Model Checking Web Specifications

- Verification of design specifications for Web applications -

JAIST/AIST Workshop Sep. 21 2005

Eun-Hye CHOI Hiroshi WATANABE

Research Center for Verification and Semantics (CVS), AIST

http://unit.aist.go.jp/cvs/

Outline of My Talk

- Background
- Proposed methods
- Experiments
- Conclusion

Background

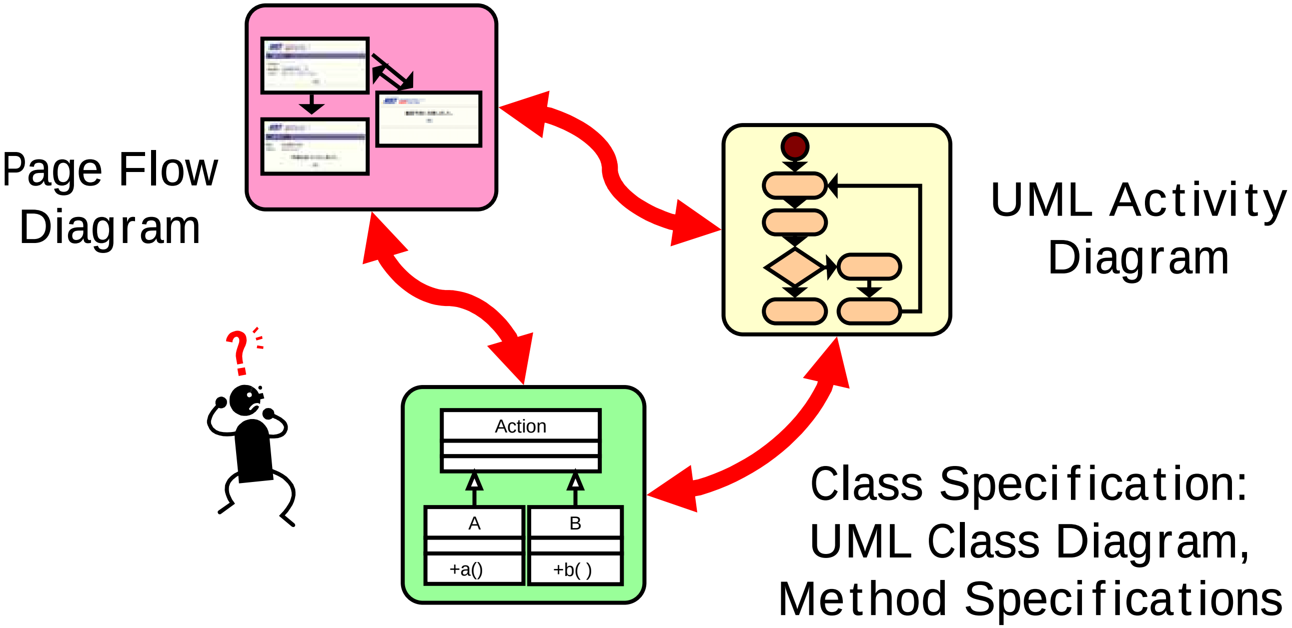
- A certain company, A, provided us a set of design specifications which was used for an actual Web-based business processing application.

- In our fieldwork, we tackled proposing a verification technique for the given design specifications that is easily applicable to existing design process for Web applications.

Design Specifications for a Web Application

Preliminary Design Phase

Detailed Design Phase


Page Flow Diagram
UML Activity Diagram
Action
A
B
+a()
+b( )
Class Specification: UML Class Diagram, Method Specifications

- Consistency checking for design specifications is important in terms of not only reliability but also maintenance and reuse of a Web application.

Our Work

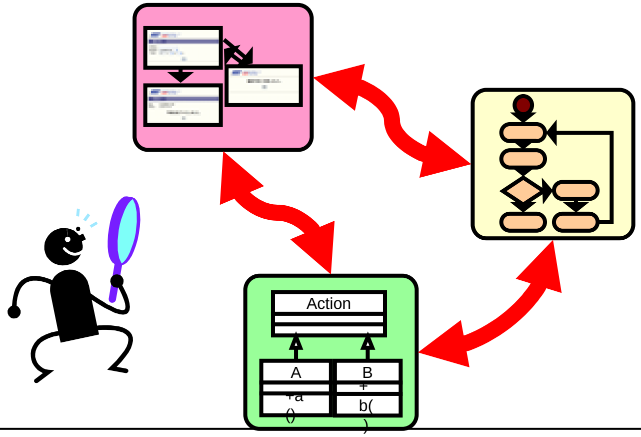
- We proposed verification methods to check

  I. Consistency between a page flow diagram & an activity diagram

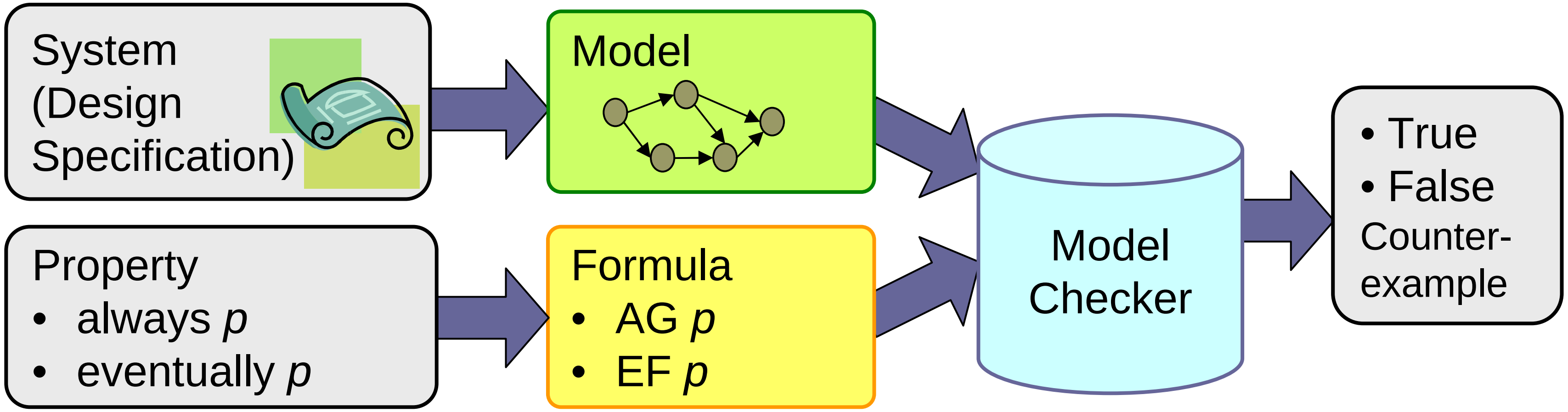
  II. Consistency between a page flow diagram & a class specification

  III. Consistency between a class specification & an activity diagram

- The proposed methods are based on a model checking technique.



Model Checking

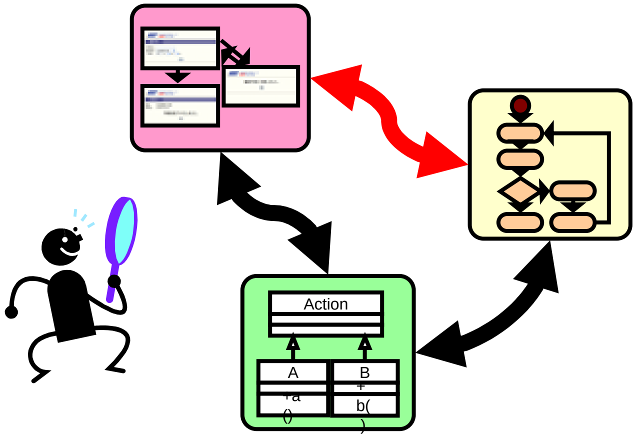


- Model checking is a verification technique that can exhaustively check whether a finite transition system satisfies a temporal logic formula or not.
- Model checking is also helpful to allocate errors because, when the system does not satisfy the property, counterexample is output with the result.

Outline of My Talk

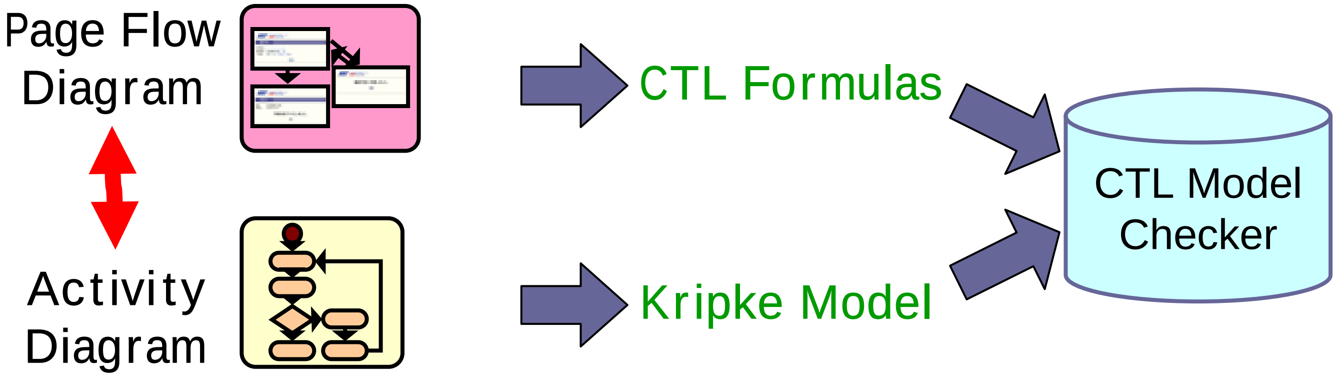
- Background
- Proposed methods to check
  1. Consistency between a page flow diagram & an activity diagram
  2. Consistency between a page flow diagram & a class specification
  3. Consistency between a class specification & an activity diagram

- Experiments
- Conclusion



Outline of the proposed method I

- Proposed method to verify the consistency between a page flow diagram and an activity diagram
  1. Define the consistency between a page flow diagram and an activity diagram.
  2. Represent the consistency using CTL formulas, which are generated from the page flow diagram.
  3. Model check if the CTL formulas hold for a Kripke model constructed from the activity diagram.



Page Flow Diagram

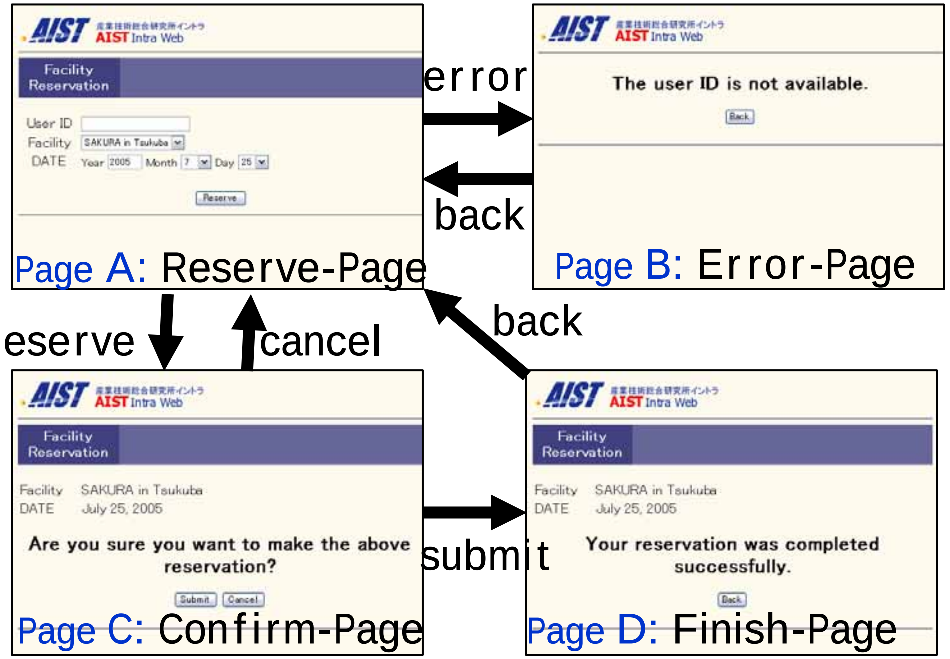
Activity Diagram

CTL Formulas

Kripke Model

CTL Model Checker

Example Application and Specifications



AIST 産業技術総合研究所イントラ AIST Intra Web

Facility Reservation

User ID

Facility SAKURA in Tsukuba

DATE Year 2005 Month 7 Day 25

Reserve

Page A: Reserve-Page

AIST 産業技術総合研究所イントラ AIST Intra Web

The user ID is not available.

Back

Page B: Error-Page

AIST 産業技術総合研究所イントラ AIST Intra Web

Facility Reservation

Facility SAKURA in Tsukuba

DATE July 25, 2005

Are you sure you want to make the above reservation?

Submit Cancel

Page C: Confirm-Page

AIST 産業技術総合研究所イントラ AIST Intra Web

Facility Reservation

Facility SAKURA in Tsukuba

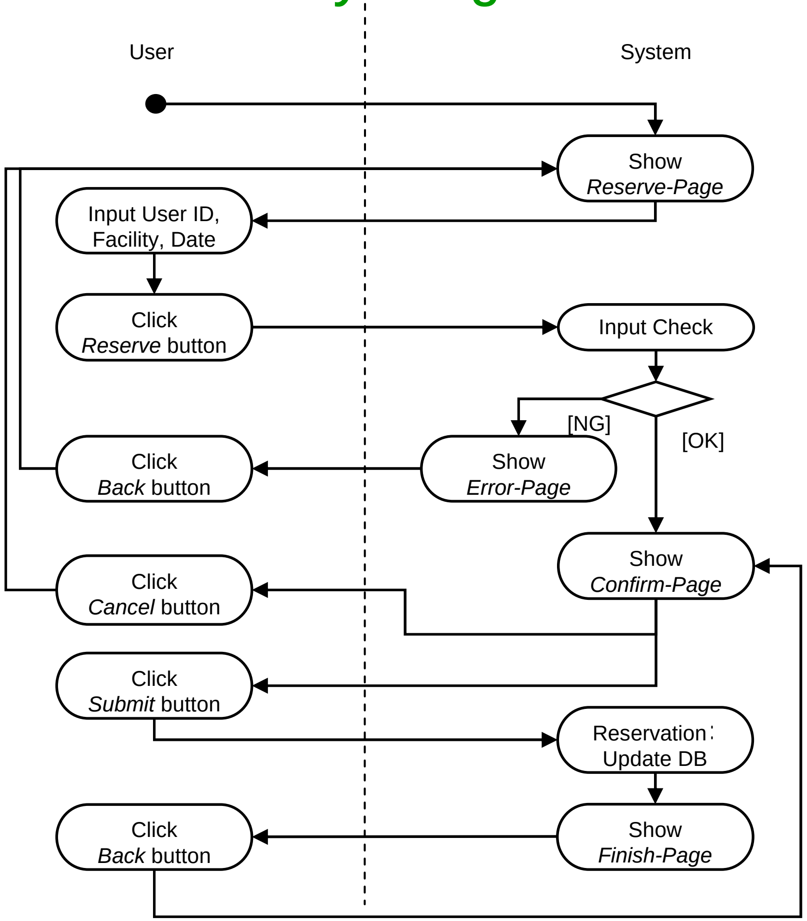
DATE July 25, 2005

Your reservation was completed successfully.

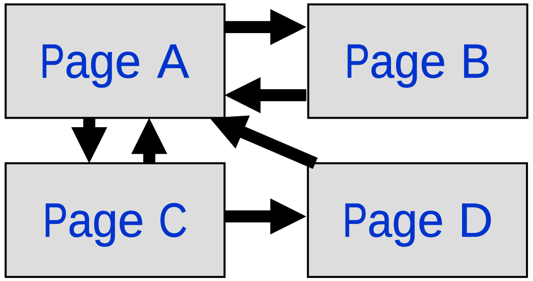
Back

Page D: Finish-Page

Activity Diagram

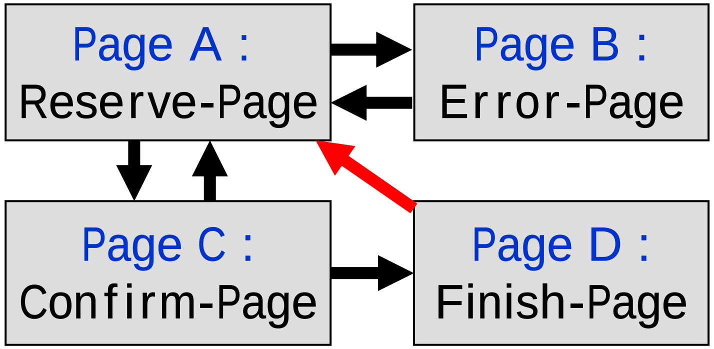


Page Flow Diagram

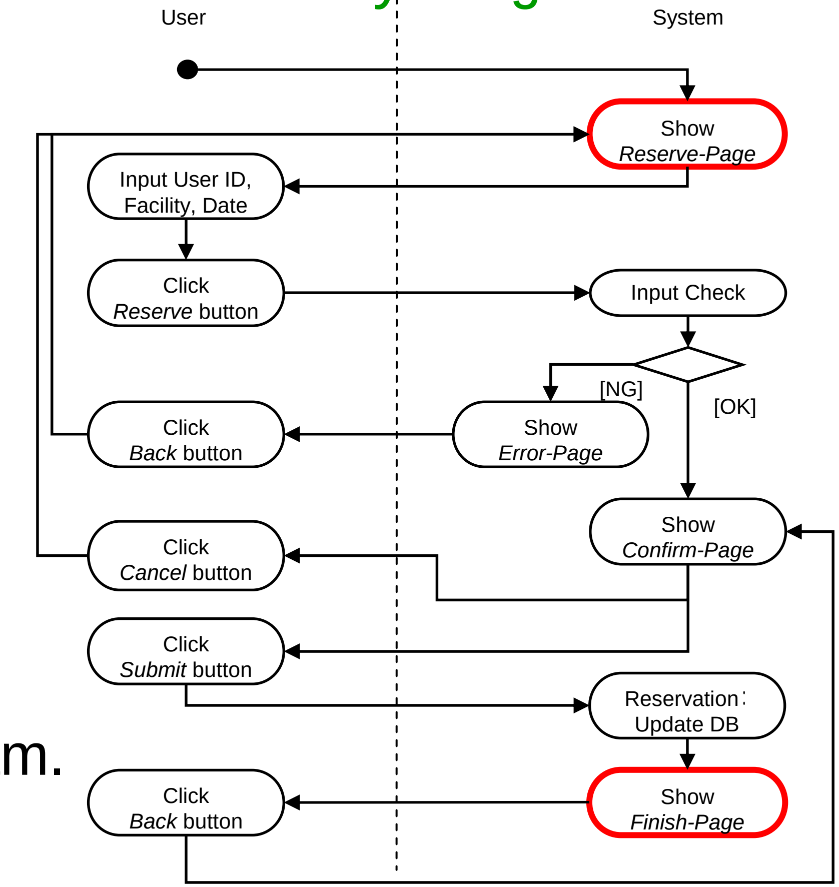


Inconsistency 1

Page Flow Diagram



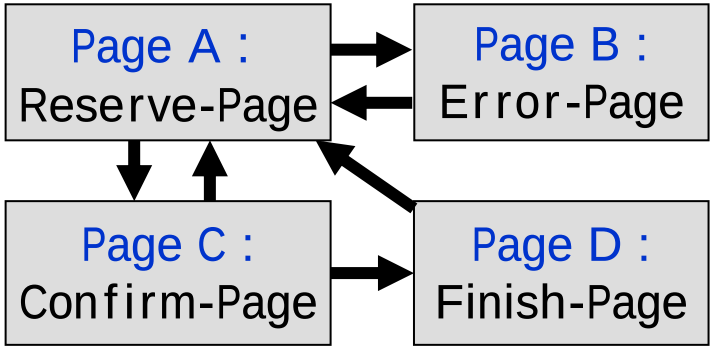
Activity Diagram



A page transition (page D, page A) in the page flow diagram does not occur in the activity diagram.

Inconsistency 2

Page Flow Diagram



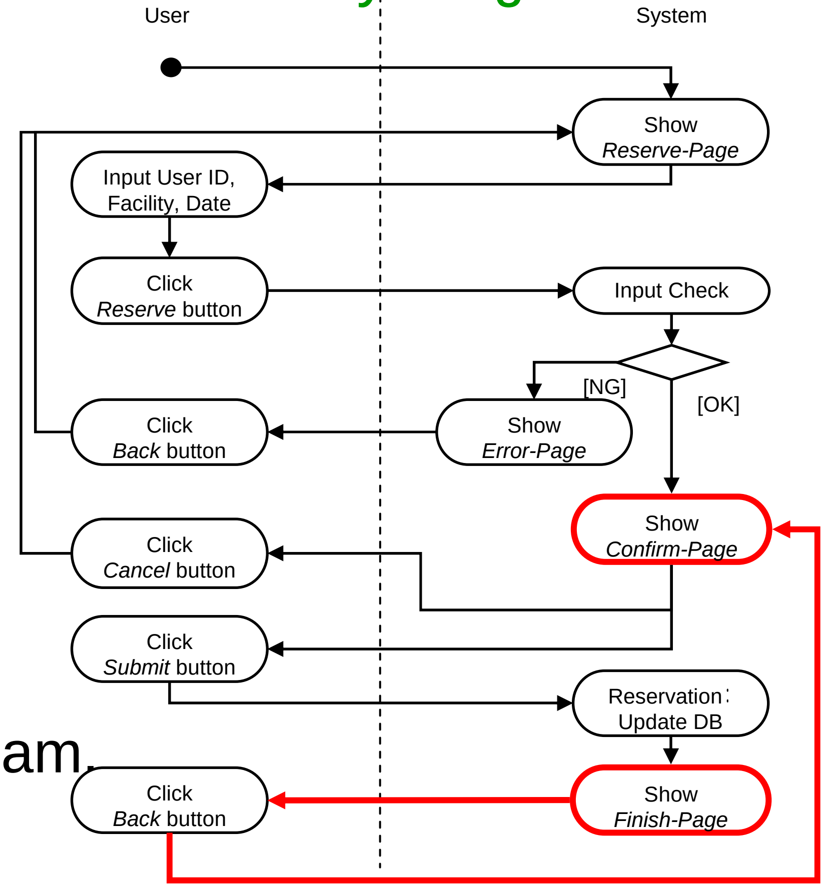
Page A : Reserve-Page

Page B : Error-Page

Page C : Confirm-Page

Page D : Finish-Page

Activity Diagram



User

System

Show *Reserve-Page*

Input User ID, Facility, Date

Click *Reserve* button

Input Check

[NG]

[OK]

Click *Back* button

Show *Error-Page*

Show *Confirm-Page*

Click *Cancel* button

Click *Submit* button

Reservation : Update DB

Click *Back* button

Show *Finish-Page*

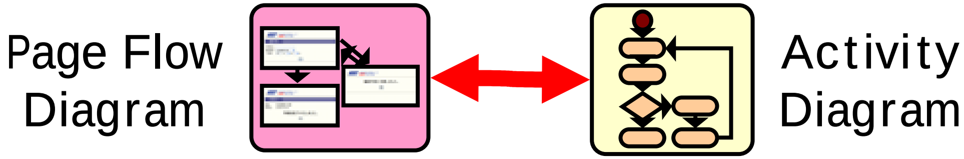
A page transition (page D, page C) in the activity diagram does not exist in the page flow diagram.

Consistency of Page Flow Diagram and Activity Diagram

- We employ the following two conditions for a definition of the consistency between a page flow diagram and an activity diagram:

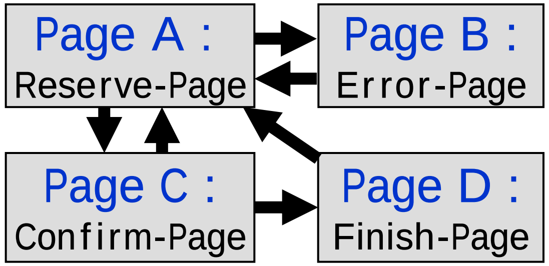
C1 : For each page transition in the page flow diagram, a transition corresponding to the page transition exists in the activity diagram.

C2 : Every transition in the activity diagram corresponds to a stuttering or a page transition in the page flow diagram.



Definition of Consistency

Page Flow Diagram: $(V,E(\subseteq V\times V))$



Page A : Reserve-Page

Page B : Error-Page

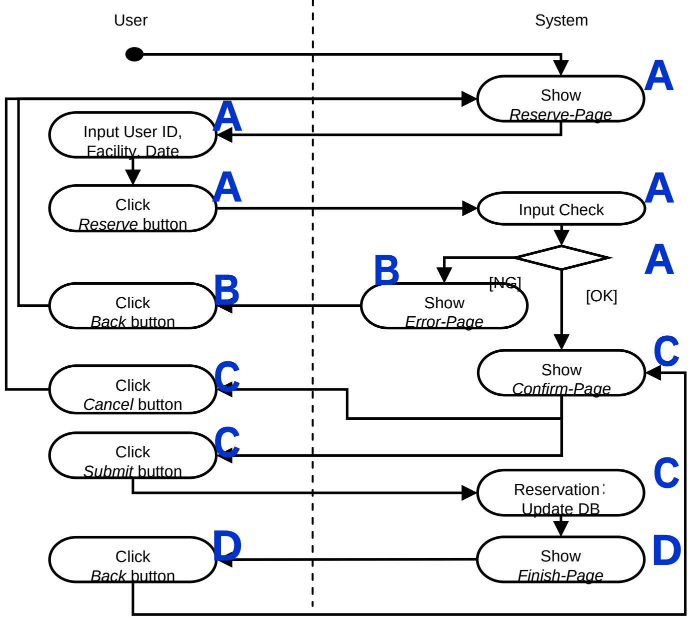
Page C : Confirm-Page

Page D : Finish-Page

$V=\{A,B,C,D\}$

Activity Diagram: $(S,T(\subseteq S\times S))$

- view:S→V



User

System

Show *Reserve-Page* A

Input User ID, Facility, Date A

Click *Reserve* button A

Input Check A

A

[NG]

[OK]

Show *Error-Page* B

Click *Back* button B

Show *Confirm-Page* C

Click *Cancel* button C

Click *Submit* button C

Reservation : Update DB C

Show *Finish-Page* D

Click *Back* button D

Def. Consistency between

(V,E) and $(S,T,view)$ holds iff

C1 : For each $(x,x')\in E$, there exists $(s,s')\in T$ such that $view(s)=x$ and $view(s')=x'$.

C2 : For each $(s,s')\in T$, $view(s)=view(s')$ or $(view(s),view(s'))\in E$.

Consistency Checking Problem

- Consider a page flow diagram (V,E) and a Kripke structure $K = (S,T, \lambda s.\{view(s)\}:S \rightarrow 2^V)$ where (S,T) denotes an activity diagram. Let s_0 denote the initial state of K.

- The two conditions for the consistency are represented using CTL (Computation Tree Logic) as follows:

  C1 : $\forall(x,x')\in E,\ \ (K, s_0) \models \mathrm{EF}\ (x \wedge \mathrm{EX}\ x')$

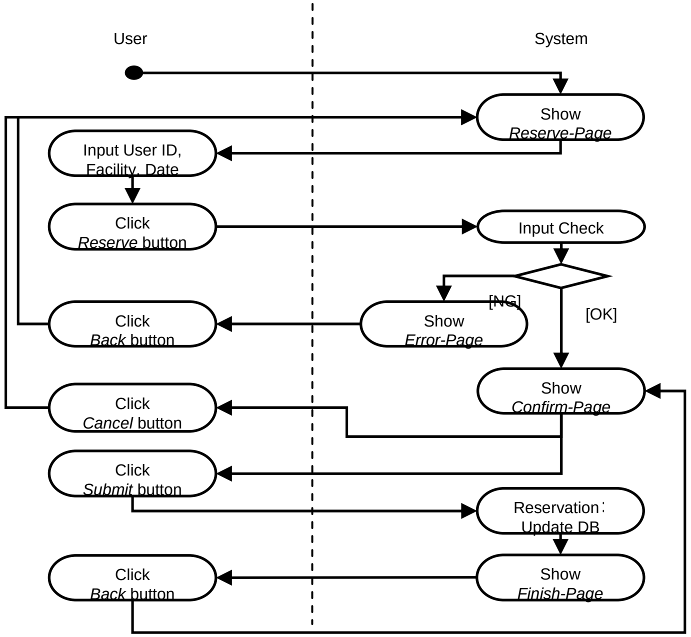
  C2 : $\forall x\in V,\ (K, s_0) \models \mathrm{AG}\ (x \rightarrow \mathrm{AX}\ (\bigvee_{(x,x')\in E} x' \ \vee\ x))$

- The consistency between a page flow diagram and an activity diagram is verified by model checking the above CTL formulas for Kripke structure K.

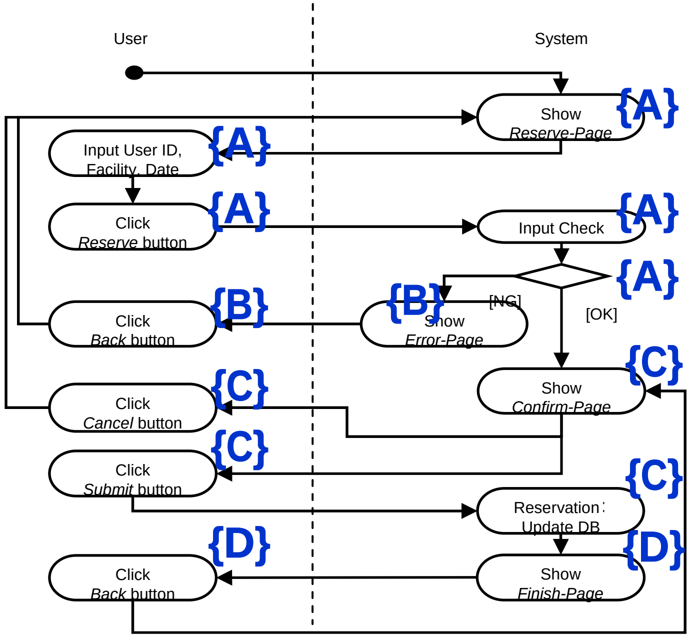
Model input to Model Checker

- Input model is constructed from an activity diagram.

Activity Diagram: (S,T)



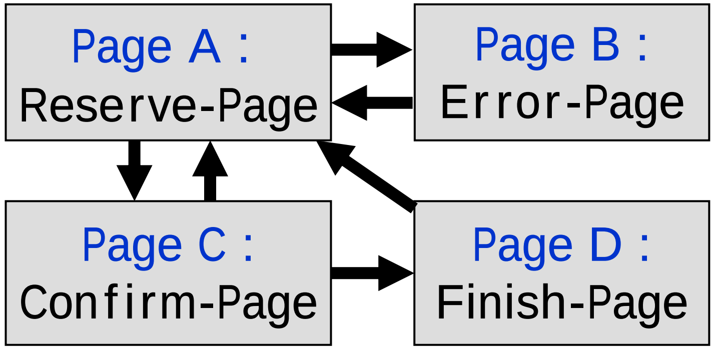
Kripke Model K: (S,T,λs.{view(s)})



Formulas input to Model Checker

- Input formulas are generated from a page flow diagram.

Page Flow Diagram

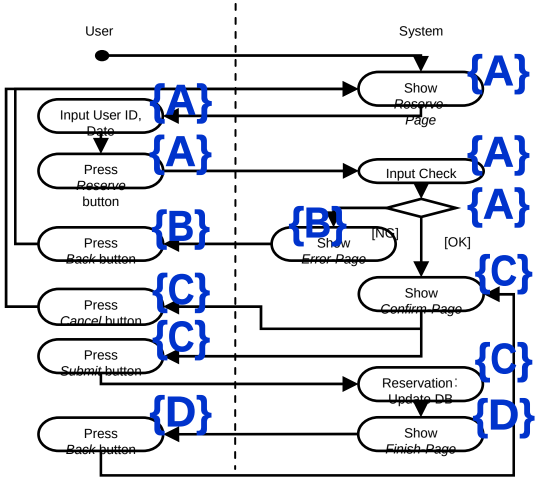


CTL Formulas

- *C1*
  1. $EF\ (A \land EX\ B)$
  2. $EF\ (A \land EX\ C)$
  3. $EF\ (B \land EX\ A)$
  4. $EF\ (C \land EX\ A)$
  5. $EF\ (C \land EX\ D)$
  6. $EF\ (D \land EX\ A)$
- *C2*
  7. $AG\ (A \rightarrow AX\ (A \lor B \lor C))$
  8. $AG\ (B \rightarrow AX\ (B \lor A))$
  9. $AG\ (C \rightarrow AX\ (C \lor A \lor D))$
  10. $AG\ (D \rightarrow AX\ (D \lor A))$

Model Checking

Kripke Model K:
(S,T,λs.{view(s)})



CTL Formulas

Result

- *Cond1*

1. $\mathrm{EF}\ (A \wedge \mathrm{EX}\ B)$ --------- True
2. $\mathrm{EF}\ (A \wedge \mathrm{EX}\ C)$ --------- True
3. $\mathrm{EF}\ (B \wedge \mathrm{EX}\ A)$ --------- True
4. $\mathrm{EF}\ (C \wedge \mathrm{EX}\ A)$ --------- True
5. $\mathrm{EF}\ (C \wedge \mathrm{EX}\ D)$ --------- True
6. $\mathrm{EF}\ (D \wedge \mathrm{EX}\ A)$ --------- False

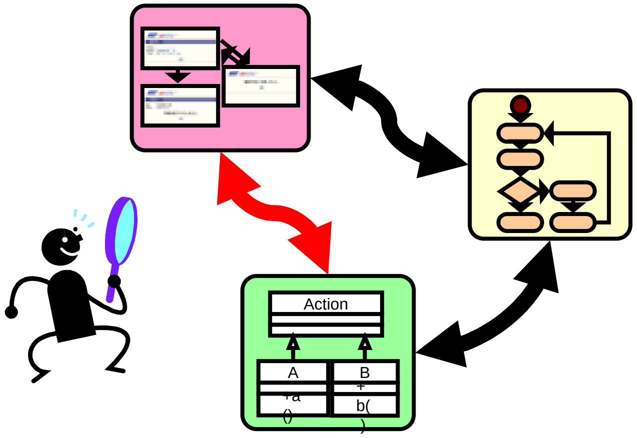
- *Cond2*

7. $\mathrm{AG}\ (A \rightarrow \mathrm{AX}\ (A \vee B \vee C))$ ----- True
8. $\mathrm{AG}\ (B \rightarrow \mathrm{AX}\ (B \vee A))$ ----- True
9. $\mathrm{AG}\ (C \rightarrow \mathrm{AX}\ (C \vee A \vee D))$ ----- True
10. $\mathrm{AG}\ (D \rightarrow \mathrm{AX}\ (D \vee A))$ ----- False

CTL Model Checker

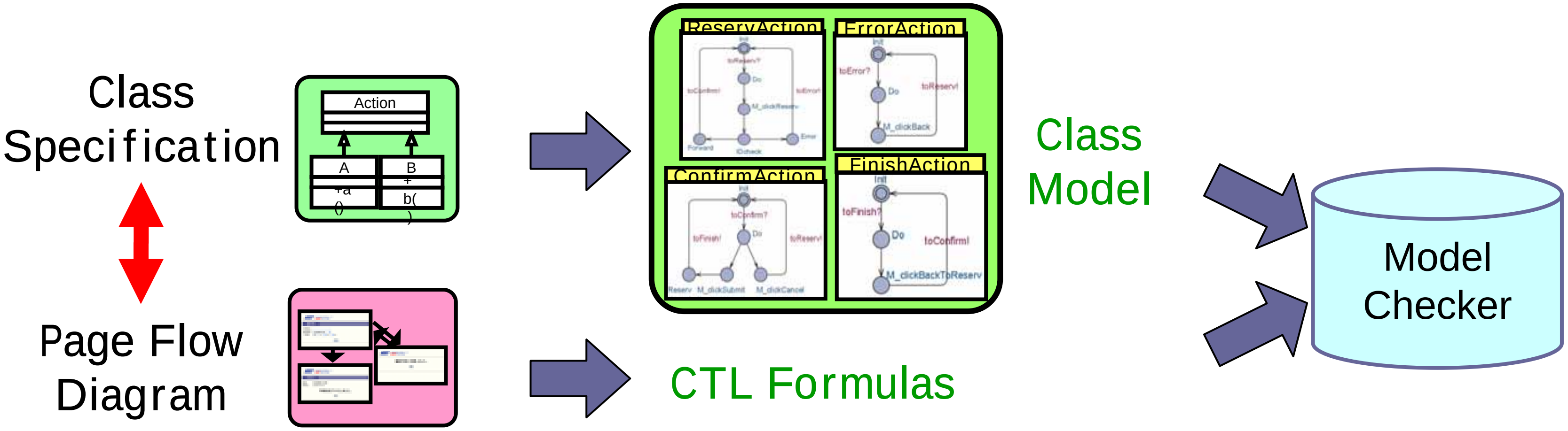
Outline of My Talk

- Background
- Proposed methods to check
  I. Consistency between a page flow diagram & an activity diagram
  II. Consistency between a page flow diagram & a class specification
  III. Consistency between a class specification & an activity diagram
- Experiments
- Conclusion



Outline of the proposed method ＩＩ

- Proposed method to verify the consistency between a page flow diagram and a class specification
  1. From the given class specification consisting of a class diagram and method specifications, we model its behavior by a parallel composition of labeled transition systems.
  2. Apply the proposed method Ｉ to the behavior model of the class specification.



Class Specification

Page Flow Diagram

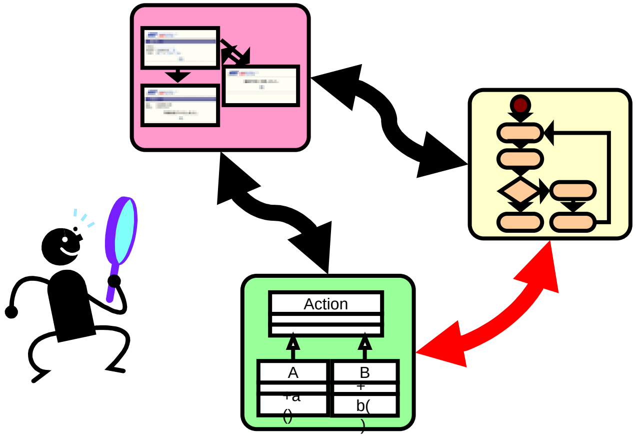
Class Model

CTL Formulas

Model Checker

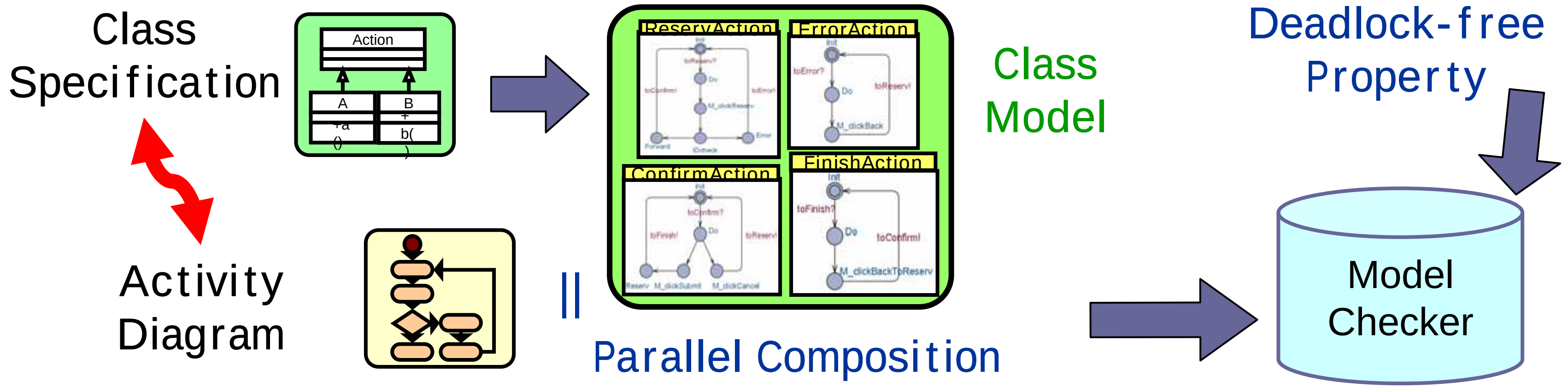
Outline of My Talk

- Background
- Proposed methods to check
  1. Consistency between a page flow diagram & an activity diagram
  2. Consistency between a page flow diagram & a class specification
  3. Consistency between a class specification & an activity diagram
- Experiments
- Conclusion



Outline of the proposed method III

- Proposed method to check the consistency between a class specification and an activity diagram
  1. Compose the class model and the activity diagram.
  2. Model check the deadlock-free property for the composed model.
- If a deadlock occurs in the composed model, there exists an inconsistency between the two specifications.



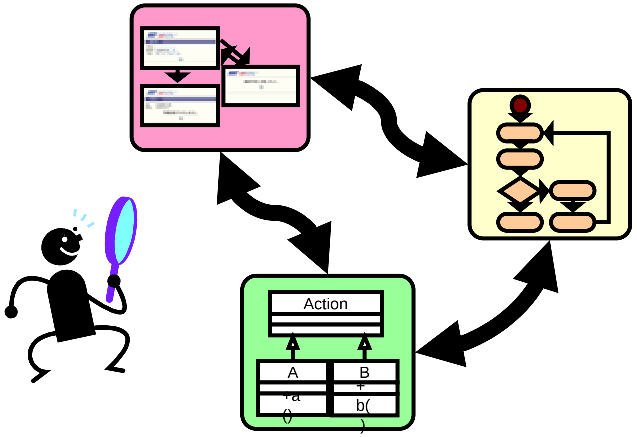
Outline of My Talk

- Background
- Proposed methods to check

  I. Consistency between a page flow diagram & an activity diagram

  II. Consistency between a page flow diagram & a class specification

  III. Consistency between a class specification & an activity diagram
- Experiments
- Conclusion



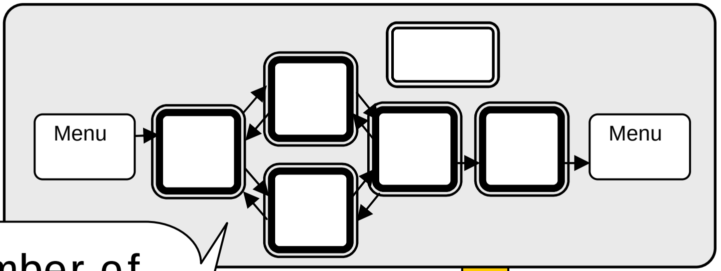
Case Study

- We applied the proposed methods to the real specifications of the given Web application.
  - Developed by Java using Jakarta Struts framework.
  - Classified into several tens of modules.

- We chose one module *M* and checked the consistencies for a page flow diagram, an activity diagram, and a class specification for module *M*.

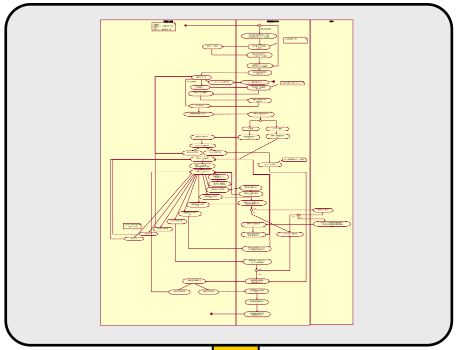
Experiment I

Page Flow Diagram



- Number of pages: 6
- Number of transitions: 9

Activity Diagram



- Number of states: 66
- Number of transitions: 83

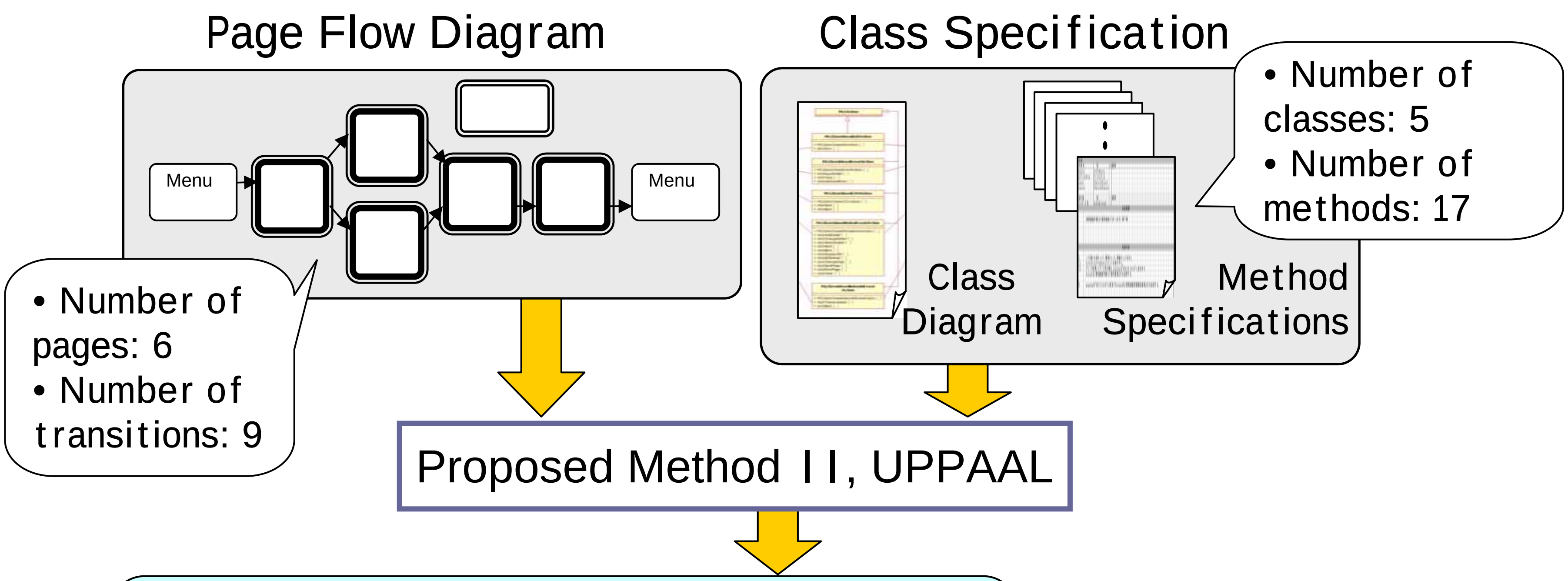
Proposed Method I, NuSMV

Proprietary Secret

X : page transition that does not exist in the activity diagram

→ : page transition that exists only in the activity diagram

Experiment II



- Number of pages: 6
- Number of transitions: 9

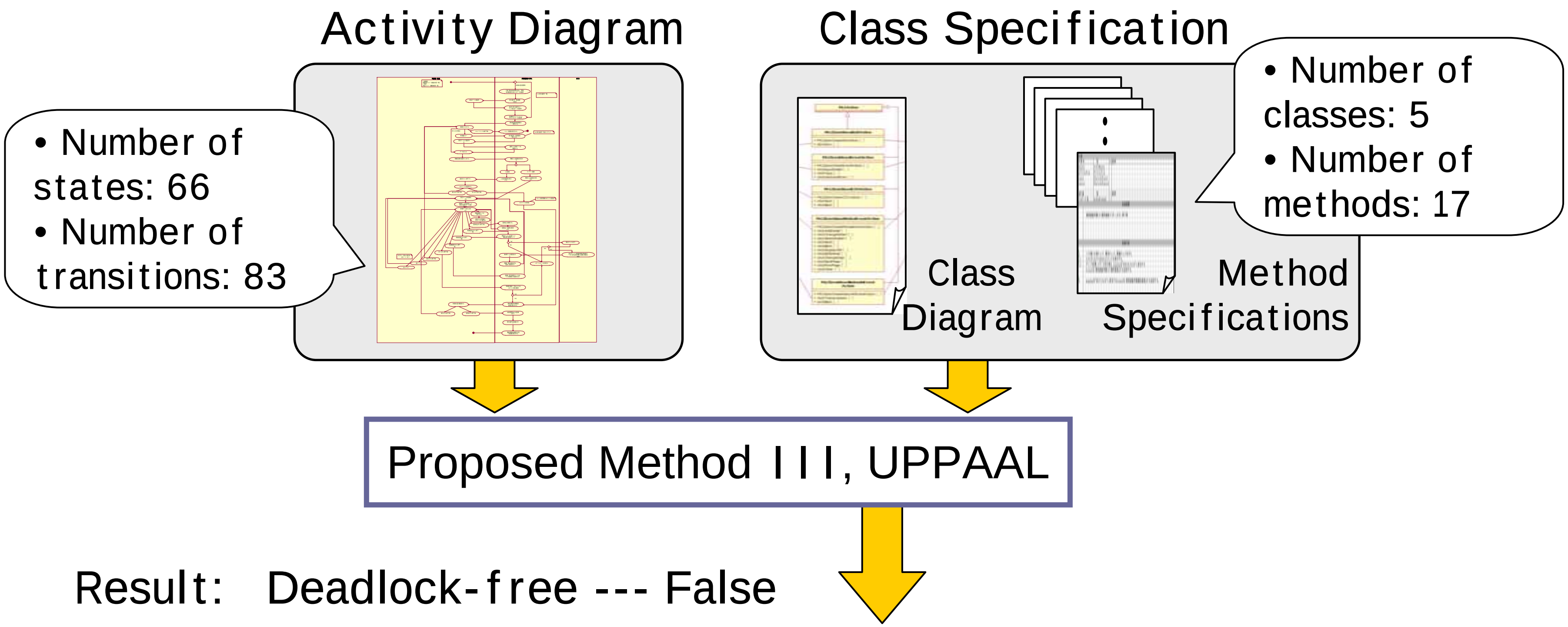
- Number of classes: 5
- Number of methods: 17

Proprietary Secret

X : page transition that does not exist in the class specification

→ : page transition that exists only in the class specification

Experiment III



Activity Diagram

- Number of states: 66
- Number of transitions: 83

Class Specification

- Number of classes: 5
- Number of methods: 17

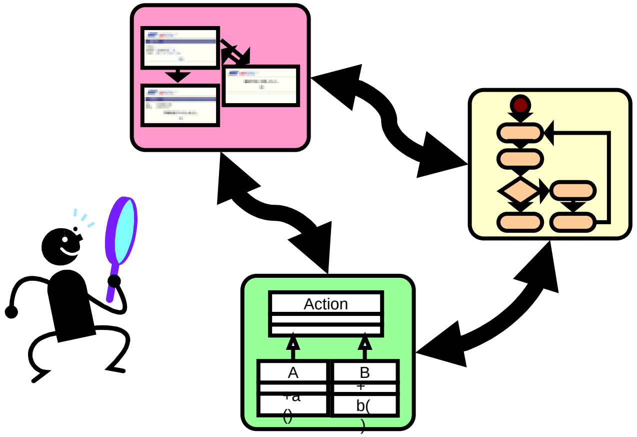
Proposed Method III, UPPAAL

Result: Deadlock-free --- False

Proprietary Secret

Outline of My Talk

- Background
- Proposed methods to check
  1. Consistency between a page flow diagram & an activity diagram
  2. Consistency between a page flow diagram & a class specification
  3. Consistency between a class specification & an activity diagram
- Experiments
- Conclusion



Conclusion

- As a fieldwork, we proposed the methods to verify the consistency between design specifications for Web applications.
- By applying the proposed methods, we found several faults in the real specifications that had not been detected in actual reviews.

- Future work includes a full automation and an evaluation of scalability and efficiency of the proposed methods.