Domain-Specific Software Architectures

- ARPA program advocated investment in infrastructure
- Software Architecture is the key to programming in the large:
  - It forms a contract between the component developers
  - "Reference Architecture" is the basis for refinement of individual components
  - When formalized, forms the specification for a composition mechanism
Three Levels of Environments

Architecture-Based Compilation

- To take advantage of the formal architecture specification:
  - Apply global compiler optimizations
  - At system linkage time
  - Using information from the specification
  - Instead of expensive analysis techniques
Problem with today's compile/compose paradigm

- Repeated development of similar components
  - premature binding of variable portions
  - inexpressive
  - hard to parameterize
- Extra layers of abstraction inefficient
  - function call / method dispatch layers
  - mediators and wrappers
  - copy incessantly
  - convert between representations
Solution

- Use global program optimization techniques
  - Currently applied piecemeal to individual components
  - Want to delay until more information is known
  - Compile at composition time
- Delay commitments until composition time
  - "Decontextualized components"
  - Basic blocks or statements
- Exploit architecture specification
  - Derive annotations
    - flow analysis
    - communication paths
  - Allow application engineer to attach annotations

Information Now
Available to ABC

- Historically, compilers are “opening up”
  - Linking
  - Debugging
  - Separate compilation
- Added to components
  - Symbol table information
  - Signature and typing information
  - Type and function call class information
  - Analysis information
  - Annotation information
New Information Used by ABC

- Abstracted (decontextualized) components
- Formal representation of architecture in ADL
  - correspondence between box and component
  - correspondence between arc and communication or control mechanism
  - component clustering information
  - interface details
  - process and platform distribution
  - analysis information derived from architecture description
  - architecture style
  - explicit annotations of architecture description by application engineer

Experience

- Need to characterize enabling analytical information
- Widened the notion of component to "decontextualized component"
  - Exception handling
  - Output
  - Input
- Instructive examples
  - SAR radar data processing
  - Simulated token ring
  - Automated teller
- Experience with ADLs
  - Initially, wanted to adopt "existing" ADL
  - Instead, contributed to ACME design and development
Progress

- "Decontextualized components"
  ==> "Embryonic programs"
- Architecture-based program generation
- Requirements imposed on ADLs by ABC's concerns

Teller Reference Architecture
Pocket Teller

Enable Signal
Continuous: 0

Teller's Assistant

T1:Teller
DAT

T2:Teller
DAT

T3:Teller
DAT

Shared Account Transactions

Account Database
TI: ATM

Account

Transactions

Trigger Activity

Possible Exceptions

- Invalid account number
- Invalid transaction type
- Overdrawn
- Unlikely deposit/withdrawal amount
- Approaching/Dropping below balance minimum
- Timeout
Responses

<table>
<thead>
<tr>
<th>Pocket Calculator</th>
<th>Teller's Assistant</th>
<th>ATM</th>
</tr>
</thead>
<tbody>
<tr>
<td>acct</td>
<td>Impossible</td>
<td>Eject card, Restart</td>
</tr>
<tr>
<td>trans</td>
<td>Use balance</td>
<td>Reobtain</td>
</tr>
<tr>
<td>over</td>
<td>Confirm/Reobtain</td>
<td>Confirm/Reobtain</td>
</tr>
<tr>
<td>? S</td>
<td>Confirm/Reobtain</td>
<td>Eject card, Restart</td>
</tr>
<tr>
<td>-&gt; X</td>
<td>Confirm/Reobtain</td>
<td>Confirm/Reobtain</td>
</tr>
<tr>
<td>time</td>
<td>Impossible</td>
<td>Retry x 5, Restart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Retry x 5, Restart</td>
</tr>
</tbody>
</table>

Teller Program

```c
#include account_transactions
void teller() {
    dollar_amount balance;
    obtain account as SS_number_format
    (obtain transaction as string
     (exception illegal_banking_transaction
      unless (transaction = "balance") II
      (transaction = "deposit") II
      (transaction = "withdraw")
      default "balance"
      report "Transaction must be one of: balance, deposit, withdraw."
    )
    if (transaction = "balance")
        balance = account_transactions.balance(account);
    else (obtain amount as dollar_amount
        if (transaction = "deposit")
            balance = account_transactions.deposit(account, amount);
        else balance = account_transactions.withdraw(account, amount);
        provide balance)))
```
Environment Activities

Architecture-Based Generation
ABC's requirements of an ADL

- Style must be a first class object
- No deviations from style
  - within a given view
  - multiple views constitute the architecture
- Styles widened to include implicit meanings
  - e.g. exceptions are ubiquitous: should not need to portray
    them explicitly in the architecture
  - similarly, communication with environment
- ADLs expanded to allow more program specification scaffolding constructs

ACME:
an Architecture Exchange Language

David Garlan (CMU)
Robert Monroe (CMU)
David Wile (ISI)