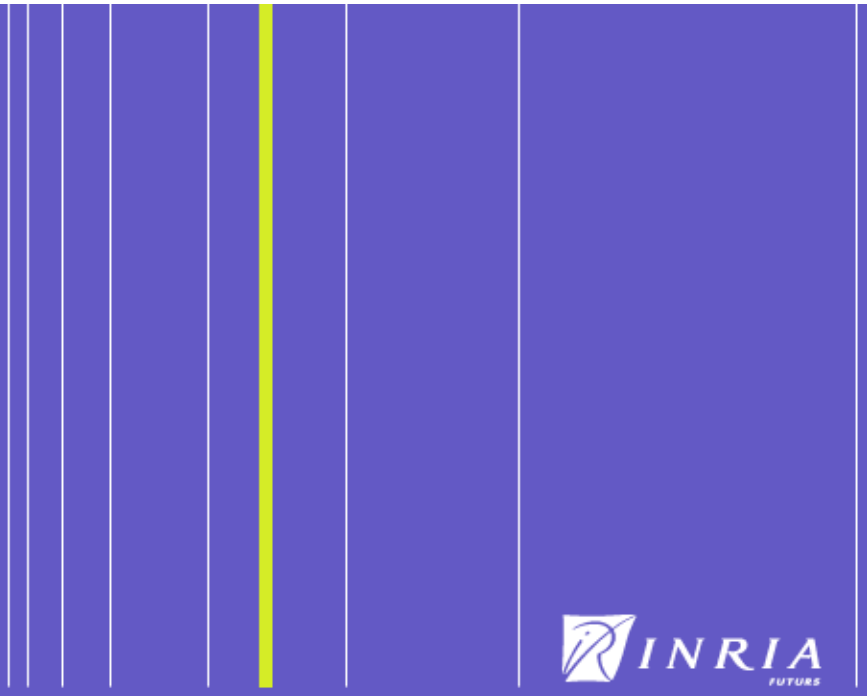
Language Technology for Internet Telephony Service Creation

Laurent Burgy[*], Charles Consel[*], Fabien Latry[*], Julia Lawall[†], **Nicolas Palix[*]**, Laurent Réveillère[*]

* Department of Telecommunications
LaBRI – INRIA, France
{burgy,consel,latry,palix,reveillere}@labri.fr

† DIKU
University of Copenhagen, Denmark
julia@diku.dk



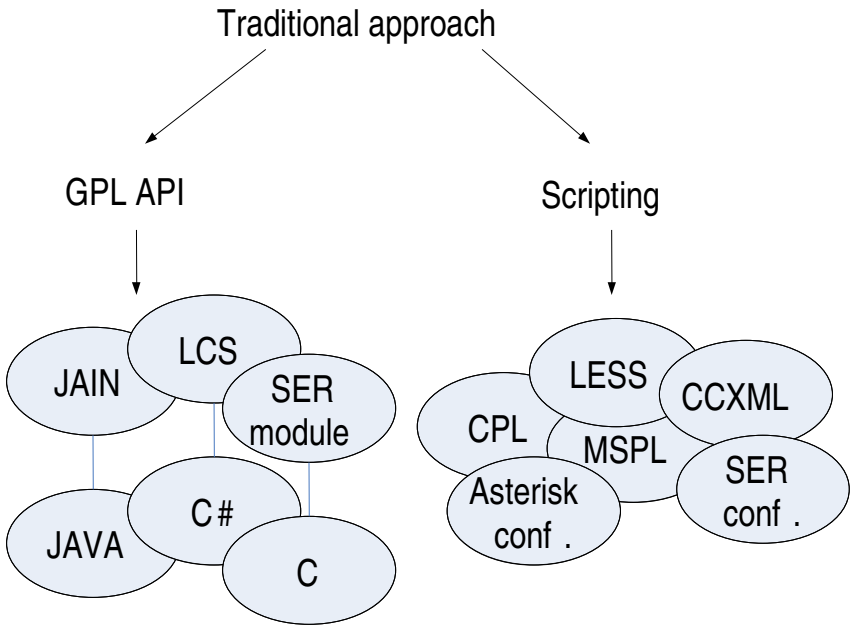
Context

- Rapidly evolving IP telephony
- Implicit CTI (Computer Telephony Integration)
- New functionalities
  - Address book
  - Calendar
  - Email
  - Databases
  - Web services

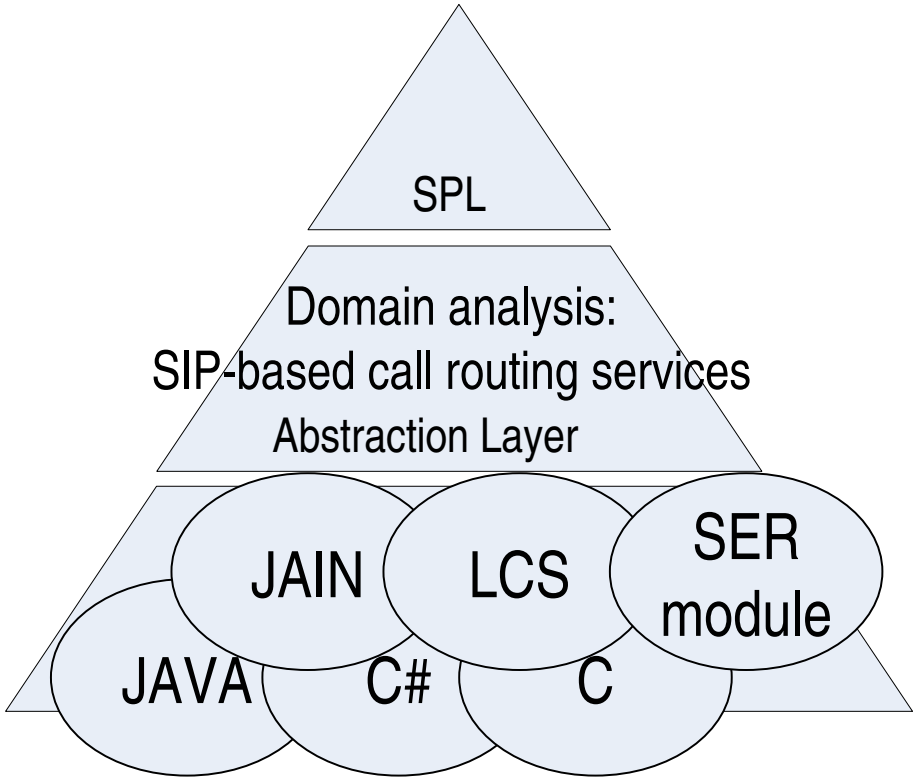
How to incorporate these functionalities into *robust* services ?

Traditional approach

- Extensive knowledge required
- Multiple languages
- Multiple protocols
- Large and complex API



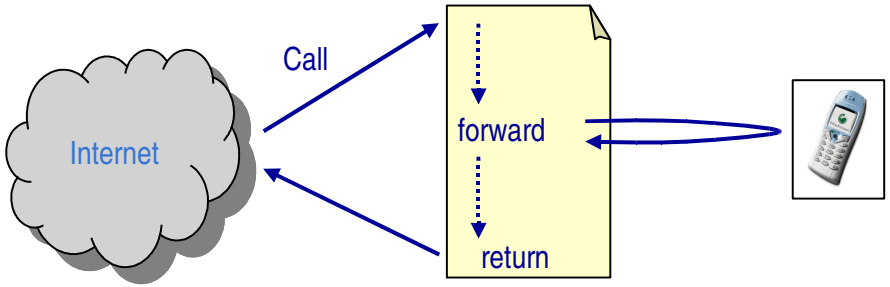
Our Approach



Session Processing Language - SPL

- Event handlers and signaling operations
- Session
- Hierarchical sessions
  - Service
  - Registration
  - Dialog
- Inter-event control flow

Event Handlers and Signaling Operations



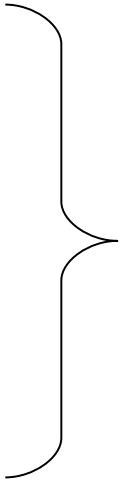
SPL Service

```
response incoming INVITE() {
    […]
    response resp = forward;
    if (resp == /ERROR) {
     resp = forward 'sip:phoenix.secretary@inria.fr';
    }
    return resp;
}
```

```
// Deny Service
response incoming INVITE() {
    return /ERROR/CLIENT/BUSY_HERE;
}
```

Session

- INVITE
- BYE
- REINVITE
- …



Dialog = ID

uri caller;
time start;
…

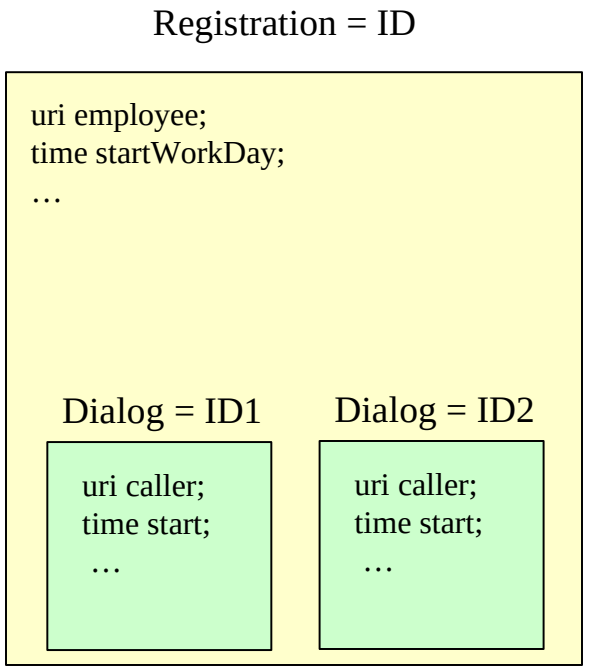
Session: The Dialog Session Example

```
dialog {
  uri caller;
  time start;

  response incoming INVITE() {
    caller = FROM;
    return forward;
  }

  void incoming ACK(){
    if(caller == 'sip:my.wife@home.fr')
      log("Personal call");
    start = getTime();
  }

  response BYE() {
    string duration = time_to_string(getTime() - start);
    log("Call: "+ duration +" "+uri_to_string(caller));
    return forward;
  }
}
```

Hierarchical Sessions

- REGISTER
- REREGISTER
- *unregister*

- INVITE
- BYE
- REINVITE
- …

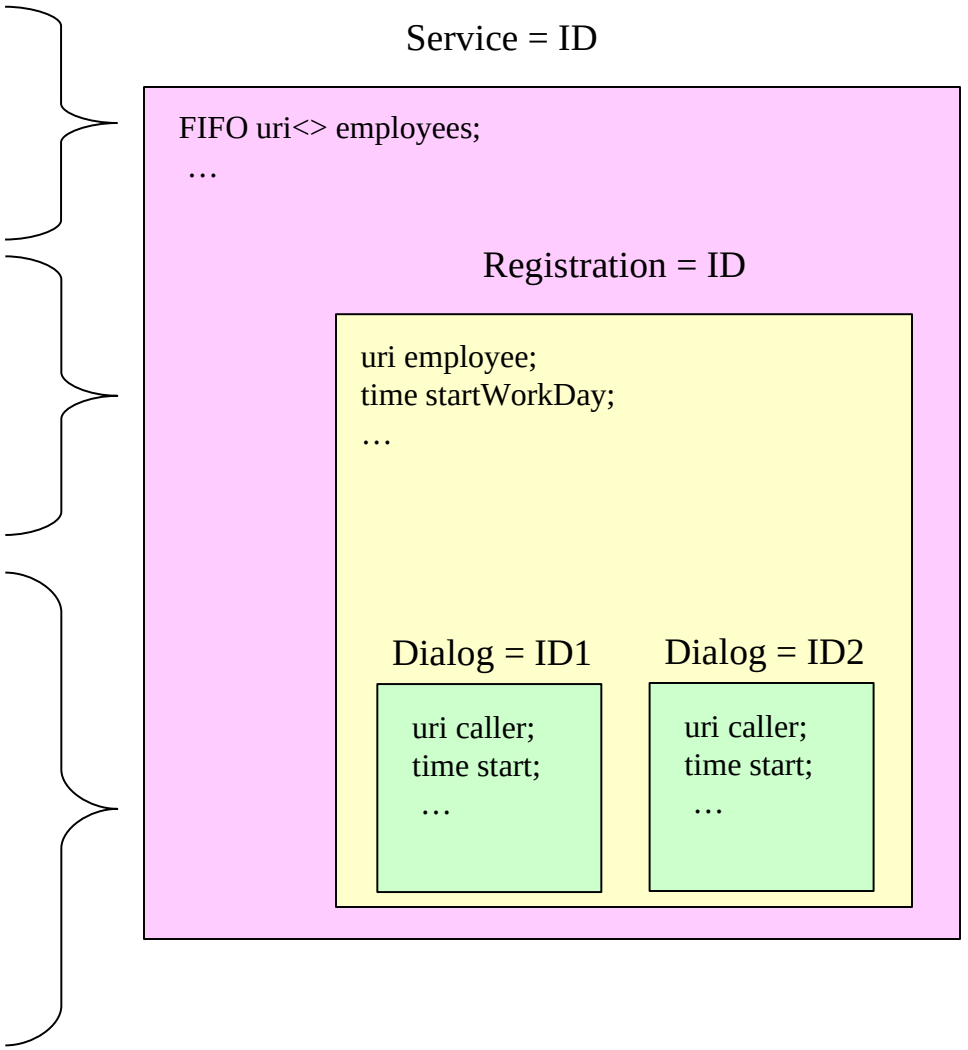


Hierarchical Sessions: Example

```
registration {
  uri employee;
  time startWorkDay;

  response outgoing REGISTER() {
    startWorkDay = getTime();
    employee = FROM;
    return forward;
  }

  void unregister() {
    string duration = time_to_string(getTime() - startWorkDay);
    log("WorkDay: "+ duration +" "+uri_to_string(employee));
    return;
  }

  dialog {
    uri caller; time start;

    ...
  }
}
```

Hierarchical Sessions

- *deploy*
- *undeploy*
  - REGISTER
  - REREGISTER
  - *unregister*
    - INVITE
    - BYE
    - REINVITE
    - …

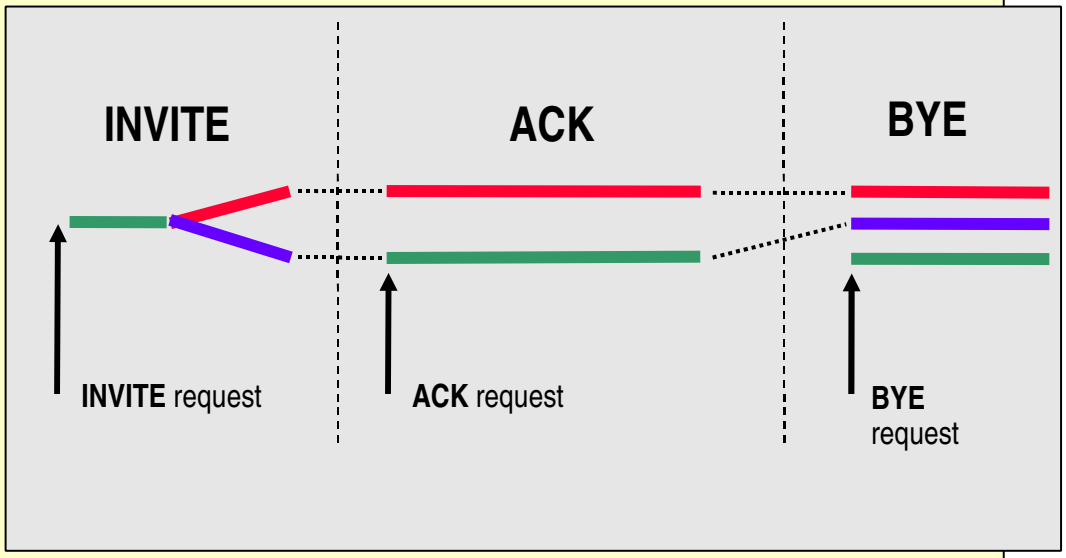


Hierarchical Sessions: Example (cont'd)

```
service hotline {
  ...
  processing {
    uri<100> employees = <>;

    void deploy() {...}
    void undeploy() {...}

    registration {...

      response outgoing REGISTER() {
        startWorkDay = getTime();
        employee = FROM;
        push employees employee;
        return forward;
      }
      ...

      dialog { ...
        response incoming INVITE() {
          return forward employees;
        }
      }
    }
  }
}
```

Inter-Event Control Flow

```
dialog {
  response incoming INVITE() {
    response r;
    ...
    if (...) {
      ...
      return r branch hotline;
    }
    else {
      ...
      return r branch personal;
    }
  }

  void incoming ACK(){
    branch hotline {... }
    branch default {... }
  }

  response BYE() {
    branch hotline  {... }
    branch personal {... }
    branch default  {... }
  }
}
```



Conclusion

- Usable, expressive, concise and safe language
  - A queuing service in about 100 lines
  - Guaranteed safety properties
    - Appropriate signaling action
    - Redirections
    - Limited access to headers
    - Inter-event control flow reachability
- Domain-Specific Languages approach
- Portability
- Application Server
  - Interpreted services (done)
  - Compiled services (in progress)

Future Work

- Raising SPL services as User Agents
- Visual programming
  - For end-users and non-programmers
- Feature Interactions between services
  - Multiple users
  - Multiple services
- Raising the abstraction level of SPL beyond SIP

Thank You For Your Attention !

Questions ?

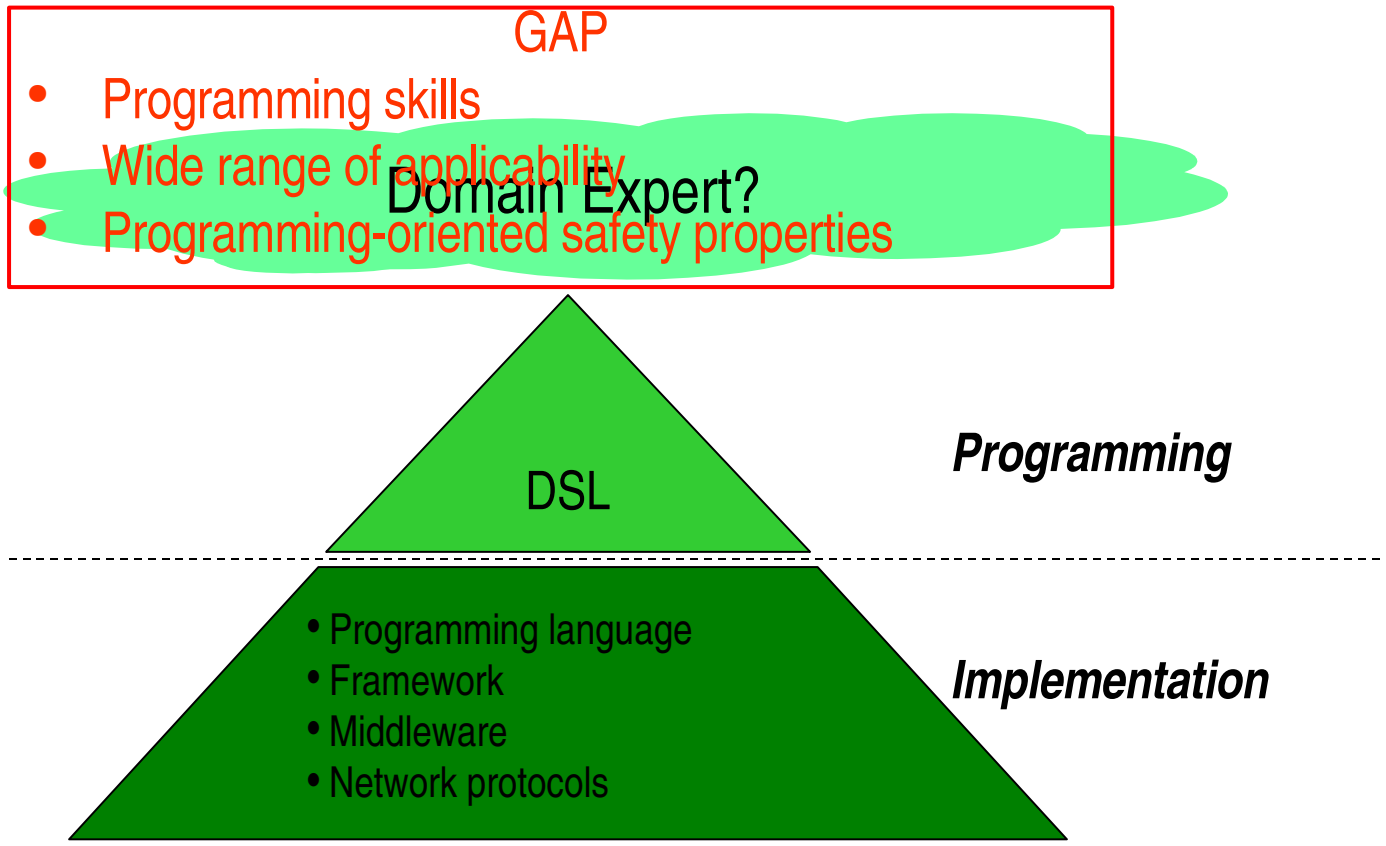
SPL

```
service example {
 processing {
  dialog {
   response incoming INVITE() {
     response r =
       forward 'sip:bob@phone.example.com';
     if (r == /ERROR/CLIENT/BUSY_HERE)
       return
         forward 'sip:bob@voicemail.example.com';
     else
      if (r == /ERROR) {
        if (FROM == 'sip:boss@example.com')
          return forward 'tel:+19175554242';
        return r;
      }
   }
  }
 }
}
```

JAIN SIP

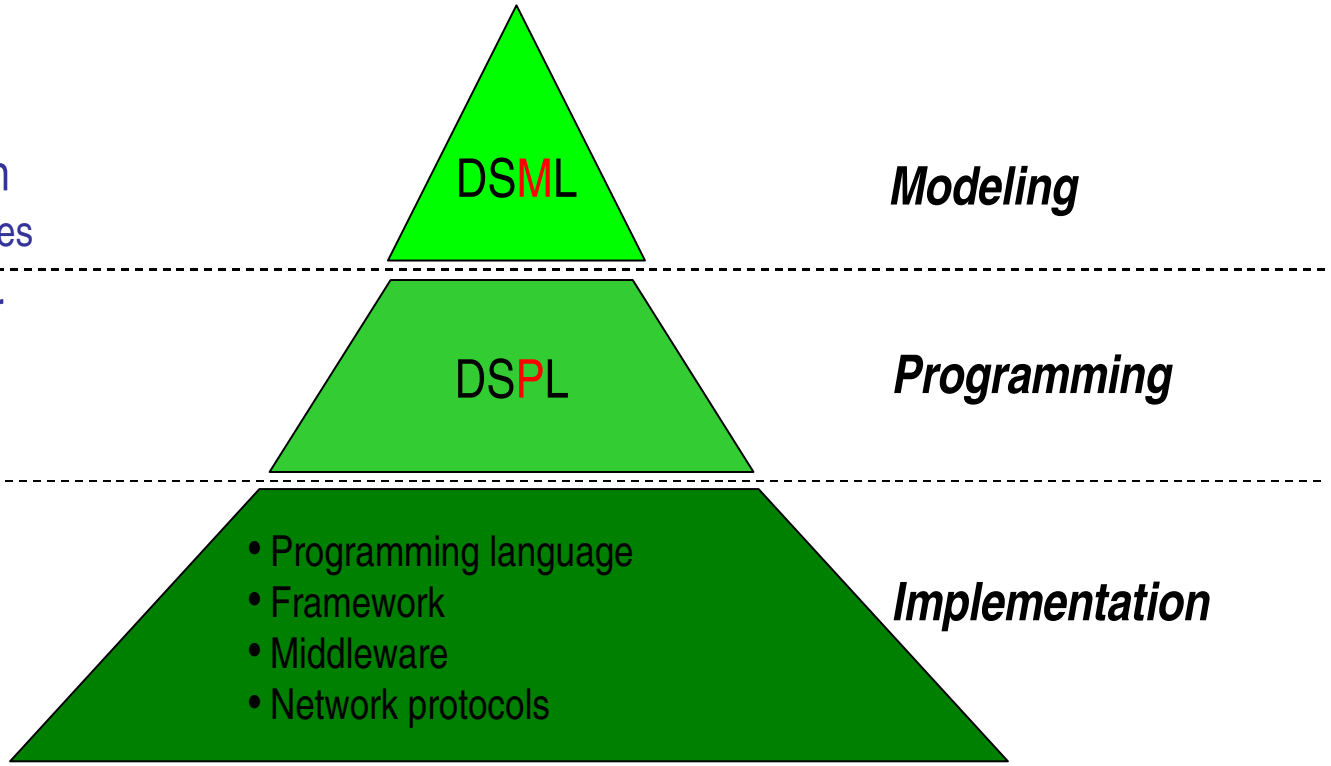
```
public class Example implements SipListener {
 [...]
 private AddressFactory factory = getAddressFactory();

 public void processRequest (RequestEvent requestEvent) {
  Request rq_request = requestEvent.getRequest();
  SipProvider rq_sipProvider = (SipProvider) requestEvent.getSource();
  String method = rq_request.getMethod();
  [...]
  if (method.equals (Request.INVITE)) {
   SipURI uri = factory.createSipURI ("bob", "phone.example.com");
   rq_request.setRequestURI (uri);
   ClientTransaction ct = rq_sipProvider.getNewClientTransaction(rq);
   ct.sendRequest (rq_request);
  ... }

 public void processResponse (ResponseEvent responseEvent) {
  ClientTransaction rs_ct = responseEvent.getClientTransaction();
  if (rs_ct != null) {
   Request rs_request = rs_ct.getRequest();
   Response rs_response = responseEvent.getResponse();
   SipProvider rs_sipProvider = (SipProvider) responseEvent.getSource();
   String method = rs_request.getMethod();
   rs_responseCode = rs_response.getStatusCode();
   if (method.equals (Request.INVITE)) {
    if (rs_responseCode == 486) {
     SipURI uri = factory.createSipURI ("bob", "voicemail.example.com");
     rs_request.setRequestURI (uri);
     rs_sipProvider.sendRequest (rs_request);
    } else if (rs_responseCode >= 300) {
     if (rs_request.getHeader("FROM").equals("sip:boss@example.com ")) {
      TelURL tel = factory.createTelURL ("tel:+19175554242");
      rs_request.setRequestURI (tel);
      rs_sipProvider.sendRequest (rs_request);
     } else {
      rs_sipProvider.sendResponse (rs_response);
     } ... }
```

Abstraction Layers



A Layered Domain-Specific Language Approach

- Defining a solution
  - High level
  - Simple
- Verifying a solution
  - Domain properties
- High-level tools for
  - Compilation
  - Verification



A Layered Domain-Specific Language Approach

- Defining a solution
  - Visual
  - Activity diagram
- Verifying a solution
  - Domain properties
- Compilation
  - SPL
  - TLA formulas