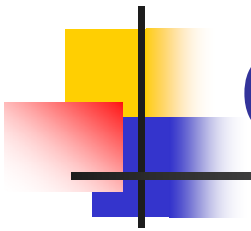
# Methods for Designing SIP Services in SDL with Fewer Feature Interactions

Presented by: Ken Y. Chan

School of Information Technology and Engineering, University of Ottawa

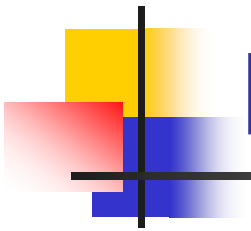
Feature Interactions Workshop 2003

Date: Wed, June 11, 2003



# Outline

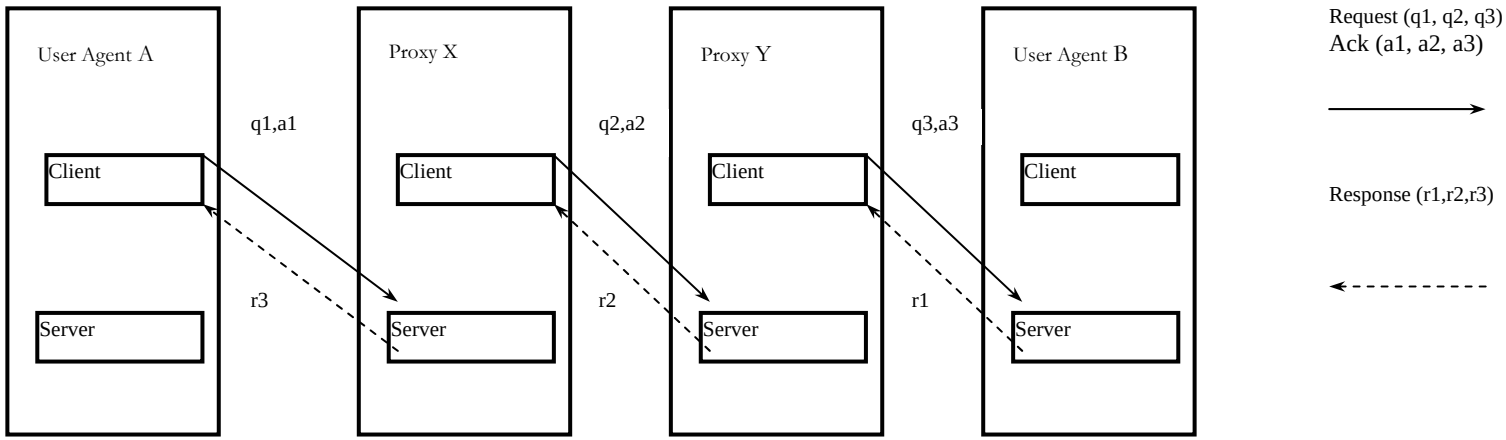
- Motivations
- Overview of SIP & FI
- SDL Model of SIP and its services
- Simulation, Verification & Validation
- Extended FI taxonomy
- Detecting & Preventing SIP FI's using Tau
- New FIs in SIP
- Conclusion & Future Works



# Motivations

- No Formal Service Specification of SIP (IETF RFC 2543 & 3261) -> To improve existing RFC and drafts.
- Feasibility of SDL/MSC (Tau) tools to model IETF signaling protocols.
- Leverage POTS FIs to prevent FIs in SIP
- New Feature Interactions in SIP

# Overview of SIP




- [back-to-back/regular] User Agent (Client & Server) & Stateful/Stateless Proxy
- Message Type: Request, Response, Acknowledge, others.

# Sample SIP Message Headers

*INVITE sip:ken@ee.uottawa.ca SIP/2.0*

*Via: SIP/2.0/UDP gtwy1.uottawa.ca;branch=8348*

*;maddr=137.128.16.254;ttl=16*

*Via: SIP/2.0/UDP gtwy.ee.uottawa.ca*

*Record-Route: gtwy.ee.uottawa.ca*

*From: Bill Gate <sip:bill@Microsoft.com>*

*To: Ken Chan <sip:ken@uottawa.ca>*

*Contact: Ken Chan <sip:ken@site.uottawa.ca>*

*Call-ID: 56258002189@site.uottawa.ca*

*CSeq: 1 INVITE*

*Subject: SIP will be discussed, too*

*Content-Type: application/sdp*

*Content-Length: 187*

*v=0*

*o=bill 53655765 2353687637 IN IP4 224.116.3.4*

*s=RTP Audio*

*i=Discussion of .Net*

*c=IN IP4 224.2.0.1/127*

*t=0 0*

*m=audio 3456 RTP/AVP 0*

*OK 200 SIP/2.0*

*Via:SIP/2.0/UDP gtwy1.uottawa.ca;branch=8348*

*;maddr=137.128.16.254;ttl=16*

*Record-Route: gtwy.ee.uottawa.ca*

*From: Bill Gate <sip:bill@Microsoft.com>*

*To: Ken Chan <sip:ken@uottawa.ca>*

*Contact: Ken Chan <sip:ken@site.uottawa.ca>*

*Call-ID: 56258002189@site.uottawa.ca*

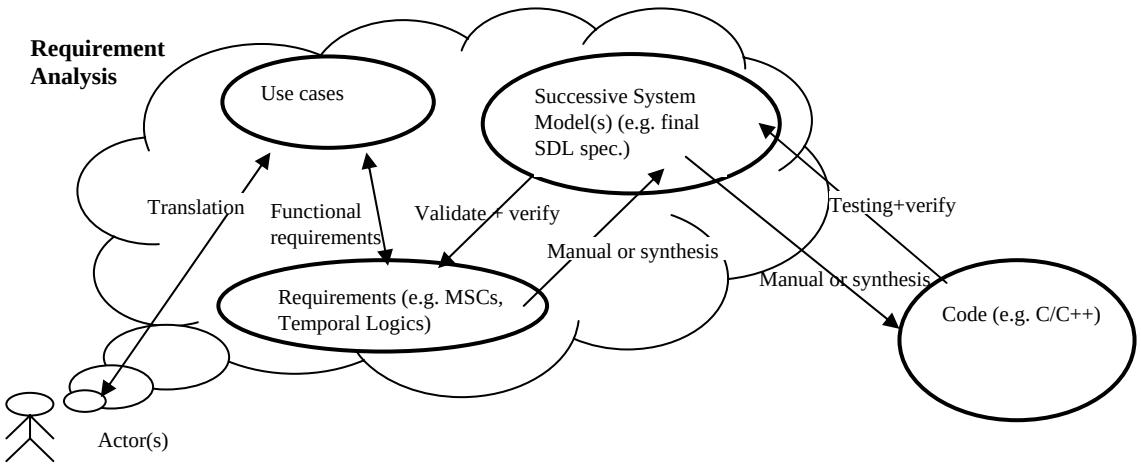
*CSeq: 1 INVITE*

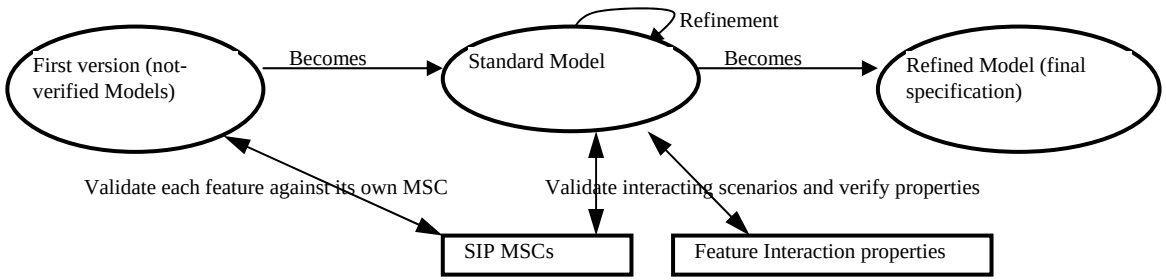
*Content-Type: application/sdp*

*Content-Length: 187*

- Significant header fields:
  - Request-URI, Method, Response Code,From, To, Contact(s), Via(s), Record-Route(s),Call-Id, CSeq, Content-type
- SDP body may contain feature commands and parameters

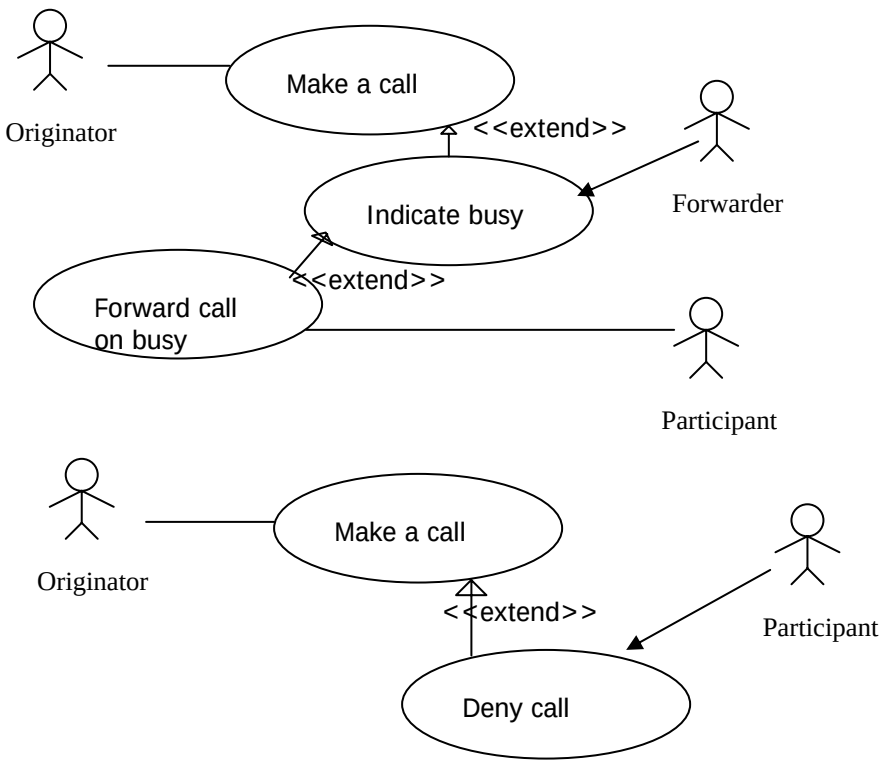
# Design Approach



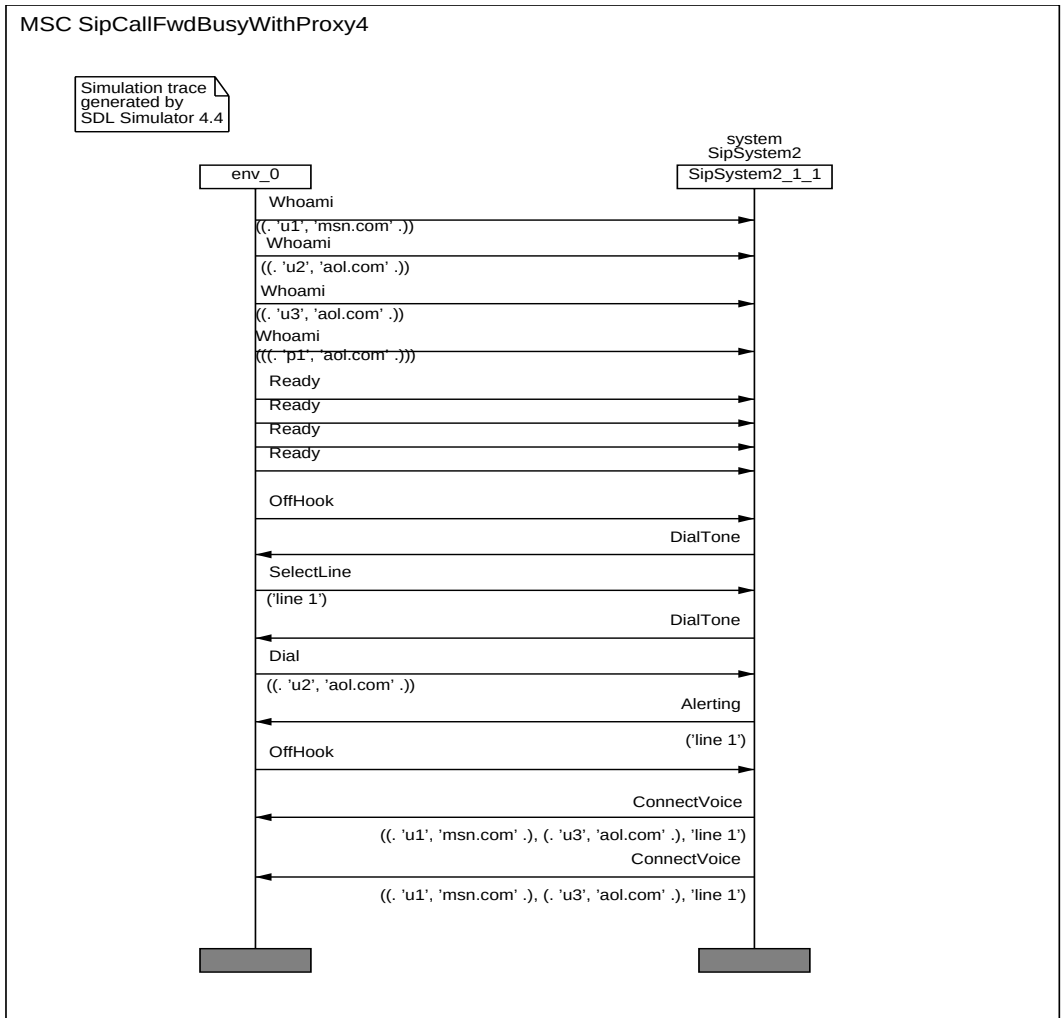

- The process is highly iterative.
- Modeling starts with SIP basic service (establishing, terminating, suspending two-party call, and ringing, alert, dial tone)
- Add advanced call features (CFB, TCS, OCS..etc) later.

# Use Cases - CFB and OCS



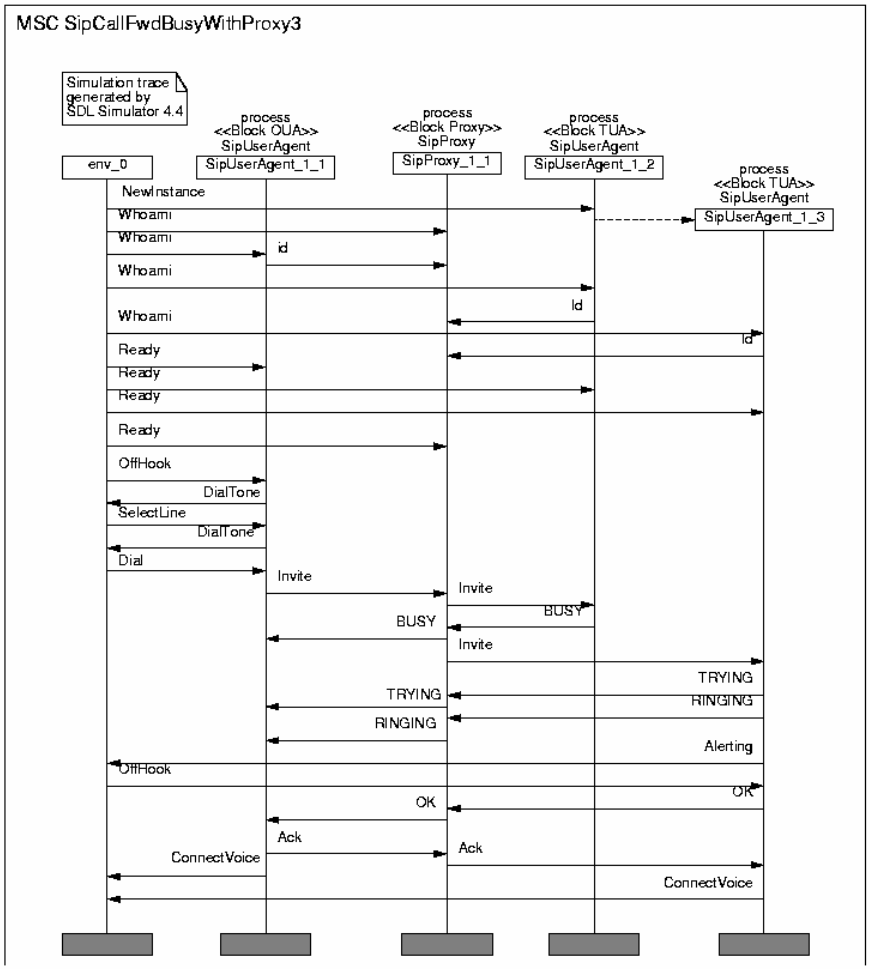

- Each actor has a role.
- Each use case represents one or more scenarios.

# Use Case Scenarios as MSC



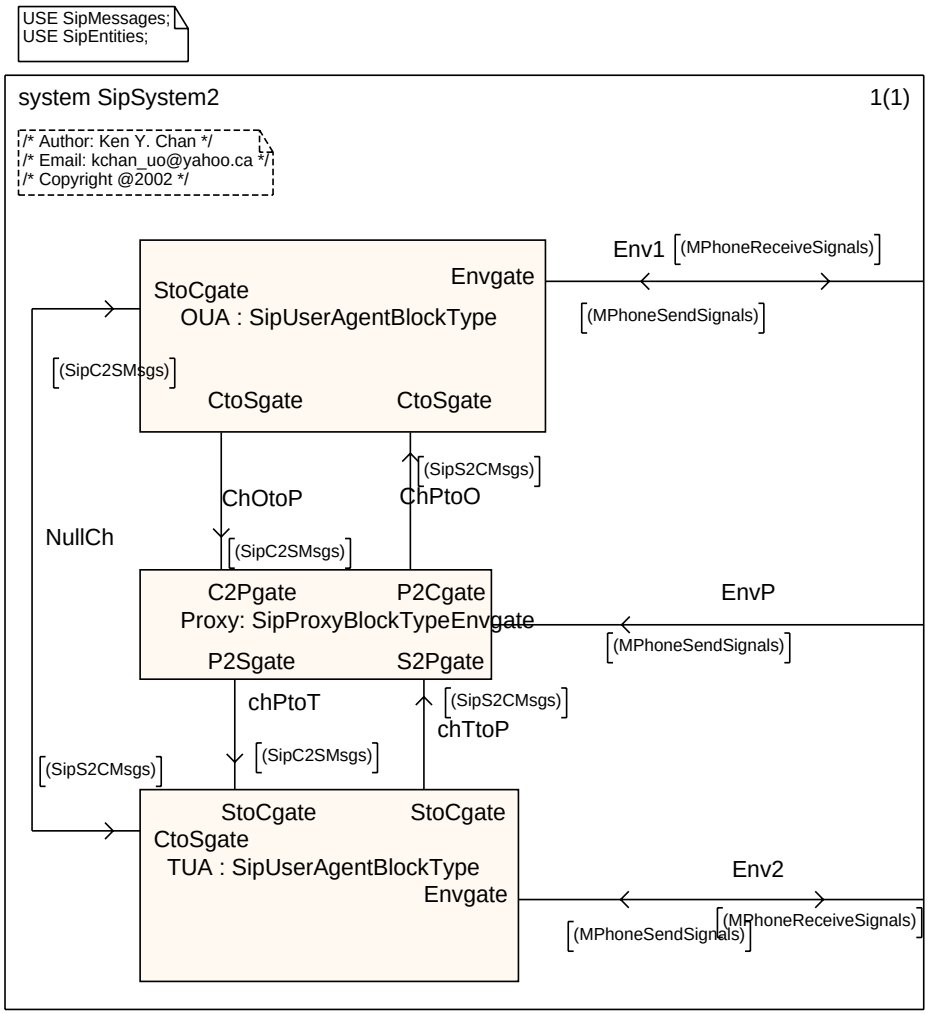

- This is Call Forward Busy (CFB).
- Call Flow Diagrams do not represent service scenarios in the sense of use cases.
- So we define service usage scenarios at the interface between the user and the system.
- Env_0 represents all users/actors.
- Interactions between the users and the SIP system describe use case scenarios of SIP services.
- Abstract User interfaces = { Whoami, OffHook, SelectLine, Dial, OnHook, Alerting, DialTone}.

# Test Scenarios as MSC



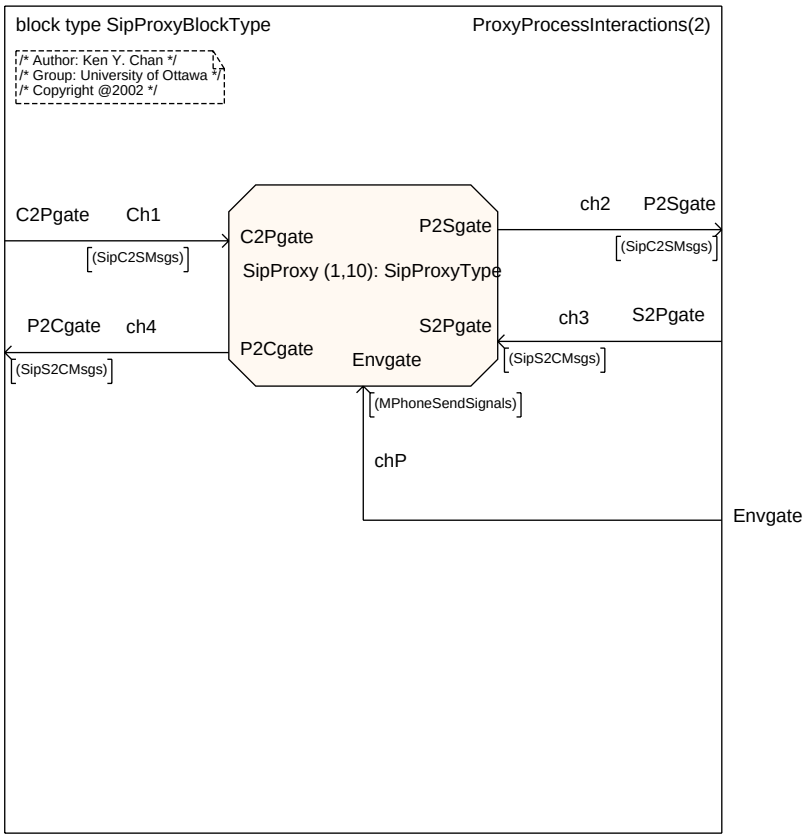

- Call Forward Busy “service and protocol scenario” .
- Test Scenario is the combination of the use case scenario with the corresponding scenario of exchanged SIP messages.
- It is a MSC for validating the SDL specification.

# Structural Definition



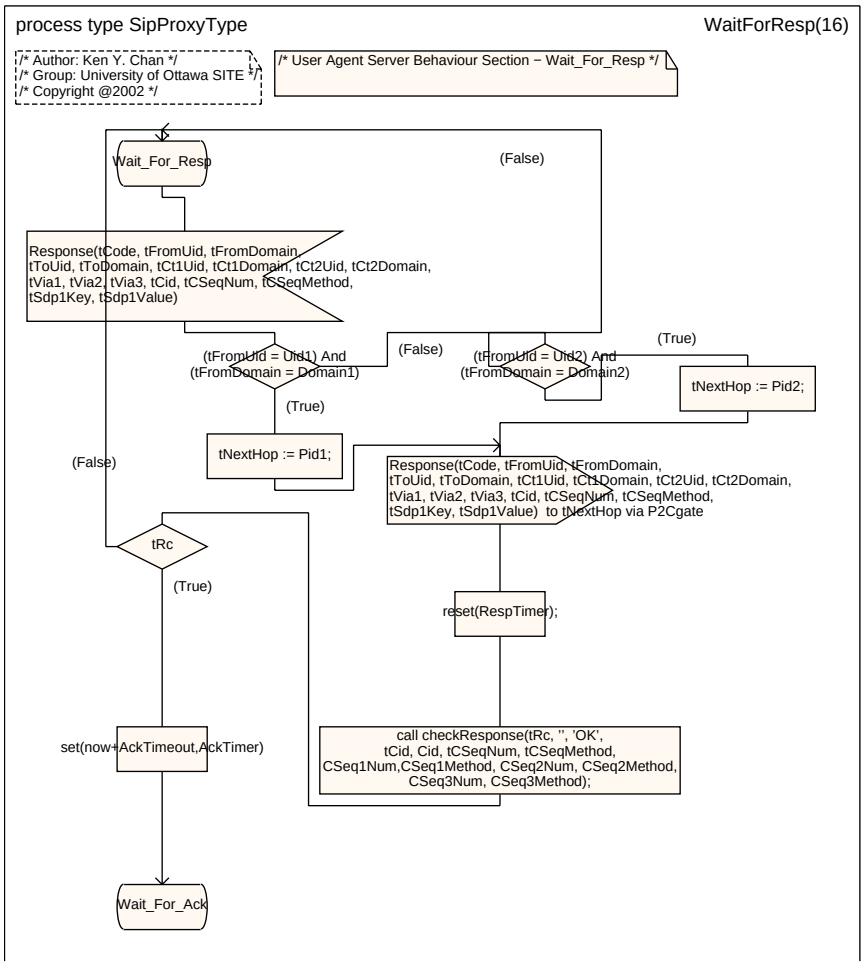

- The “Envgate” gate manages the sending and receiving of “Abstract User” signals between the user agent (UA) and the environment.
- It has:
  - an originating UA block,
  - a proxy block,
  - a terminating UA block.
- Only the originating user agent and proxy instances can send SIP requests.
- Initialize each user agent and proxy instance using ‘whoami’ and ‘id’ signals.

# SipProxyBlockType




- All blocks are initialized with one process instance.
- During the simulation, a ‘NewInstance’ “Abstract User” signal can be sent to a process instance to create a new process instance.

# SipProxyType

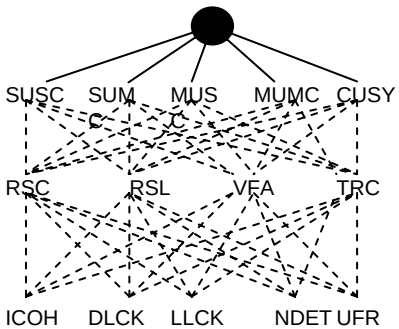


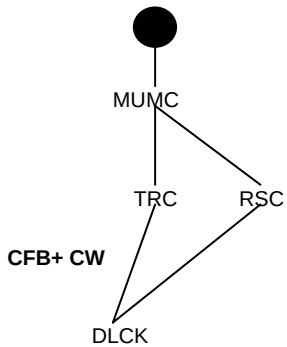

- trigger events are expressed as:
  - incoming signals;
  - pre-condition, post-conditions, constraints are expressed as:
    - enabling conditions or decision.
- A state transition occurs when:
  - an “Abstract User” signal is received from the environment,
  - a request or response message is received, or
  - a continuous signal is enabled.
- To add additional features to a process type:
  - Subtype a “basic” process type. The derived type has the same interfaces and also additional state transitions
- SDL timers and a combination of ‘*’ and ‘-‘ state symbols for error handling and response timer expirations.

# Simulation, Verification and Validation in Tau

- Tau offers bit-state, exhaustive, random walk bit state exploration and verification of MSC.
- Use “Verify MSC” option to check whether the model would be able to realize specific interaction scenarios (MSC).
- Tau may report three types of results:
  - verification of MSC,
  - violation of MSC, and
  - deadlock.
- An MSC is verified if there exists an execution path in the SDL model such that the scenario described by the MSC can be satisfied.
- If “Verify MSC” crashes, we can simulate the model to produce a matching MSC.

# Extended FI Taxonomy




- Feature Interaction Tree (FIT, on left) has three hierarchies: by nature, by cause, by effect.
- FIT is a visualization of the extended taxonomy.
- By Effect category:
  - Incoherent
  - Deadlock
  - Livelock
  - Race Condition
  - Unexpected Non-determinism
- Preventive measures are associated to each effect.

# Detecting FIs

- Specify incoherencies as MSC:
  - In case of CFB and OCS, How can we express in an MSC that user A cannot call user C?
  - If m is a scenario that should never happen, Tau can check whether this MSC m can be satisfied.
  - If the result is that m cannot be satisfied by the model, this verifies the property.
  - However, not possible to verify OCS with current versions of Tau because Tau needs the MSC to be a complete trace.
- Specify incoherencies as Observer Process Assertions:
  - The observer processes remain idle until all the observed processes have made their transitions.
  - Then, each observer process would make one transition and conditions (assertions) would be checked.
  - A violation of an assertions would stop the process -> generate a report!
  - Liveness and faireness property may be checked using counters.
- Observer Process is the more viable for FI detection with current Tau.

# Results of FI test cases

| | CW | OCS | TCS | CFB | ACB | AR |
|---|---|---|---|---|---|---|
| **CW** | - | No | No | No | No | No |
| **OCS** | No | - | No | ICH | No | No |
| **TCS** | No | No | - | No | No | No |
| **CFB** | No | ICH | No | - | No | No |
| **ACB** | No | No | No | No | - | LCK |
| **AR** | No | No | No | No | LCK | - |

- "-" means no tests for that feature pair.
- "No" indicates that one of the FI tests (livelocking, deadlocking, or incoherent) -> found no FIs for that feature pair.
- "ICH" denotes incoherent interaction
- "LCK" denotes livelocking interactions.
- Intuitively no need for all possible FI tests for all feature pairs
- We wrote test scenarios:
  - MSCs for CFB and OCS.
  - Observer Process Assertions for OCS and TCS, and AR and ACB.

# Preventing (Resolving) FIs

- We have made progress since the submission of our FI paper.
- J. Rosenberg has proposed an IETF draft which describes a caller preference extension to SIP.
- An example of a feature predicate for caller preference:

```
(& (audio=TRUE)
    (video=TRUE)
    (msgserver=TRUE)
    (automata=TRUE)
    (attendant=TRUE)
    (mobility=fixed)
    (| (methods=INVITE) (methods=BYE) (methods=OPTIONS) (methods=ACK)
      (methods=CANCEL))
    (uri-user="user")
    (uri-domain=host.example.com) )
```

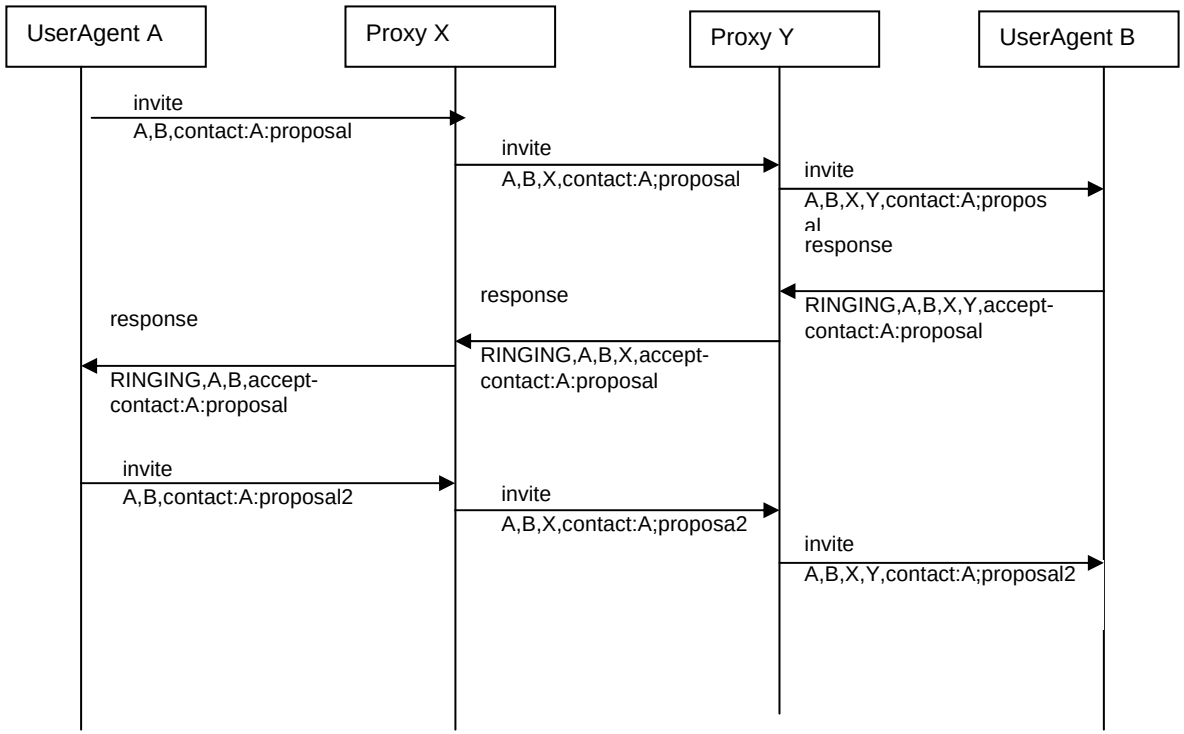
- Our feature negotiation framework for resolving feature interactions at run-time:
  - **Griffeth's Negotiating agent approach,**
  - **Gorse's logic-based formalism,**
  - **Glyne's feature set RFC 2533,**
  - **IETF draft SIP caller preferences.**

# Feature Negotiation Framework

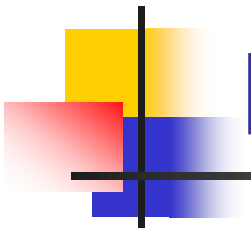
- Many researchers such as Gorse and Kamoun have described a feature as a predicate
  - feature-name ([Preconditions],[TriggerEvents],[Results]).

- Instead, we add the feature participants (user agents and proxies bound to the feature) to this predicate form, which is then used as the signature of a feature.
  - feature-name ([Participants],[Preconditions],[TriggerEvents], [Results]).

# Example of proposal and re-proposal

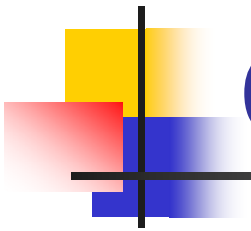



- The caller detects feature interaction(s) and re-proposes



# New FIs in SIP

- Cooperative Interactions:
  - Request Forking (RF) and Auto-Answer (voicemail)
- Adversarial Interactions:
  - Timed ACD and Timed Terminating Call Screening
  - Call Screening and Register
  - Dynamic Addressing and User Mobility and Anonymity

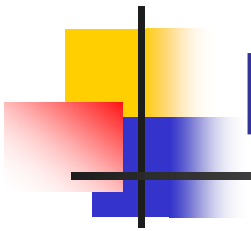


# Conclusion 1

- We believe SIP or any IETF application protocols should be specified from a user-centric perspective (e.g. Abstract User Interface).
- Feature Interaction Tree is currently a catalog of FIs. Useful for giving us the intuition on the new FIs.
- Our Feature Negotiation Framework can resolve many known feature interactions (e.g. MUMC).
- It also allows distributing the resolution decision making around.
- Our Feature Negotiation extension to SIP is compatible to caller preferences, and SIP 1.x/2.x.
- Should be compatible to all sorts of call features (e.g. mid-call, multi-user call), and web services.

# Conclusion 2

- Enhance SDL and Tau:
  - MSCs have limitations in terms of expressing quantification of instances and their behaviors → LSC??
  - Observer Process Assertions is the only viable approach for detecting FI.
  - To model SIP messages as SDL signals, we cannot easily insert, remove, search, and modify values from the optional and/or variable size header fields.
    - The SDL language could be extended with additional built-in ADTs, e.g. linked list and hash table like Java and C++.
    - String processing facilities like the *int indexOf(String substring)*
  - To incorporate model checking of the SDL system using temporal logic formula -> easier to specify distributed properties like liveness.
  - Too many crashes in Tau Validation Engine -> complex data type or model size??



# Future Works and References

- Submit an IETF draft on our Feature Negotiation extension to SIP.
- Explore properties of FIT.
- Investigate LSC for specifying FI (test scenarios).
- Perhaps modify the model to support RFC 3261.

# Acknowledgements and References


- Thanks Dr. G. v. Bochmann for his supervision, and Dr D. Amyot and L. Logrippo for their insights in FI and Telephony Service Specifications.

- More info can be found on:
  - My research web site is: http://www.geocities.com/kchan_uo
  - Or my university web site: http://www.site.uottawa.ca/~kchan
- Publications:
  - My thesis: K. Chan, “Ken Chan University of Ottawa Thesis Page”, http://beethoven.site.uottawa.ca/DSRG/PublicDocuments/REPORTS-THESES/Thesis-Chan/, accessed on April 30, 2003.


- [1] K. Chan, "Methods for Designing Internet Telephony Services with Fewer Feature Interactions", Master Thesis, University of Ottawa, Ottawa, ON, Canada, May 2003.
- [2] K. Chan, and G. v. Bochmann, "Methods for Designing IP Telephony Services with Fewer Feature Interactions", Feature Interactions in Telecommunications and Software Systems VII, IOS Press, June 2003.
- [3] K. Chan, and G. v. Bochmann, "Modeling IETF Session Initiation Protocol and its services in SDL", In Proceeding of Eleventh SDL Forum, LNCS, Springer-Verlag Heidelberg, Stuttgart, Germany, July 1-4, 2003.