Homework 1: Three in a row game UI

- **Due:** Next Thursday 2/27, 9 AM EST
- **Design:** Aim for open-closed principle
  - But should at least improve encapsulation
- **Implementation:** Refactor MVC and 4 violations of best practices
  - Don’t need to implement Observer or proposed extension
- **Office hours:** This Friday AM (Heather), next Monday PM (William), and next Tuesday AM (Heather)

Today

- Requirements engineering
- Specification
- Specification mining (i.e. model inference for inferring processes)
- Final project selection

Requirements engineering: What is a software requirements specification?

- Documents the assumptions about, features requested, and behavior of a given software application excepted by the users

- Defines a set of requirements that must be satisfied by the software application
Requirements engineering: Two key types of requirements

• Non-functional requirement: A quality constraint on the software application
  – e.g., Performance (in terms of space or time), security
• Functional requirement: An intended (or unintended) behavior of the software application
  – e.g., The Three in a Row game must use the model-view-controller architecture pattern.

NOTE) There are other types of requirements to describe assumptions, features, and usage scenarios (e.g., UML use cases).

Requirements engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

Requirement → Design → Implementation
Requirements engineering: Phases

1. Elicitation
2. Specification
3. Analysis
4. Management

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Design</th>
<th>Implementation</th>
</tr>
</thead>
</table>

Specification: Natural language

- Elicitation often produces requirements written in natural language
- Sample:
  1. The ‘Three in a Row’ game must use the model-view-controller architecture pattern.
     a. There must be a single game model.
     b. Once the new game model is created, that model can have its state changed.
     c. If the game model has a state change, then its game view must be updated.
     d. ...
  2. ...

Specification: Disadvantages of natural language

- Natural language is often ambiguous
- Such ambiguity can lead to human misunderstandings
- The ambiguity also means validation & verification cannot be carried out by applying automated tools, e.g.,
  - Testing, model checking

Therefore mathematical notations (e.g., finite state automata) are commonly used because they are rigorously defined
Specification:
Finite state automaton (FSA)

Event sequence over an alphabet (meaning a set of events)

Finite state automaton (FSA)

or

accept ✓

reject ✗

FSA: Sample 1a

There must be a single game model.

Event sequence:
[gameModelNew, gameModelNew]
There must be a single game model.

Event sequence: [gameModelNew, gameModelNew]

accept

Event sequence: [gameModelNew]

reject

Event sequence: []
FSA: Sample 1a

There must be a single game model.

Event sequence: []

reject

FSA: Sample 1b

gameModelNew  gameModelStateChange

FSA: Sample 1b

gameModelNew  gameModelStateChange

Some event sequences that are accepted (i.e. positive examples)?

FSA: Sample 1b

gameModelNew  gameModelStateChange

Some event sequences that are accepted (i.e. positive examples)?

[gameModelNew]
[gameModelNew, gameModelStateChange]
Other event sequences that are rejected (i.e. negative examples)?

[]

[gameModelStateChange]

[gameModelNew, gameModelNew]

Once the new game model is created, that model can have its state changed.

Represented as a 5 tuple:

• \( E \): The alphabet as a set of events
• \( S \): The set of states
• \( T \): The set of transitions where each transition is a directed edge from a source state \( s \) on event \( e \) to target state \( t \)
• \( s_0 \): The start state
• \( A \): The set of accepting states

NOTE: Can be automatically converted to a regular expression
FSA: Disadvantages

- Natural language is generally more accessible than FSAs
- Hand writing FSAs can be error prone and time consuming

FSA: Disadvantages

- Natural language is generally more accessible than FSAs
- Hand writing FSAs can be error prone and time consuming

Specification patterns have been identified of commonly occurring intended (or unintended) behaviors of software applications

Property specification pattern: Formal definition

- Consists of:
  - **Behavior** captures occurrence or order of events/propositions (e.g., Bounded existence of `gameModelNew`)
  - **Scope** captures parts of the trace where behavior must be satisfied (e.g., `Globally`)
- Provides mapping to various property specification languages (e.g., regular expressions)
  - e.g., `gameModelNew`

(Property specification pattern: Behaviors)

- **Occurrence of event/proposition A**
- **Order of events/propositions A and B**
- **Absence**
- **Universality**
- **(Bounded) Existence**
- **(Chained) Precedence**
- **(Chained) Response**
GameModelNew must precede GameModelStateChange.

Scope: Globally
Property specification patterns: Disadvantages

- Use 80/20 rule
  - May need to hand write FSAs
- Don’t provide support for real-time or probabilistic constraints
  - Has been extended [See http://ps-patterns.wikidot.com]
- Designed for experts
  - Can be difficult for novices to select among patterns and customize them

Property specification patterns: Disadvantages

- Use 80/20 rule
  - May need to hand write FSAs
- Don’t provide support for real-time or probabilistic constraints
  - Has been extended [See http://ps-patterns.wikidot.com]
- Designed for experts
  - Can be difficult for novices to select among patterns and customize the selected pattern

PROPEL tool: What?

- Builds on the property specification patterns specified as finite state automata
- Provides guidance to select among the patterns and customize that pattern

PROPEL tool: User interface

Available from http://laser.cs.umass.edu/release/
PROPEL tool: Demonstration

Specify the following: “If the game model has a state change then its game view must be updated”

PROPEL tool: Sample FSA 1c

Behavior: `gameModelStateChange` must have response `gameViewUpdate`  
Scope: Globally

PROPEL tool: Sample DNL 1c


- Log represented as an event sequence of method calls
- Generated model represented as a finite state automation (FSA)

Specification mining:
Example log

<table>
<thead>
<tr>
<th>Log:</th>
</tr>
</thead>
<tbody>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.232.201</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
<tr>
<td>174.15.155.103</td>
</tr>
</tbody>
</table>

Specification mining:
Example generated model

Specification mining:
How to diff two FSAs

- Execute a given event sequence
- Generate a set of positive examples (i.e. a set of events sequences that are accepted)
  - Generate a set of negative examples (i.e. a set of events sequences that are rejected)
- ...

Specification mining:
How to diff two FSAs
Final project: Topic selection

- Form team of 4 or 5 students
- Select one of the following 4 topics:
  1. MSR 2020 mining challenge
  2. Replication study
  3. Model inference for inferring processes (i.e. specification mining)
  4. EleNa: Elevation-based navigation
- Due: Tuesday March 3, 2020 9 AM EST

https://people.cs.umass.edu/~hconboy/class/2020Spring/CS520/finalProject.pdf

Final project: Selected topic

1. Read some papers
2. Start to develop
3. Create and give a mid-point presentation
4. Continue to develop
5. Create and give a final presentation
6. Write a final report

Final project: MSR 2020 mining challenge objectives

- Read 8-10 papers
- Select one or more research questions
- Propose an approach to investigate the research question(s)
- Develop experiments to evaluate the proposed approach by applying to the provided dataset
- Study the experimental results


Final project: Replication study objectives

- Read 4-5 papers
- Select one of the debugging/testing tools (e.g., SOSRepair)
- Learn about the benchmark (i.e. input data) for the selected tool
- Replicate the experiments to evaluate the selected tool by applying to the same data
- Extend the experiments to further evaluate the selected tool on additional data
Final project: Specification mining objectives

- Read 4-5 papers
- Select a model inference tool or tools (e.g., Synoptic, InvariMint)
- Select a reasonable way to generate traces
- Develop experiments to evaluate the selected tool(s) by applying to the generated traces
- Study the experimental results
  - Develop an automated approach to diff two FSAs

https://github.com/ModelInference/synoptic

Final project: Elevation-based Navigation (EleNa)

- **Goal:** Develop a software system that determines, given a start and an end location, a route that maximizes or minimizes elevation gain, while limiting the total distance between the two locations to \( x \% \) of the shortest path
- **Components:**
  - Data model that represents the geodata
  - A component that populates the data model, querying, e.g., https://www.openstreetmap.org
  - The actual routing algorithm that performs the multi-objective optimization
  - Another component that outputs or renders the computed route

Final project: EleNa objectives

- Read any necessary technical documents
- Design one or more components
- Implement the designed component(s)
- Build a test plan for the implemented component(s) and carry out that test plan