Tisa: A Language Design and Modular Verification Technique for Temporal Policies in Web Services

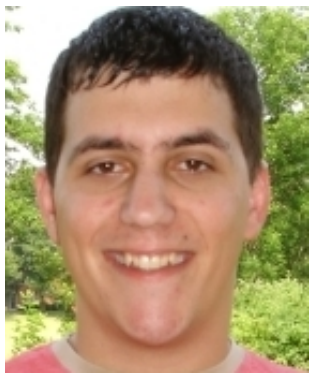
Hridesh Rajan

Computer Science @ Iowa State University

This work was supported in part by the US National Science Foundation (NSF) under grant CNS 0627354, CNS 0808913, and CAREER-08-46059.

Mahantesh Hosamani
Iowa State U.

Jia Tao
Iowa State U.



Steve Shaner
Iowa State U.

Gary T. Leavens
U. of Central Florida

Specification and Verification for Web Services

- Web-services important paradigm for organizing systems
- Demonstrating that a web-service complies with certain non-functional policies is important
  - e.g. to satisfy government regulations such as HIPAA

Problem: Conflict between Compliance and Modularity

- Compliance is helped by showing details
- . . . . . Modularity is helped by hiding details

Tisa: A Language Design and Verification Technique

- Greybox specifications: hide/expose details
- Modular and scalable verification technique

Specification and Verification for Web Services

- Web-services important paradigm for organizing systems
- Demonstrating that a web-service complies with certain non-functional policies is important
  - e.g. to satisfy government regulations such as HIPAA

Problem: Conflict between Compliance and Modularity

- Compliance is helped by showing details
- . . . . . Modularity is helped by hiding details

Tisa: A Language Design and Verification Technique

- Greybox specifications: hide/expose details
- Modular and scalable verification technique

Specification and Verification for Web Services

- Web-services important paradigm for organizing systems
- Demonstrating that a web-service complies with certain non-functional policies is important
  - e.g. to satisfy government regulations such as HIPAA

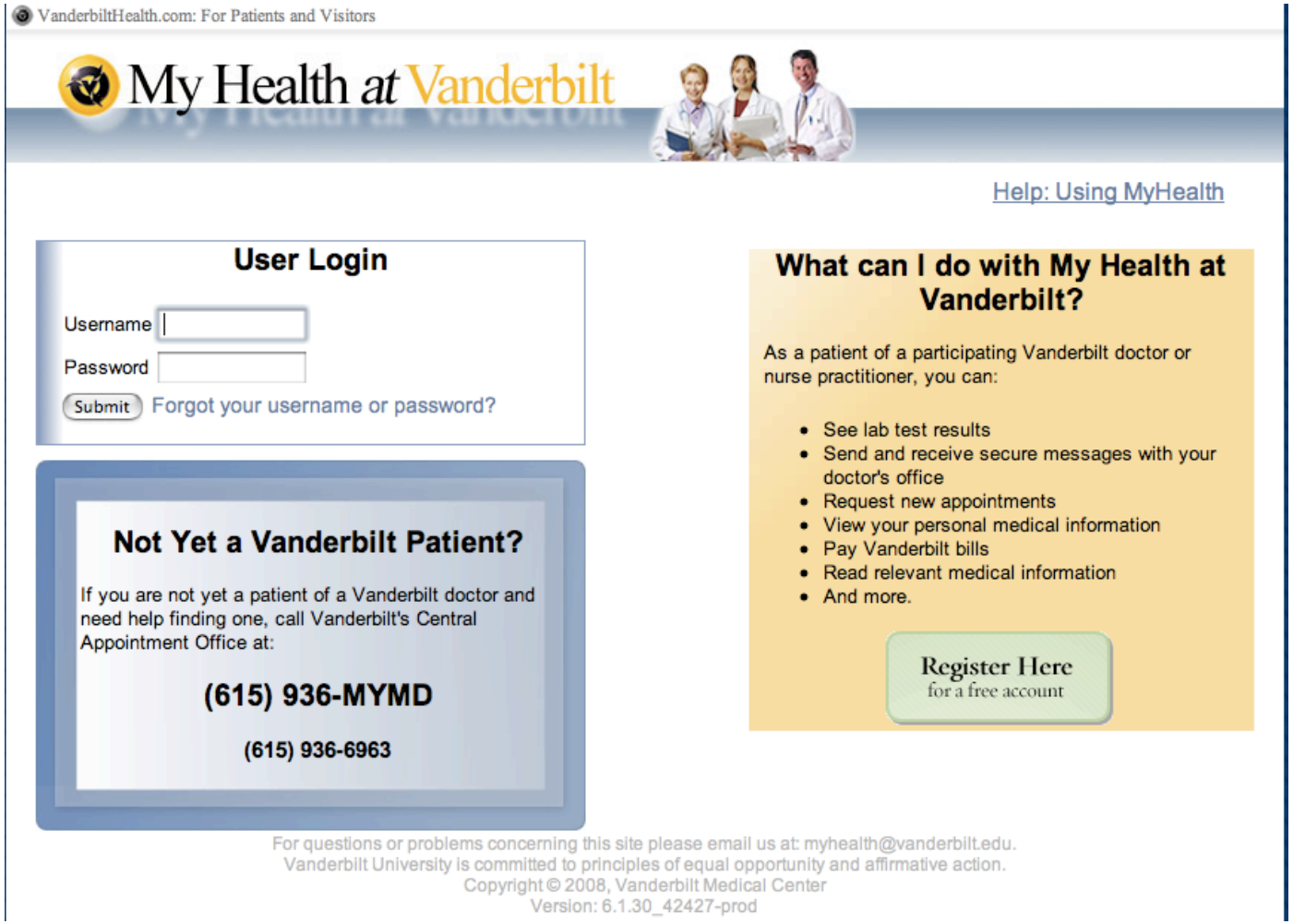
Problem: Conflict between Compliance and Modularity

- Compliance is helped by showing details
- . . . . . Modularity is helped by hiding details

Tisa: A Language Design and Verification Technique

- Greybox specifications: hide/expose details
- Modular and scalable verification technique

MyHealth Service at Vanderbilt University



Business Rules for MyHealth [Barth et al.'07]

A Contract for Patient Service

```
service Patient {

  /*@ requires pId >= 0; ensures result >=0; @*/
  int query(int pId, int msg);

  /*@ requires qId >= 0; ensures result >=0; @*/
  int retrieve(int qId);

}
```

Specification in a form similar to JML [Leavens *et al.*2006]

A Contract for Patient Service

```
service Patient {

  /*@ requires pId >= 0; ensures result >=0; @*/
  int query(int pId, int msg);

  /*@ requires qId >= 0; ensures result >=0; @*/
  int retrieve(int qId);

}
```

A Contract for Patient Service

```
service Patient {

  /*@ requires pId >= 0; ensures result >=0; @*/
  int query(int pId, int msg);

  /*@ requires qId >= 0; ensures result >=0; @*/
  int retrieve(int qId);

}
```

Behavioral Contracts Insufficient

Consider an example policy ϕ (Similar to HIPAA):

> A health question should ONLY be answered by the doctor.

Behavioral Contracts Insufficient

Now consider interactions made visible by behavioral contracts

Client	$\longrightarrow$	query(myId, msg = 99)	$\longrightarrow$	Patient Service
Client	$\longleftarrow$	qID	$\longleftarrow$	Patient Service
Client	$\longrightarrow$	retrieve(qID)	$\longrightarrow$	Patient Service
Client	$\longleftarrow$	101	$\longleftarrow$	Patient Service

Message	Meaning
99	Do I have AIDS?
101	Yes, What were you thinking?

Compliance to policy ϕ MAY NOT be demonstrated.

Behavioral Contracts Insufficient

- Demonstrating compliance requires providings access to internal states of the service
- Ability to make business rules explicit
- Providing such access MAY compromise modularity
- Clients will come to depend on the internal details
- Makes it harder to change them without breaking clients

Behavioral Contracts Insufficient

- Demonstrating compliance requires providings access to internal states of the service
- Ability to make business rules explicit
- Providing such access MAY compromise modularity
- Clients will come to depend on the internal details
- Makes it harder to change them without breaking clients

Is there a balance between compliance and modularity?

Tisa Language Design and Verification Technique

Modular Compliance via Greybox Specification of Services

- Greybox features expose (a bit more) detail
- . . . clients get to write more expressive assertions
- Rest of the details are hidden
- . . . so services are free to change it

Decoupled Verification via Greybox Specifications

- . . . client reasoning concerns policy + service spec
- . . . Services compliance concerns service spec + impl

An Example Tisa Specification : Patient Service

```
01 service Patient {
02
03
04
05
06
07
08
09
10
11
12
13 }
```

An Example Tisa Specification : Patient Service

```
01 service Patient {
02   int query(int pId, int msg) {
03
04   }
05   int retrieve(int qId) {
06
07
08
09
10
11
12   }
13 }
```

An Example Tisa Specification : Patient Service

```
01 service Patient {
02  int query(int pId, int msg) {
03   query(pId,msg)@Secretary //Fwd Questions
04  }
05  int retrieve(int qId) {
06
07
08
09
10
11
12  }
13 }
```

An Example Tisa Specification : Patient Service

```
01 service Patient {
02  int query(int pId, int msg) {
03   query(pId,msg)@Secretary //Fwd Questions
04  }
05  int retrieve(int qId) {
06
07   if ((qId/1000) == 1) {//Appointments
08    retrieve(qId)@Secretary
09   } else if((qId/1000) == 2) {//Health Q?
10    retrieve(qId)@Doctor
11   }
12  }
13 }
```

An Example Tisa Specification : Patient Service

```
01 service Patient {
02  int query(int pId, int msg) {
03   query(pId,msg)@Secretary //Fwd Questions
04  }
05  int retrieve(int qId) {
06   preserve qId > 0;
07   if ((qId/1000) == 1) {//Appointments
08    retrieve(qId)@Secretary
09   } else if((qId/1000) == 2) {//Health Q?
10    retrieve(qId)@Doctor
11   }
12  }
13 }
```

Another Example : Secretary Service

```
01 service Secretary {
02
03
04
05
06
07
08
09
10
11
12
13
14 }
```

Another Example : Secretary Service

```
01 service Secretary {
02   int query(int pId, int msg) {
03
04
05
06
07
08
09
10   }
11   int retrieve(int qId) {
12
13   }
14 }
```

Another Example : Secretary Service

```
01 service Secretary {
02  int query(int pId, int msg) {
03
04   if (msg >= 2) {//Health Q?
05    query(pId,msg)@Doctor
06   }
07   else {//Apointments
08
09   }
10  }
11  int retrieve(int qId) {
12
13  }
14 }
```

Another Example : Secretary Service

```
01 service Secretary {
02  int query(int pId, int msg) {
03   preserve pId > 0 && msg > 0;
04   if (msg >= 2) {//Health Q?
05    query(pId,msg)@Doctor
06   }
07   else {//Apointments
08    establish result > 0
09   }
10  }
11  int retrieve(int qId) {
12   requires qId > 0 ensures result > 0
13  }
14 }
```

Interaction Sequence

Client	$\rightarrow$	query(myId, msg = 99)@Patient	$\rightarrow$	Patient Service
Patient Service	$\rightarrow$	query(myId, msg)	$\rightarrow$	Secretary Service
Secretary Service	$\rightarrow$	query(myId, msg)	$\rightarrow$	Doctor Service
Secretary Service	$\leftarrow$	queryID	$\leftarrow$	Doctor Service
Patient Service	$\leftarrow$	queryID	$\leftarrow$	Secretary Service
Client	$\leftarrow$	queryID	$\leftarrow$	Patient Service

Client	$\rightarrow$	retrieve(queryID)@Patient	$\rightarrow$	Patient Service
Patient Service	$\rightarrow$	retrieve(queryID)@Doctor	$\rightarrow$	Doctor Service
Patient Service	$\leftarrow$	101	$\leftarrow$	Doctor Service
Client	$\leftarrow$	101	$\leftarrow$	Patient Service

Message	Meaning
99	Do I have AIDS?
101	Yes, What were you thinking?

Compliance to policy ϕ MAY NOW be demonstrated.

Client Verification: Overview

- Convert service specification $\mathcal{S}$ to an automaton $\boldsymbol{FSM(\mathcal{S})}$
- Express LTL policy $\phi \in \boldsymbol{\Phi(\mathcal{S})}$ as an automaton $\boldsymbol{FSM(\phi)}$
- Specification satisfies the policy if $\boldsymbol{FSM(\mathcal{S})} \cap \neg \boldsymbol{FSM(\phi)}$ is empty, which is a standard problem [Vardi and Wolper].

Finite-state Machine Construction I

Production relation: $NT \vdash se \rightsquigarrow (Z, z_0, R, \Delta), NT$ where
$NT \in \mathcal{NT} = \mathcal{W} \times \mathcal{M} \rightarrow Z$

(If Exp FSM)

$$\frac{NT \vdash se' \rightsquigarrow (Z', z', R', \Delta'), NT' \qquad NT' \vdash se'' \rightsquigarrow (Z'', z'', R'', \Delta''), NT'' \qquad Z = Z' \cup Z'' \cup \{z\} \qquad \Delta = \Delta' \uplus \Delta'' \uplus \{(z', \{sp\}), (z'', \{!sp\})\} \qquad R = R' \cup R'' \cup \{(z, z'), (z, z'')\}}{NT \vdash \texttt{if}\ (sp)\ \{se'\}\ \texttt{else}\ \{se''\} \rightsquigarrow (Z, z, R, \Delta), NT''}$$

(Web Method Call FSM 1)

$$\frac{\neg(\exists z :: NT(w, m) = z) \qquad NT' = NT \cup ((w, m), z) \qquad m(t_1, \ldots t_n)\{se\} = find(w, m) \qquad NT' \vdash se \rightsquigarrow (Z', z', R', \Delta'), NT'' \qquad Z = Z' \cup \{z\} \qquad \Delta = \Delta' \uplus \{(z', \{m@w\})\} \qquad R = R' \cup \{(z, z')\}}{NT \vdash m(v_1, \ldots, v_n)@w \rightsquigarrow (Z, z, R, \Delta), NT''}$$

(Web Method Call FSM 2)

$$\frac{z = NT(w, m)}{NT \vdash m(v_1, \ldots, v_n)@w \rightsquigarrow (\{z\}, z, \{\}, \{\}), NT}$$

Finite-state Machine Construction II

(SPEC EXP FSM)

$$Z = \{z_1, z_2, z_3, z_4\} \qquad R = \{(z, z_1), (z, z_2), (z_1, z_3), (z_1, z_4), (z_3, z')\}$$
$$\Delta_{pre} = \{(z_1, \{sp_1\}), (z_2, \{!sp_1\})\}$$
$$\Delta = \Delta_{pre} \uplus \{(z_3, \{sp_1, sp_2\}), (z_4, \{sp_1, !sp_2\})\}$$

$$\frac{}{NT \vdash \texttt{requires}\ sp_1\ \texttt{ensures}\ sp_2 \rightsquigarrow (Z, z, R, \Delta), NT}$$

Client Verification: Scalability Advantages

- Service spec. ($\mathcal{S}$) is simpler than Service impl. ($\mathcal{P}$)
- State space of $\boldsymbol{FSM(\mathcal{S})}$ is smaller compared to $\boldsymbol{FSM(\mathcal{P})}$
- . . . determining $\boldsymbol{FSM(\mathcal{S}) \cap \neg \mathcal{FSM}(\phi)}$ is more scalable.
- Clients solve easier problem.

Service Compliance Demonstration : Overview

Given a service spec. ($\mathcal{S}$) and a service impl. ($\mathcal{P}$)

- prove that $\mathcal{S} \sqsubseteq \mathcal{P}$
- method similar to [Shaner, Leavens, Naumann 2007]
- uses structural refinement rules to compare $\mathcal{S}$ and $\mathcal{P}$

Inference Rules for Proving Tisa Refinement $\sqsubseteq$

(PROGRAM REF)

$$\frac{\forall i \in \{1..m\}\ \exists j \in \{1...n\}\ decl_j \in servicedecl \wedge servicespec_i \sqsubseteq decl_j}{servicespec_1 \ldots servicespec_m \sqsubseteq decl_1 \ldots decl_n}$$

(SP REF)

$$\frac{sp = e}{sp \sqsubseteq e}$$

(SERVICE REF)

$$\frac{\forall i \in \{1..m\}\ \exists j \in \{1...n\}\ wmspec_i \sqsubseteq meth_j}{\texttt{service}\ w\ \{wmspec_1 \ldots wmspec_n\} \sqsubseteq \texttt{service}\ w\ \{field_1 \ldots field_f\ meth_1 \ldots meth_n\}}$$

(WEB METHOD REF)

$$\frac{se \sqsubseteq e}{t\ m(form_1 \ldots form_n)\ \{se\} \sqsubseteq t\ m(form_1 \ldots form_n)\ \{e\}}$$

(SEQ EXP REF)

$$\frac{se_1 \sqsubseteq e_1 \qquad se_2 \sqsubseteq e_2}{se_1\ ;\ se_2 \sqsubseteq e_1\ ;\ e_2}$$

(IF EXP REF)

$$\frac{sp \sqsubseteq e_b \qquad se_T \sqsubseteq e_T \qquad se_F \sqsubseteq e_F}{\texttt{if}\ (sp)\ \{se_T\}\ \texttt{else}\ \{se_F\} \sqsubseteq \texttt{if}\ (e_b)\ \{e_T\}\ \texttt{else}\ \{e_F\}}$$

(DEF EXP REF)

$$\frac{sp \sqsubseteq e_{init} \qquad se \sqsubseteq e_{body}}{form = sp\ ;\ se \sqsubseteq form = e_{init}\ ;\ e_{body}}$$

Service Compliance Checking: Advantages

- Refinement checking ($\mathcal{S} \sqsubseteq \mathcal{P}$) easy to automate
- Independent of the number and policies of clients
- Need to be repeat only when service impl. changes

Related Work

- Greybox Specification and Verification
  - Büchi and Weck[**?**], Barnett and Schulte[**?**], Wasserman and Blum[**?**], Tyler and Soundarajan[**?**], Shaner, Leavens and Naumann[**?**]
- Behavioral Contract Refines Desired Requirements
  - Castagna *et al.*[**?**], Bravetti *et al.*[**?**]
- Verifying Behavioral Contracts for Web Services
  - Acciai[**?**], Kuo *et al.*[**?**], Baresi *et al.*[**?**], Barbon *et al.*[**?**]

Future Work

- **Theory**
  - Relaxing rules for refinements
  - Proving refinement for other logic (besides LTL)
- **Realization**
  - Checking integrity of Refinement Mechanism
    - Some work already [Rajan-Hosamani IEEE SOC 08]
    - More work needed, e.g. apply zero knowledge proofs
  - Language-independent incarnation of Tisa
    - Important from the industrial perspective
    - Expressing Tisa features in XML-like syntax
  - Auto-generation of Tisa Specs based on property
    - To decrease annotation burden
    - Appears possible to some extent

Specification and Verification for Web Services

- Web-services important paradigm for organizing systems
- Demonstrating that a web-service complies with certain non-functional policies is important
  - e.g. to satisfy government regulations such as HIPAA

Problem: Conflict between Compliance and Modularity

- Compliance is helped by showing details
- . . . . . Modularity is helped by hiding details

Tisa: A Language Design and Verification Technique

- Greybox specifications: hide/expose details
- Modular and scalable verification technique

Questions?

http://www.cs.iastate.edu/~tisa/

References I

Overview of the Tisa Project