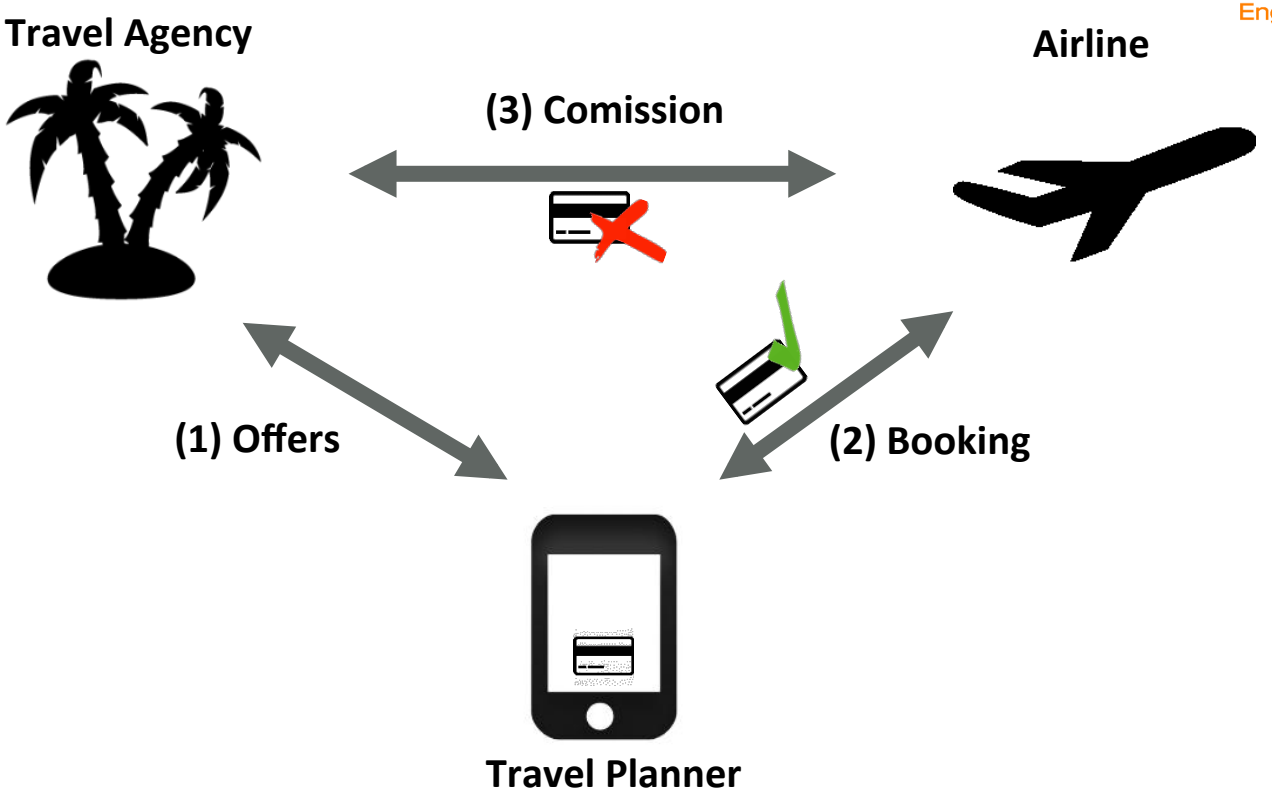
IFlow – Model driven development of information flow secure systems

Wolfgang Reif, Kuzman Katkalov, Marian Borek, Kurt Stenzel

Where does your information go?

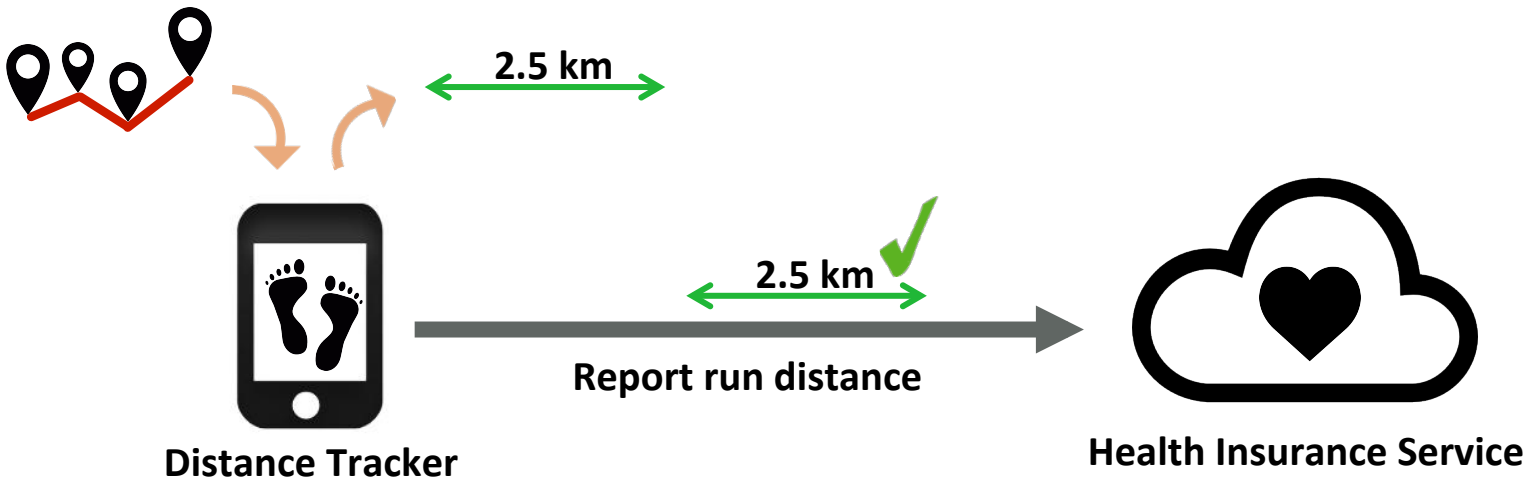
Institute for
Software & Systems
Engineering



- **Travel Agency** never learns the user's credit card data
- **Airline** only learns the user's credit card data after user confirmation

How much information are you sharing?

Institute for Software & Systems Engineering



- **Health Insurance Service never learns** the user's location
- **Distance Tracker properly anonymizes** the user's location data as distance

Overview

Institute for Software & Systems Engineering

- **IFlow: a model driven approach for developing distributed apps with secure information flow**
- Software/security co-design in distributed apps
  - Modeling
  - Automatic check and verification
- Special aspect: limited release of information
  - Modeling
  - Verification

IFlow: a model-driven approach to IFC

Institute for Software & Systems Engineering

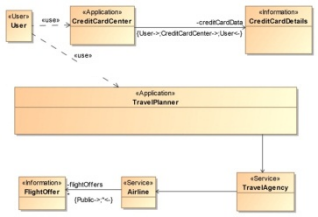
MODEL » FLOW

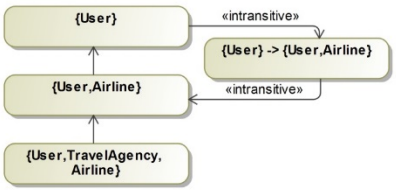
Modeling language

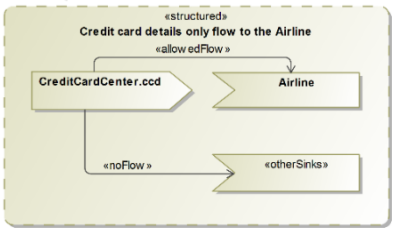
- *principle: security by design*
- *graphical, domain specific language for IFC*
- *distributed apps & services*
- *with tool support*
- *and code generation*

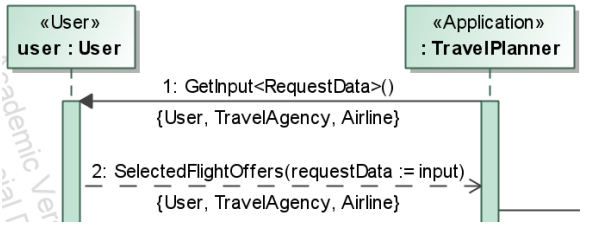
Formal foundation

- *formal semantics (ASMs & predicate logic)*
- *intransitive noninterference (extension of Rushby/Meyden)*
- *declassification ("what", "where", "when")*









IFlow: a model-driven approach to IFC

Institute for Software & Systems Engineering

MODEL FLOW

UML model

automatic transformation

Monolithic code skeleton

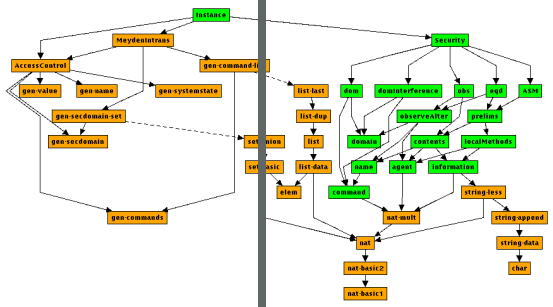
automatic language-based IF check

Final, distributed application

automatic check of manually implemented code

verification of information release

Formal model



Result:
runnable application with guaranteed information flow properties

A hybrid approach to provide guarantees

Institute for Software & Systems Engineering

Information flow:
Automatic code check

Travel Agency ***never learns*** the user's credit card data

Health Insurance Service ***never learns*** the user's location

Information release:
Formal verification

Airline only learns the user's credit card data ***after user confirmation***

Distance Tracker ***properly anonymizes*** the user's location

JOANA

KIV

Overview

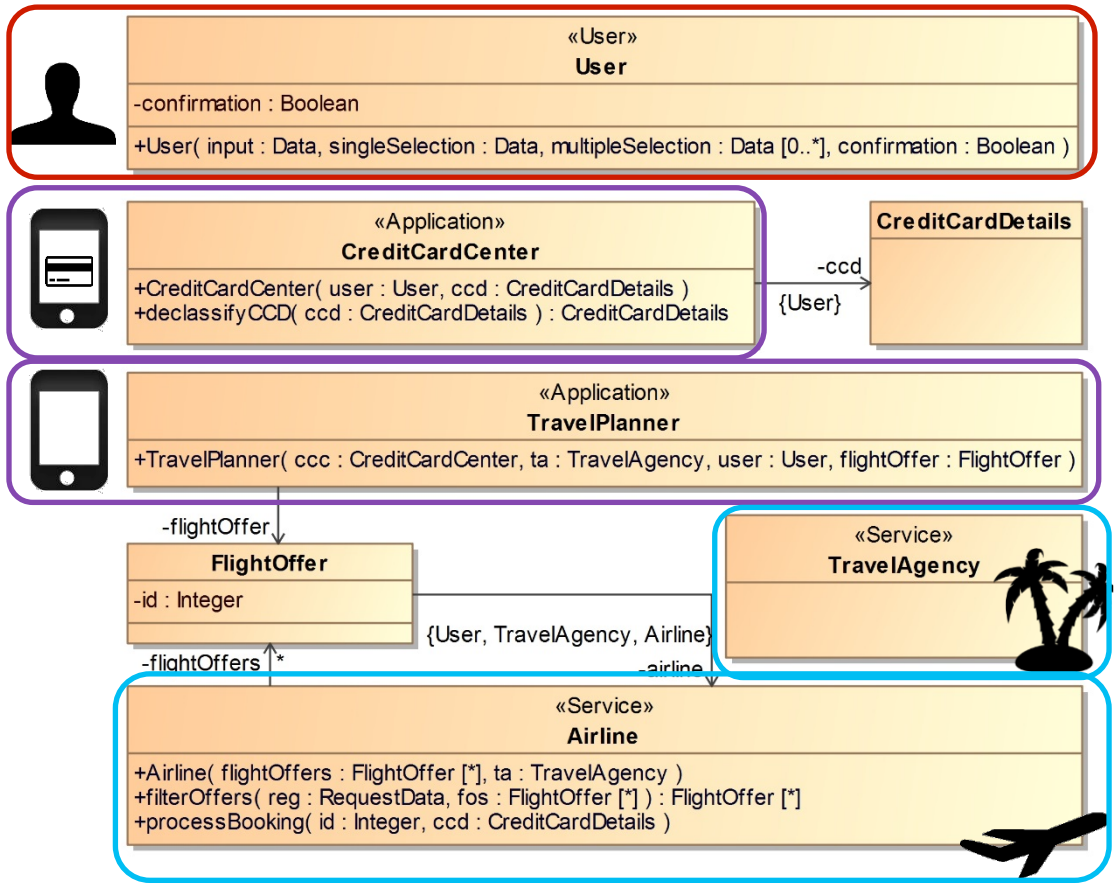
Institute for Software & Systems Engineering

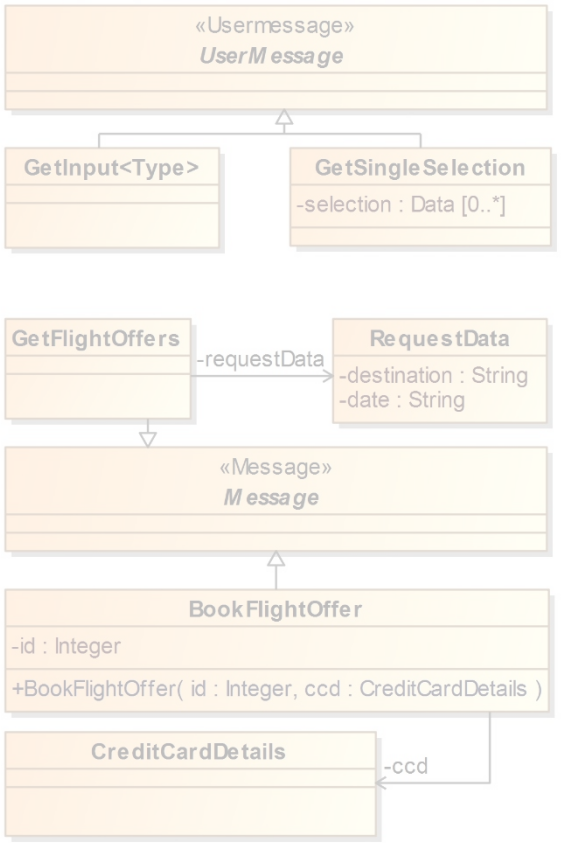
- IFlow: a model driven approach for developing distributed apps with secure information flow
- **Software/security co-design in distributed apps**
  - Modeling
  - Automatic check and verification
- Special aspect: limited release of information
  - Modeling
  - Verification

TravelPlanner: **components**

Institute for Software & Systems Engineering

Application components: **user interface**, **mobile apps** and **web services**

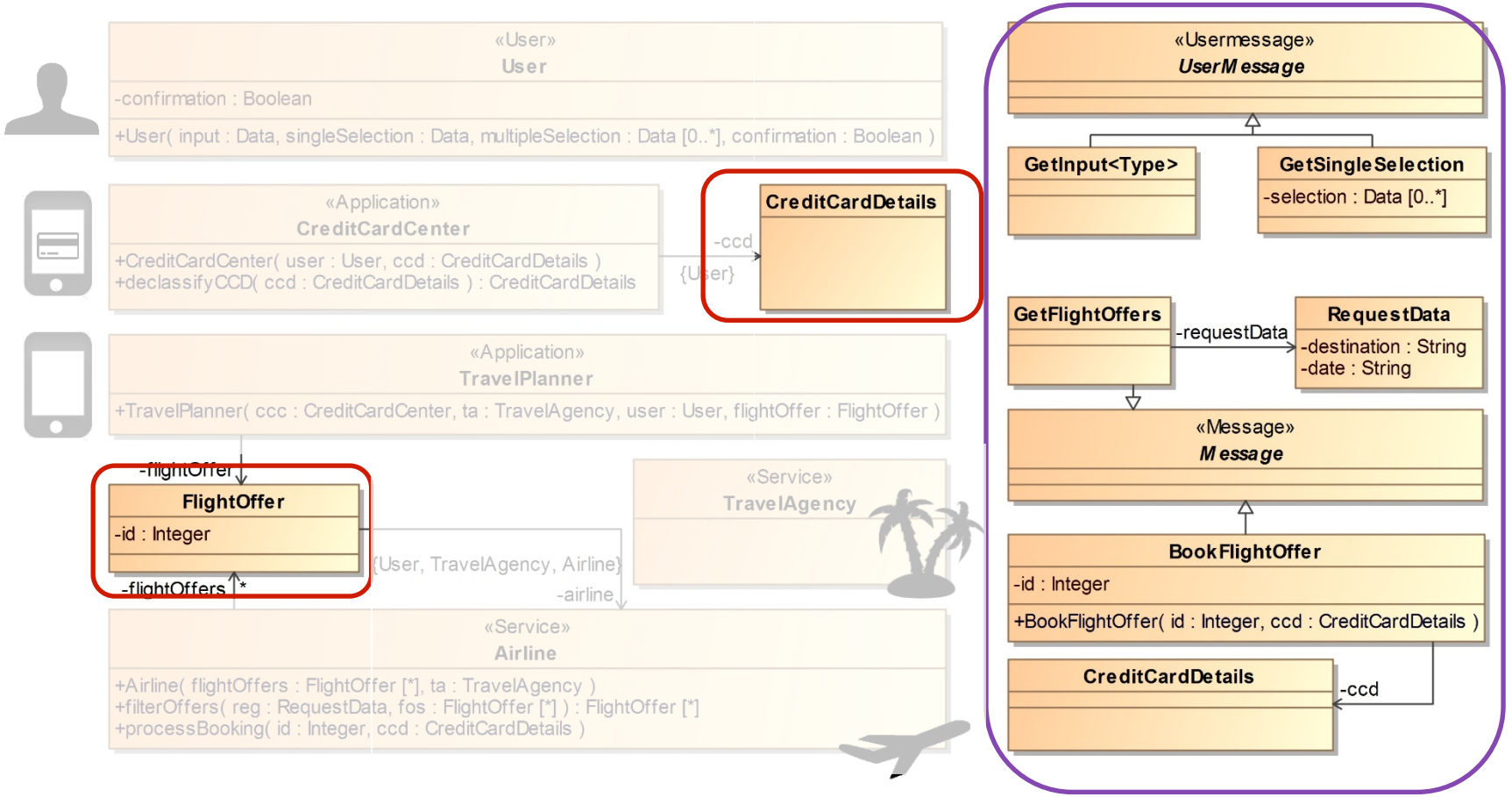




TravelPlanner: **data types**

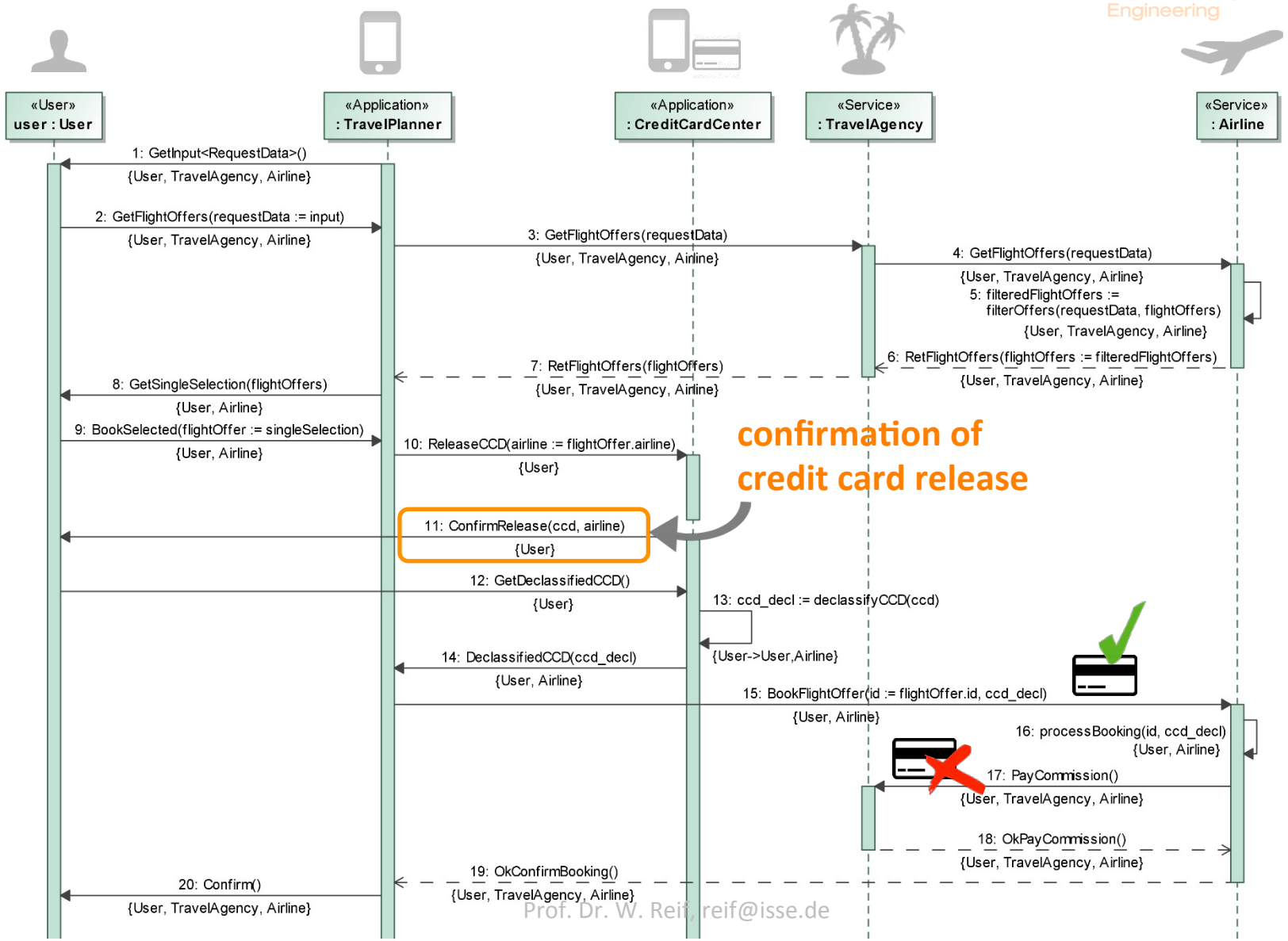
Institute for Software & Systems Engineering

Data types and **message classes**



TravelPlanner: **behaviour**

Institute for Software & Systems Engineering



Travel Planner: **security**

Institute for Software & Systems Engineering

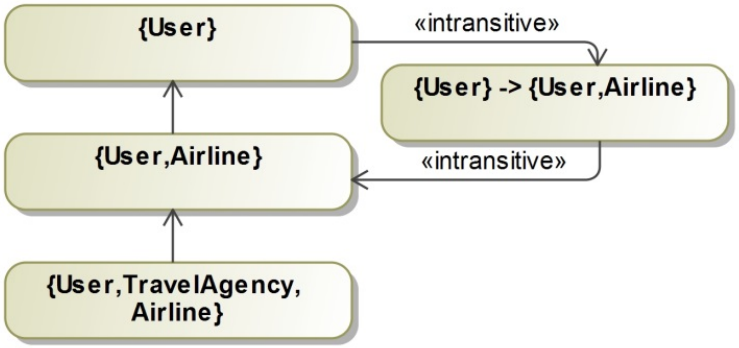
- **Subjects** (user, apps & web services): *S*

  $S_{TravelPlanner}$ = *{User, Airline, TravelAgency}*

- **Security domains**: $d \subseteq S$, $d \subseteq S \times S$

  {User}, {User, Airline}, {User, Airline, TravelAgency}, {User}->{User, Airline}

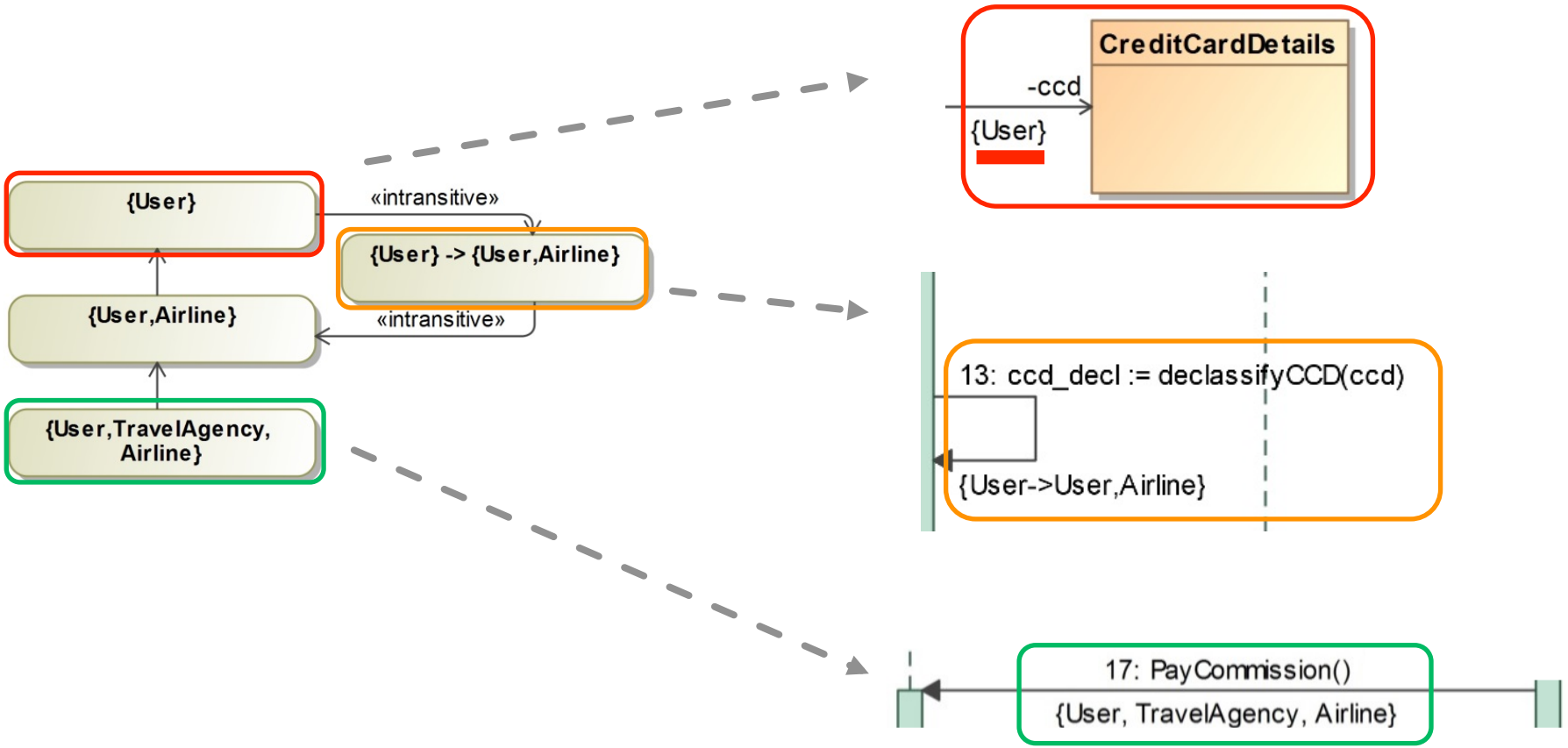
- **Security policy**:



Travel Planner: **security (cont.)**

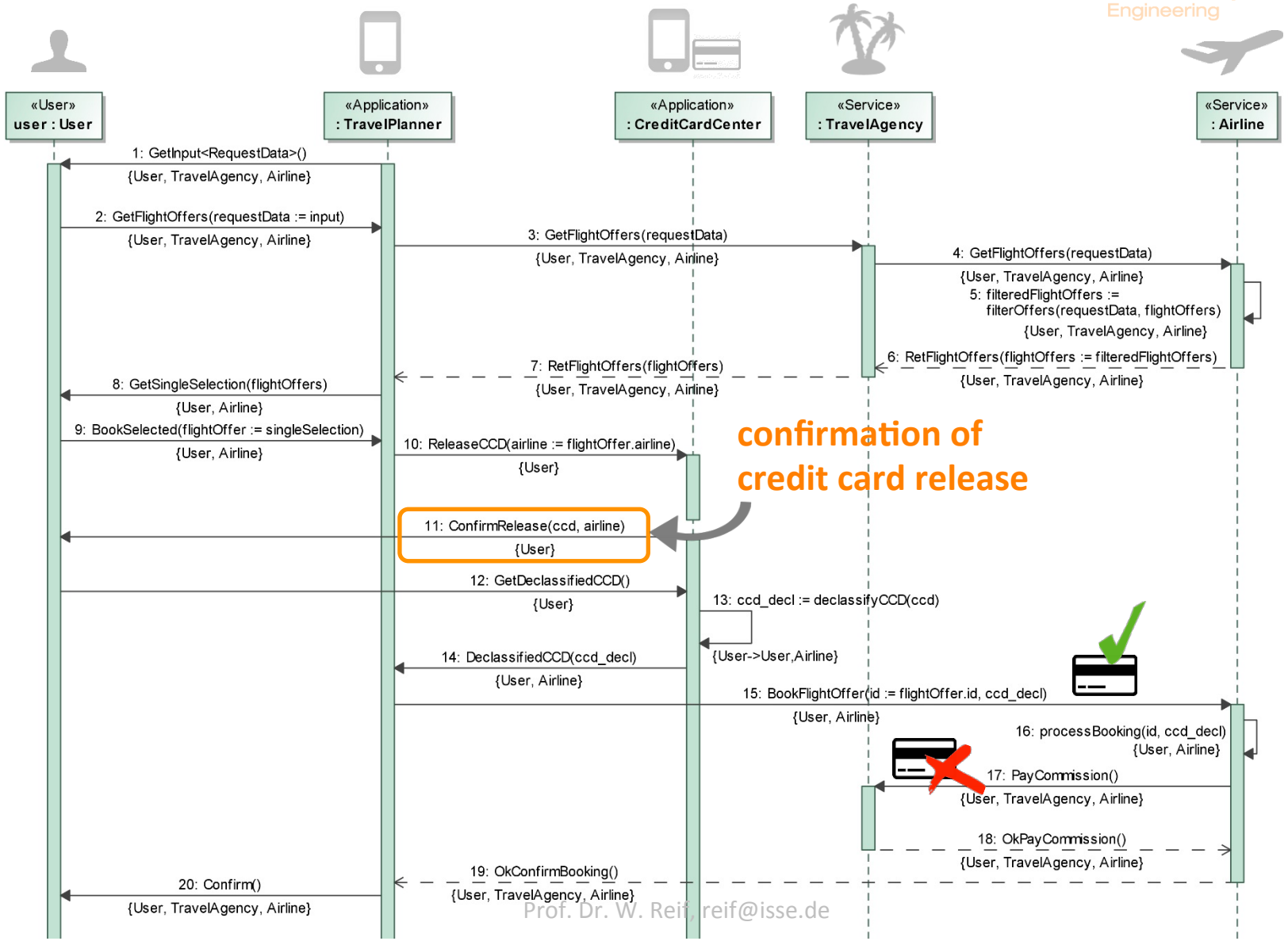
Institute for Software & Systems Engineering

Assigning security domains to data & messages:



TravelPlanner: **behaviour**

Institute for Software & Systems Engineering



Travel Planner: **security (cont.)**

Institute for Software & Systems Engineering

Noninterference theorem

(extension of Rushby/van der Meyden):

Security policy satisfied, if

RM1: $s_1 \approx_d s_2 \rightarrow output(s_1, d) = output(s_2, d)$ (true by construction) ✓

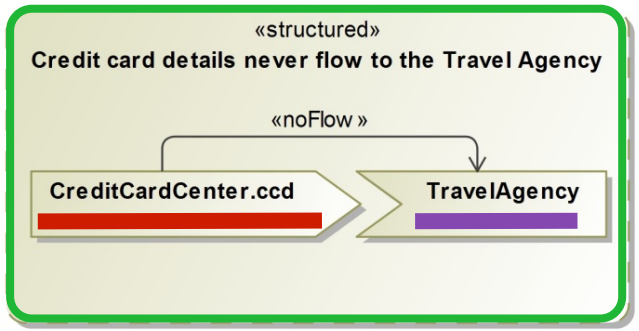
RM2: $s_1 \approx_{dom(a)} s_2 \wedge s_1(l) = s_2(l) \wedge l \in alter(dom(a)) \rightarrow step(s_1, a)(l) = step(s_2, a)(l)$ (true by construction) ✓

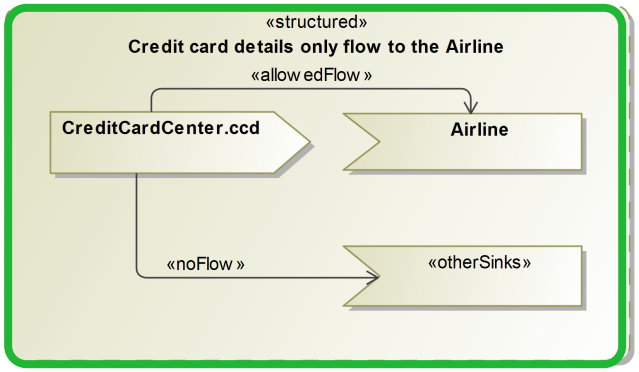
RM3: $step(s, a)(l) \neq s(l) \rightarrow l \in alter(dom(a))$ (true by construction) ✓

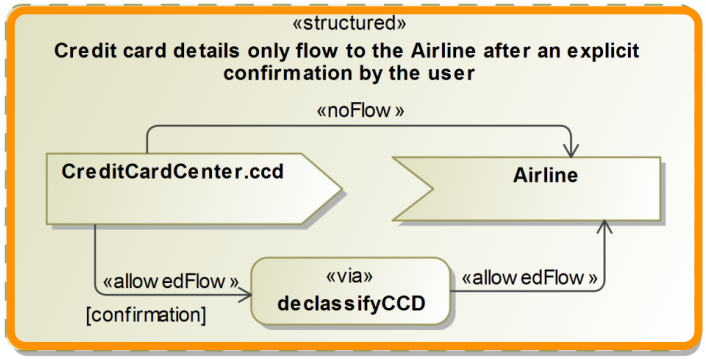
AOI: $alter(d_1) \cap observe(d_2) \neq \emptyset \rightarrow d_1 \rightsquigarrow d_2$ **(prove)**

TravelPlanner: **security properties**

Institute for Software & Systems Engineering



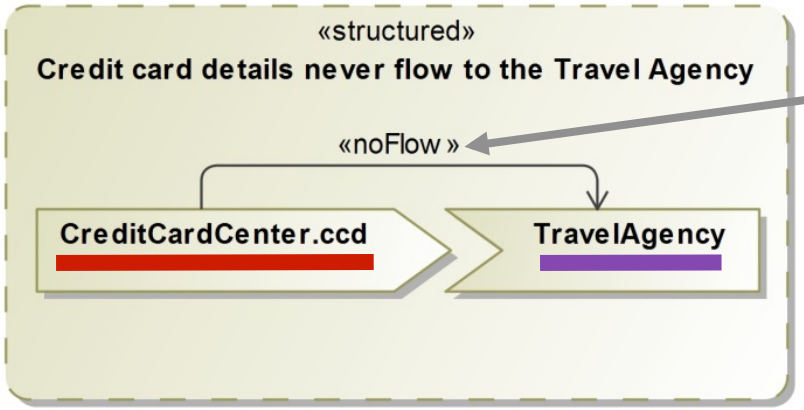




- Components (apps/services) and their attributes can be **sources** and **sinks** of information
- Information flow is either
  - **disallowed** (*«noFlow»*), or
  - **allowed** (*«allowedFlow»*)
    … via a specific method (*«via»*)
    … after **user confirmation** (*[confirmation]*)
- Properties can be
  - **automatically checked** on the code level, or
  - **verified interactively** using the formal model

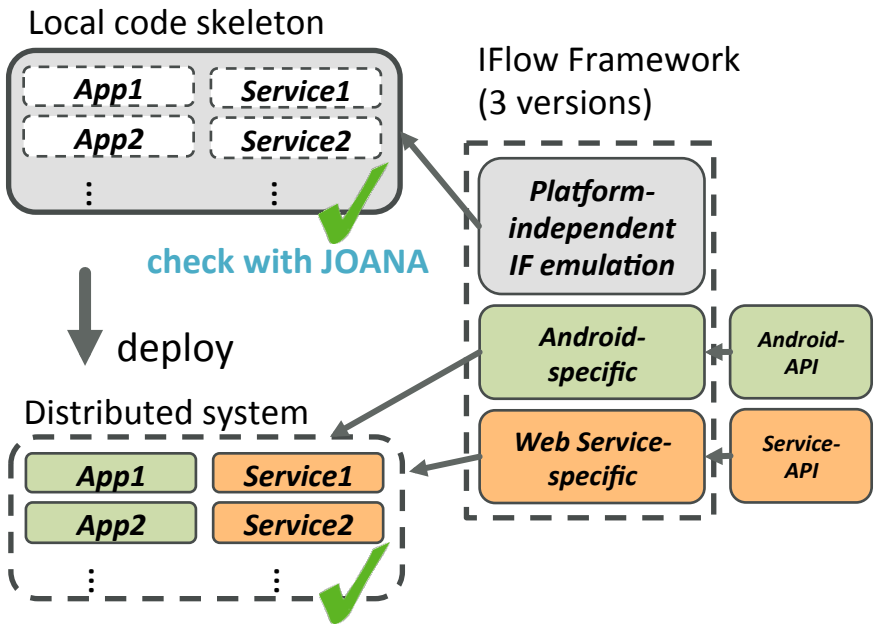
TravelPlanner: **automatic IF check**

Institute for Software & Systems Engineering



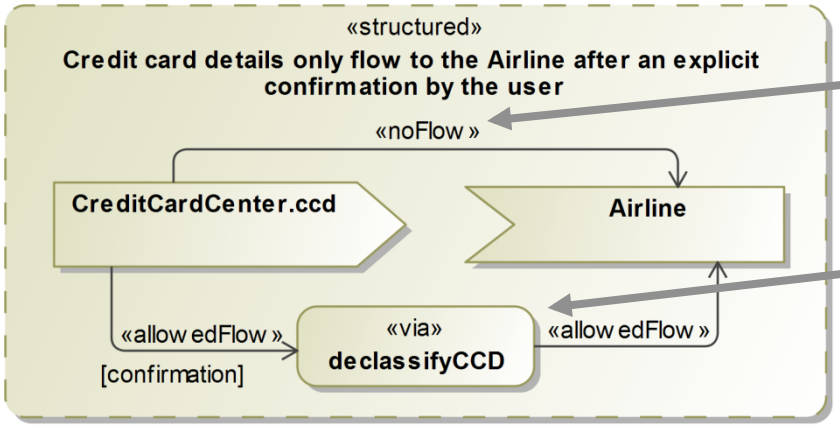
disallowed information flow

1. Generate a code skeleton as a **monolithic Java application**
2. Generate and run **information flow check** with JOANA (PDG-based IFC)
3. Distribute application as **Android apps** and **Java web services**



TravelPlanner: **verification**

Institute for Software & Systems Engineering



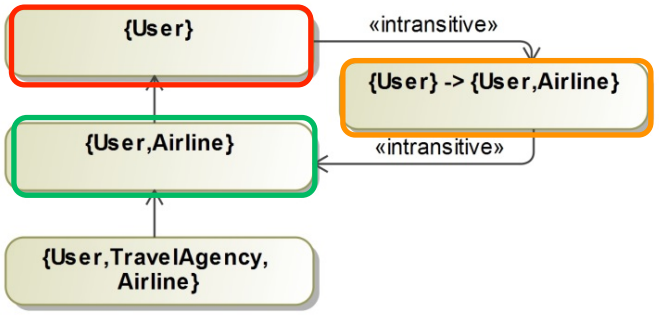
disallowed **direct** information flow

explicitly allowed information flow **via a specific method**

Proof obligations:

1. Security policy implies property
2. **Application model satisfies** security policy (noninterference theorem)
3. **declassifyCCD** is only called **after confirmation**

$\neg$ domain(**ccd**) $\Rightarrow$ domain(**Airline**) $\wedge$
domain(**ccd**) $\Rightarrow$ domain(**declassifyCCD**) $\wedge$
domain(**declassifyCCD**) $\Rightarrow$ domain(**Airline**)

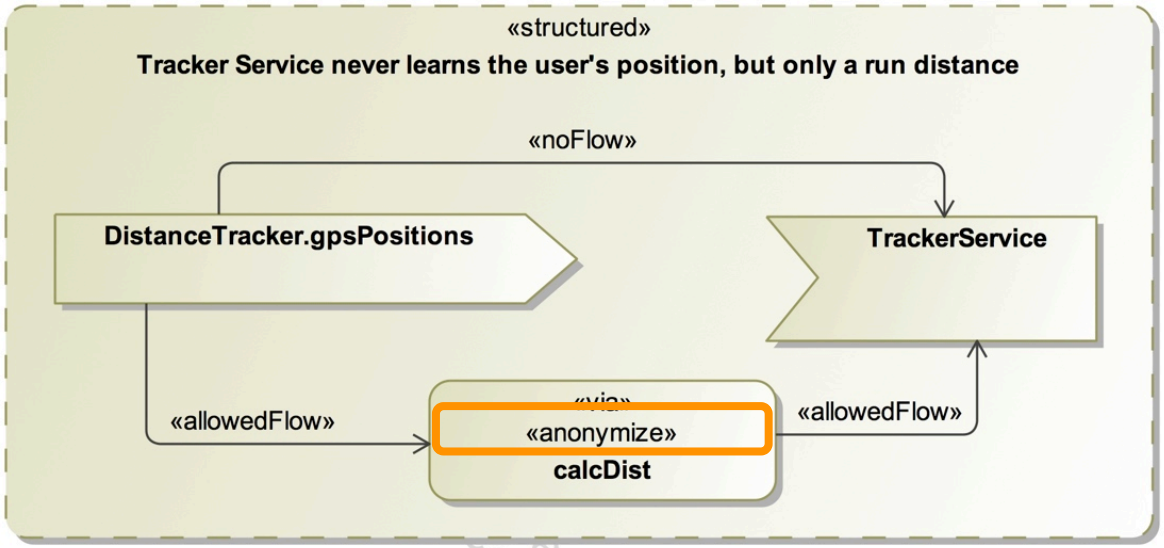


Overview

Institute for Software & Systems Engineering

- IFlow: a model driven approach for developing distributed apps with secure information flow
- Software/security co-design in distributed apps
  - Modeling
  - Automatic check and verification
- **Special aspect: limited release of information**
  - Modeling
  - Verification

Distance Tracker: **property**



- **Tracker Service never learns** the user's location
- **Distance Tracker properly anonymizes** the user's location data as distance

What do we prove?

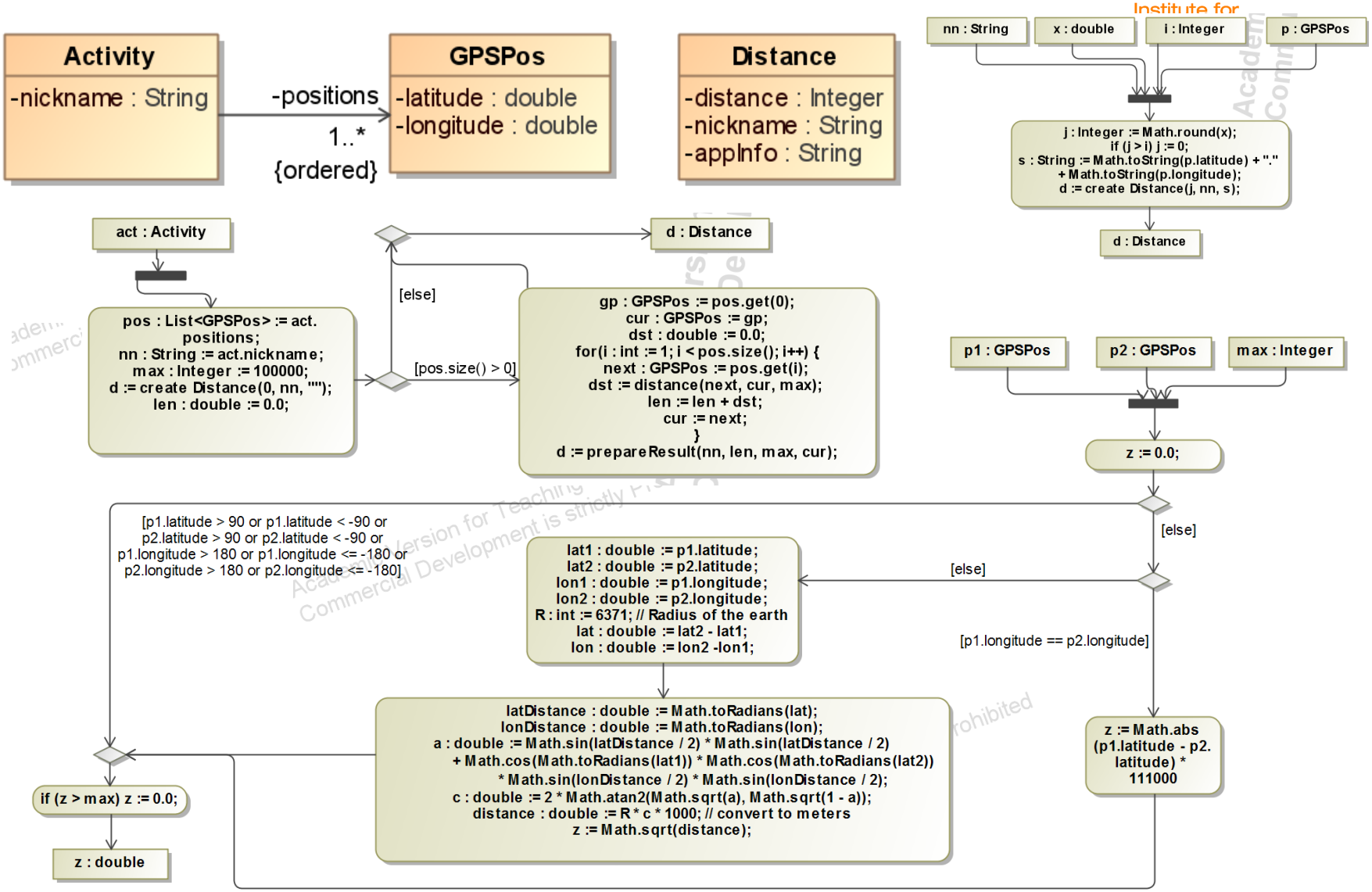
Institute for Software & Systems Engineering

- define equivalence class of all inputs that map to the same output
- reason about size of the classes

„For every valid output there are enough (***e***) different inputs with **mutually disjoint positions**“:

$$\forall\, x, d.\ \mathrm{calcDist}(x) = d \rightarrow$$
$$\exists\, a\_set.\ \#\, a\_set \geq \boldsymbol{e} \wedge \mathbf{disjoint}(a\_set) \wedge$$
$$\forall\, y.\ y \in a\_set \rightarrow \mathrm{calcDist}(y) = d$$

calcDist(act : Activity) : Distance



Summary

Institute for Software & Systems Engineering

IFlow integrates

- a model driven approach (ModelFlow)
- formally verified information flow properties
- automatic information flow control