Transformation of Spatio-Temporal Role Based Access Control Speciation to Alloy

MEDI 2012

Emsaieb Geepalla
E.M.E.Geepalla@cs.bham.ac.uk

Behzad Bordbar
B.Bordbar@cs.bham.ac.uk

Joel Last
Jxl745@cs.bham.ac.uk

School of Computer Science
University of Birmingham
UK

Presentation Foot Steps

1. Introduction

- STRBAC
- Alloy
- Motivation

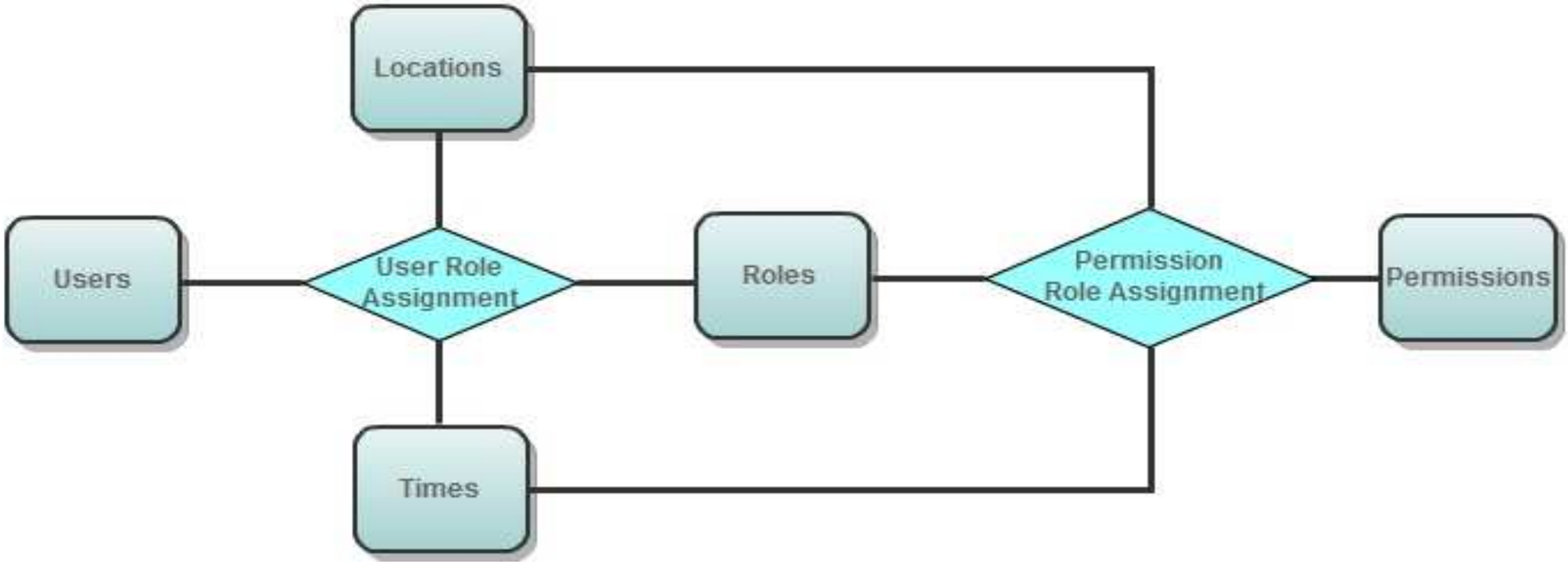
2. Concluded Work

- STRBAC to Alloy Transformation
- AC2Alloy
- Results

3. Conclusions

The Basic Concepts of STRBAC

STRBAC: An extension of the original RBAC model that supports temporal and location constraints.



Software Engineering Group
School of Computer Science
University of Birmingham, UK

Cont…

- Users, Roles, Permissions, Times & Locations

  finite & non-empty sets

- User Role Assignment (URA): Users are assigned to roles based on time and location constraints.

  URA={(u, r, t, l), …….,}

- Permission Role Assignment (PRA): Permissions are assigned to roles based on time and location constraints.

  PRA={(r, p, t, l), …….,}

- Role Hierarchy

  RH={$(r_1 \geq r_2)$, $(r_1 \geq_t r_2)$, $(r_1 \geq_l r_2)$, $(r_1 \geq_{t,l} r_2)$, …….,}

- Location Hierarchy

  LH={$(l_1 \geq l_2)$, …….,}

Software Engineering Group
School of Computer Science
University of Birmingham, UK

The Basic Concepts of STRBAC

- Separation of Duty between Roles

  SoDR={(r_1, r_2, t, l), …….,}
- Separation of Duty between Permissions

  SoDP={(p_1, p_2, t, l), …….,}
- Cardinality Constraints over Roles

  CCR={(r_1, t, l, N), …….,}
- Cardinality Constraints over Permissions

  CCP={(p_1, t, l, M), …….,}

- Separation of Duty and Cardinality constraints can also be unrestricted, Timed dependent, Location dependent, and Time-Location dependent.

Software Engineering Group
School of Computer Science
University of Birmingham, UK

Inconsistencies in STRBAC Specification

- Due to the complexity and the size of modern system, it is possible that several STRBAC polices conflict with each other

Software Engineering Group
School of Computer Science
University of Birmingham, UK

Alloy

- **Alloy is a formal specification language that has developed by Daniel Jackson [1]**
-
- **Alloy is supported by a fully automated constraint solver, called Alloy Analyser, that analyses system properties**
- **Alloy analyser search for model instances that violate checks about them.**
- **Alloy Analyser translates the model into a Boolean expression, and analyses it using embedded SAT-solvers.**

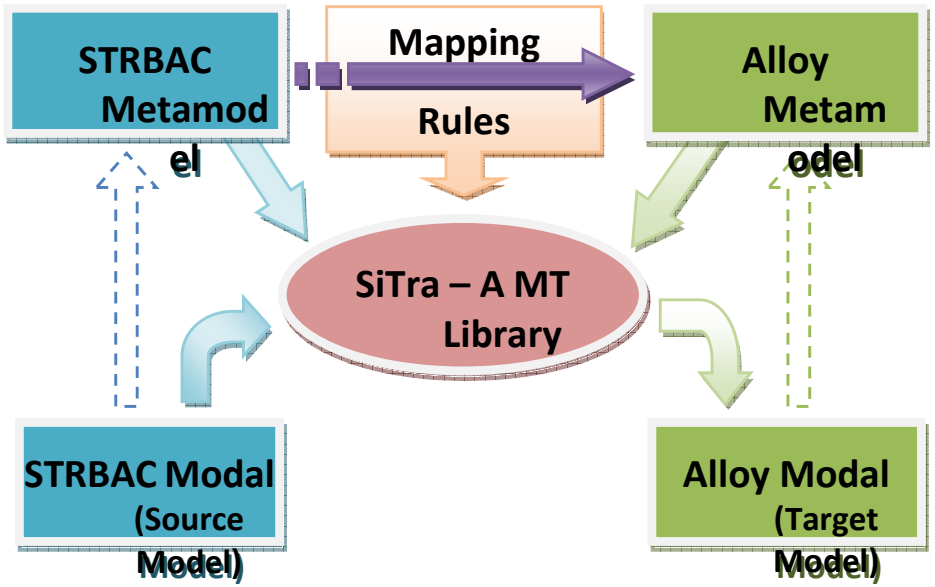
[1] Jackson.Daniel (2006), Software Abstractions Logic, Language, and Analysis, Cambridge: The Mit Press.

Motivation

- The prevention of fraud is more beneficial to organisations than its detection after it had occurred. “Wrong access might result in a huge lose”
- Alloy has been used successfully for automatic verification of Access Control policies.
- Some practically minded software engineers find Alloy hard .
- The Complexity and the size of modern Access control systems.

Idea: provide a tool to assist in writing Alloy to simplify process of Model Analysis

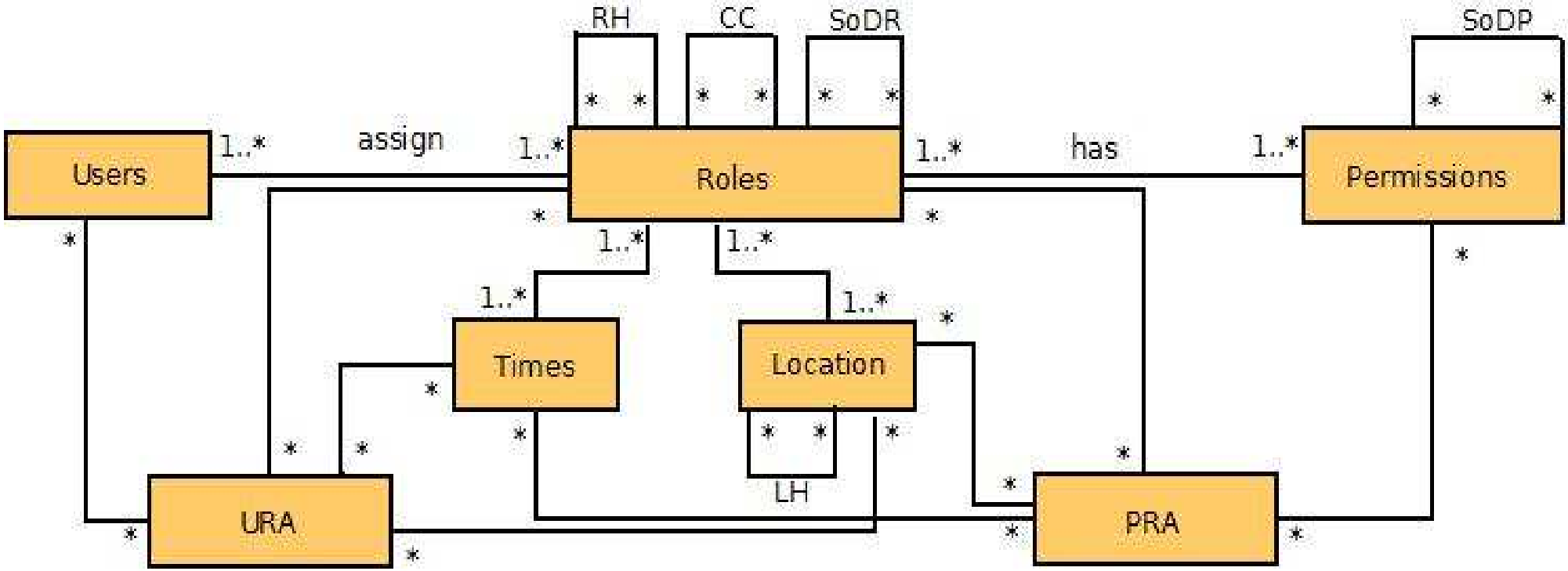
STRBAC to Alloy Transformation



A Model Transformation from STRBAC to Alloy

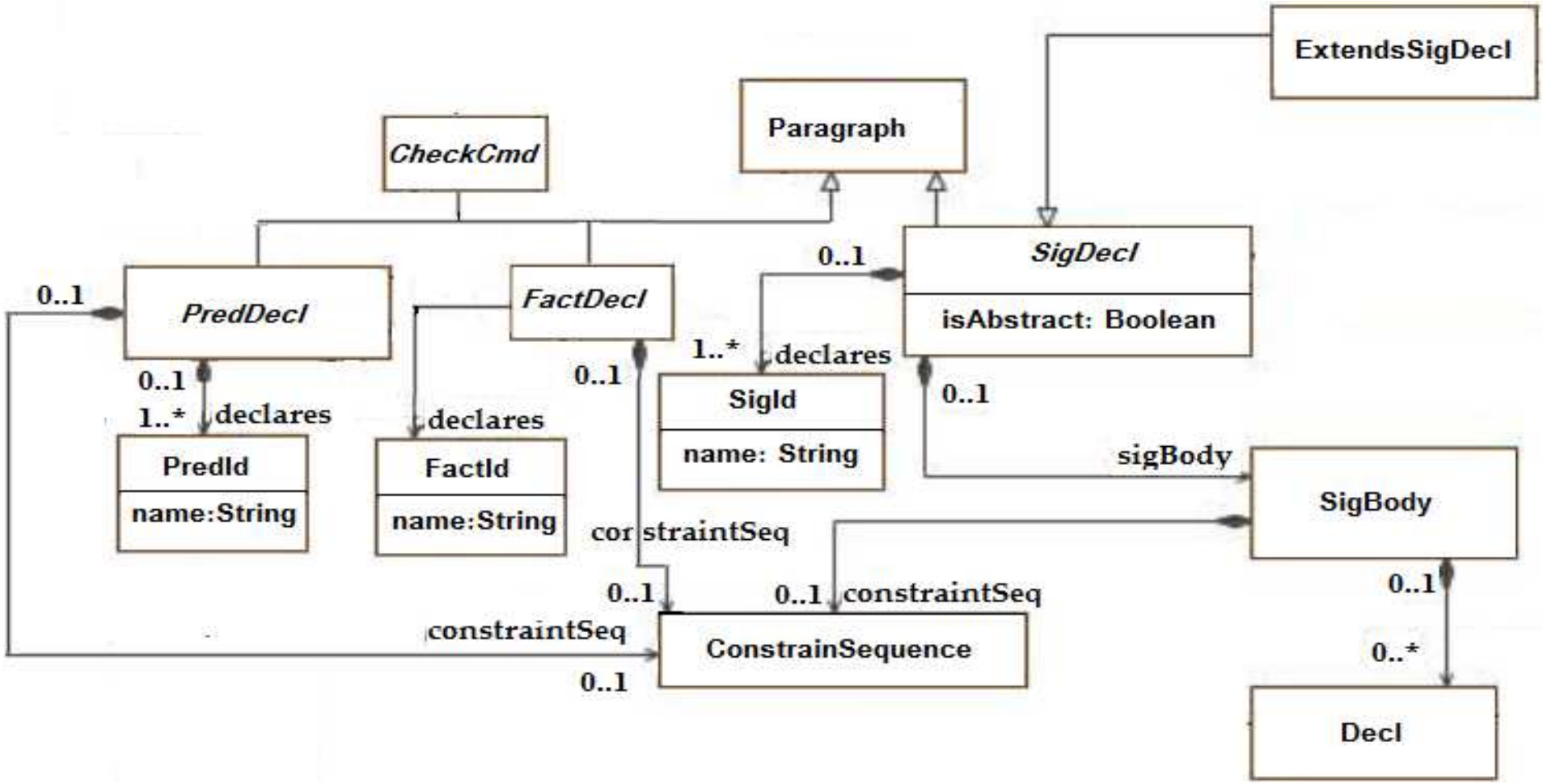
School of Computer Science
University of Birmingham, UK

STRBAC Meta Model (A Fragment)



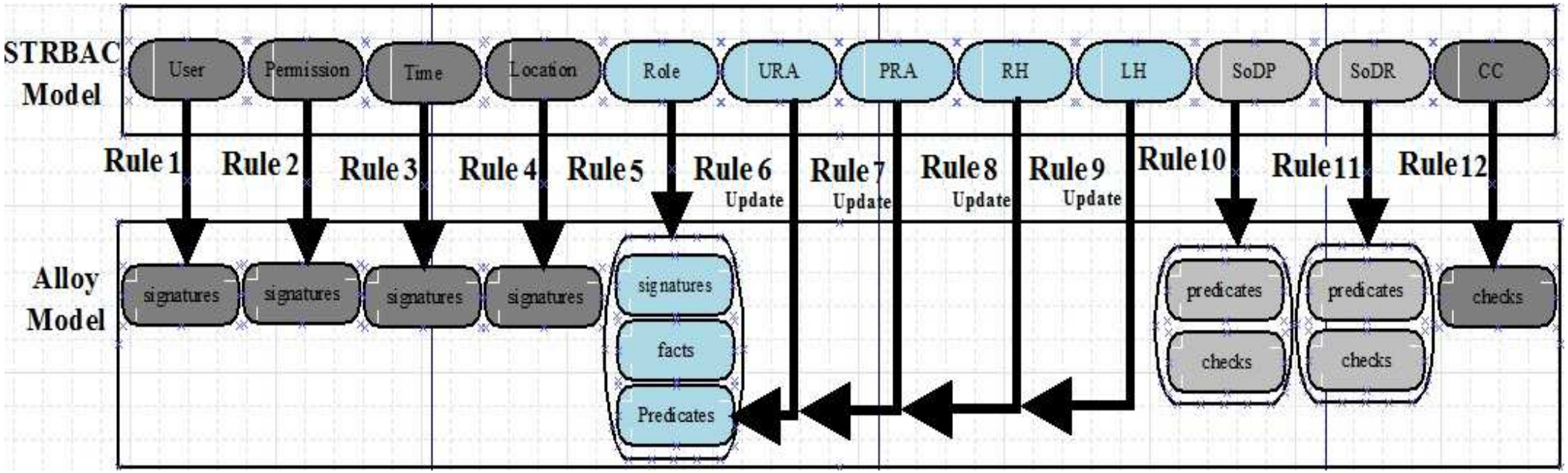
Software Engineering Group
School of Computer Science
University of Birmingham, UK

Alloy Meta Model (A Fragment)



Software Engineering Group
School of Computer Science
University of Birmingham, UK

Transformation Rules



Software Engineering Group
School of Computer Science
University of Birmingham, UK

Example

Consider an Access control model system which consists of the following Users: Dave, Mark, Sarah, the following Locations: office1 and office2, the following Times: DayTime and NightTime, the following Permissions: Read and Write Teller Files, Read and Write Loan Files and Read and Write System Operator Manager Files, and the following Roles: Teller, Loan Officers and System Operator Manager (SOM) .

Rule 1:Transformation of Users

```
abstract sig Users {}
One sig Dave, Mark, Sarah extends Users {}
```

Rule 2:Transformation of Permissions

```
abstract sig Permissions{}
One sig ReadWriteTellerFiles, ReadWriteLoanFiles,
ReadWriteSOMFiles extends Users {}
```

Cont…

Rule 5: Transformation of Roles

```
abstract sig Roles{time: lone Times,
        location: lone Locations,
        permissions: set Permissions,
        users: set Users}
one sig teller, LoanOfficer, SOM extends Users {}
fact SOMFact {all self:SOM | SOMCondition[self]}
pred SOMCondition[self:SOM]{((self.permissions=none)&&
(self.location=none)&&(self.time=none)&&(self.users=none))}
```

The transformation of URA, PRA, RH & LH will update the predicates for every role with new user to role assignment and permission to role assignment.

Rule 6: Transformation of User Role Assignment (URA)

For example, URA={(Mark, SOM, DayTime, Office1)

```
pred SOMCondition[self:SOM]{((self.permissions=none)&&
(self.location=Office1)&&(self.time=DayTime)&&(self.users=Mark))}
```


Software Engineering Group
School of Computer Science
University of Birmingham, UK

Cont…

The other STRBAC elements such as SoDR, SoDP, CCR, & CCP will be transformed to Alloy checks.

Rule 10: Transformation of Separation of Duty over Permissions (SoDP)

For example, SoDP={(Read and Write Teller Files, Read and Write Loan Files, DayTime, Office1)

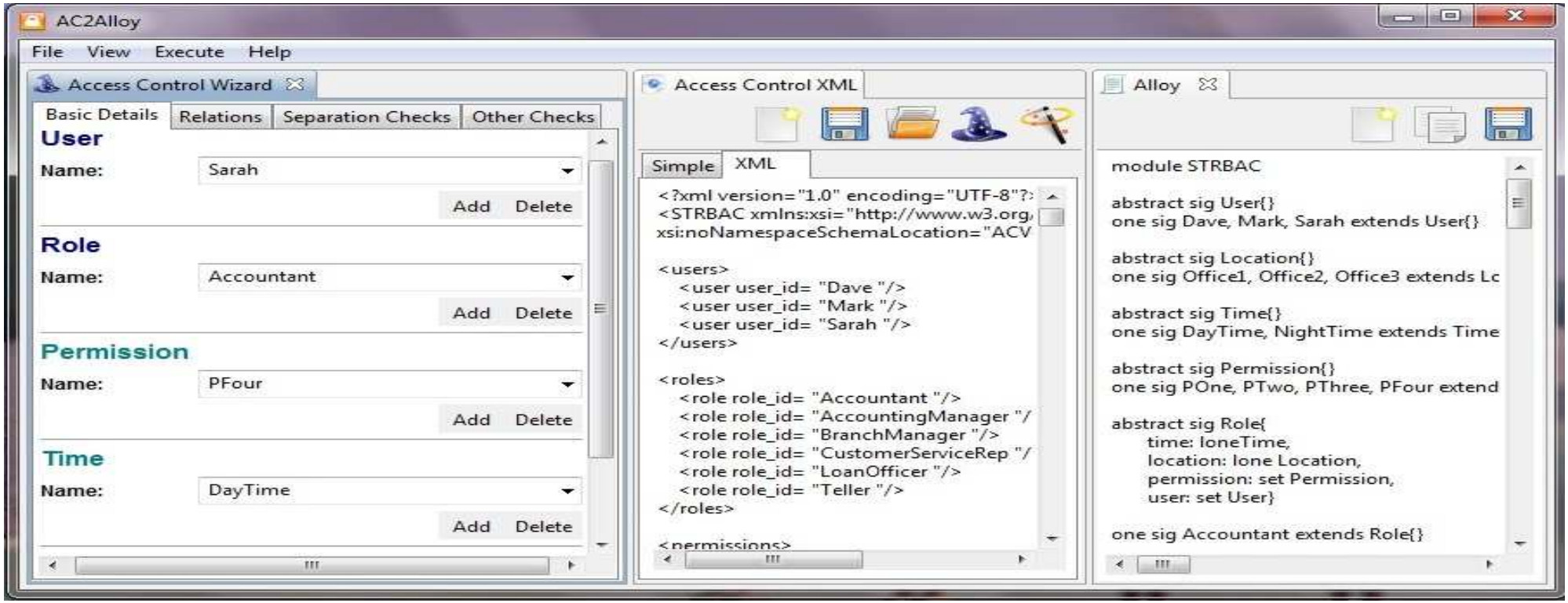
```
pred SODP[p1, p2:Permission, l:Location, t:Time] {all r1:Role |
((p1 in r1.permissions)&&(t in r1.time)&&(l in r1.location)))
((p2 not in r1.permissions)&&(t in r1.time)&&(l in r1.location))}

SoDP1 :check { SODP[ ReadandWriteTellerFiles, ReadandWriteLoanFiles,
DayTime, Office1]}
```


Software Engineering Group
School of Computer Science
University of Birmingham, UK

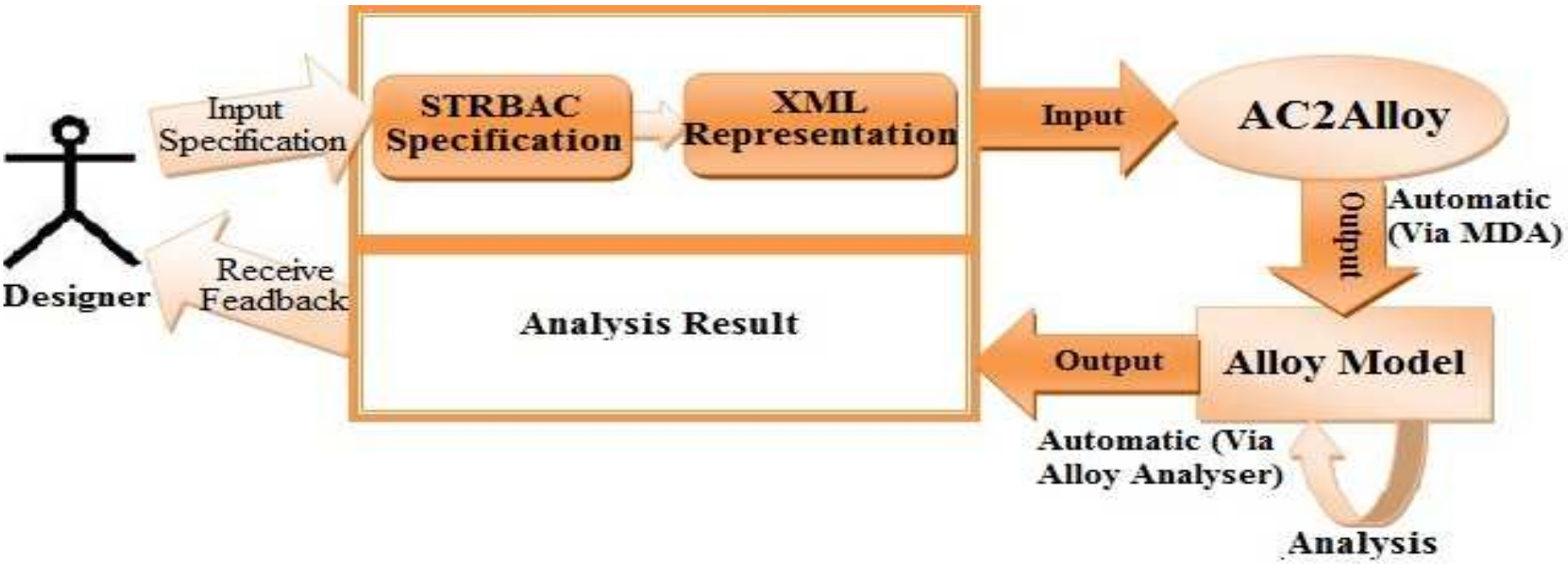
AC2Alloy

- AC2Alloy is a tool for integrating STRBAC and Alloy into a single tool.
- It enables the user to create STRBAC specification and conduct the analysis via Alloy.



Software Engineering Group
School of Computer Science
University of Birmingham, UK

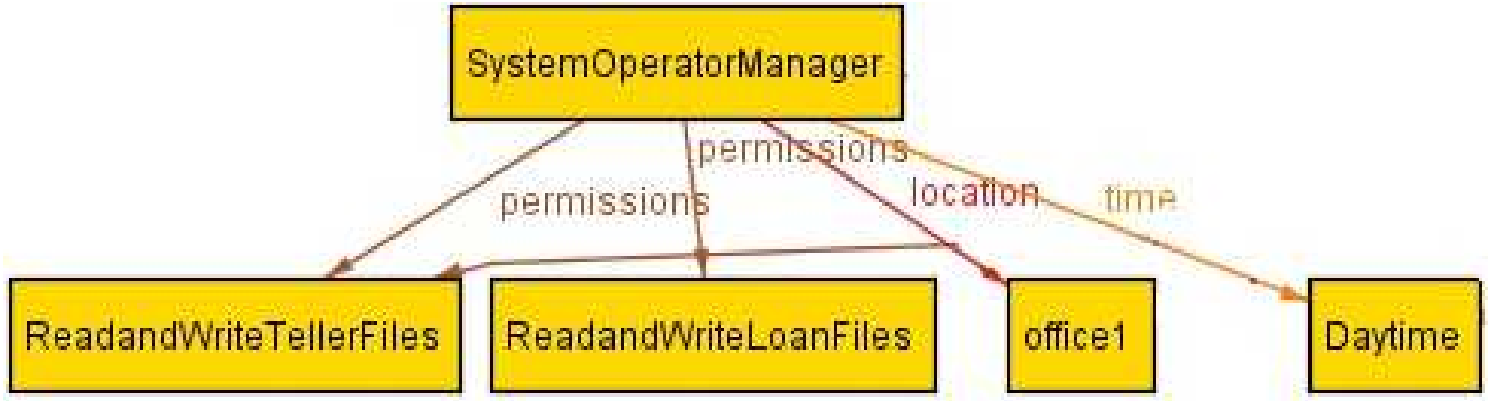
How AC2Alloy works?



Software Engineering Group
School of Computer Science
University of Birmingham, UK

Results

- **Our method has been tested using a simplified SECURE Bank System**
  - **Transform the SECURE Bank Specification successfully and very quickly to Alloy code**
  - **Inconsistencies in the SECURE Bank Specification have been detected**



Software Engineering Group
School of Computer Science
University of Birmingham, UK

Conclusion

- The proposed approach also ensures the creation of consistent Alloy codes.
- AC2Alloy softens the process of Alloy creation
- STRBAC to Alloy Transformation can provide a simple means of analysis and verification of the given STRBAC specification.

School of Computer Science
University of Birmingham, UK

Thank You!

Software Engineering Group
School of Computer Science
University of Birmingham, UK