

An LLM-based Method to Generate Instructions for Users How to Navigate Web-based User Interfaces

Doruk Gerçel

21st October 2024, Master Thesis Kick Off Presentation

Chair of Software Engineering for Business Information Systems (sebis)
Department of Computer Science
School of Computation, Information and Technology (CIT)
Technical University of Munich (TUM)
www.matthes.in.tum.de

Outline



Motivation

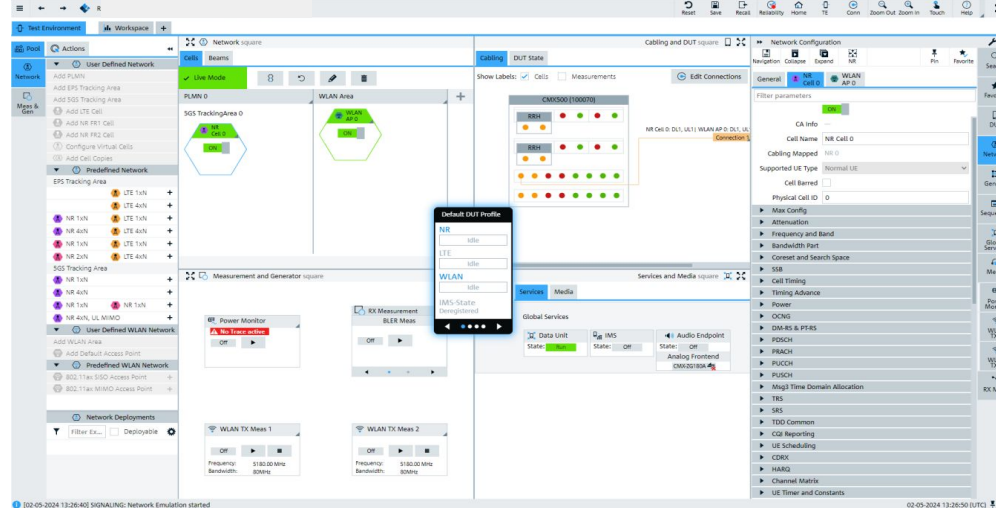
Research Questions

Initial Findings

Evaluation

Timeline

- ▶ Modern Technology Company has a Web-based UI Application:
 - ▶ With lots of Components, Options and Configurations
 - ▶ Accessible from Standard Web Browser
- ▶ They want to Design and (Partially) Implement LLM Chatbot for their Web UI Application:
 - ▶ Make Web UI Setup Easier with Virtual Assistance
 - ▶ Guide User in the UI according to the User's Prompt
 - ▶ Integration of LLM to the Product



- ▶ **RQ1)** How to model UI components to feed LLM in order to generate required steps to perform UI actions?
- ▶ **RQ2)** How to generate UI instructions from the textual output generated from the LLM?
- ▶ **RQ3)** How to integrate this chatbot to the product by following and considering software development and deployment principles?
 - Source Code (Repository) Structure
 - Code Integration

Research Question 1

- ▶ **RQ1)** How to model UI components to feed LLM in order to generate required steps to perform UI actions?
- ▶ **Steps:**
 - ▶ Searching for Related Work to Find Appropriate Textual Data Model for UI Components
 - ▶ Comparing and Contrasting Related Work to Implement a Data Model
 - ▶ Uniquely and Semantically Identifying UI Components
 - ▶ Describing UI Components' Attributes and Properties

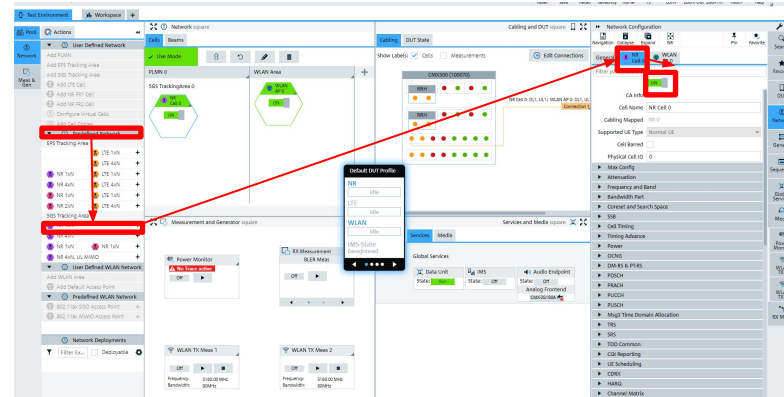


Research Question 2

- ▶ **RQ2)** How to generate UI instructions from the textual output generated from the LLM?
- ▶ **Steps:**
 - ▶ Converting text responses from LLM to semantic UI operations with causal chaining
 - ▶ Finding an abstract textual format for UI operations
 - ▶ Designing an appropriate approach to perform user guidance with LLM

LLM

Convert from text to UI operations



Research Question 3

- ▶ **RQ3)** How to integrate this chatbot to the product by following and considering software development and deployment principles?

- ▶ **Steps:**
 - ▶ Design and (partially) implement interface to access UI components
 - ▶ Implement adapter for conversions from text to UI operations
 - ▶ Design and implement chatbot for user interactions
 - ▶ Test and validate whether it produces the desired results

► Two Approaches for Textual Representation *from Academic Papers*

► **Model Driven:**

- Use Cases defined with Necessary Components
- Defines Navigation from/to Components
- Use Cases are State Machines where Components are States
- Use Cases Defined Directly
- Component' Mission Description not Explicitly Defined
- Components are not Well Structured

► **Event Driven:**

- Components with Initial States, Forwarded States & Transitions
- Defines Navigation with State Transitions
- Each Component is a State Machine (with Connections to Other Machines)
- Navigation are Hard to Define

► **Both:**

- Mostly used in Internal Communication in Company
- Do not Leverage the Intelligence of LLMs

Model Driven:

```
Model:
  (includes += Include)*
  (elements += ModelElement)*;
ModelElement:
  ViewSet | UseCase | Include;
Include:
  "include" model = URI;
ViewSet:
  "viewset" name = ID (title = STRING)?
  (description = STRING)?
  (elements += InteractionElement)*
  "end";
InteractionElement:
  InputElement | EventElement | ListElement;
```

Grammar

```
usecase MainMenu "Main menu"

  interaction ShowPhoneBook withviewset MainMenu
    User can :open_phonebook(people:getpersons)
    and the >ManagePhonebook section
    will be showed.
  end
end
```

Example

Event Driven:

```
model <model name>

entity <interaction element> {
  actions: <list of actions>
  states: <list of states>
  properties: <list of properties>
}
```

Grammar

```
entity widget {
  actions: move
  states: displayed, shown, available, enabled, disabled
}

entity CheckBox is widget {
  actions: check, pick, select, choose, uncheck
  states: checked, unchecked, picked, selected, chosen
  properties: option, options
}
```

Example

► **Proposed Grammar Attributes:**

- Include Relevant Component Attributes
- Include Navigation from/between Components
- Uniquely Identify Components with IDs
- Include Component Descriptions for Use Cases

- **Make it Recognizable by LLM**
- **Intelligence of LLM Shall be Leveraged**

Initial Findings - RQ1 - Proposed Grammar

```
ComponentName:
  ID: #id
  ComponentSelector: #css-selector
  Label: #label-str | null
  Description: #usage-str | null
  UserAssistanceID: #id | null
  ComponentType: #component-key | null
  IsGlobalView: #true | false | null
  IsExpandable: #true | false | null
  CanBeDisabled: #true | false | null
  Attributes:
    HasValues:
      - #value1
      - #value2
    Opens: #component-key | null
    NavigatesTo: #component-key | null
    ChangesTo: #component-key | null
    EnablesDisables:
      - #component1
      - #component2
    Contains:
      - #comp1
      - #comp2
  HasDynamicValues: #true | false | null
  UserInput:
    Type: #text | number
    Placeholder: #value | null
```

Proposed Grammar

```
HamburgerBtn:
  ComponentType: Button
  IsGlobalView: true
  Attributes:
    Opens: MenuSection

MenuSection:
  ComponentType: Section
  Attributes:
    Contains:
      - Home
      - Current Workspaces
      - Message Analyzer
      - Sequencer
      - Use Cases
      - Library
      - File Explorer
      - License
      - Settings
      - About
      - Maintenance
      - User Assistance
      - Installation Service
      - Data Services
      - Utilization/Security

SettingsSection:
  ComponentType: Section
  Attributes:
    Contains:
      - Device
      - Reference Frequency
      - Trigger
      - Date & Time
      - Display
      - Restart
      - SCPI Logging
      - Detailed Device API Logging
      - Detailed Database Logging
```

Example Description of UI with Grammar

```
DisplaySection:
  ComponentType: Section
  Attributes:
    Contains:
      - Color Set

ColorSetDiv:
  ComponentType: Div
  Attributes:
    Contains:
      - ColorSetText
      - ColorSetDropdown

ColorSetText:
  ComponentType: Text
  Attributes:
    Value: Color Set

ColorSetDropdown:
  ComponentType: Dropdown
  Attributes:
    HasValues:
      - Light background
      - Dark background

DeviceSection:
  ComponentType: Section
  Attributes:
    Contains:
      - HostnameDiv
      - DeviceGroupDiv
      - oneOf:
        - EditButton
        - DeviceModifyDiv
```

Input:

You are in an arbitrary page, can you change the background color

Output:

Sure, let's go through the steps to change the background. Here are the detailed steps:

Step 1: Click the Hamburger Button

Locate the HamburgerBtn on the UI. This button is typically represented by three horizontal lines (≡).
Click on the HamburgerBtn to open the MenuSection.

Step 2: ...

Step 3: ...

Step 4: ...

Step 5: Change the Color Set

In the ColorSetDiv, locate the ColorSetDropdown.

Click on the ColorSetDropdown to see the available options.

Select either "Light background" or "Dark background" from the dropdown menu.

► Robot Framework:

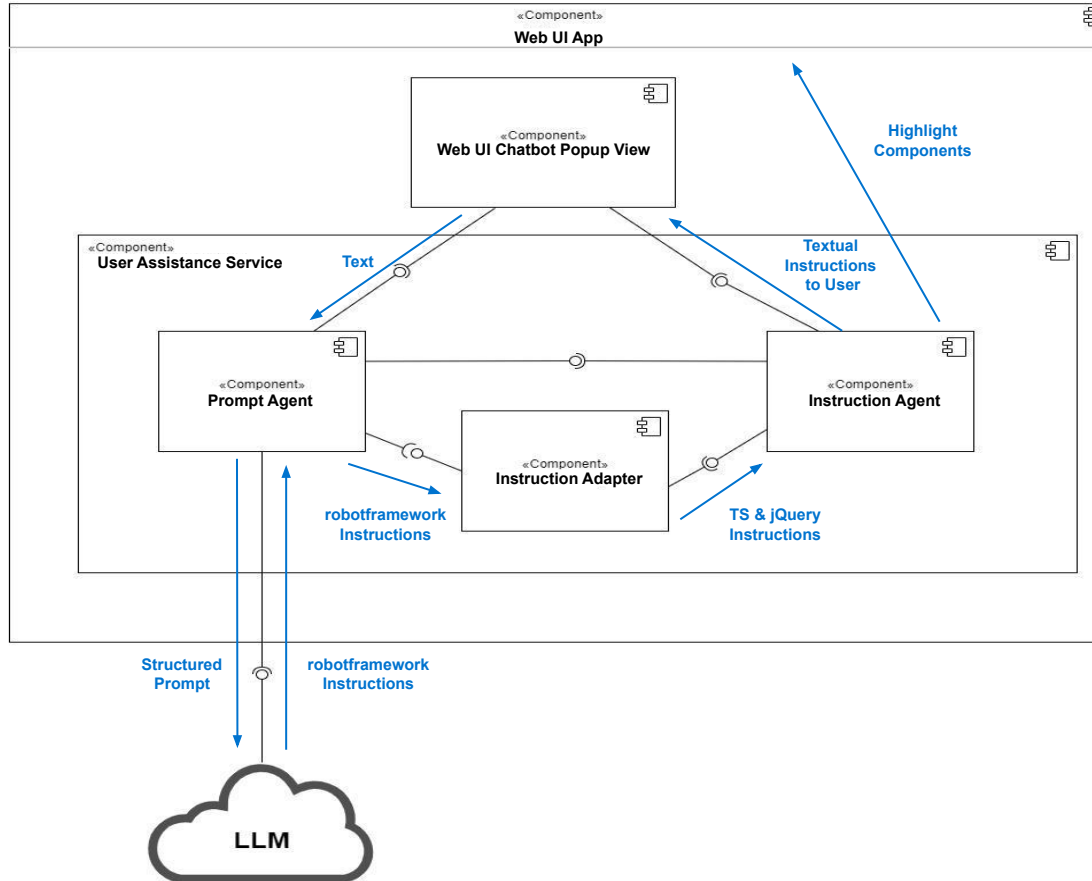
► Properties:

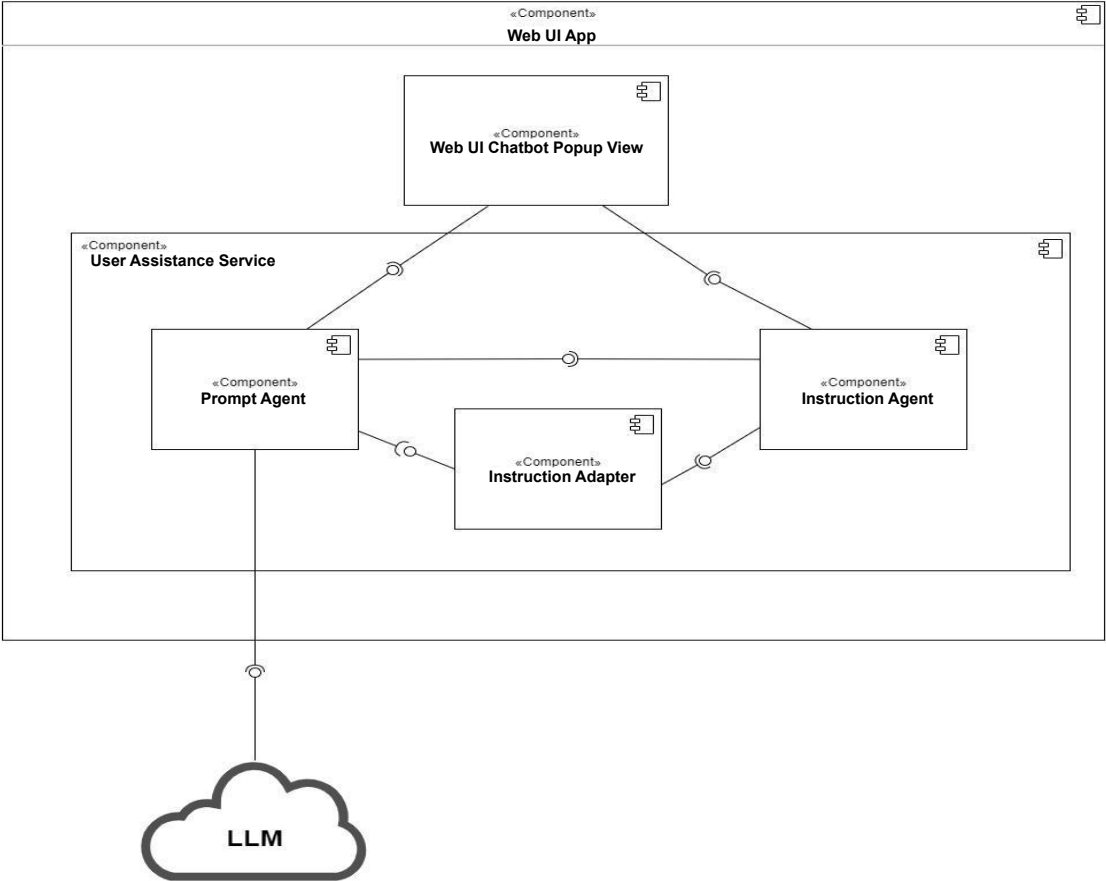
- Designed for robotic process automation (RPA)
- Uses an abstract language to interact with UI components
- ChatGPT can produce desired script for Robot Framework
- Easy to extract TS and jQuery code from the instructions
- Open source and widely used

► Potential Usage:

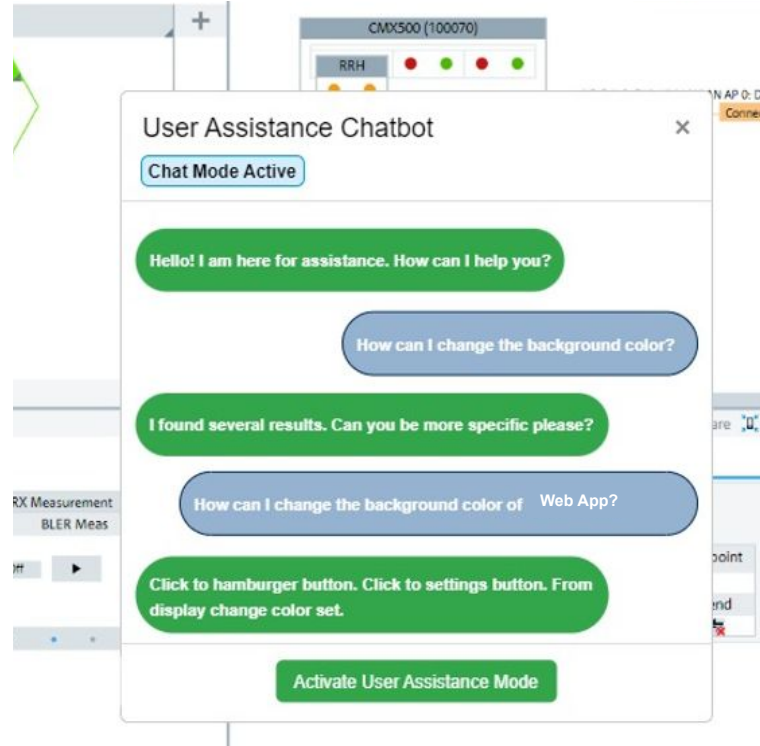
- Define necessary UI instructions
- Easy to convert from Robot Framework instructions to other programming languages
- Resourceful to obtain Robot Framework instruction from the LLM

Initial Findings - RQ3 - Architecture Overview



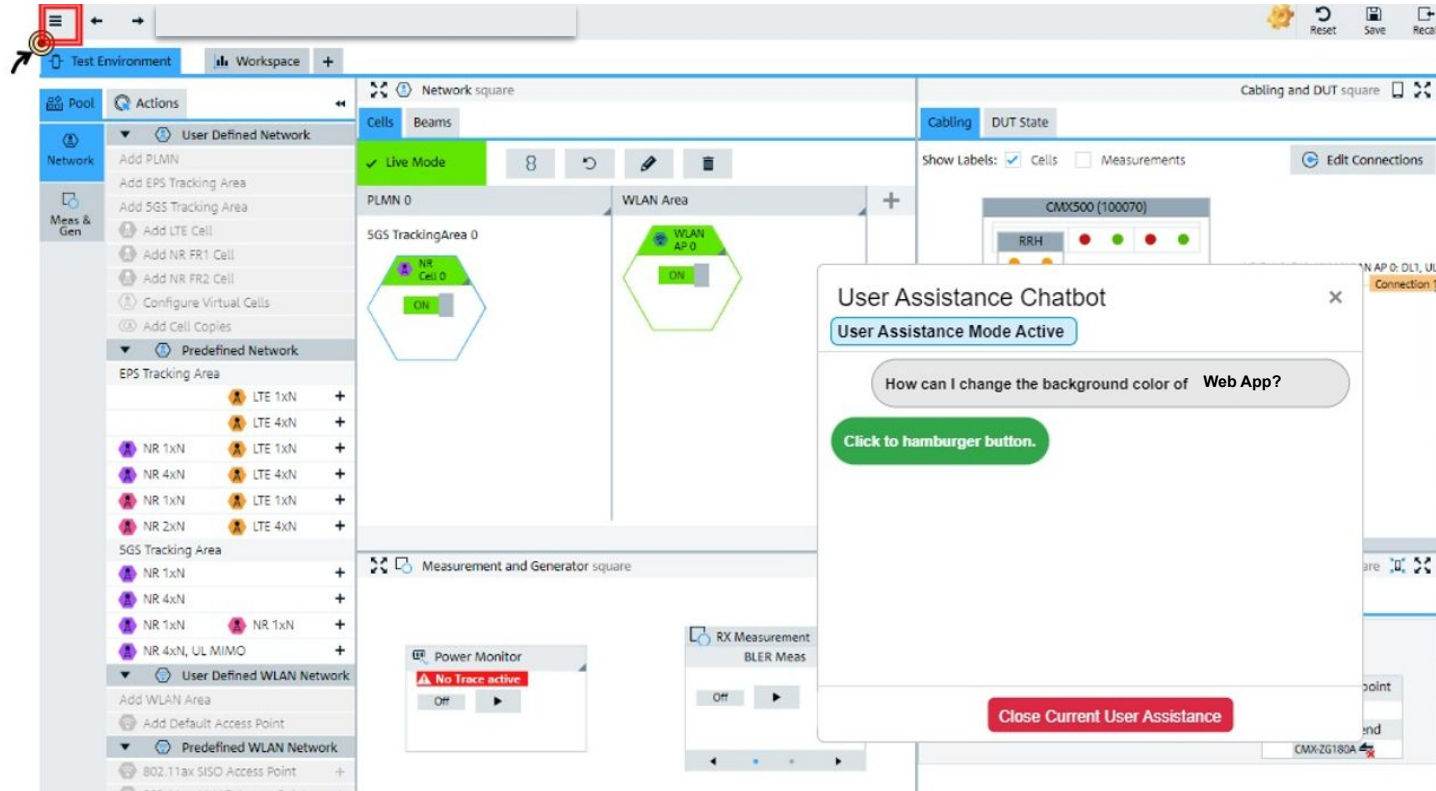


Initial Findings - RQ3 - Prototype Proposal



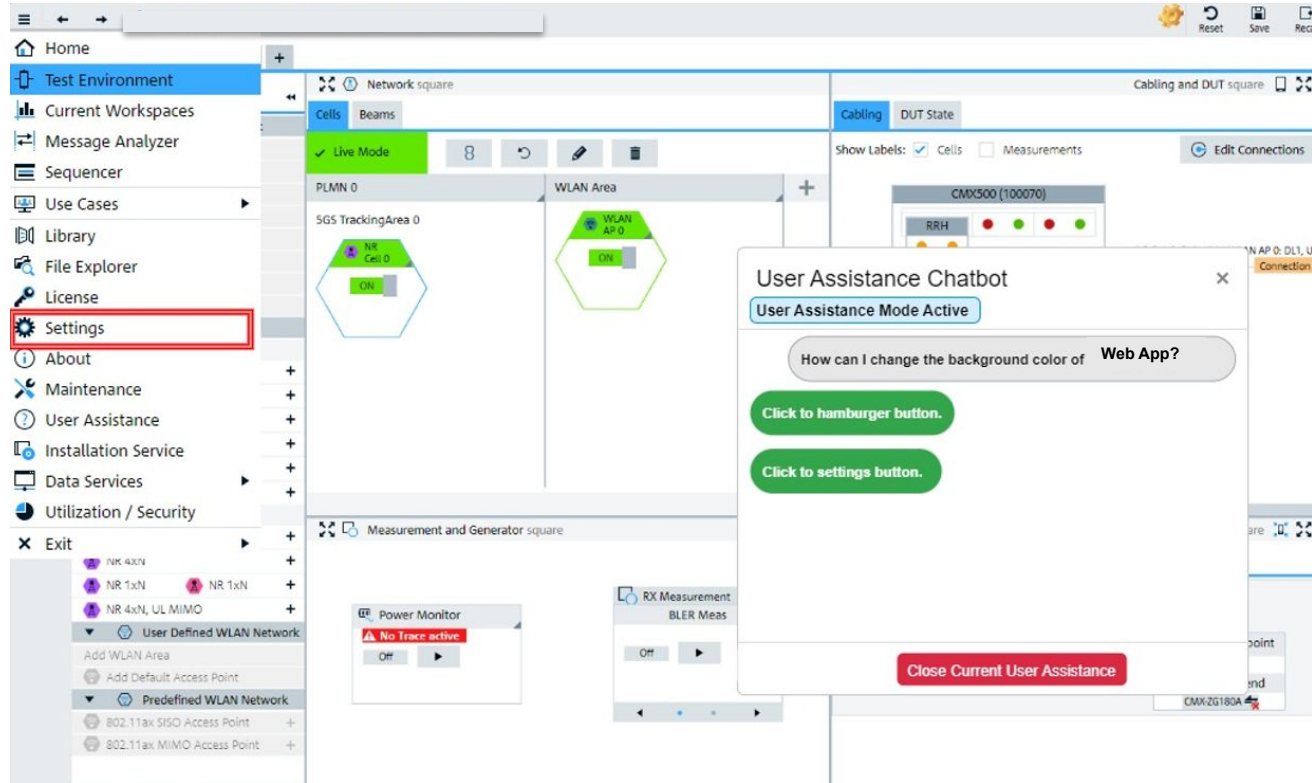
User Interaction with Prototype - 1

Initial Findings - RQ3 - Prototype Proposal



User Interaction with Prototype - 2

Initial Findings - RQ3 - Prototype Proposal



User Interaction with Prototype - 3

- ▶ Testing Test Cases:
 - ▶ From Different Starting Pages
 - ▶ Cases with Configuration Requirements

- ▶ UI and UX (Usability):
 - ▶ Feedback from Internal Customers with Surveys
 - ▶ Feedback from Product Managers and UX Researchers with Interviews

	Sep	Oct	Nov	Dec	Jan	Feb
Literature Review						
Design						
Implementation						
Experiments & Analysis						
Writing						
Special Tasks	• Thesis Registration					• Thesis Submission • Thesis Presentation



Doruk Gercel

doruk.gercel@tum.de

Technical University of Munich (TUM)
TUM School of CIT
Department of Computer Science (CS)
Chair of Software Engineering for
Business Information Systems (sebis)

Boltzmannstraße 3
85748 Garching bei München

+49.89.289.17132
matthes@in.tum.de
www.matthes.in.tum.de



- [1]** Grönninger, Hans & Krahn, Holger & Rumpe, Bernhard & Schindler, Martin & Völkel, Steven. (2014). Textbased Modeling.
- [2]** Karu, Mart. (2013). A Textual Domain Specific Language for User Interface Modelling. 10.1007/978-1-4614-3558-7_84.
- [3]** Robot Framework. robotframework.org. Accessed 14 October 2024.
- [4]** Ryan, S. (2024, May 21). Reduce LLM costs and increase speed: consider switching to YAML instead of JSON. Medium.
<https://medium.com/@mr.sean.ryan/reduce-llm-costs-and-increase-speed-consider-switching-to-yaml-instead-of-json-62af2f7a37c0#:~:text=Generally%2C%20YAML%20is%20faster%20and,More%20Efficient%20for%20Language%20Models%3F>
- [5]** Silva, Thiago. (2022). Towards a Domain-Specific Language to Specify Interaction Scenarios for Web-Based Graphical User Interfaces. 48-53. 10.1145/3531706.3536463.
- [6]** Vanderdonckt, Jean & Limbourg, Quentin & Michotte, Benjamin & Bouillon, Laurent & Trevisan, Daniela & Florins, Murielle. (2004). USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces. W3C Workshop on Multimodal Interaction.

Additional Slides

Initial Findings - RQ1 - Related Works - Grammar Examples

Model Driven:

```
Model:
  (includes += Include)*
  (elements += ModelElement)*;
ModelElement:
  ViewSet | UseCase | Include;
Include:
  "include" model = URI;
ViewSet:
  "viewset" name = ID (title = STRING)?
  (description = STRING)?
  (elements += InteractionElement)*
  "end";
InteractionElement:
  InputElement | EventElement | ListElement;
InputElement:
  (isSecureEntry ?= "secure")?
  "input" name = ID (title = STRING)?
  (description = STRING)?
  ("(" objectName = ID
    ("." property = ID )? ")")?;
EventElement:
  "event" (forElement = [InteractionElement] ".")?
  name = ID
  (title = STRING)? (description = STRING)?;
ListElement:
  "list" name = ID (title = STRING)?
  (description = STRING)?
  ("(" (columnMappings += Mapping)* ")");
Mapping:
  name = ID ":" value = ID;
UseCase:
  "usecase" name = ID (description = STRING)?
  (interactions += Interaction)*
  "end";
Interaction:
  "interaction" name = ID
  ("withviewset" viewSet = [ViewSet])?
  (navigations += NavigationRule)*
  "end";
AbstractInteraction:
  UseCase | Interaction;
NavigationRule:
  (beforeStrings += FillString)*
  ":" event = [EventElement]
  ( "(" (navigationParameters += Mapping " ") )?
  (punctuationAfterEvent += Punctuation)*
  (centerStrings += FillString)*
  ":" interaction = [AbstractInteraction]
  (punctuationAfterInteraction += Punctuation)*
  (afterStrings += FillString)*;
FillString:
  name = ID (Punctuation)*;
Punctuation:
  "." | "," | "!" | "...";
```

Source: Grammar

- (Karu, 2013)
- (Silva, 2022)

```
usecase MainMenu "Main menu"

interaction ShowPhoneBook withviewset MainMenu
  User can :open_phonebook (people:getpersons)
  and the >ManagePhonebook section
  will be showed.
end
end

usecase ManagePhonebook "Phonebook Manager"

interaction ListOfPersons withviewset PhoneBook
  User needs to :select a person from the
  list of persons to >Edit the person data.
  If user wants to :add_new person then the
  >AddNewPerson form must be filled. User
  can :exit the phonebook and to return
  to the >MainMenu.
end

interaction Edit withviewset EditPhoneBookEntry
  User modifies the Person data and
  initiates :save (people:getpersons) to store
  the changes. The system responds
  with >ListOfPersons. If the user wants
  to :cancel (people:getpersons) editing, no
  changes are stored and the user will be
  directed to >ListOfPersons. To remove a
  person user must :delete (people:getpersons)
  and after that the >ListOfPersons will
  be shown.
end

interaction AddNewPerson
  withviewset EditPhoneBookEntry
  To :create (people:getpersons) a new person,
  user enters and saves person data and the
  >ListOfPersons will be shown. If the user
  wishes to :cancel (people:getpersons) new
  person creation the >ListOfPersons
  will be displayed.
end
end

viewset MainMenu
event open_phonebook "Open phonebook"
end

viewset PhoneBook
list people
  "People in the phonebook:"
  (Name:name Number:number)
event people.select "Edit"
event add_new "Add new person"
event exit "Exit"
end

viewset EditPhoneBookEntry "Person data entry"
input name "Name" "Name of the person"
input number "Phone number"
  "Phone number of the person"
event save "Save"
event create "Create"
event delete "Delete"
event cancel "Cancel"
end
```

Example

Event Driven:

```
model <model name>

entity <interaction element> {
  actions: <list of actions>
  states: <list of states>
  properties: <list of properties>
}
```

Grammar

```
model widgets

entity widget {
  actions: move
  states: displayed, shown, available, enabled, disabled
}

entity CheckBox is widget {
  actions: check, pick, select, choose, uncheck
  states: checked, unchecked, picked, selected, chosen
  properties: option, options
}
```

Example

► **YAML or JSON:**

- Response Time:
 - YAML: 5,000ms vs JSON: 20,000ms
 - Switching from JSON to YAML boost 2.5 times
 - YAML reduces tokens and characters
 - Curly brackets in the JSON increases response time of LLM
 - YAML more appropriate for natural language representation so better for LLM as well
 - Both JSON and YAML produce semantically similar responses
- **Conclusion:** Choose YAML as it produces correct responses in shorter time

Input:

You are in an arbitrary page, can you change the background color

Output:

Sure, let's go through the steps to change the background. Here are the detailed steps:

Step 1: Click the Hamburger Button

Locate the HamburgerBtn on the UI. This button is typically represented by three horizontal lines (≡).
Click on the HamburgerBtn to open the MenuSection.

Step 2: Navigate to Settings

In the MenuSection, you should see a list of items.
Find the item labeled Settings in the list.
Click on Settings to open the SettingsSection.

Step 3: Navigate to Display

In the SettingsSection, you should see a list of settings options.
Find the item labeled Display in the list.
Click on Display to open the DisplaySection.

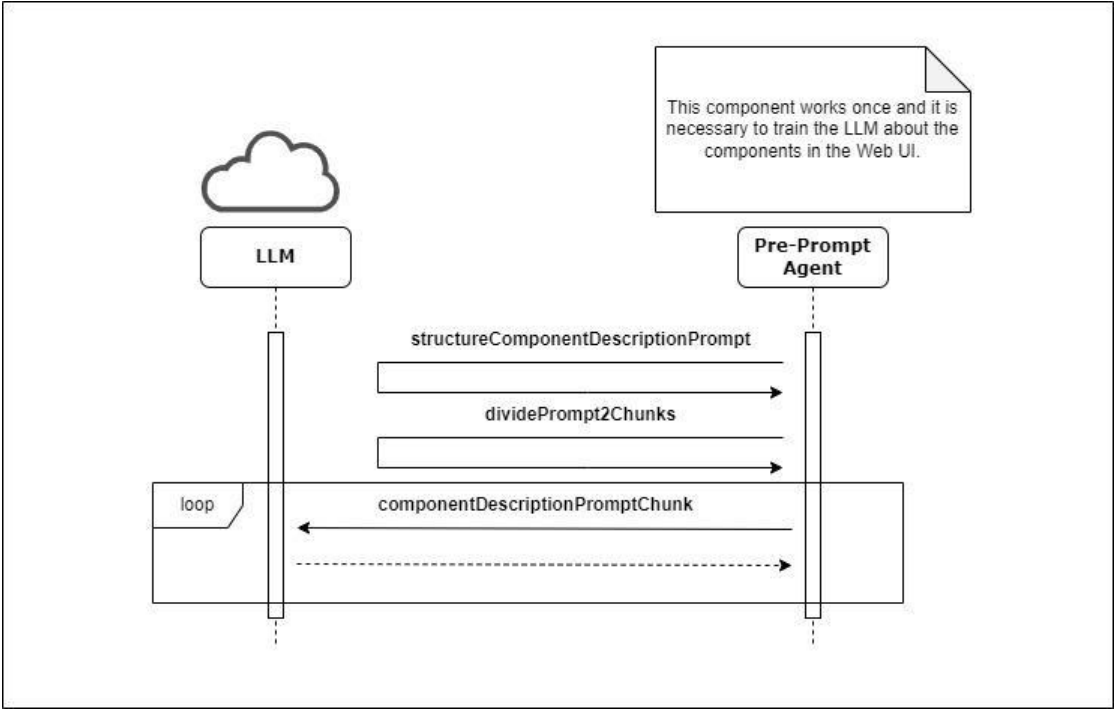
Step 4: Locate the Color Set

In the DisplaySection, locate the ColorSetDiv.
The ColorSetDiv contains both ColorSetText and ColorSetDropdown.

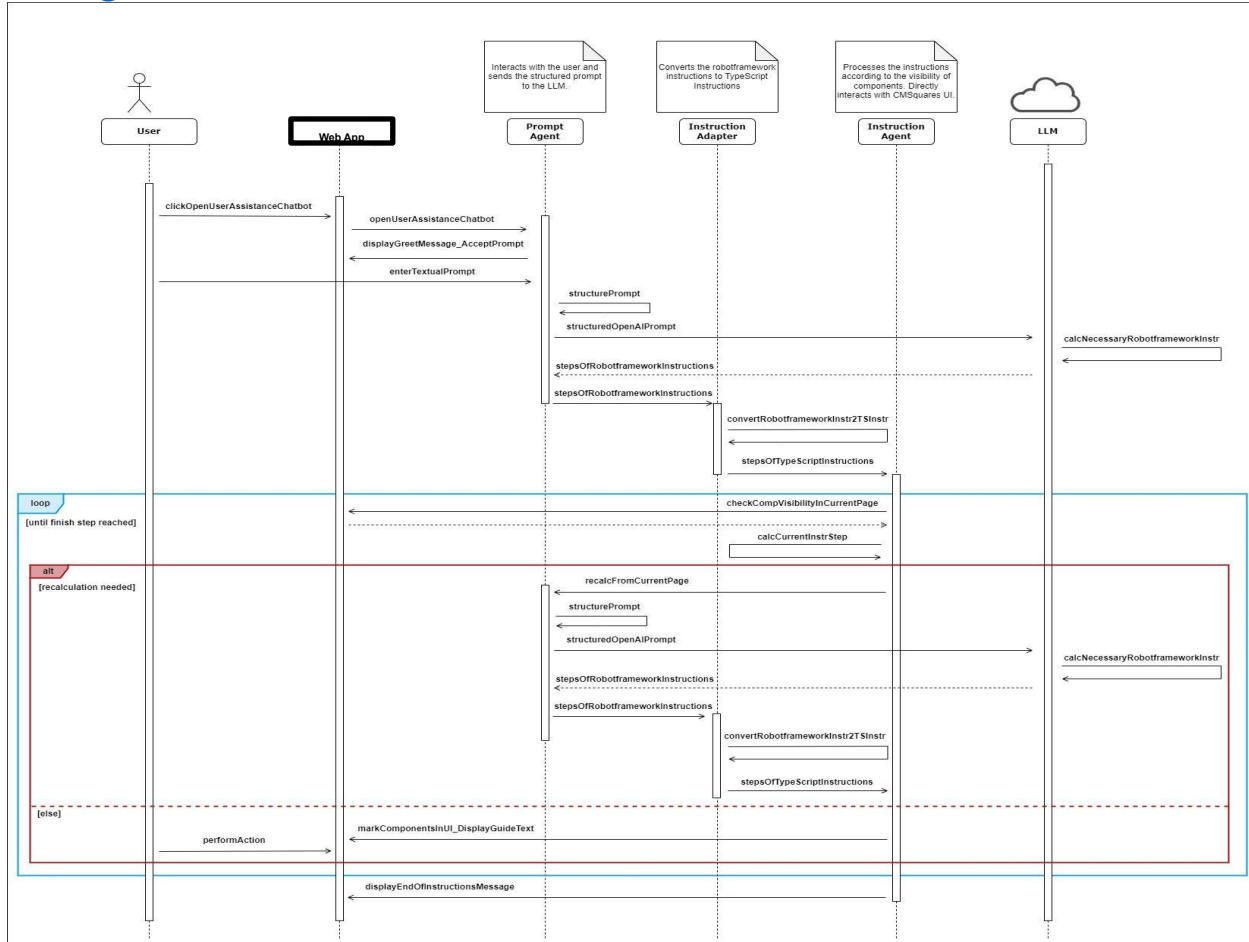
Step 5: Change the Color Set

In the ColorSetDiv, locate the ColorSetDropdown.
Click on the ColorSetDropdown to see the available options.
Select either "Light background" or "Dark background" from the dropdown menu.

Sequence Diagram - Feed LLM Phase



Sequence Diagram - User Interaction



Finite State Machine Diagram - Algorithm for User Guidance

