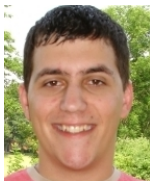




Hridayesh Rajan  
Iowa State U.



Jia Tao  
Iowa State U.



Steve Shaner  
Iowa State U.



Gary T. Leavens  
U. of Central Florida

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

18th European Symposium on Programming (ESOP '09)

Rajan and Tao were supported in part by the US National Science Foundation (NSF) under grant CNS 0627354. Rajan, Shaner and Leavens were supported in part by the NSF under grant CNS 0808913.

## 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

VanderbiltHealth.com: For Patients and Visitors



[Help: Using MyHealth](#)

### User Login

Username

Password

[Forgot your username or password?](#)

### Not Yet a Vanderbilt Patient?

If you are not yet a patient of a Vanderbilt doctor and need help finding one, call Vanderbilt's Central Appointment Office at:

**(615) 936-MYMD**

**(615) 936-6963**

### What can I do with My Health at Vanderbilt?

As a patient of a participating Vanderbilt doctor or nurse practitioner, you can:

- See lab test results
- Send and receive secure messages with your doctor's office
- Request new appointments
- View your personal medical information
- Pay Vanderbilt bills
- Read relevant medical information
- And more.

[Register Here](#)  
for a free account

For questions or problems concerning this site please email us at: [myhealth@vanderbilt.edu](mailto:myhealth@vanderbilt.edu).  
Vanderbilt University is committed to principles of equal opportunity and affirmative action.  
Copyright © 2008, Vanderbilt Medical Center  
Version: 6.1.30\_42427-prod

# 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  $\phi$  (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  →  query(myId, msg = 99)  →  Patient Service
Client  ←           qID          ←  Patient Service
Client  →           retrieve(qID) →  Patient Service
Client  ←           101          ←  Patient Service
  
```

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

Compliance to policy  $\phi$  MAY NOT be demonstrated.

## Behavioral Contracts Insufficient

- ▶ Demonstrating compliance requires providing 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 providing 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 }
```

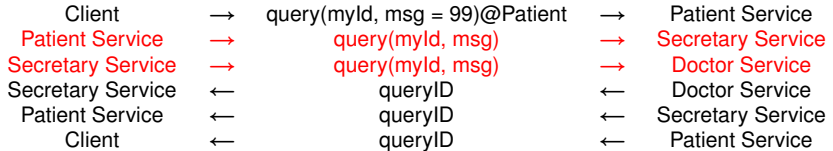
## Another Example : Secretary Service

```
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



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

Compliance to policy  $\phi$  MAY NOW be demonstrated.



# Client Verification: Overview

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

# Finite-state Machine Construction I

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

(IF EXP FSM)

$$\begin{array}{l}
 NT \vdash \mathbf{se}' \rightsquigarrow (Z', z', R', \Delta'), NT' \quad NT' \vdash \mathbf{se}'' \rightsquigarrow (Z'', z'', R'', \Delta''), NT'' \\
 Z = Z' \cup Z'' \cup \{z\} \quad \Delta = \Delta' \uplus \Delta'' \uplus \{(z', \{\mathbf{sp}\}), (z'', \{\!|\mathbf{sp}\})\} \\
 R = R' \cup R'' \cup \{(z, z'), (z, z'')\} \\
 \hline
 NT \vdash \mathbf{if}(\mathbf{sp}) \{ \mathbf{se}' \} \mathbf{else} \{ \mathbf{se}'' \} \rightsquigarrow (Z, z, R, \Delta), NT''
 \end{array}$$

(WEB METHOD CALL FSM 1)

$$\begin{array}{l}
 \neg(\exists z :: NT(w, m) = z) \quad NT' = NT \cup ((w, m), z) \\
 m(t_1, \dots, t_n)\{\mathbf{se}\} = \mathbf{find}(w, m) \quad NT' \vdash \mathbf{se} \rightsquigarrow (Z', z', R', \Delta'), NT'' \\
 Z = Z' \cup \{z\} \quad \Delta = \Delta' \uplus \{(z', \{m@w\})\} \quad R = R' \cup \{(z, z')\} \\
 \hline
 NT \vdash m(v_1, \dots, v_n)@w \rightsquigarrow (Z, z, R, \Delta), NT''
 \end{array}$$

(WEB METHOD CALL FSM 2)

$$\begin{array}{l}
 z = NT(w, m) \\
 \hline
 NT \vdash m(v_1, \dots, v_n)@w \rightsquigarrow (\{z\}, z, \{\}, \{\}), NT''
 \end{array}$$

# Finite-state Machine Construction II

(SPEC EXP FSM)

$$Z = \{z_1, z_2, z_3, z_4\} \quad R = \{(z, z_1), (z, z_2), (z_1, z_3), (z_1, z_4), (z_3, z')\}$$

$$\Delta_{pre} = \{(z_1, \{sp_1\}), (z_2, \{!sp_1\})\}$$

$$\Delta = \Delta_{pre} \uplus \{(z_3, \{sp_1, sp_2\}), (z_4, \{sp_1, !sp_2\})\}$$

---


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

## Client Verification: Scalability Advantages

- ▶ Service spec. ( $\mathcal{S}$ ) is simpler than Service impl. ( $\mathcal{P}$ )
- ▶ State space of  $FSM(\mathcal{S})$  is smaller compared to  $FSM(\mathcal{P})$
- ▶ ... determining  $FSM(\mathcal{S}) \cap \neg FSM(\phi)$  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)

$$\frac{\forall i \in \{1..m\} \exists j \in \{1..n\} \text{decl}_j \in \text{servicedecl} \wedge \text{servicespec}_i \sqsubseteq \text{decl}_j}{\text{servicespec}_1 \dots \text{servicespec}_m \sqsubseteq \text{decl}_1 \dots \text{decl}_n}$$

(SP REF)

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

(SERVICE REF)

$$\frac{\forall i \in \{1..m\} \exists j \in \{1..n\} \text{wmspec}_i \sqsubseteq \text{meth}_j}{\text{service } w \{ \text{wmspec}_1 \dots \text{wmspec}_n \} \sqsubseteq \text{service } w \{ \text{field}_1 \dots \text{field}_f \text{meth}_1 \dots \text{meth}_n \}}$$

(WEB METHOD REF)

$$\frac{se \sqsubseteq e}{t \ m(\text{form}_1 \dots \text{form}_n) \{se\} \sqsubseteq t \ m(\text{form}_1 \dots \text{form}_n) \{e\}}$$

(SEQ EXP REF)

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

(IF EXP REF)

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

(DEF EXP REF)

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

## 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[6], Barnett and Schulte[4], Wasserman and Blum[11], Tyler and Soundarajan[10], Shaner, Leavens and Naumann[9]
- ▶ Behavioral Contract Refines Desired Requirements
  - ▶ Castagna *et al.*[7], Bravetti *et al.*[5]
- ▶ Verifying Behavioral Contracts for Web Services
  - ▶ Acciai[1], Kuo *et al.*[8], Baresi *et al.*[3], Barbon *et al.*[2]



# 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

-  L. Acciai and M. Boreale.  
XPi: A typed process calculus for XML messaging.  
*Science of Computer Programming*, 71(2):110–143, 2008.
-  F. Barbon, P. Traverso, M. Pistore, and M. Trainotti.  
Run-time monitoring of instances and classes of web service compositions.  
In *ICWS*, pages 63–71, 2006.
-  L. Baresi, C. Ghezzi, and S. Guinea.  
Smart monitors for composed services.  
In *ICSOC*, pages 193–202, 2004.

## References II



M. Barnett and W. Schulte.

Runtime verification of .net contracts.

*Journal of Systems and Software*, 65(3):199–208, March 2003.



M. Bravetti and G. Zavattaro.

Towards a unifying theory for choreography conformance and contract compliance.

*In Software Composition*, pages 34–50, 2007.






M. Büchi and W. Weck.

The greybox approach: When blackbox specifications hide too much.

Technical Report 297, Turku Center for Computer Science, August 1999.

## References III

-  G. Castagna, N. Gesbert, and L. Padovani.  
A theory of contracts for web services.  
In *POPL*, pages 261–272, 2008.
-  D. Kuo, A. Fekete, P. Greenfield, S. Nepal, J. Zic,  
S. Parastatidis, and J. Webber.  
Expressing and reasoning about service contracts in  
service-oriented computing.  
In *ICWS*, pages 915–918, 2006.
-  S. M. Shaner, G. T. Leavens, and D. A. Naumann.  
Modular verification of higher-order methods with  
mandatory calls specified by model programs.  
In *OOPSLA*, pages 351–368, 2007.

## References IV



B. Tyler and N. Soundarajan.  
Black-box testing of grey-box behavior.  
In *FATES*, pages 1–4, 2003.



H. Wasserman and M. Blum.  
Software reliability via run-time result-checking.  
*J. ACM*, 44(6):826–849, 1997.





# Overview of the Tisa Project

