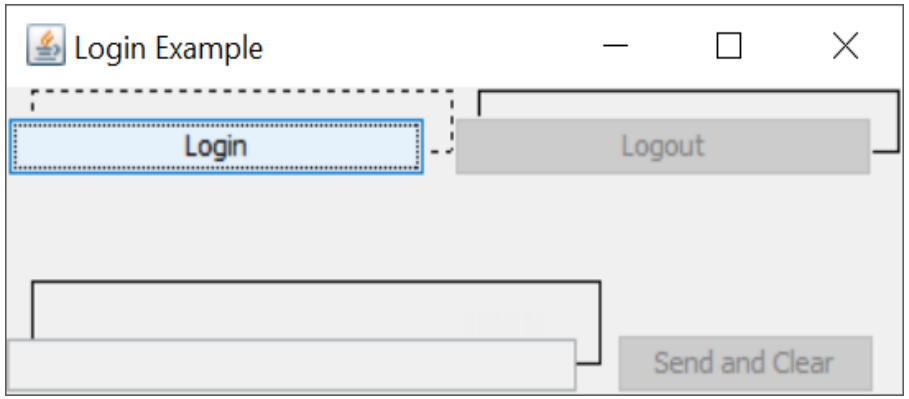




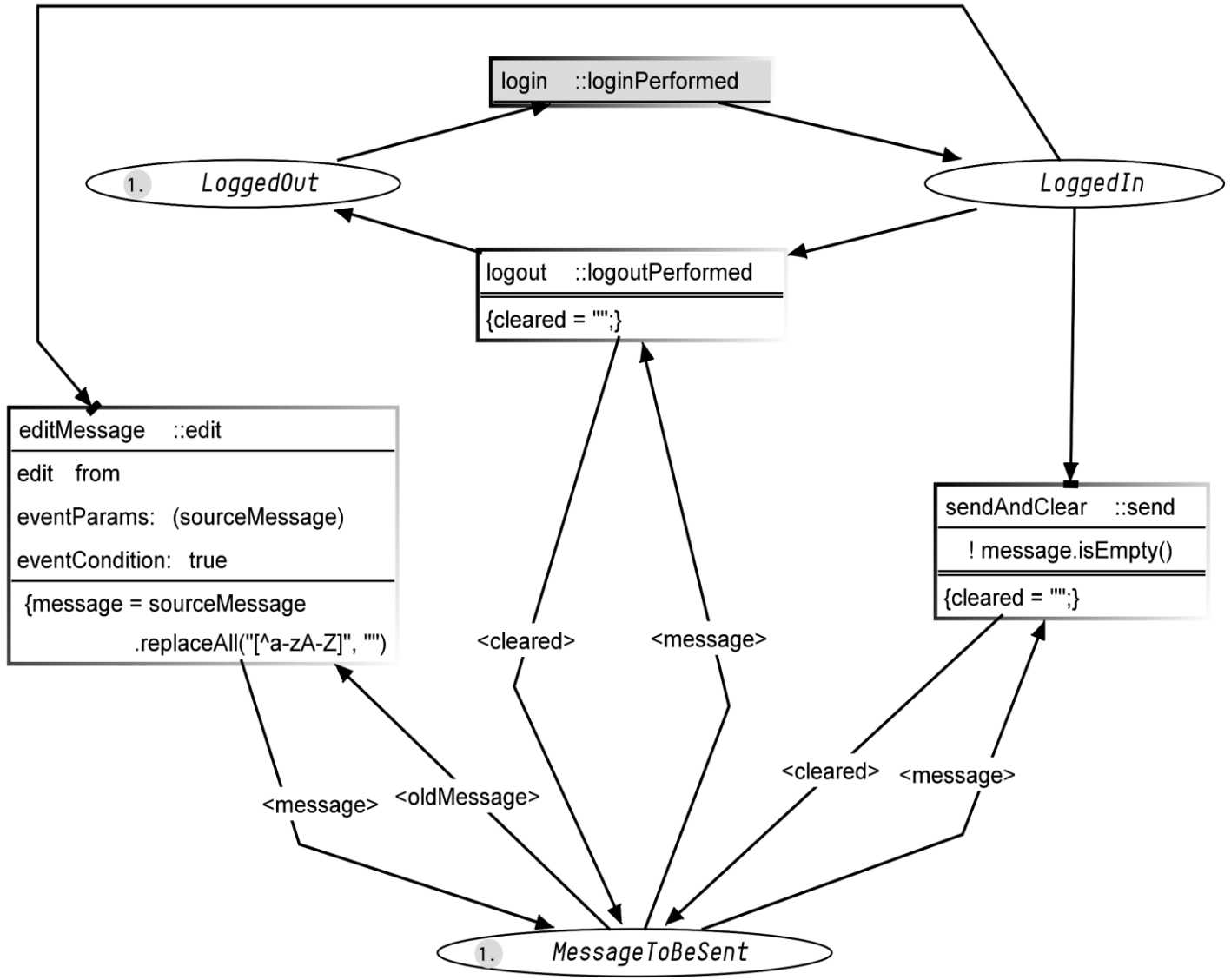
Login – Logout: an Illustrative Example

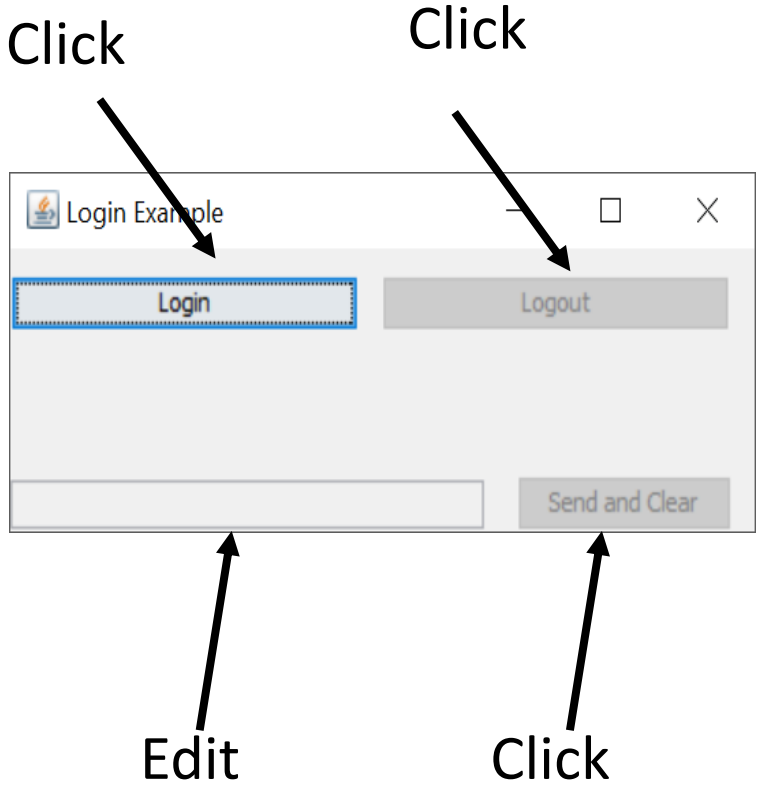




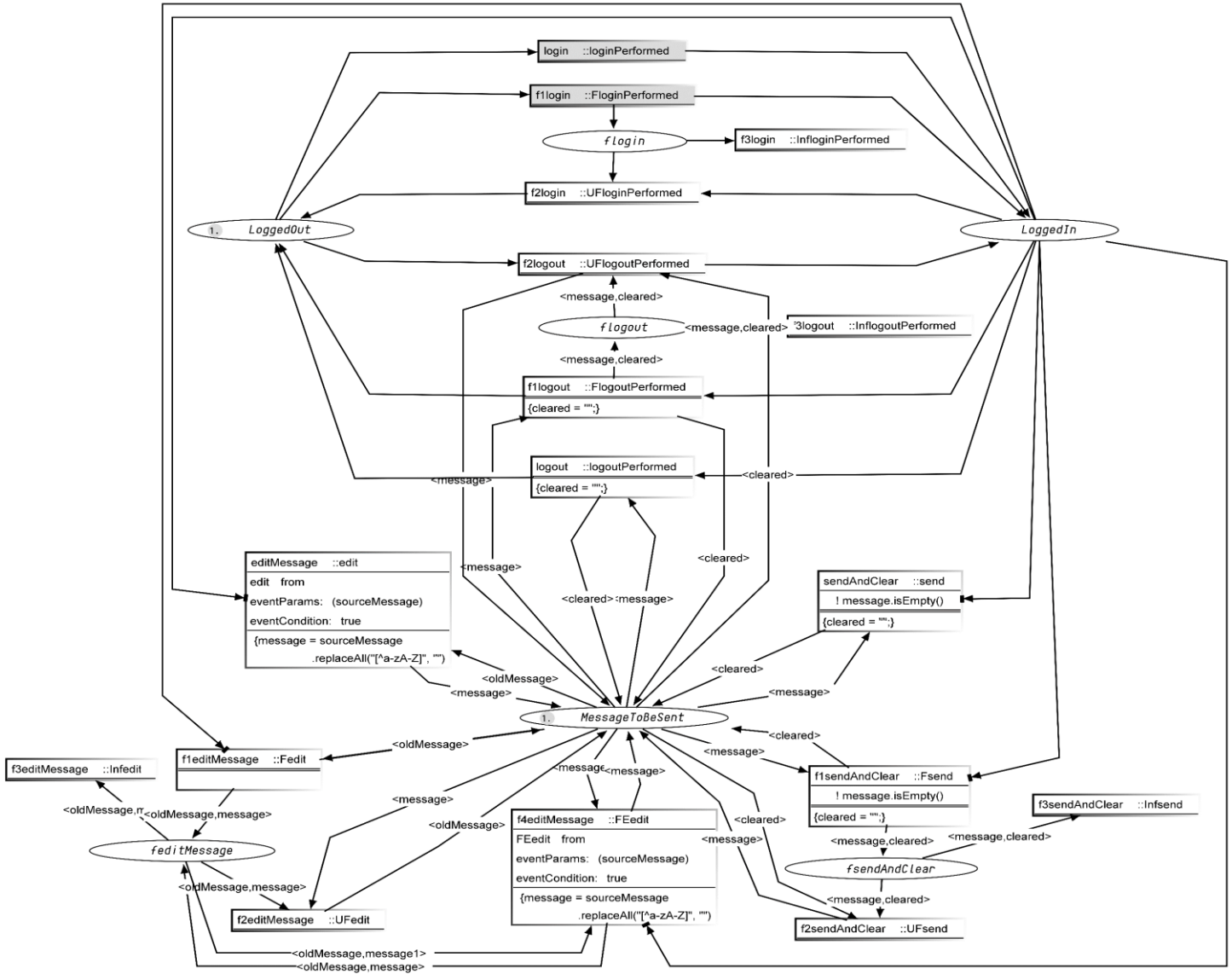


Petri net Behavior of the Application

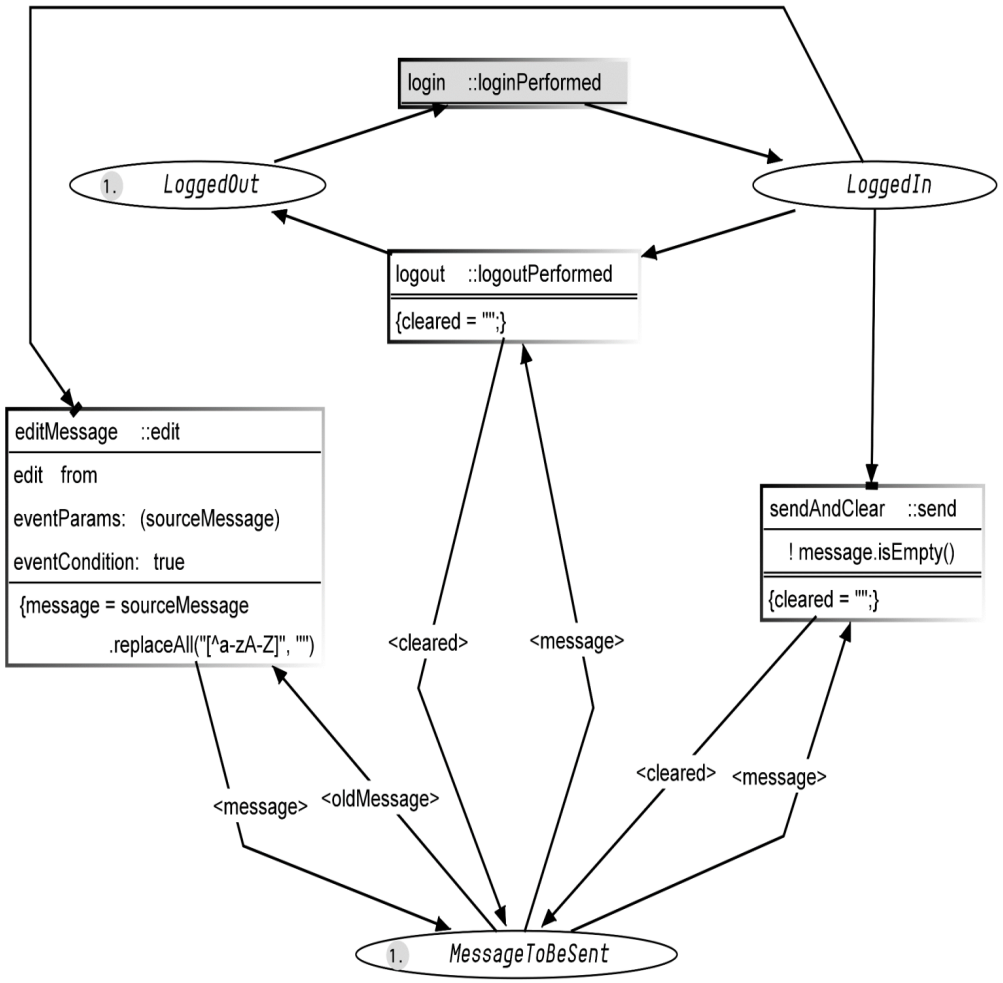


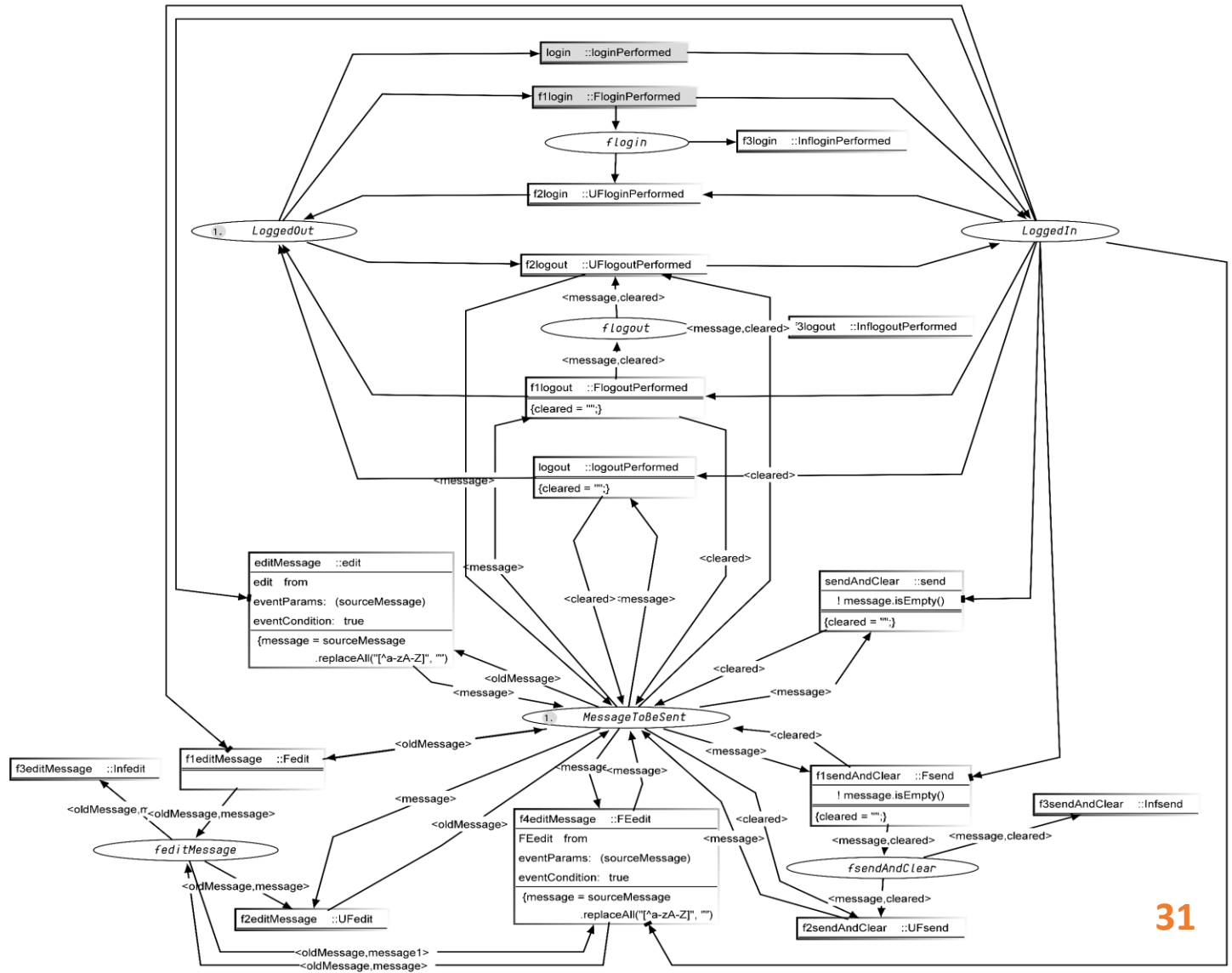


Fortune net of the Application

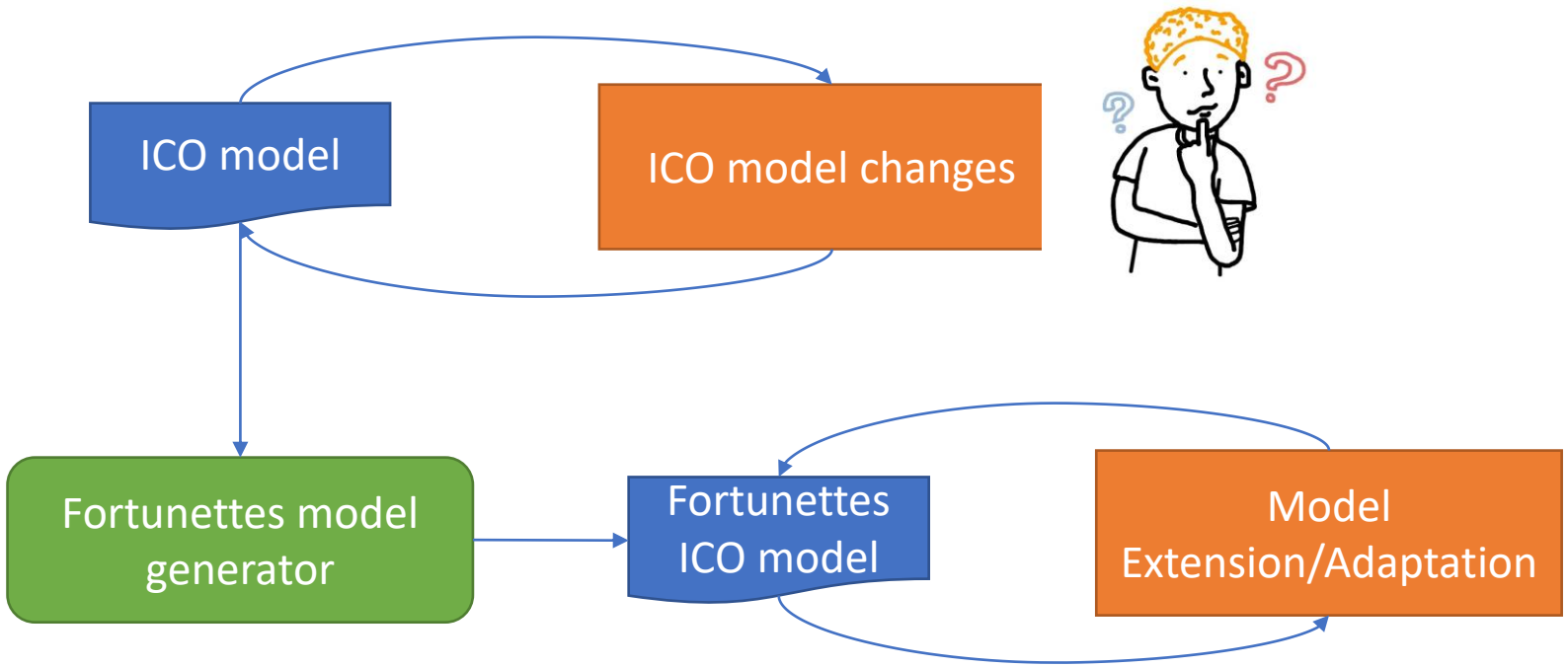


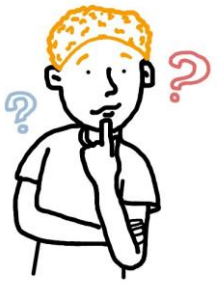
Comparing the Application Model and Fortune net of the Application





Generation Process: how the Fortunettes model is produced

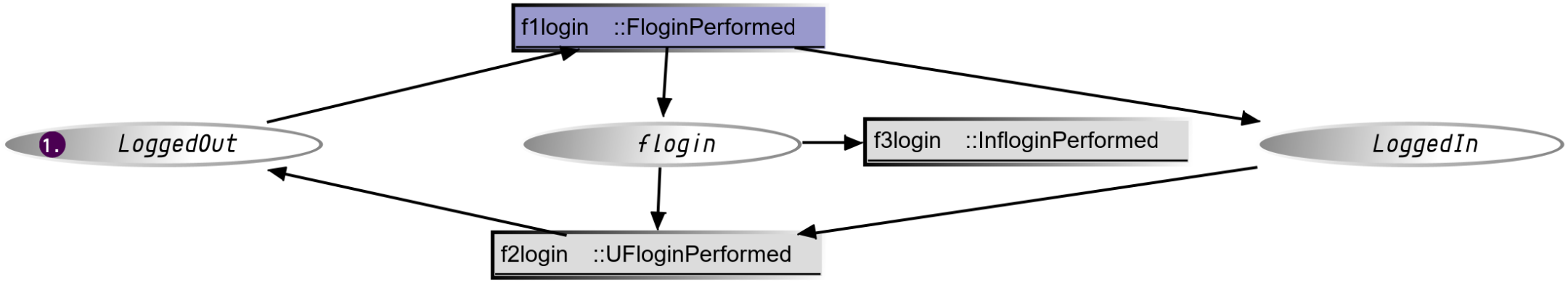




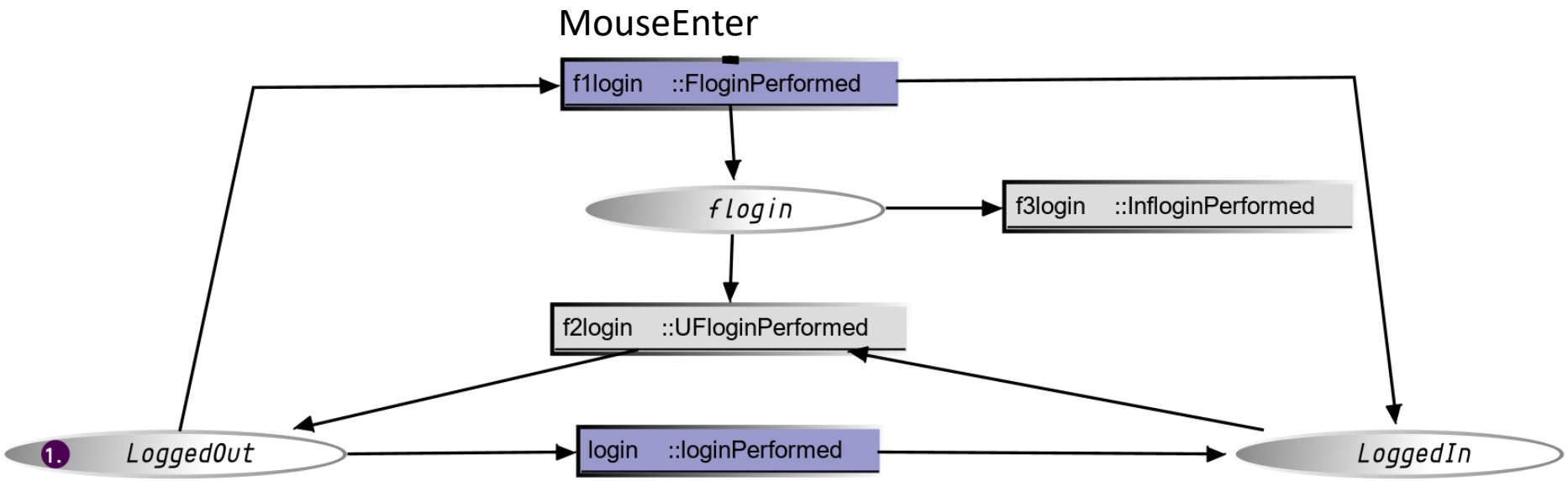
Fine tuning based on NO display based on functional core methods not triggered

Login Widget (button)

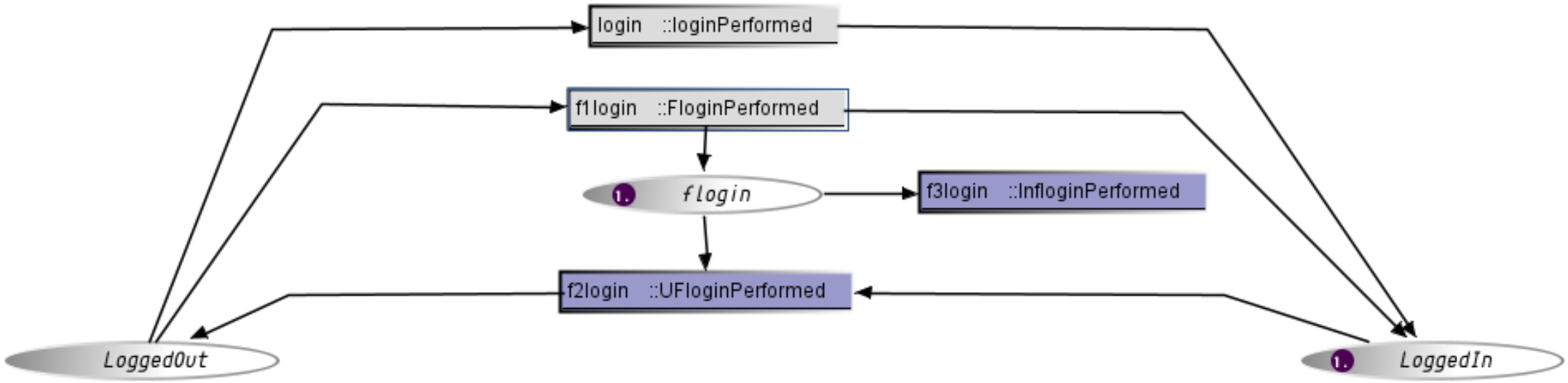
Fortunettes Login Simplified



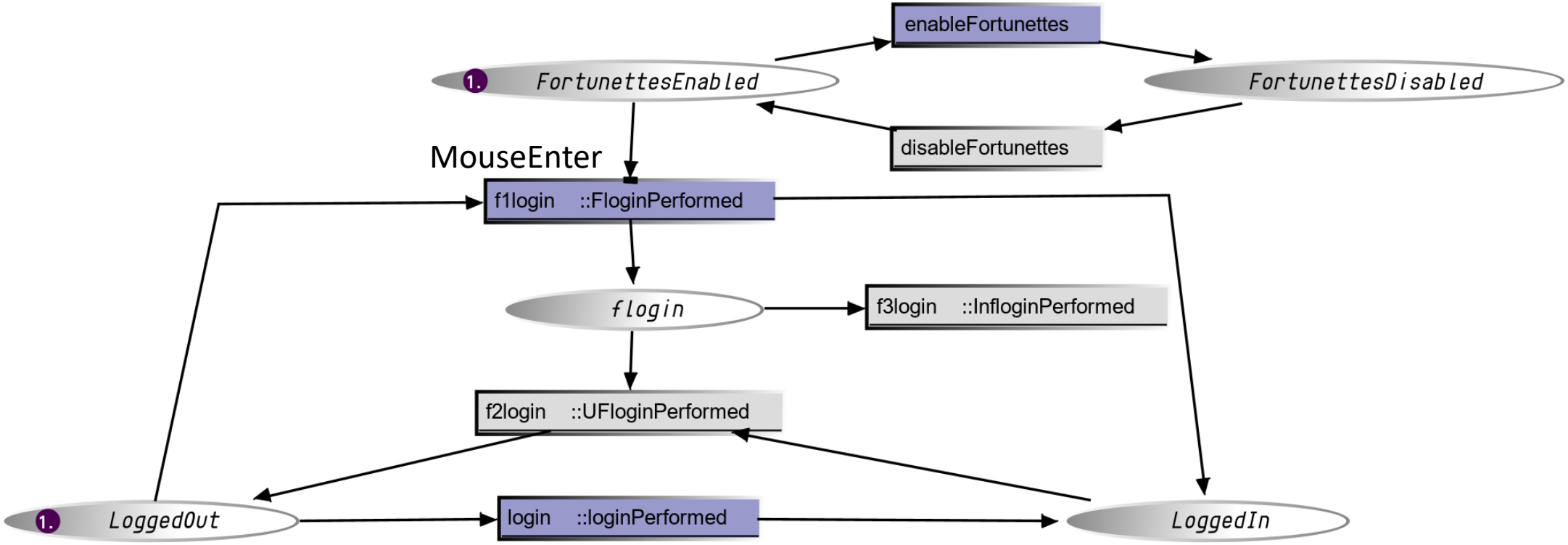
Complete behaviour of the widget



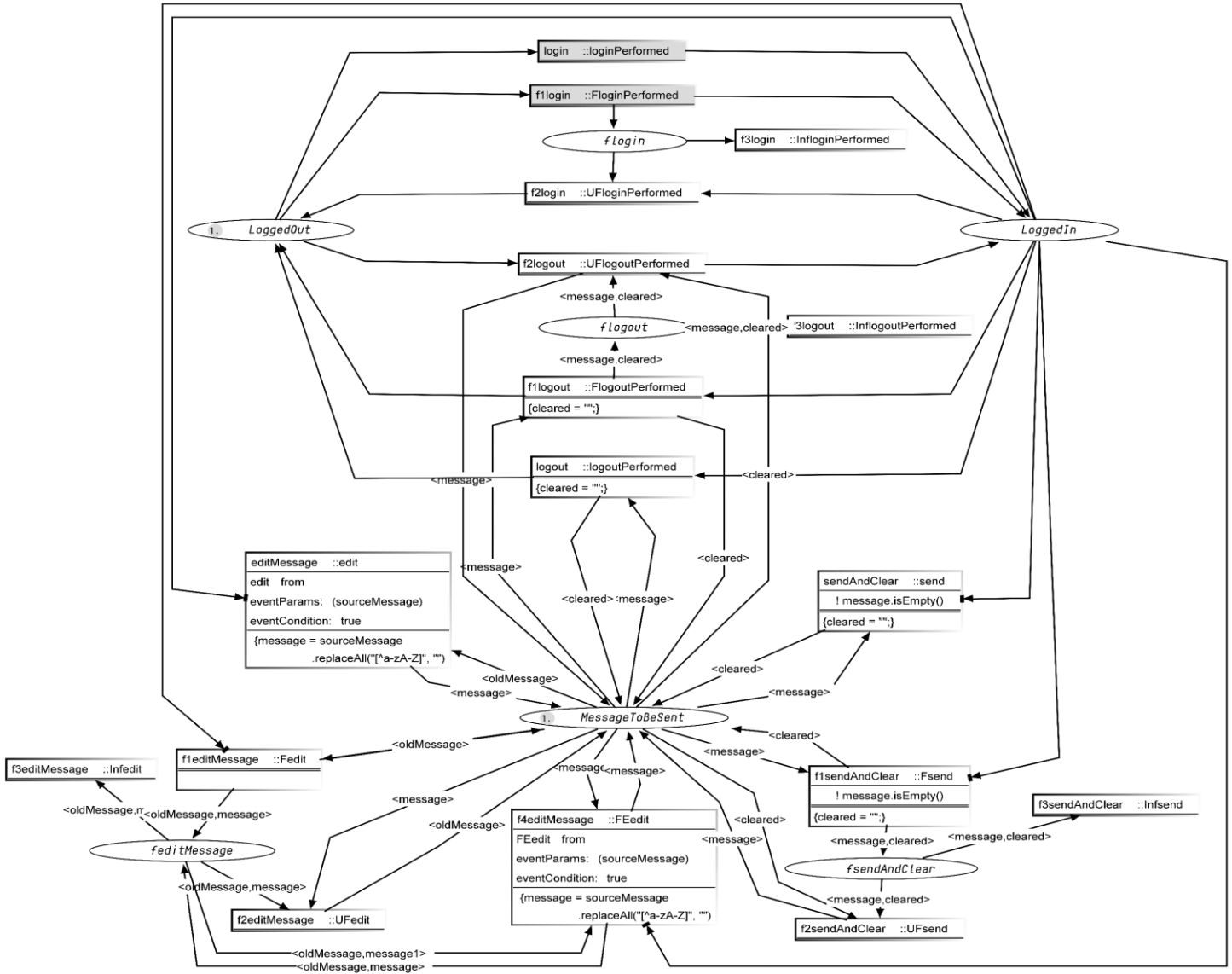
Complete behaviour of the widget



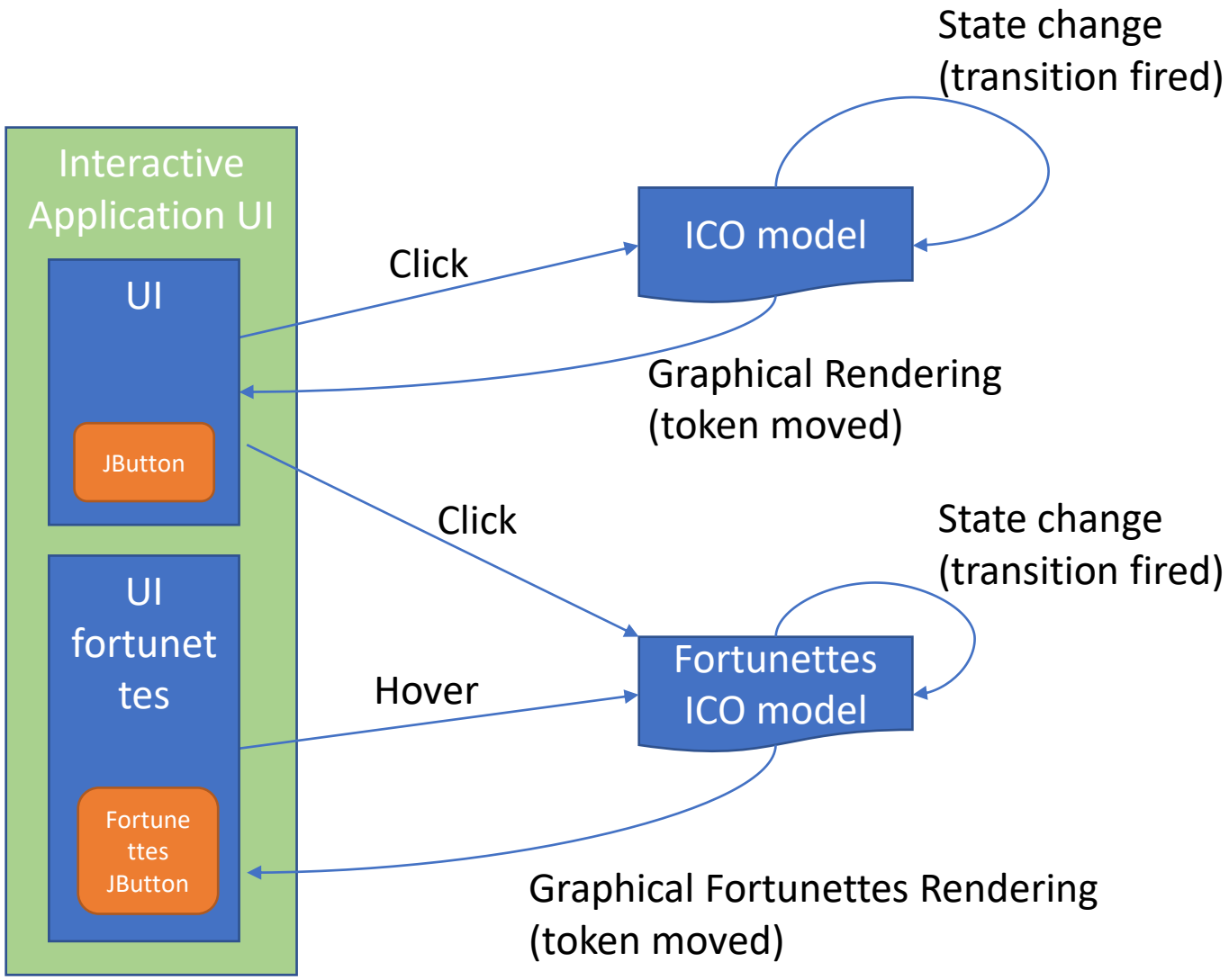
Complete behaviour of the widget



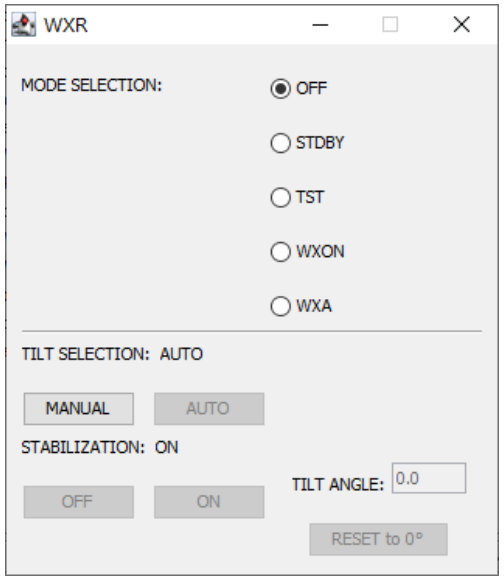
Fortune net of the Application

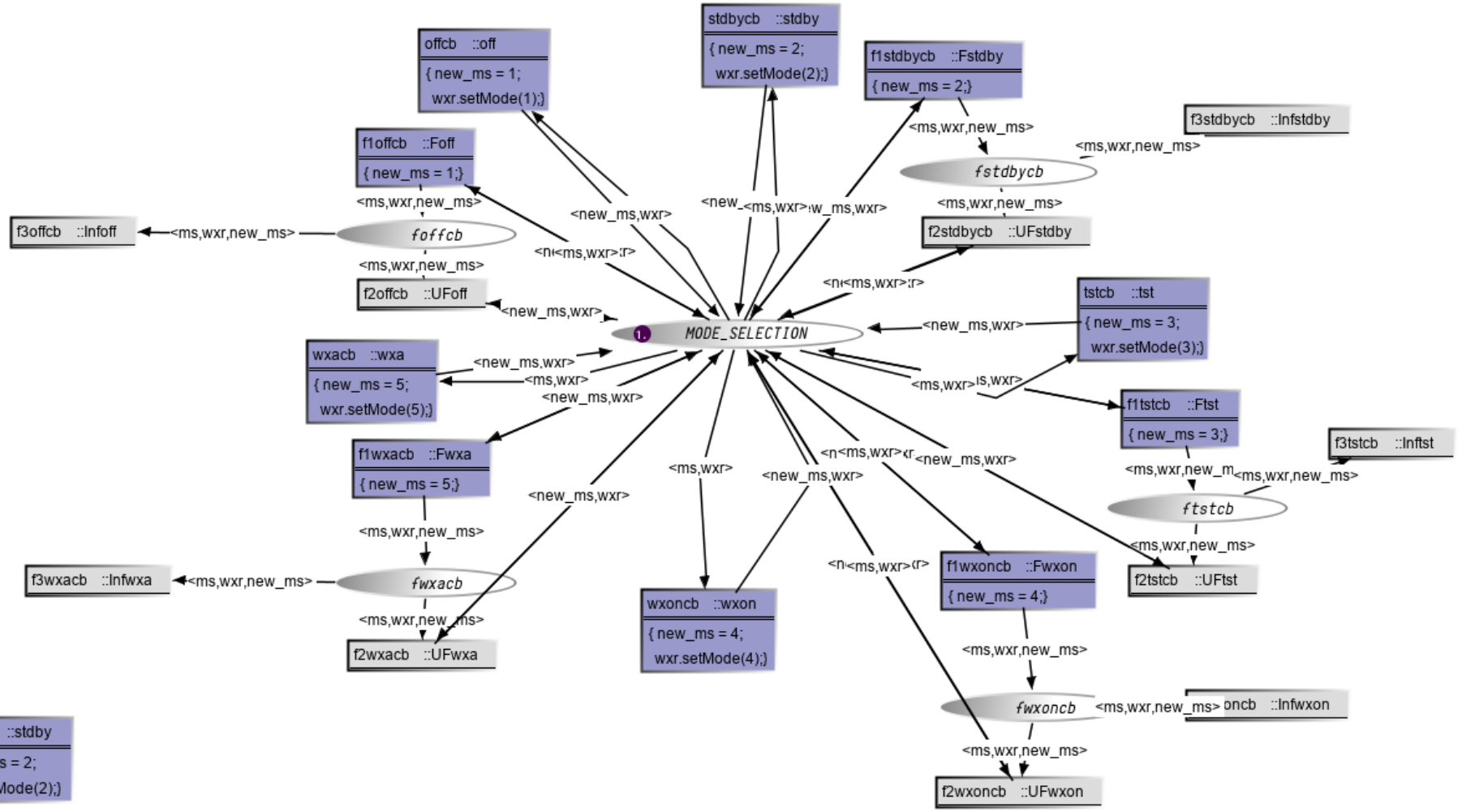


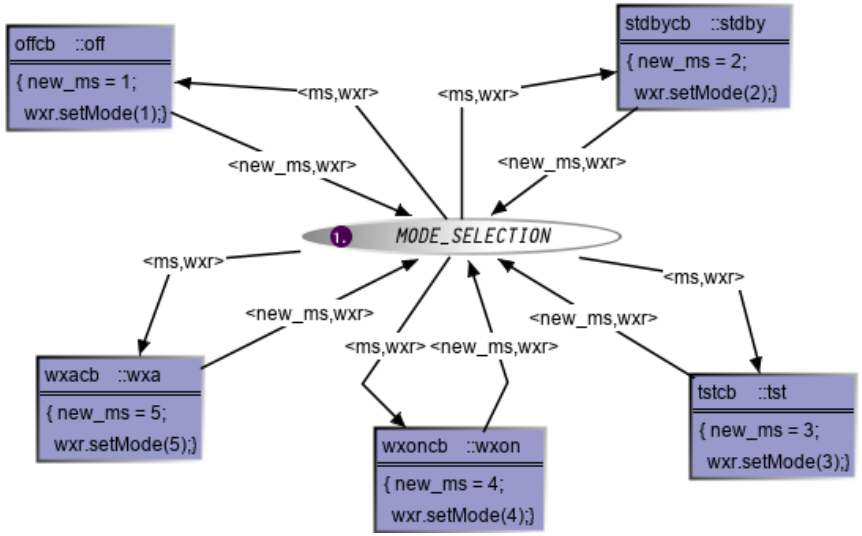
Execution Process (simplified)



Demo


WXR
MODE SELECTION:
OFF
STDBY
TST
WXON
WXA
TILT SELECTION: AUTO
MANUAL
AUTO
STABILIZATION: ON
TILT ANGLE: 0.0
OFF
ON
RESET to 0°


offcb ::off
{ new_ms = 1;
wxr.setMode(1);}
stdbycb ::stdby
{ new_ms = 2;
wxr.setMode(2);}
f1stdbycb ::Fstdby
{ new_ms = 2;}
f3stdbycb ::Infstdby
f1offcb ::Foff
{ new_ms = 1;}
foffcb
fstdbycb
f3offcb ::Infoff
f2stdbycb ::UFstdby
f2offcb ::UFoff
MODE_SELECTION
tstcb ::tst
{ new_ms = 3;
wxr.setMode(3);}
wxacb ::wxa
{ new_ms = 5;
wxr.setMode(5);}
f1tstcb ::Ftst
{ new_ms = 3;}
f3tstcb ::Inftst
f1wxacb ::Fwxa
{ new_ms = 5;}
ftstcb
f3wxacb ::Infwxa
fwxacb
f2tstcb ::UFtst
wxoncb ::wxon
{ new_ms = 4;
wxr.setMode(4);}
f1wxoncb ::Fwxon
{ new_ms = 4;}
fwxoncb
f3wxoncb ::Infwxon
f2wxacb ::UFwxa
f2wxoncb ::UFwxon
<ms,wxr,new_ms>
<new_ms,wxr>


offcb ::off
{ new_ms = 1;
wxr.setMode(1);}
stdbycb ::stdby
{ new_ms = 2;
wxr.setMode(2);}
MODE_SELECTION
wxacb ::wxa
{ new_ms = 5;
wxr.setMode(5);}
wxoncb ::wxon
{ new_ms = 4;
wxr.setMode(4);}
tstcb ::tst
{ new_ms = 3;
wxr.setMode(3);}
<ms,wxr>
<new_ms,wxr>

Formal Analysis

- Formal Analysis
  - Based on the underlying Petri net
  - Some extensions have been added
  - Interactive view of the analysis results
- Formal, Petri nets-based analysis of the application
  - T and P invariants
  - Traps and siphons
  - Possible to deduct meaningful information (e.g. reinitialisability, absence of deadlocks, mutual exclusion, presentation of each state change …)
- Formal, Petri nets-based analysis of the Fortunettes application
  - Behavior must be “compatible” with the one of the application
  - Some specific properties are relevant (e.g. always possible to leave future and come back to present)

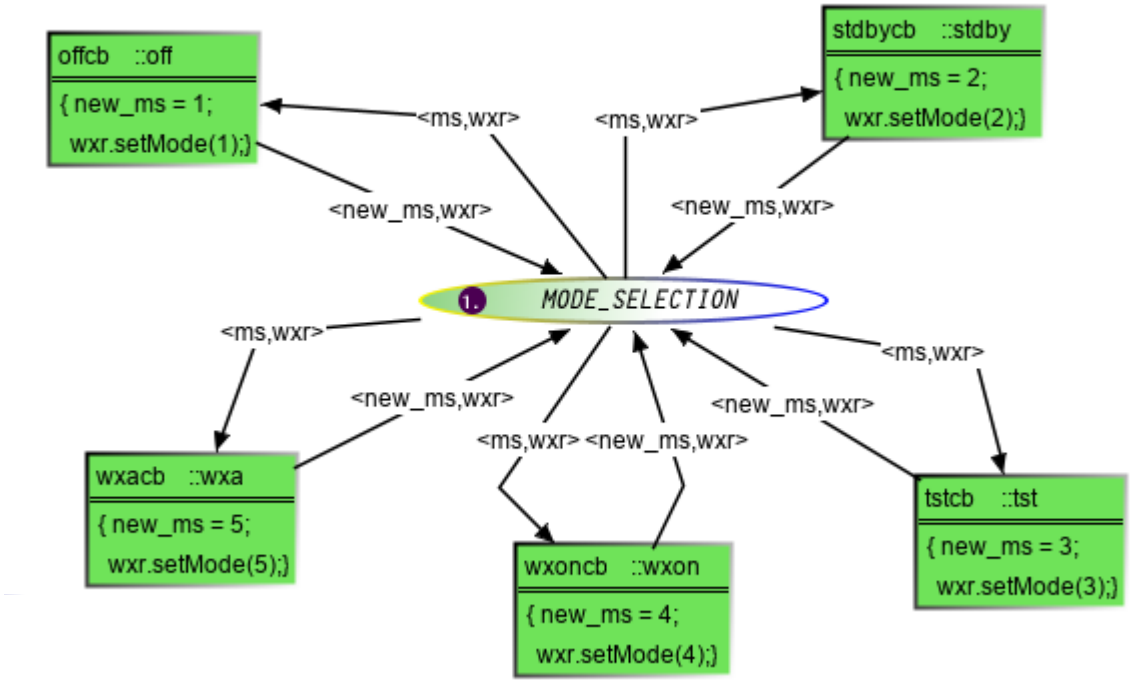
Formal Analysis Results in Petshop (Application)

Siphons
- TILT_ANGLE
- MODE_SELECTION
- AUTO, NOT_AUTO
- STABILIZATION_ON, STABILIZATION_OFF

Transition Invariants
- 1*angleIsLow, 1*changeAnglecb
- 1*wxoncb
- 1*wxacb
- 1*angleIsCorrect, 1*changeAnglecb
- 1*stdbycb
- 1*switchStabOffcb, 1*switchStabOncb
- 1*angleIsHigh, 1*changeAnglecb

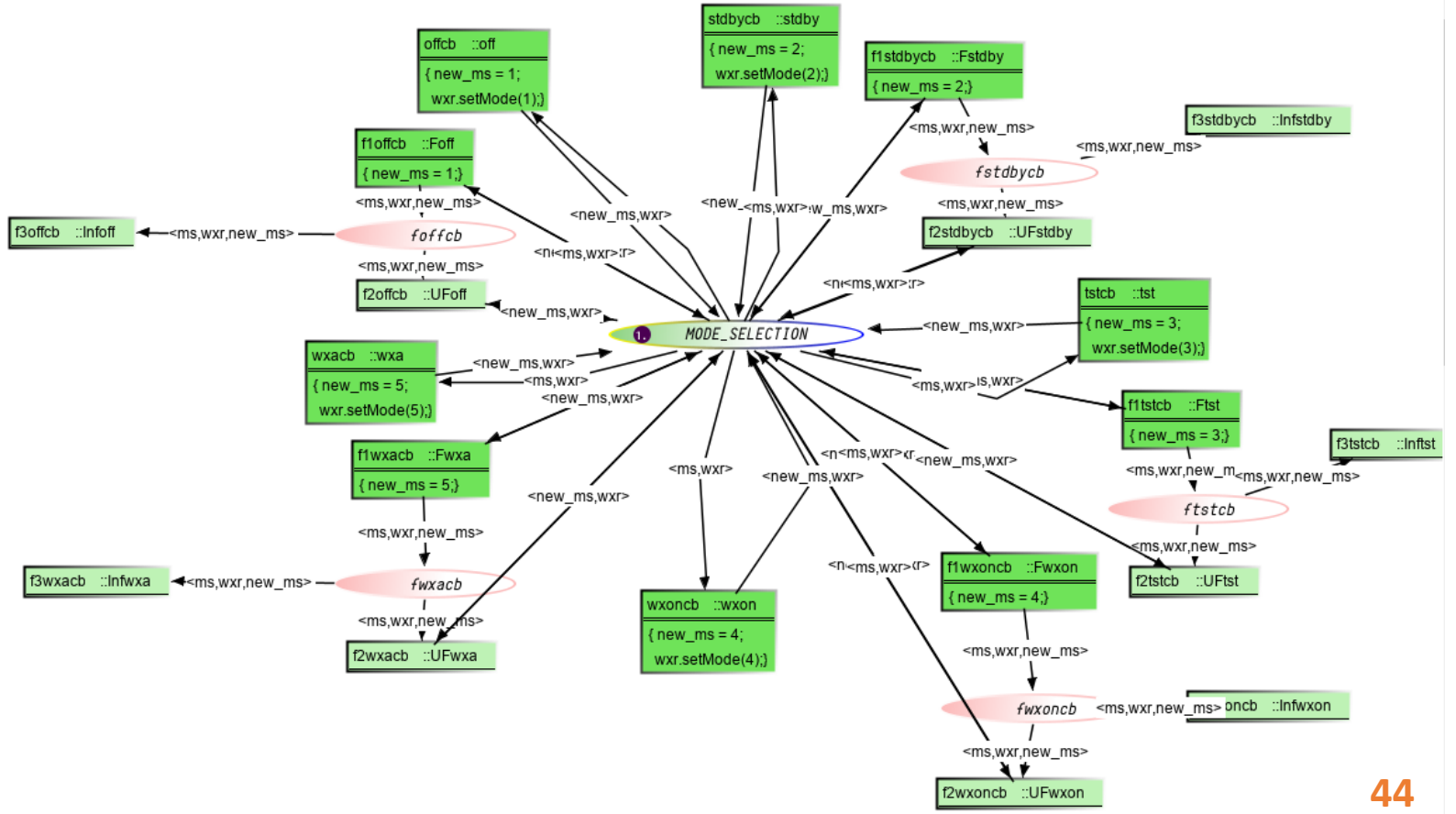
Traps
- TILT_ANGLE
- MODE_SELECTION
- AUTO, NOT_AUTO
- STABILIZATION_ON, STABILIZATION_OFF

Place Invariants
- 1*AUTO, 1*NOT_AUTO
- 1*TILT_ANGLE
- 1*STABILIZATION_OFF, 1*STABILIZATION_ON
- 1*MODE_SELECTION

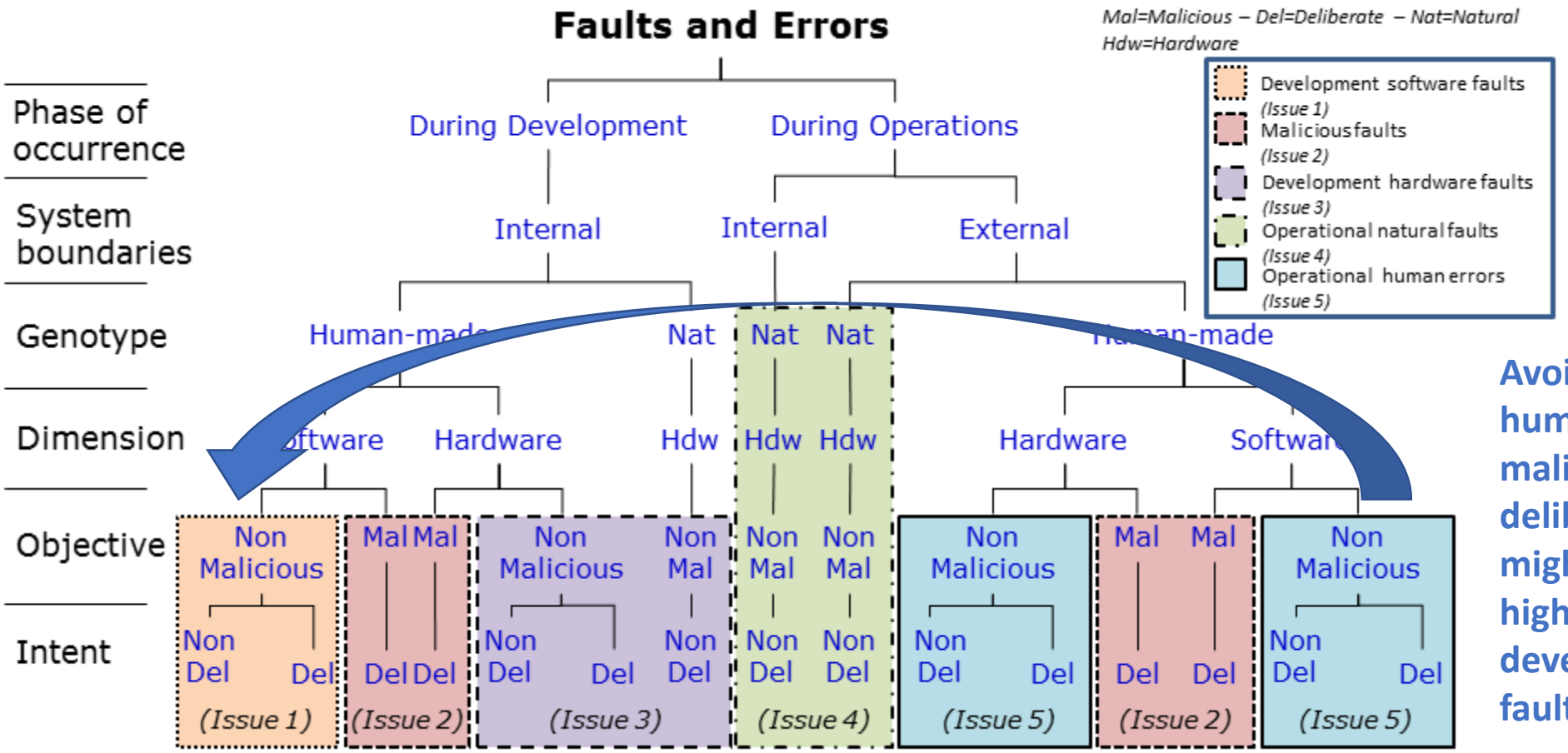


Formal Analysis Results in Petshop (Fortune net)

- Siphons
  - NOT_AUTO, AUTO
  - TILT_ANGLE
  - STABILIZATION_OFF, STABILIZATION_ON
  - MODE_SELECTION
- Transition Invariants
  - 1*f1tstcb, 1*f3tstcb
  - 1*f1wxacb, 1*f2wxacb
  - 1*f1offcb, 1*f2offcb
  - 1*f1switchAutocb, 1*f1switchManualcb, 1*f3switchAutocb, 1*f3switchManualcb
  - 1*wxoncb
  - 1*wxacb
  - 1*f1switchStabOffcb, 1*f3switchStabOffcb, 1*switchStabOncb
- Traps
  - NOT_AUTO, STABILIZATION_OFF, AUTO, fswitchStabOncb
  - NOT_AUTO, AUTO, fswitchStabOffcb
  - TILT_ANGLE
  - STABILIZATION_OFF, STABILIZATION_ON
  - MODE_SELECTION
- Place Invariants
  - 1*TILT_ANGLE
  - 1*STABILIZATION_OFF, 1*STABILIZATION_ON
  - 1*MODE_SELECTION



Rationale: **Induced** Fault Detection and Removal by a Zero-Default Approach



Avoiding operation human made non malicious non deliberate error might induce higher number of development faults

C. Fayollas et al. **An Approach for Assessing the Impact of Dependability on Usability: Application to Interactive Cockpits**, *2014 Tenth European Dependable Computing Conference*, Newcastle, 2014, pp. 198-209.

Take Away Message

- Feedforward (guidance)
  - is known as one a key design rule for UI (Scapin & Bastien 89, Nielsen's heuristic 94 Rule 1: Visibility of system status & Rule 6: Recognition rather than recall)
  - is not consistently offered in environments
  - is limited to one step (next one)
- Engineering guidance
  - requires the exploration of state space of the interactive applications
  - requires presentation and interaction design
  - Is a complex task not supported by programming environments
  - State-based formal methods can be of great help and bring Fortunettes "for free"
- Design aspects of Fortunettes are critical (not supported by standards yet) but benefits are very high
  - The formal method must be able to encompass new input/output devices
  - The formal method must be able to deal with post-WIMP interactions

Arnaud Hamon et al. 2014. **Formal modelling of dynamic instantiation of input devices and interaction techniques: application to multi-touch interactions**. *2014 ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '14). ACM, New York, NY, USA, 173-178.

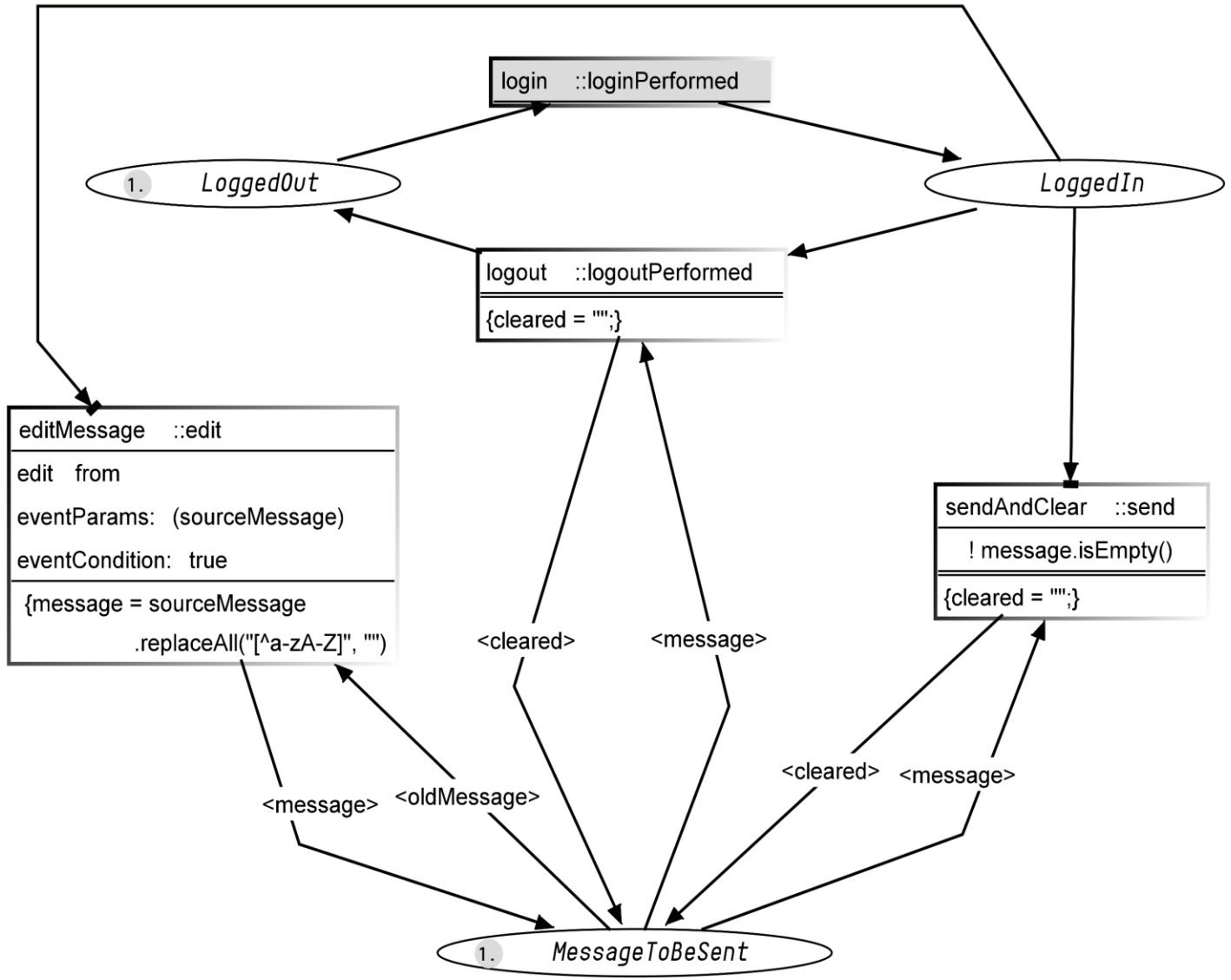
Future Work

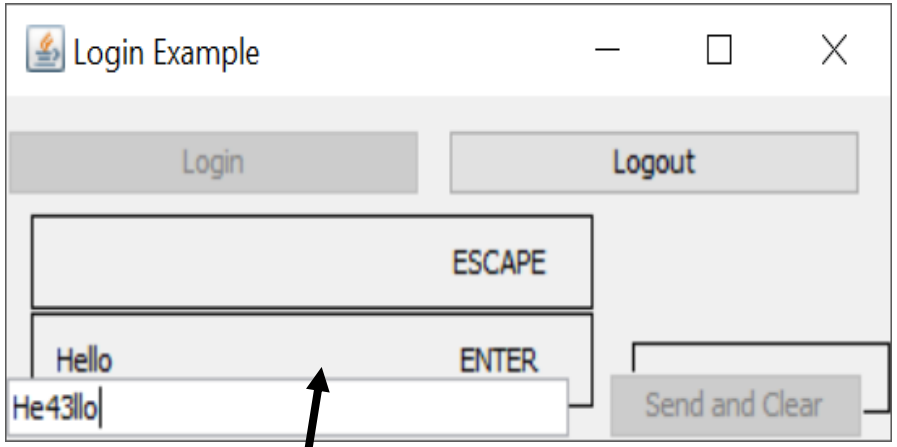
- Dependability of Fortunettes
  - Self-checking Fortunettes (tolerance to natural faults)
  - Reuse of dependability of input devices (no specific input and output)
  - Extend Fortunettes to touch interactions
- Use of Fortunettes
  - Conformance with ARINC 661 Specification supplement 7 (just out)
  - For future cockpit applications
- Impact of Fortunettes on training processes and material
- Investigate other interactions techniques
- Investigate generic fortunettes (similar to undo)
  - Multiple levels
  - UI properties levels

Thank you very much … for your attention

Questions?

Fortune net Behavior of the Application





Text Filtering