A JavaPathfinder Extension to Analyze Human Machine Interactions

Sébastien Combéfis (UCLouvain), Dimitra Giannakopoulou (NASA), Charles Pecheur (UCLouvain), Peter Mehlitz (NASA)
HMI issues

- automation surprises
  - non-determinism, mode confusion

- system abstractions for human operators
  - user / pilot training, procedure generation, test-case generation

- jpf-hmi
  - supports the definition of hmi systems
  - provides a number of hmi-specific analysis capabilities
HMI LTS of a countdown system

idle

running
val=4

tick

running
val=3

tick

running
val=2

tick

running
val=1

tick

running
val=0

---

command

tau

unobservable

observation

mode1

mode2
public class CountDown extends Model
{
    @Override
    public List<Action> getActions() {
        List<Action> actions = new ArrayList<Action>();
        actions.addAll(Arrays.asList(
            new Action("start", COMMAND),
            new Action("stop", COMMAND),
            new Action("tick", OBSERVATION)
        ));
        return actions;
    }
}

public static class Behaviour extends State {
    private static final int MAX = 4;

    public class Idle extends State {
        public void start() ...
    }

    public class Running extends State {
        int val = 0;
        public void stop() ...
        public void tick() ...
    }
}
abstraction $M_U$ allows full control of system $M_M$ if at any time, when using the system according to $M_U$:

1. the set of available commands is exactly the same for the two models
2. abstraction allows at least all the observations that can be produced by the system
HMI analyses during generation

System model is not full control deterministic:

CEX: [start, tick, tick, tick, tick]

FC determinism

Mode confusion

Modes are self-loop transitions treated like commands. If CEX ends in mode action, then it represents mode confusion.
where would we be without abstraction?

- @FilterField

- public static class ValAbs1 extends AbstractionAdapter {
  public int getAbstractValue (int v) {
    if (v > 0) {
      return 0;
    } else if (v == 0) {
      return 1;
    }
    return -1;
  }

  public String getName (int v) {
    int i = getAbstractValue (v);
    return i == 0 ? "(>0)" : "(=0)";
  }
}

conclusions & extensions

more input sources / analyses / scalability, more users...
system vs mental models

- **system model** describes complete behavior of a system
- **mental model** describes user’s view of the system

- user does not need to distinguish states with the same color
- the focus of this work is to generate mental models automatically
what is a good mental model?
  – it should be as compact as possible
  – the user should have enough information to control the system

mental model $M_U$ allows full control of a system $M_M$ if at any time, when using the system according to the mental model:
  – the set of available commands is exactly the same for the two models
  – the mental model allows at least all the observations that can be produced by the system