

# PIRRO as a reactive system

Team C: Andreas, Gerben, Joanna, Pieter

Slot 13

# Approach

- ▶ We use a state machine to describe the behaviour of the system.
- ▶ The state of the system is the combined states of each component.
- ▶ To specify navigation between screens, we model a state machine that keeps the required information in the state of the system. We denote this component as the *Navigation Manager*.

# Assumptions

- ▶ We assume all the components of the system are specified as state machines.
- ▶ The input and output actions of all the components are distinct.
- ▶ The system is composed of :  $S = \{ \text{Compositor, Version Management, Piece Management, ASS, Communication System, User Management, Navigation Manager} \}$ .

# Naming conventions

For each components reactive system we use the following name convention:

- ▶ The state is formed by adding the suffix *State* to the name of the component.
- ▶ The set of input events is formed by adding the suffix *Input* to the name of the component.
- ▶ The set of output events is formed by adding the suffix *Output* to the name of the component.

We denote this by the functions:

$\text{StateName} : S \rightarrow \text{State } S$	$\text{StateName}(\text{Compositor}) = \text{CompositorState}$
$\text{InputName} : S \rightarrow \text{Input } S$	$\text{InputName}(\text{Compositor}) = \text{CompositorInput}$
$\text{OutputName} : S \rightarrow \text{Output } S$	$\text{OutputName}(\text{Compositor}) = \text{CompositorOutput}$

## Data and Transitions

The *state* of the system is a tuple of each component reactive systems state.

$$\prod_{c \in S} \text{StateName}(c)$$

The set of *input events* of the system is the union of each component reactive systems set of input events.

$$\bigcup_{c \in S} \text{InputName}(c)$$

The set of *output events* of the system is the union of each component reactive systems set of output events.

$$\bigcup_{c \in S} \text{OutputName}(c)$$

The transitions are the transitions of each component where the state is replaced by the state of the system. This is possible because the input and output events are distinct.

# Component specification

- ▶ For the specifications of the Compositor, Version Management and Piece Management see the presentation of Team A and Team B.
- ▶ For the User Management component consult the specification of a similar system.
- ▶ We specify the Automatic Suggestion System, Communication Manager and the Navigation Manager.

# Automatic Suggestion System (ASS)

ASS analyses pieces to find matching between them. Each matches is rated with a value between *maxQuality* (perfect match) and *minQuality* (no match). This value is called quality index.

The state has type (  $\underbrace{\mathbb{R}}_{\text{minQuality}}$  ,  $\underbrace{\mathbb{R}}_{\text{maxQuality}}$  ,  $\underbrace{\mathbb{P}}_{\text{piece data}}$  ).

The input events are: *matchPiece*, *qualityInterval*.

The output events are: *matches*, *range*.

## Transitions for ASS

PRE :  $(\min, \max, P)$   
IN : *matchPiece* Piece  
OUT: *matches* Seq ( $\mathbb{P}, \mathbb{R}$ )  
POST:  $(\min, \max, P)$

PRE :  $(\min, \max, P)$   
IN : *qualityInterval*  
OUT: *range*  $(\min, \max)$   
POST:  $(\min, \max, P)$

# Communication Manager

The Communication Manager ensures communication between users.

The state has type  $(\overbrace{\mathbb{U}}^{\text{user contact data}}, \overbrace{\mathbb{F}}^{\text{forum data}})$ , where  $\mathbb{F} = \{(\text{Topic}, \text{Message})\}$ .

The input events are: *sendIM*, *postMessage*, *createTopic*, *removeTopic*.

This output events are: *send*.

# Transitions for Communication Manager

PRE :  $(U, F) \wedge \text{User} \in U$   
IN : *sendIM* User Message  
OUT: *send addressOf*(User,  $U$ ) Message  
POST:  $(U, F)$

PRE :  $(U, F)$   
IN : *postMessage* Message Topic  
OUT: -  
POST:  $(U, F \cup (\text{Topic}, \text{Message}))$

PRE :  $(U, F) \wedge \exists M . (\text{Topic}, M) \in F$   
IN : *removeTopic* Topic  
OUT: -  
POST:  $(U, F - \{(\text{Topic}, M) \mid (\text{Topic}, M) \in F\})$

# Navigation Manager

The Navigation Manager is responsible for the transitions between the screens of the system.

The screens are those defined by Team C in Slot 11, where the events in each screen can be either commands to other components or navigation commands (i.e. commands that change the screen).

The state is denoted by the current screen the user is in.

The input events are the union of all screen input events.

The output events are the union of all of the commands to other components.

## Transition for Navigation Manager

If a navigation command is received then the current state is modified by changing the screen, and the output is empty.

Else the state remains the same and the input is copied to the output for another component to handle.