# Supporting the Interaction Designing Phase of the Current Mobile Paradigm

*Remote Talk at Tel Aviv University*


**Shah Rukh Humayoun**


Monday, May 19, 2014


Computer Graphics and HCI Group

University of Kaiserslautern, Germany

*http://hci.uni-kl.de/*

# Outline

1. Computer Graphics and HCI Group
2. The current mobile paradigm
3. Building interactive mockups
4. Formalizing user gesture interaction
5. Evolving prototypes towards the best-suited design and interaction Schema

# Part – I

## The Current Mobile Paradigm

# Mobile Phones

- First hand-held cell phone demonstration by Motorola in 1973 (2.2 pounds: 1 KG)
- NTT launched the first commercial cellular network in Japan in 1979
- 1983, DynaTAC 8000x was available commercially
  - 30 minutes talk time and 8 hours of standby
  - Price: 3,995 US dollars


DynaTAC 8000X

# Evolution of Mobile Phones

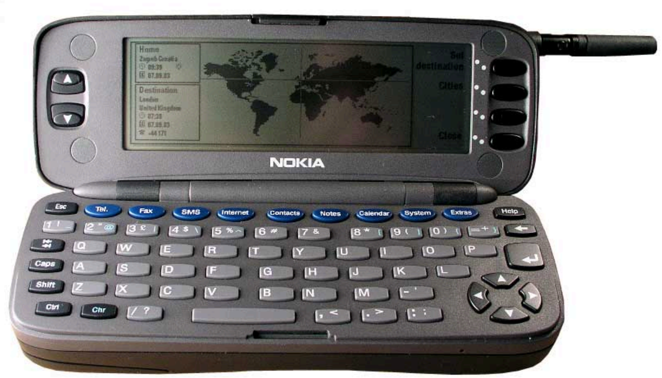
# Smartphones

- First smartphone: IBM Simon
  - Initially produced in 1992
  - Launched commercially in 1993 by BellSouth
  - Touch screen
  - Applications: *calendar, email-client, calculator, games, etc.*

IBM Simon

- Other example of initial smartphones are Apple MessagePad and Nokia 9000 Communicator

Apple MessagePad 100




Nokia 9000 Communicator

# Current Mobile Paradigm

- Although, many advancements had been done in mobile phones domain over these years

*however;*

- the current mobile paradigm is mostly influenced by:
  - The launch of Apple iPhone in 2007
    - Touch screen with the support of multi-touch gestures
  - The launch of Apple iPad in 2010, *and*
  - The launch of AppStore for mobile apps

# Current Smart Devices
## Smartphones and Smart Tablets



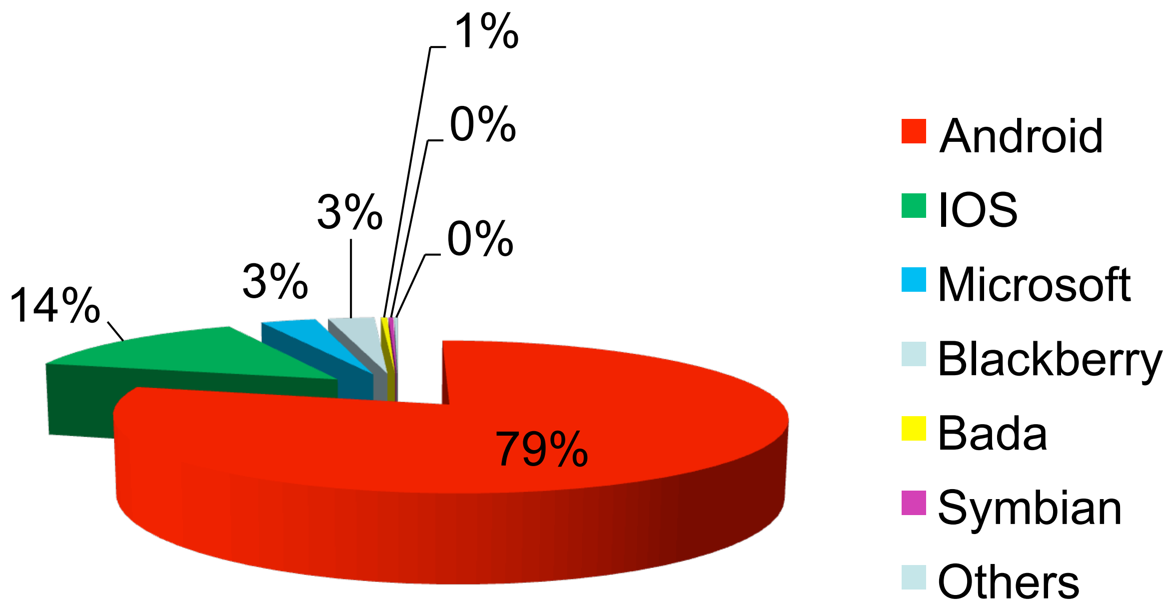

# Current Usage

World Population: **over 7 Billion**
Number of mobile phones : **over 6.8 Billion**

→ **More than 50% are Smartphones!!!**

# Mobile Operating Systems

- World-wide smartphone sales by OS, Gartner (2Q 2013)
  - [source: http://www.digitaltrends.com/mobile/smartphone-sales-for-q2-2013/]



# Mobile Paradigm *vs.* Desktop Paradigm

Current mobile paradigm

*vs.*

Conventional desktop paradigm

- Fundamentally differences at multi levels
  - Business model
  - Development
  - Consumer
  - Functionality

# Mobile Paradigm *vs.* Desktop Paradigm

## Business Model Level:

- Single task-focused apps rather than multi-tasks software
- Availability of apps through online apps stores
- Consumer market

# Mobile Paradigm *vs.* Desktop Paradigm

## Development Level:

- Multiple platforms and device classes
- Short-time spam development cycle
- New interactions techniques

# Mobile Paradigm *vs.* Desktop Paradigm

## Consumer level:

- Mobility
- From entertainment to commerce and from daily life activities (e.g., bus timings) to learning (m-learning)
- Apps availability

# Mobile Paradigm *vs.* Desktop Paradigm

## Functionality level:

- Context-awareness
- Sensor-based functionality
- New interaction paradigm, such as multi-touch gestures
- User interface (e.g., less text, more buttons, etc.)

# Challenges

- These all factors bring new challenges not only for:
  - Stakeholders
  - Members of development teams (e.g., architecture, designers, developers, etc.)
  - Users
- But, also at other levels, e.g.:
  - Software development
  - Management
  - Marking
  - etc.

# Challenges

*To tackle these challenges,*

- We envision that there is a need to start research for new approaches, methods and techniques at different levels (from lower to high abstract level),
- As well as dynamical change and advancement in previous approaches, techniques, and methods in order to accommodate them properly for the current mobile paradigm.

# Our Focus!

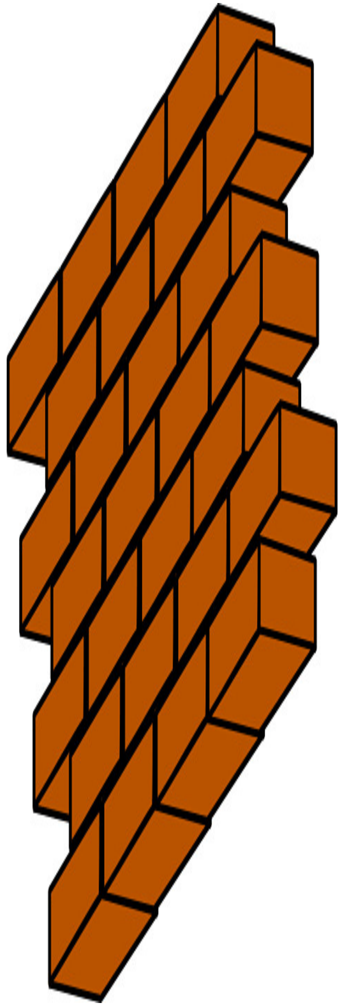
- At Computer Graphics and HCI Group, we mainly focus towards the current mobile paradigm from the ***interaction*** point of view

# Part – II

## Building Interactive Mockups

# Communication Problem in Software Development

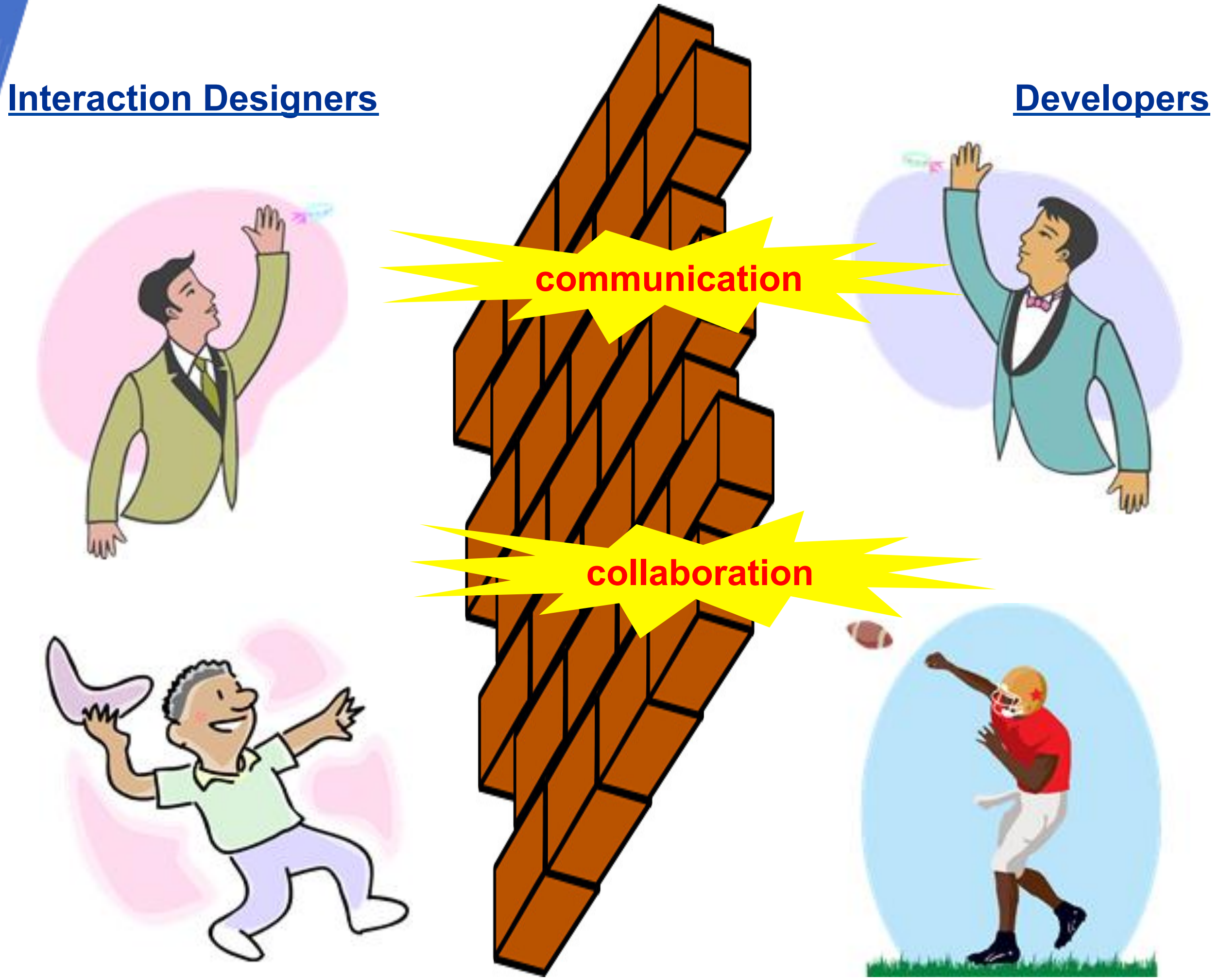
**Interaction Designers**



**Developers**

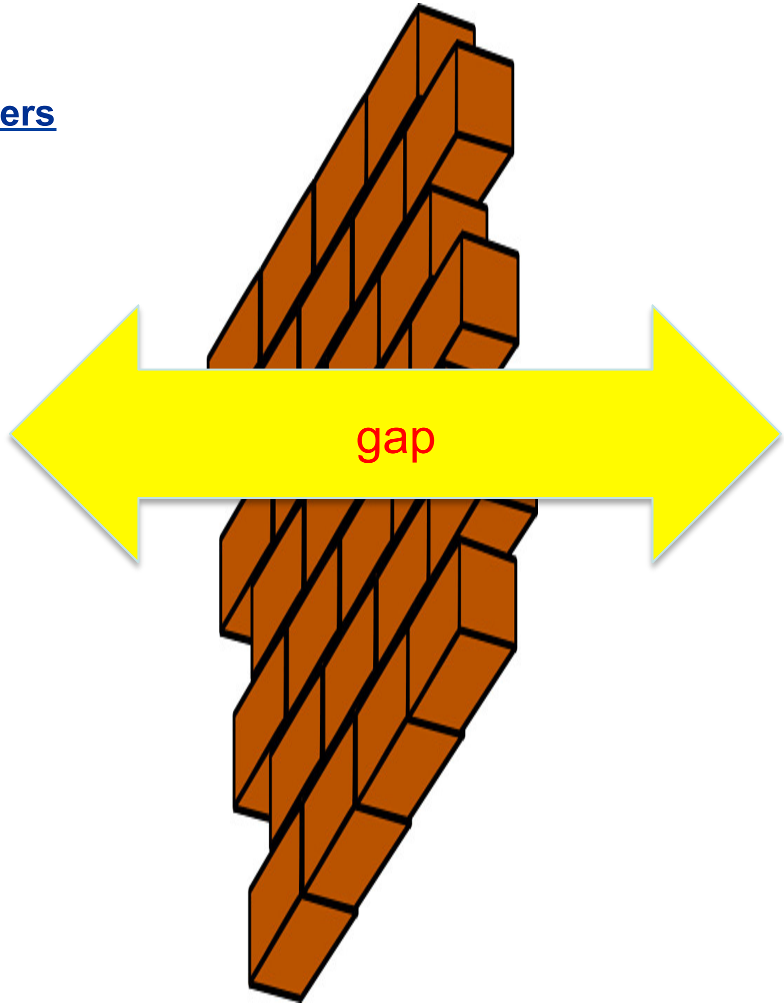
Computer Graphics and HCI Group

# Communication Problem in Software Development



# Communication Problem in the Current Mobile Domain

**Interaction Designers**

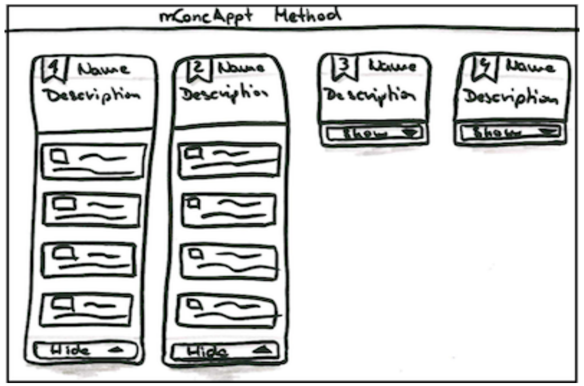



**Developers, Customers, Stakeholders, etc.**

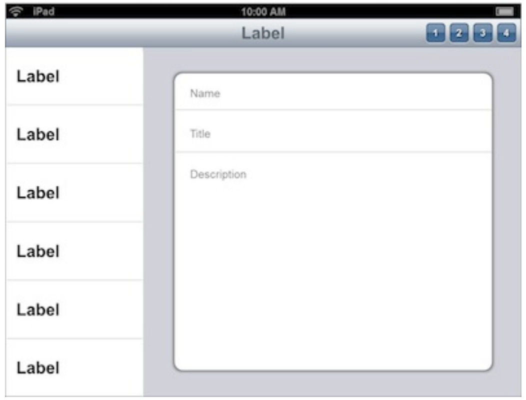
# Challenges in User Interface Designing

- Interaction designers use sketches/mockups/prototypes for designing UI and for communicating their ideas and thoughts
- Challenges with regard to the current mobile paradigm:
  - Users' direct interaction (e.g., multi-touch gestures)
  - Multiple platforms
  - Different device classes
  - Context-aware services
  - ….

# Conventional Lightweight Prototyping Approaches




Paper-based sketches




Digital representation

- One of the main limitations is that they lack multi-touch gestures interaction and screen transitions

# Existing Commercial Tools

- Few examples:
  - Antetype, Axure, Fluid UI
- Main drawbacks:
  - Complex processes not suitable for rapid building interactive mockups and prototypes
  - Do not support all major platforms
  - Many of them also lack multi-touch gestures interaction support in the generated prototypes

# Requirements

- We worked with interaction designers at **Fraunhofer IESE** for considering their requirements
- Requirements:
  - Light-weight solution
  - Freedom of expression through their existing ways (e.g., paper sketching, etc.)
  - Interactive mockups with concrete mobile interaction schema
    - to communicate with other parties (e.g., customers, developers, etc.)
    - To enable early user testing
  - Support of multiple platforms and device classes

# Our Solution!

- **i2ME** (**i**nteractive **M**ockup-Building for **M**obile **E**nvironment) Framework
- Provides an environment for building interactive mockups targeting the current mobile interaction paradigm

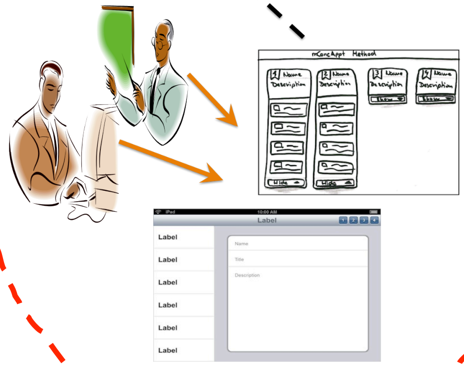
Shah Rukh Humayoun, Steffen Hess, Felix Kiefer, and Achim Ebert
**i2ME: A Framework for Building Interactive Mockups.** *ACM MobileHCI '13*. ACM, New York, NY, USA, 2013.

# The **i2ME** Framework

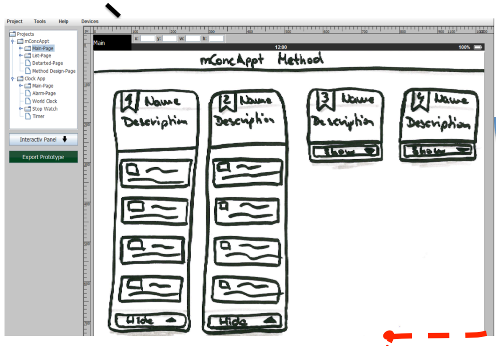
- Consists:
  - **iMocBuilder:** a mockup-building tool
  - **MTGest:** a multi-touch JavaScript-based library
  - **iMocTester:** a mockup-simulating mobile app
- Enhances the static mockups (handmade sketches or tool-generated wireframes) with concrete mobile interaction elements
- The generated **HTML5+JavaScript** based interactive mockups can be simulated on multiple platforms and mobile device classes

# The i2ME Workflow

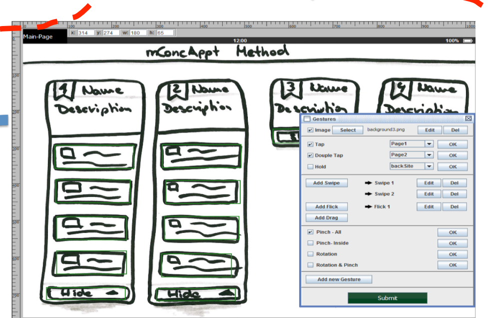
**Phase 1:**
Interaction designers create handmade sketches or tool-oriented wireframes.



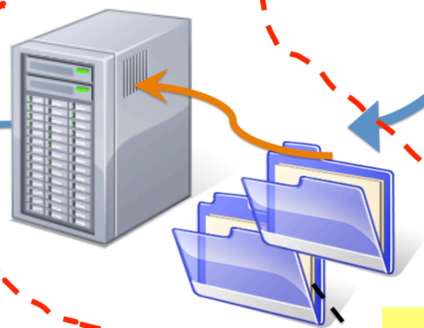
**Phase 2:**
The mockups/wireframes images are used as input to the ***iMocBuilder*** tool.



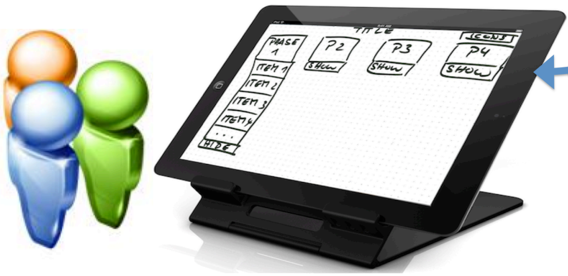
**The iMocBuilder Tool**



**Phase 3:**
Designers specify ***interactive-areas*** and attach to them gestures and the screen transitions schema.



**Phase 4:**
The generated ***HTML5 + JavaScript*** based interactive mockups are stored on a server as a project.



**Phase 5:**
Users/developers/customers can interact directly with these interactive mockups on the mobile device using the ***iMocTester*** app or through the device own browser.

**The iMocTester App**

# iMocBuilder
# **i**nteractive **Moc**kups **Builder**

- An easy, quick and efficient solution
- A Java-based tool
- Takes input hand-made sketches or tool-generated prototypes as images
- Generates interactive mockups using *MTGest library* in HTML5+Javascript that can be run on any device and platform

# iMocBuilder: Main Capabilities

- Freedom to use *handmade sketches* to *tool-generated mockups* as input
- Any area in the input mockups can be defined as an *interactive-area*
- Facility to attach set of multi-touch gestures and screen-transitions schema to a defined interactive area
- Automated scaling of the generated mockup according to the underlying device and platform

# iMocTester
# **i**nteractive **Moc**kups **Tester**

- iMocBuilder adds meta-tags for enabling the mockups to run on the target device in full screen mode
- Only few browsers have this functionality
- iMocTester app simulates these interactive mockups on mobile devices as mobile apps without any additional browser bar
- *Online testing:* directly from the server during testing
- *Offline testing*:  stored inside the device and then they are run locally

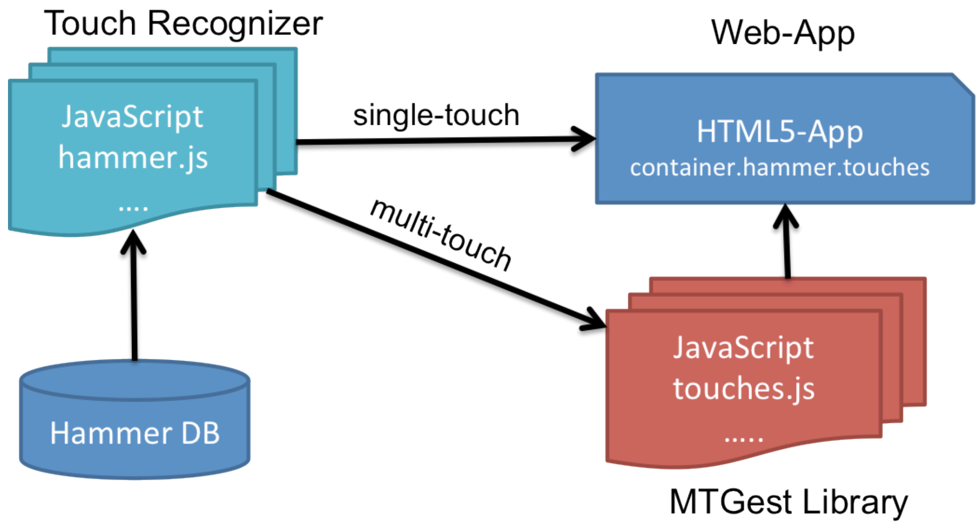
# MTGest
# Multi-Touch Gesture Library

- HTML5 includes a set of interfaces for touch events, but lacks built-in tags for the functionality of multi-touch gestures
- MTGest is based on JavaScript + JQuery and built on the top of hammer.js library
- Current supported gestures:
  - Standard gestures: *tap, double tap, drag, swipe, transformation (pinch), rotation, flick,* and *shake*
  - Customized: *e.g., three-fingers tapping, multi-fingers swiping, etc.*

Shah Rukh Humayoun, Franca-Alexandra Rupprecht, Steffen Hess, Achim Ebert
**Adding Multi-Touch Gesture Interaction in Mobile Web Apps.** In M. Kurosu (Ed.): Human-Computer Interaction, Part II, HCII 2014, LNCS 8511, pp. 48–57, 2014.

# How it works!

- Each function in the library represents a gesture
- The specific function is attached to a container, which represents a specific area in the HTML document
- *hammer.js* is also attached to the same container for getting the touch events
- More than one gesture can be attached to the same container



# Mobile Web Apps

- Based on web technologies (HTML, CSS, JavaScript, etc.)
- HTML5 enables offline browsing and possibility of accessing many device resources
- Mobile web apps can be an alternative in many cases
  - Require less efforts and resources for developing
  - Support of multiple platforms
  - Provide more consistent user experience across different platforms
- One of the main lacks is built-in tags for the functionality of multi-touch gestures support

# User Evaluation Study

- A controlled experiment
- The aim was to compare the multi-touch gesture interaction support provided by:
  - MTGest library for the mobile web apps

  *vs.*
  - Native platform library for the native mobile apps

# User Evaluation Study

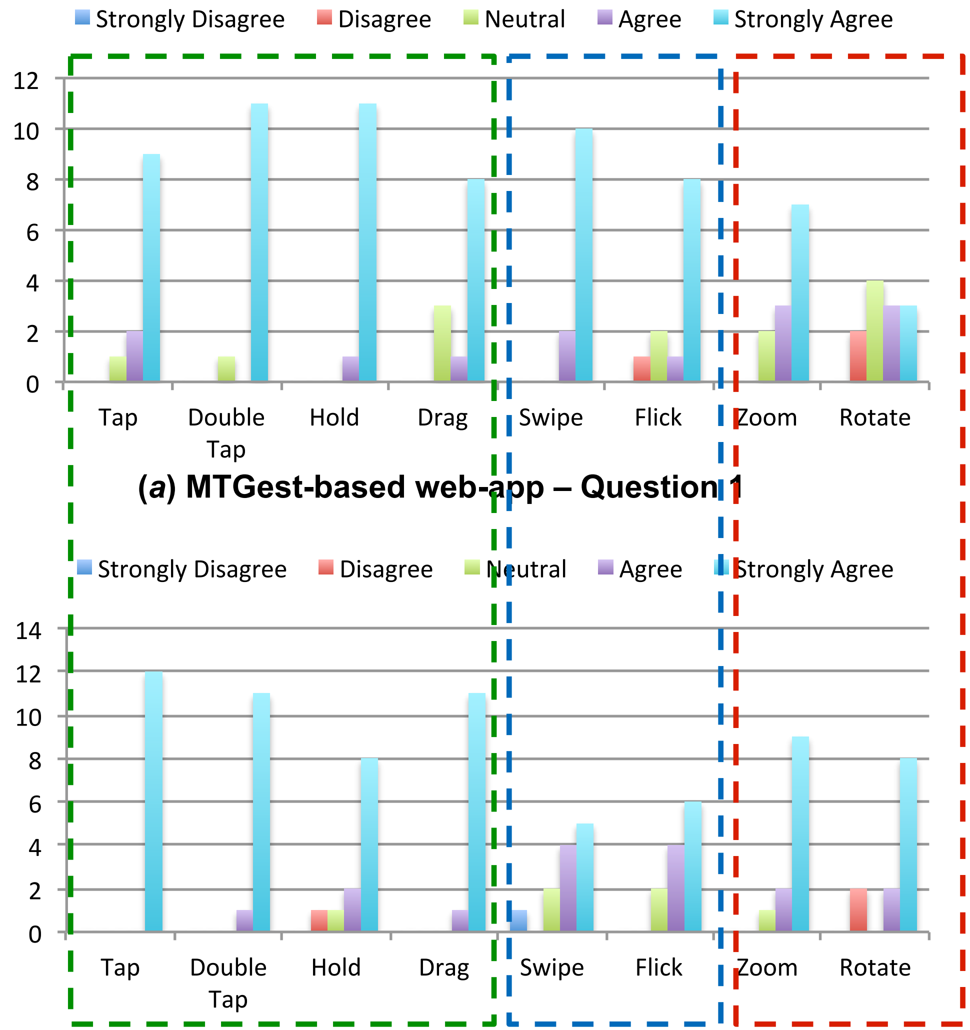
- We developed two simple apps:
  - a mobile web app
  - a native iOS app
- Both apps were identical in providing functionality
  - On each page, both apps provided few tasks to apply a targeted gesture
- Used gestures: *tap*, *double tap*, *hold*, *drag*, *swipe*, *flick*, *zoom-in*, *zoom-out*, and *rotation*

# User Evaluation Study

- 12 participants
  - 3 females
  - 9 males
- User groups:
  - iOS experienced users
  - Android experienced users
  - Non-experienced users
- Age between 20 to 36 years with a 27.5 mean
- We compared the results from the *efficiency* and *user satisfaction* perspectives

# User Evaluation Study Results

- Users feedback about the accurately work of the gesture

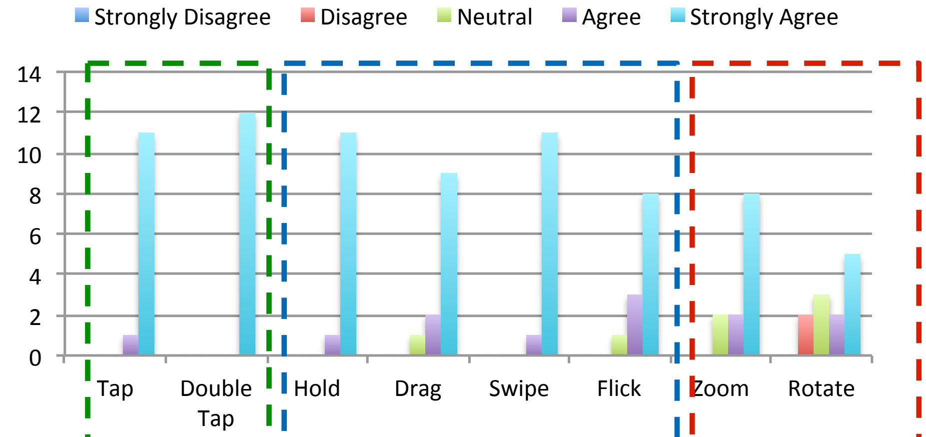



(c) iOS-based mobile-app – Question 1

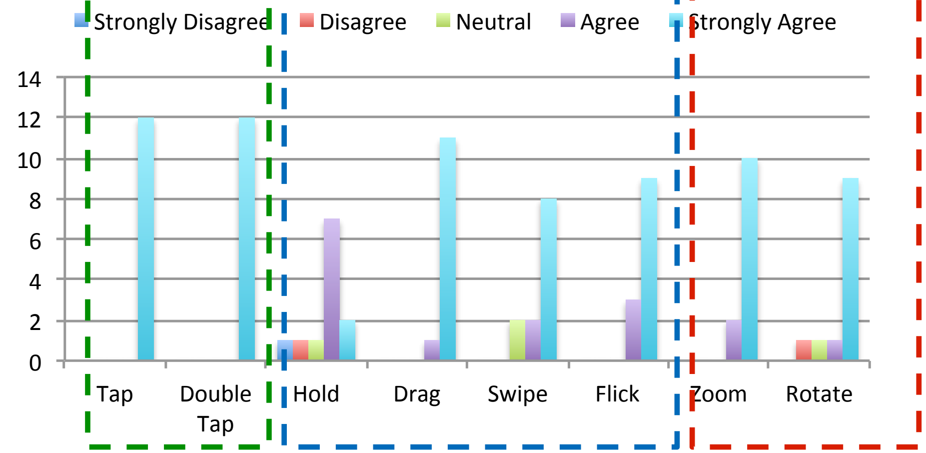
Computer Graphics and HCI Group

# User Evaluation Study Results

- Users feedback about the interaction response of the gesture



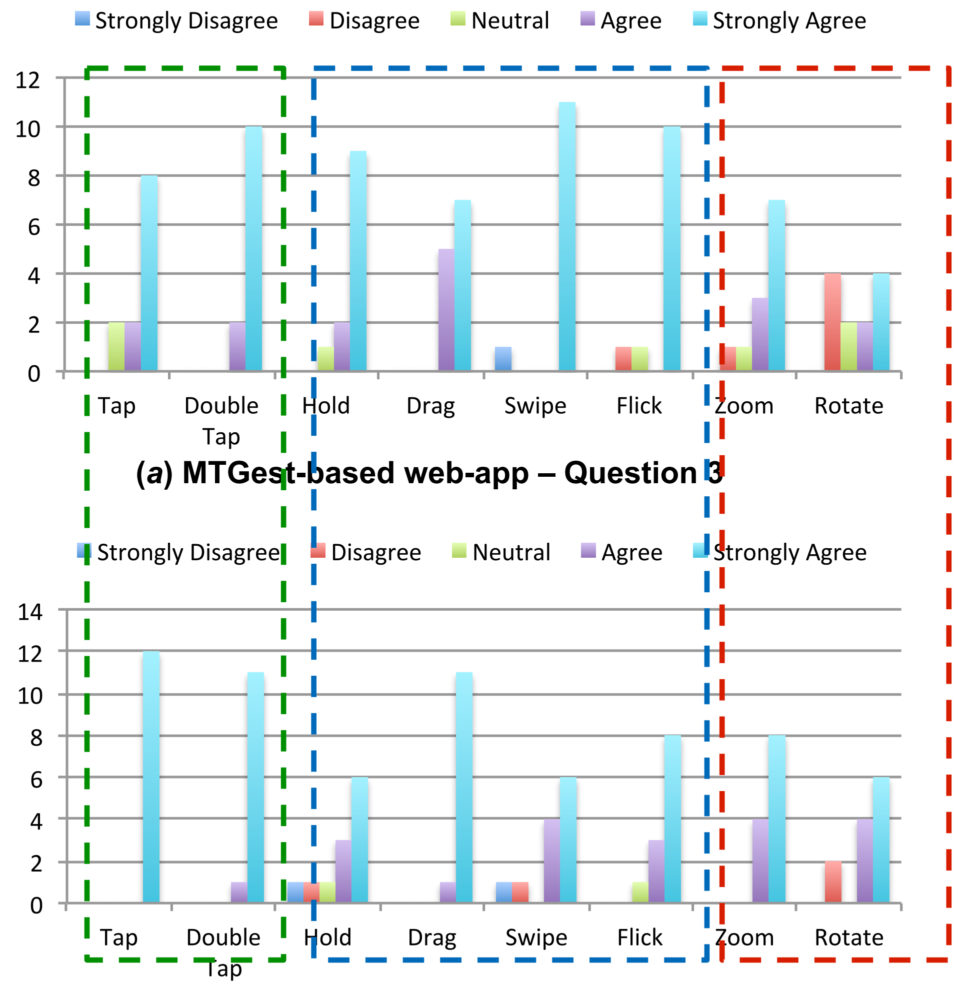

(*b*) **MTGest-based web-app – Question 2**




(*d*) **iOS-based mobile-app – Question 2**

# User Evaluation Study Results

- Users feedback about the satisfaction of the gesture facility




(*a*) MTGest-based web-app – Question 3



(*c*) iOS-based mobile-app – Question 3

# User Evaluation Study Results

- Users feedback about the intention to use the gesture facility in future

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

# User Evaluation Study Results

- There are many factors that can affect users' satisfaction level:
  - *users' expectations, curiosity, their interests in new experiences, their expertise with gestures, their positive attitude towards Apple, their low expertise with MTGest library, etc.*
- Our conclusion:
  - Mobile web apps have the potential of providing an alternative to the native apps in many cases
  - But, they need to provided improved functionalities, especially the user gestures interaction
- However, the study targeted only the iOS platform and there is a need to perform further studies with all platforms to generalize the results

# Part – III

## Formalizing User Gestures Interaction

# Formalizing User Gestures Interaction

- User gestures interaction with mobile devices and apps is one of the most important factors
- A formal definition of each user interaction is required for many purposes, such as:
  - Communication between different involved groups
  - Unambiguous requirements specification
  - Automated user evaluation
  - Automated testing and verification
  - …
- We will show in next part how we are planning to use it for generating prototypes in the initial design phase

# In Past!

- In past:
  - We provided specification of TaMoGolog (Task Modeling Golog) formal task modeling language
  - It was built on top of the foundations of Situation Calculus and the Golog-family of high-level programming languages
- We also used TaMoGolog for performing automated user-based ongoing product evaluation

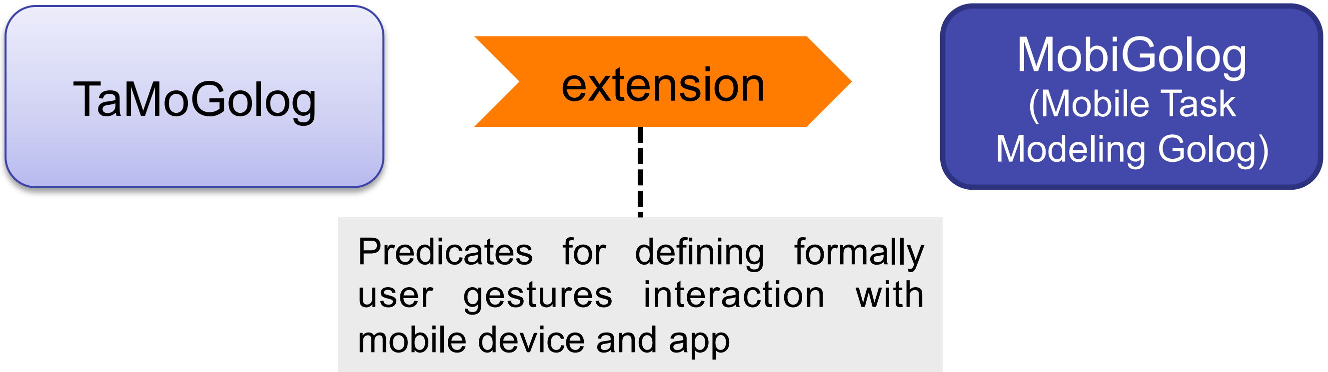

- Shah Rukh Humayoun, Tiziana Catarci, Yael Dubinsky
  A Dynamic Framework for Multi-View Task Modeling. *CHItaly ' 11*, ACM, New York, NY, USA, 185-190.
- Shah Rukh Humayoun, Yael Dubinsky, Tiziana Catarci, Eli Nazarov, Assaf Israel
  A Model-based Approach to Ongoing Product Evaluation. *AVI '12*, ACM, New York, NY, USA, 596-603, 2012.

# TaMoGolog Characteristics

- Expressiveness and dynamicity
- Formally well-defined (syntactically & semantically)
- Rich set of constructs
- Domain knowledge representation in task models
- Ability to write task models *as per* defined by *framework concepts*
- Representation of human users and external applications/systems interaction in task models
- Executable task models
- Customizability & extensibility

# Our Approach

- TaMoGolog lacks in providing specification of user multi-touch gestures interaction targeted at mobile domain
- Our solution:




**Shah Rukh Humayoun, Yael Dubinsky**
MobiGolog: Formal Task Modelling for Testing User Gestures Interaction in Mobile Applications.
*ACM MOBILESoft 2014*, Hyderabad, India, June 2-3, 2014.

# TaMoGolog Predicates

| Predicate | Description |
| --- | --- |
| ViewModel(n) | n is a view model |
| ViewType(p) | p is a view type |
| TaskModel(m) | m is a task model |
| ModelContainer(n, p) | the view model n contains view type p |
| ViewModelTask(p, m) | the view type p is realized by a task model m |
| UnitTask($\mu$) | $\mu$ is a unit task |
| WaitingTask($\omega$) | $\omega$ is a waiting task |
| CompositeTask($\Gamma$ ) | $\Gamma$ is a composite task |
| Task($\alpha$) | $\alpha$ is a task |
| Type(t) | t is a task type, normally represents behavioral category |
| TaskType($\alpha$, t) | $\alpha$ is a task of type t |
| Agent(agt) | agt is an external entity (called as agent) that interacts with the system |
| Responsible(agt, $\alpha$) | external entity agt is responsible for executing task $\alpha$ |
| Fluent(f) | f is a fluent either functional or relational |
| FluentDef(f, d) | fluent f is defined by definition d |
| FluentInit(f, x) | initially, fluent f has value x |
| InitialState(m) $\equiv$ $\Omega$m | Formula $\Omega$m is conjunction of all fluents' initial states and may have other axioms for task model initial state |
| Precondition($\mu$) $\equiv \Pi_\mu$ | Formula $\Pi_\mu$ is a conjunction of all condition under which a unit task is possible to execute and equivalent to situation calculus predicate Poss($\alpha$,s) $\equiv \Pi_\mu$(s) |
| Postcondition | Executing task $\mu$ has an effect on fluent F under any condition $\Omega$F and new value of fluent F is according to formula $\Phi$F. This is equivalent to situation calculus formula |
| Goal(g) $\equiv \Delta$g | Formula $\Delta$g defines the goal g. |

# TaMoGolog Set of Constructs

| | |
|---|---|
| Golog | basic operators and procedure definition |
| ConGolog | operators for concurrency |
| IndiGolog | off-line search operator |
| GameGolog | nondeterministic operators for external entities |
| Golog-BDI | failure handling operator |
| TaMoGolog | additional nondeterministic operators for external entities and off-line failure handling operator |

Computer Graphics and HCI Group

| Constructs | Meaning | Origin |
|---|---|---|
| | *1 - Basic task category* | |
| $v$ | *unit* task | Golog |
| $\varpi$ | *waiting* task | Golog/TaMoGolog |
| $\Gamma$ | *composite* task | Golog |
| | *2 - Condition* | |
| $\phi?$ | waiting/testing action | Golog |
| | *3 - Sequence* | |
| $\Gamma_1; \Gamma_2$ | sequence | Golog |
| | *4 - Optional category* | |
| $[\Gamma_1 \mid \Gamma_2]$ | internal nondeterministic choice | Golog |
| $[agt\ \Gamma_1 \mid \Gamma_2]$ | external nondeterministic choice | GameGolog |
| $[\textbf{if}\ \phi\ \textbf{then}\ \Gamma_1\ \textbf{else}\ \Gamma_2]$ | *if-then-else* condition | ConGolog |
| $[\pi x.\Gamma(x)]$ | internal nondeterministic choice of arguments | Golog |
| $[agt\ \pi x.\Gamma(x)]$ | external nondeterministic choice of arguments | GameGolog |
| | *5 - Cycling/iteration category* | |
| $[\Gamma]^*$ | internal nondeterministic iteration | Golog |
| $[agt\ \Gamma]^*$ | external nondeterministic iteration | GameGolog |
| $[\textbf{while}\ \phi\ \textbf{do}\ \Gamma]$ | conditional (*while-do*) loop | ConGolog |
| | *6 - Time-sharing category* | |
| $[\Gamma_1 \parallel \Gamma_2]$ | normal concurrency | ConGolog |
| $[\Gamma_1 \rangle\rangle \Gamma_2]$ | concurrency with priority | ConGolog |
| $[\Gamma]^{\parallel}$ | concurrent iteration | ConGolog |
| $[agt\ \Gamma_1 \parallel \Gamma_2]$ | external every/step decision concurrency | GameGolog |
| $[agt\ \Gamma_1 \langle\rangle \Gamma_2]$ | external selected priority concurrency | TamoGolog |
| $[agt\ \Gamma_1 \langle\mid\rangle \Gamma_2]$ | external first-step decision concurrency | TaMoGolog |
| $[agt\ \Gamma]^{\parallel}$ | external selected concurrent iteration | TaMoGolo |
| | *7 - Off-line search* | |
| $\Sigma(\Gamma)$ | off-line search block | IndiGolog |
| | *8 - Interrupt* | |
| $< \phi \rightarrow \Gamma >$ | interrupt call to a task | ConGolog |
| | *9 - Failure handling* | |
| $[\Gamma_1 \rhd \Gamma_2]$ | online alternative execution | Golog-BDI |
| $[\Gamma_1 \rhd_{\Sigma} \Gamma_2]$ | off-line alternative execution | TaMoGolog |

# TaMoGolog Contributions

**3 – Framework for External Non-deterministic Constructs**

**2 – Additional Set of Constructs**

**1 – Set of predicates (syntax and semantics) for Multi-View Task Modeling**

Golog-Family | Situation Calculus

# MobiGolog Contributions

**MobiGolog:** Support of mobile multi-touch interaction paradigm

- Framework for External Non-deterministic Constructs
- Additional Set of Constructs
- Set of predicates (syntax and semantics) for Multi-View Task Modeling

TaMoGolog Support

Golog-Family | Situation Calculus

# MobiGolog Predicates

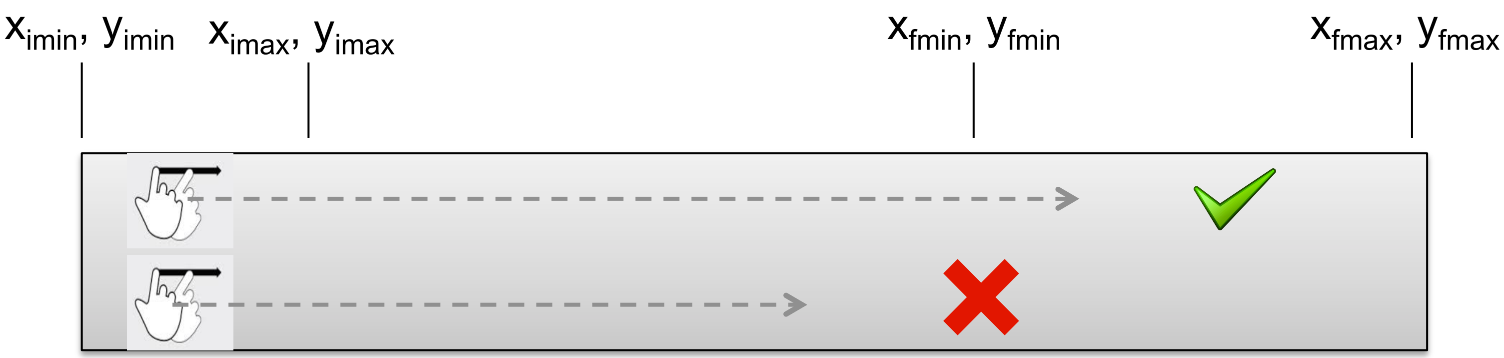
- ***UserInteraction(u)***:
  - *u* is a kind of user interaction with the mobile device or app
- ***PostconditionUserInt(u, Ʊ, Ψ(u, Ʊ)) ≡ Θ(u, Ʊ)***:
  - formula *Θ(u, Ʊ)* defines the effects of a specific user interaction *u* with the mobile device or app on the set Ʊ of related variables under any conditions Ψ*(u,* Ʊ*)*
- ***UI-Element*(*ε*)**:
  - *ε* is an UI element (either a software UI element in the mobile app or a hardware button on the mobile device)
- ***mInteractionTask*(*u, ε, T*) ≡ *Λ(u, ε, T)***:
  - It defines the execution of a specific task *T* based on user interaction with the mobile device or app.

# MobiGolog Predicates: *User Interaction*

- A specific type of user interaction with the mobile device/ app result in execution of a particular task
- First step:
  - Recognizing the type of user interaction
- MobiGolog predicate:
  - ***UserInteraction(u)***
  - *u* is a kind of user interaction with the mobile device or app

# MobiGolog Predicates: Effects of *User Interaction*

- User interaction with the mobile device/app changes the values of different related variables to show the effects
- These variables are then used to determine whether the happened user interaction is correct




Swiping area

# MobiGolog Predicates: Effects of *User Interaction*

- User interaction with the mobile device/app changes the values of different related variables to show the effects
- These variables are then used to determine whether the happened user interaction is correct
- MobiGolog predicate:
  - ***PostconditionUserInt(u, ℧, Ψ(u, ℧)) ≡ Θ(u, ℧)***
  - formula *Θ(u, ℧)* defines the effects of a specific user interaction *u* with the mobile device or app on the set ℧ of related variables under any conditions Ψ*(u, ℧)*

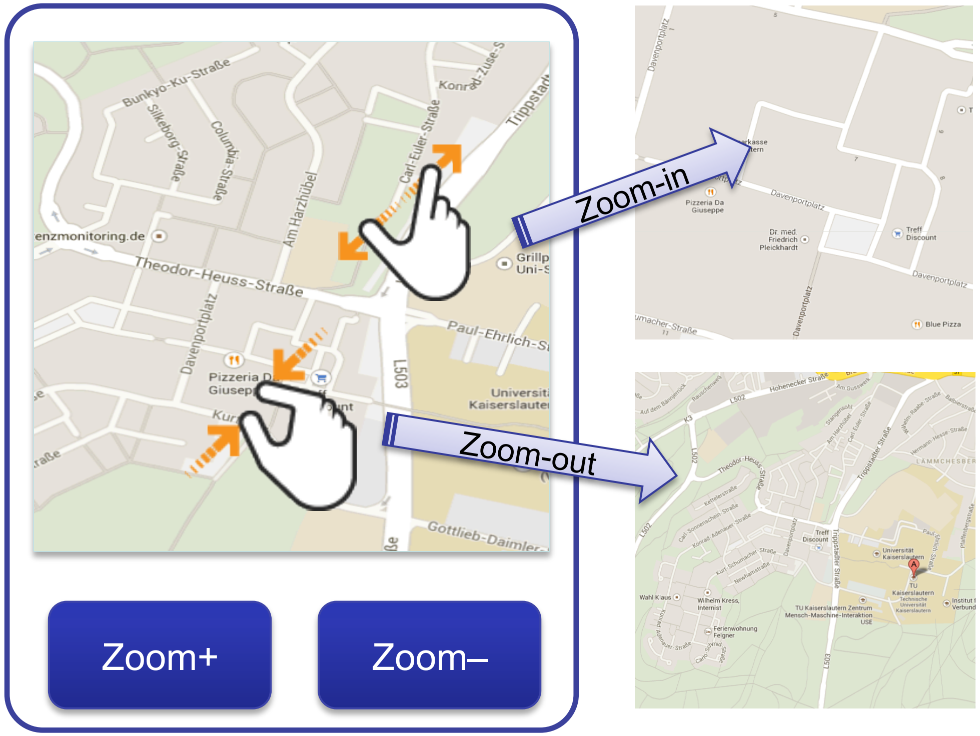
# MobiGolog Predicates:
# *User Interaction Effects on Variables*

- Different platforms provides their own specification for a particular gesture
  - iOS provides *flick* and *swipe*
  - Android provides only *swipe*
- MobiGolog predicate *postconditionUserInt* is used for specifying each gesture
- Example:
  - UserInteraction(tap).
  - PostconditionUserInt(u, (x, y), null) $\stackrel{def}{=}$
    $\exists$ tap.{UserInteraction(tap) $\wedge$ u = tap} $\wedge$
    $\exists$ x'.{touchX(x') $\wedge$ x = x'} $\wedge$
    $\exists$ y'.{touchY(y') $\wedge$ y = y'}

# MobiGolog Predicates: *User Interaction Effects on Tasks*

- Outcome of interacting with a particular UI element depends on how a user interact with it
  - Different interaction types with the same UI element may provide different results
  - A particular user interaction on a particular UI element normally results in execution of a particular task or set of tasks
- MobiGolog predicate:
  - ***mInteractionTask*(*u, ε, T*) ≡ *Λ(u, ε, T)***
  - It defines the execution of a specific task *T* based on user interaction with the mobile device or app

# Example: *Map-Viewing App*




- Functions:
  - *Zoom-in*
  - *zoom-out*
- Interaction:
  - Pressing the *plus* and *minus* buttons
  - Direct pinching –in or -out the map

TaskModel(map-viewing).
Fluent(zoom-ratio).
UI-Element(map).
UI-Element(zoom+).
UI-Element(zoom-).
UserInteraction(pinchOpen).
UserInteraction(pinchClose).
UserInteraction(tap).
UnitTask(zoom-in).
UnitTask(zoom-out).

basic predicates

CompositeModel(map-viewing).
Agent(User).
mInteractionTask(pinchOpen, map, zoom-in).
mInteractionTask(pinchClose, map, zoom-out).
mInteractionTask(tap, zoom+, zoom-in).
mInteractionTask(tap, zoom-, zoom-out).

user interaction on a particular UI element

Precondition(zoom-in) ≡ zoom-ratio < 10 .
Precondition(zoom-out) ≡ zoom-ratio > 1.
Postcondition(zoom-in, zoom-ratio, true) ≡ zoom-ratio + 1.
Postcondition(zoom-out, zoom-ratio, true) ≡ zoom-ratio – 1.

precondition and postcondition axioms

**Proc** map-viewing
[*User* ( mInteractionTask(pinchOpen, map, zoom-in) |
mInteractionTask(pinchClose, map, zoom-out) |
mInteractionTask(tap, zoom+, zoom-in) |
mInteractionTask(tap, zoom-, zoom-out) ]*
**end**

# Part – IV

## Evolving Prototypes Towards the Best-Suited Design and Interaction Schema

# Problem Description

- Challenges for the mobile app development teams:
  - Advancements in the mobile domain
    - e.g., multi-touch gesture interaction
  - Market pressure
  - Short development time
  - Limited resources
  - etc.

# Problem Description

- Selecting the final prototype with the *best-suited* design and interaction schema requires:
  - Generation of a number of initial prototypes
  - Detailed evaluation
  - Time and efforts
  - Resources
- Even the selected prototype *may not* be the best one!
  - because, it may provide better design and interaction for some parts while less for the remaining parts

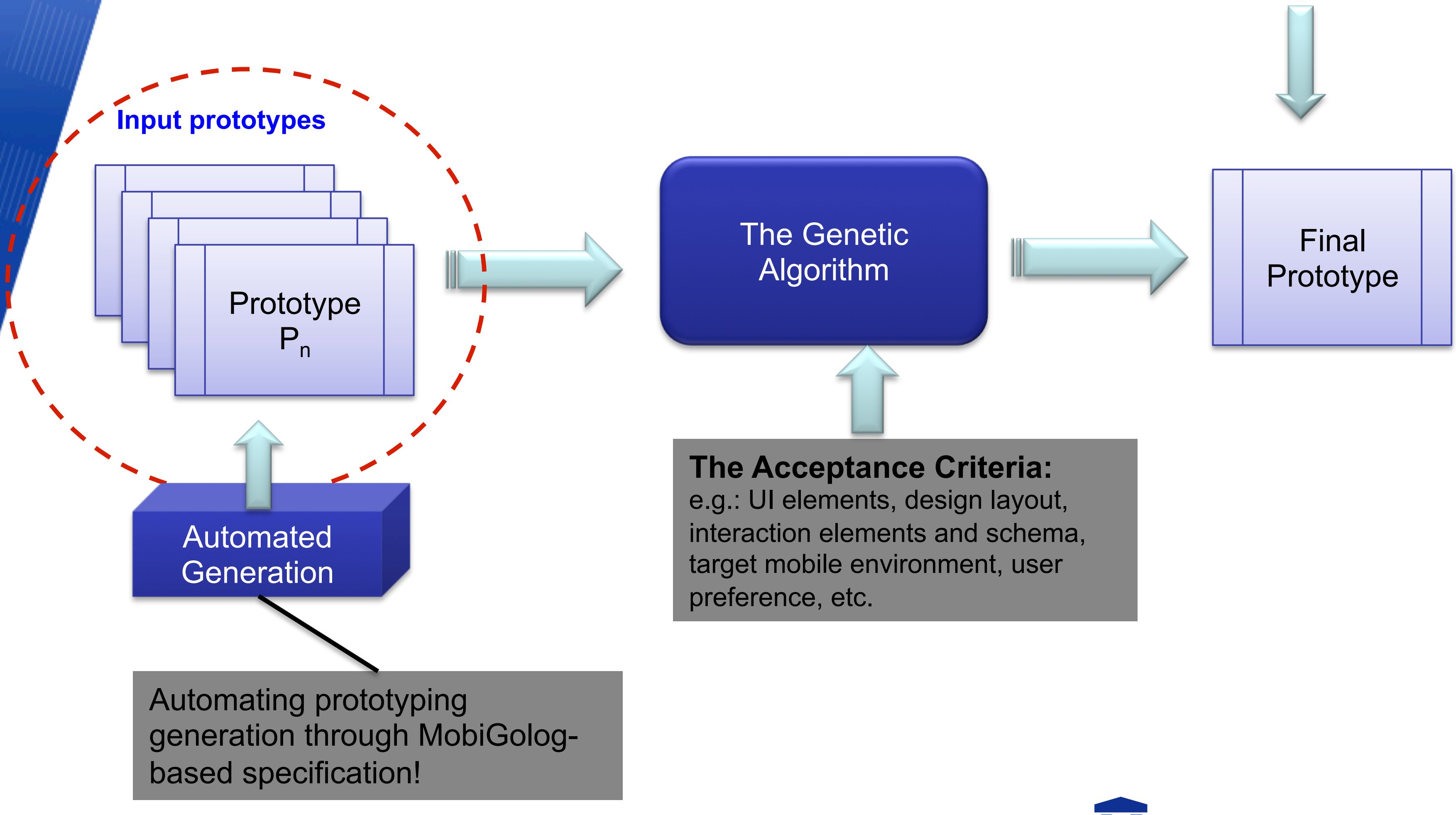
Computer Graphics and HCI Group

# Our Approach

- Evolving prototypes towards the final prototype with the possible *best-suited* design and mobile interaction schema
- Two steps:
  - Automated creation of the potential candidate solutions (i.e., target mobile app UIs) using MobiGolog-based specification
  - Application of the Genetic Algorithm for reaching to the best solution through the evolutionary process

# Our Approach

- Towards the final prototype with the possible *best-suited* design and mobile interaction schema

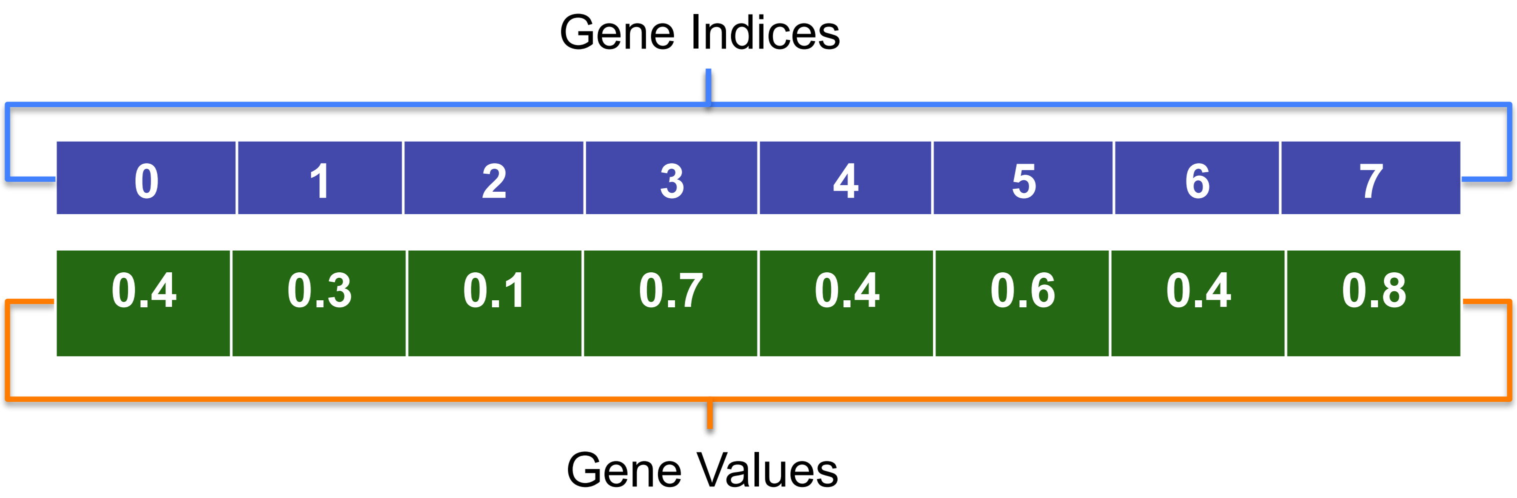


# The Genetic Algorithm

- Searching Algorithm
- Applies the natural evolutionary process on a set of potential solutions
- Generates a pool of solutions to select one among them
- Each generated solution represents one possible chromosome in the final representation
- The process consist of four steps:
  - 1 - Chromosome Encoding
  - 2 - Crossover
  - 3 - Mutation
  - 4 - Elitism

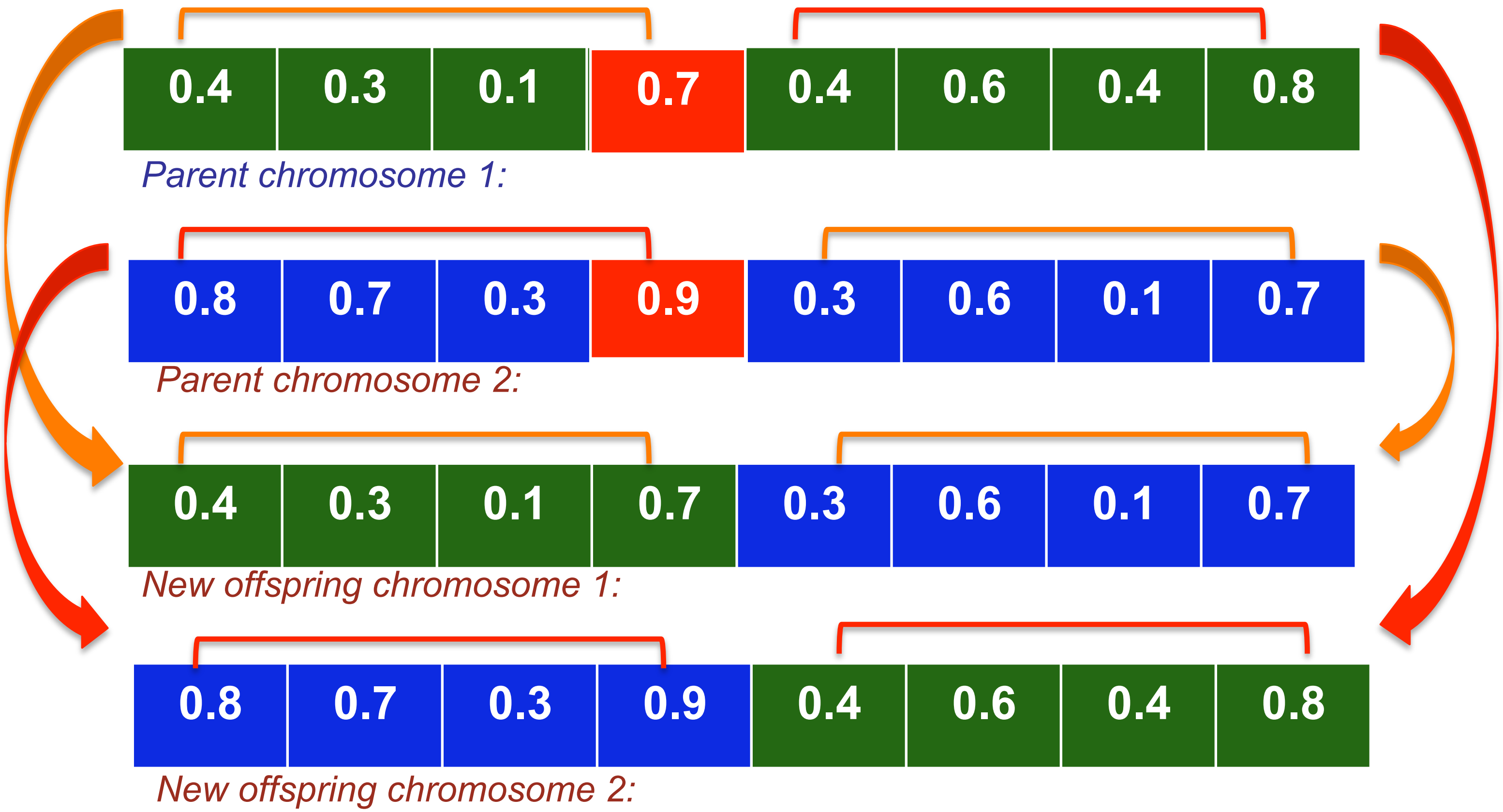
# Step 1: Chromosome Encoding

- Representing the data into chromosomes
- Each chromosome represents one of the candidate solutions in the search space



# Step 2: Crossover

- Genes are selected from different parent chromosomes, and then new offsprings are created



# Step 3: Mutation

- The mutation step changes randomly the new offspring
- This prevents falling all solutions in the population into a local optimum of solved problems

***Chromosome 1:***

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.4 | 0.3 | 0.1 | 0.7 | 0.4 | 0.6 | 0.4 | 0.8 |

***After mutation:***

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.4 | 0.3 | 0.1 | 0.7 | 0.4 | 0.9 | 0.4 | 0.8 |

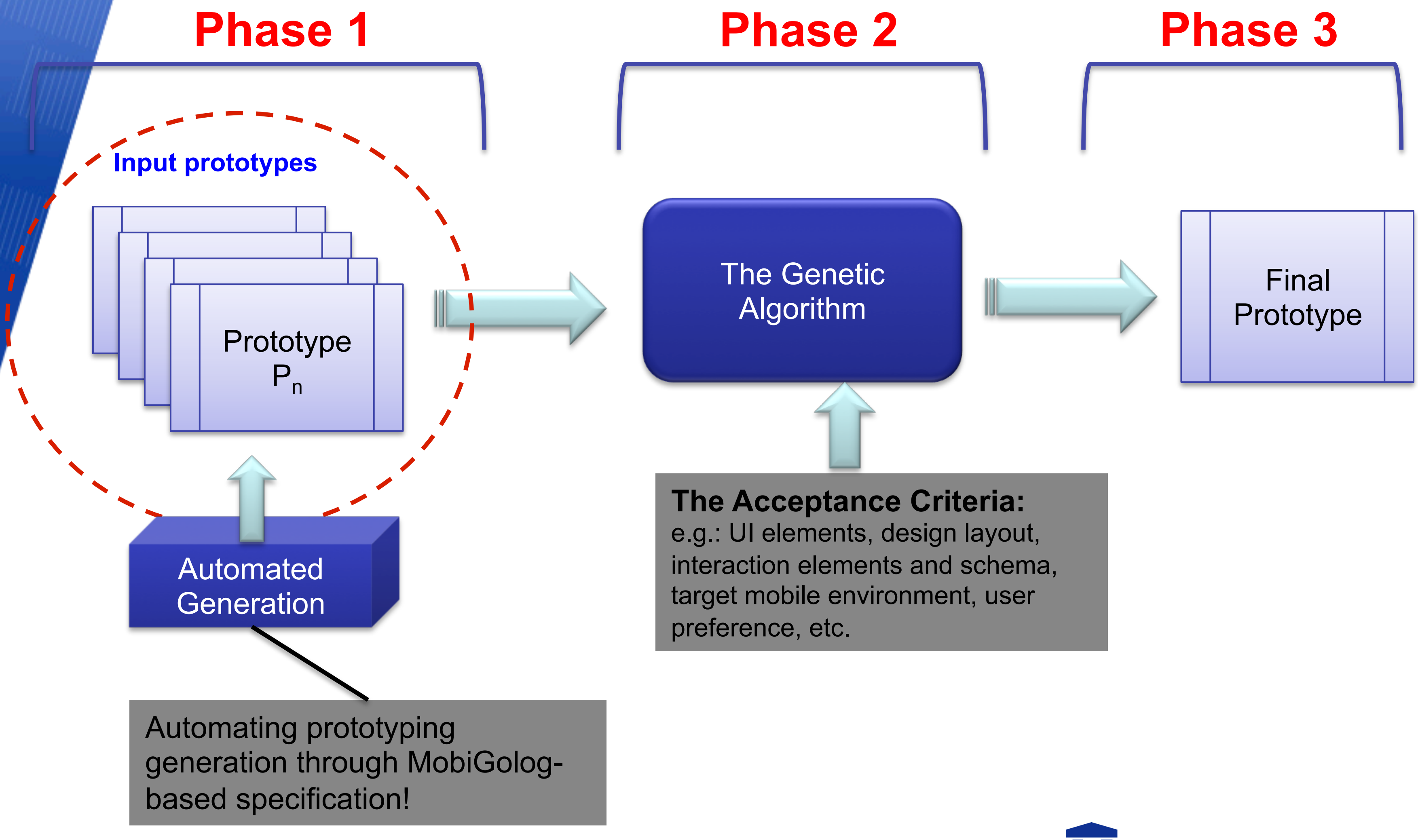
# Step 4: Elitism

- The best chromosomes (or the few best ones) are first copied and then are replaced with the old population in order to eliminate the bad chromosomes
- The GA proceeds till the last three stages have repeated to the maximum number of iterations or the GA reaches to the optimal solution

# The Methodology

- **Phase 1:**
  - Mobile app specification is defined formally through MobiGolog
  - All possible combinations of UI elements and interaction schemas are generated automatically based on the required set of functionality
- **Phase 2:**
  - The genetic algorithm is applied on the generated UIs
- **Phase 3:**
  - The final UI specification is generated based on the final solution produced by phase 2

# The Methodology



# Automation of Candidate UIs Generation

- In the current mobile domain, many factors are important in user interface, such as:
  - UI elements
  - Design layout
  - Interaction schema (e.g., multi-touch gestures)

# The Genetic Algorithm Application

- The ***best solution*** is based on the highest acceptance ratio

- The ***highest acceptance ratio*** is measured using the weight value of the acceptance criteria, which is:
  - *A combination of the design layout, the UI elements, the mobile interaction elements and schema, the target mobile environment, and the target users and their preference*

- The ***weight value*** of a particular functionality depends on the how this is formulated in the underlying prototype

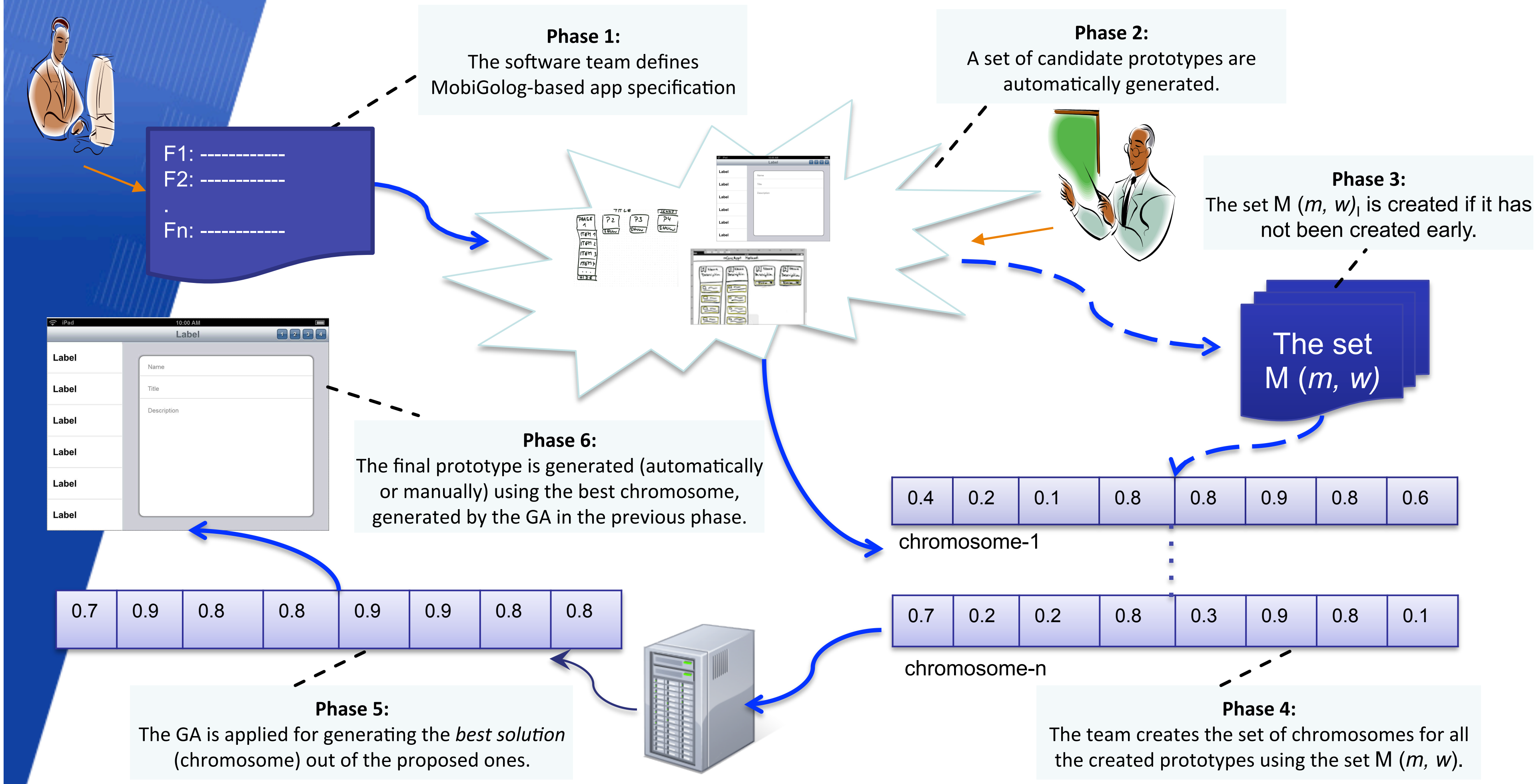
- The different variations between the weight value, due to the different formulation of combinational elements, define the fitness of the proposed solution

# Example: Map-Viewing App

- An example of weighting value based on formulation possibilities:
  - Only through plus-and-minus button
  - Only through pinching gesture with two figures
  - Through combination of above two

| Functionality name | Formulation | Weight Value |
|---|---|---|
| Zooming | plus-minus button | 0.5 |
| Zooming | pinching gesture | 0.7 |
| Zooming | both | 0.9 |

# The Workflow



# Concluding Remarks

# Concluding Remarks

- The current mobile paradigm is fundamentally different than of the conventional desktop paradigm
- It brings new problems and challenges at different levels
- Interaction designing phase is one of the most effected phases
- New approaches, methods and techniques

# Concluding Remarks

- Work has been started in many directions
  - We presented part of our work
- However, it is just a start and a long way is ahead!

# Acknowledgement!


- The works presented here were in collaboration with a number of people. I especially would like to mention few of them (alphabetically):
- Ragaad AlTarawneh (Uni. of Kaiserslautern, Germany)
- Prof. Dr. Achim Ebert (Uni. of Kaiserslautern, Germany)
- Steffen Hess (Fraunhofer IESE, Germany)
- Dr. Yael Dubinsky (IBM Research – Haifa, Israel)
- Franca-Alexandra Rupprecht (Uni. of Kaiserslautern, Germany)

# Thank You!

## Questions