Validating Software Integration With Coordination Analysis

Universidade do Minho
DI-CCTC
Braga, Portugal

CIC'09 – Braga, Portugal – 7 May 2009

Nuno F. Rodrigues – nfr@di.uminho.pt

Agenda

- The Software Integration Problem
- Coordination Analysis Process Overview
- Coordination Pattern Representation
- Coordination Pattern Discovery
- Conclusions and Future Work

Software Integration Problem

- More than 30% of all IT investment is spend on integrating existing software application
  - Company Fusion/Acquisition
  - *Webization* of companies
  - Internationalization
  - Stock reduction…
- Most of the problems found in IT systems reside on the **glue code** and not on the systems being integrated

Software Integration Problem

- **Glue Code**
  - Multithreaded
    - In order to keep a low impact on the integrating systems
  - Has to deal with many more exceptional situations
    - Network problems
    - Misbehaviour of integrating systems
    - Inconsistent data
  - Many special cases have to be encoded on this glue code
  - Its “easy” to get something wrong, and usually with great impact over the integrating systems

What is this Glue Code?

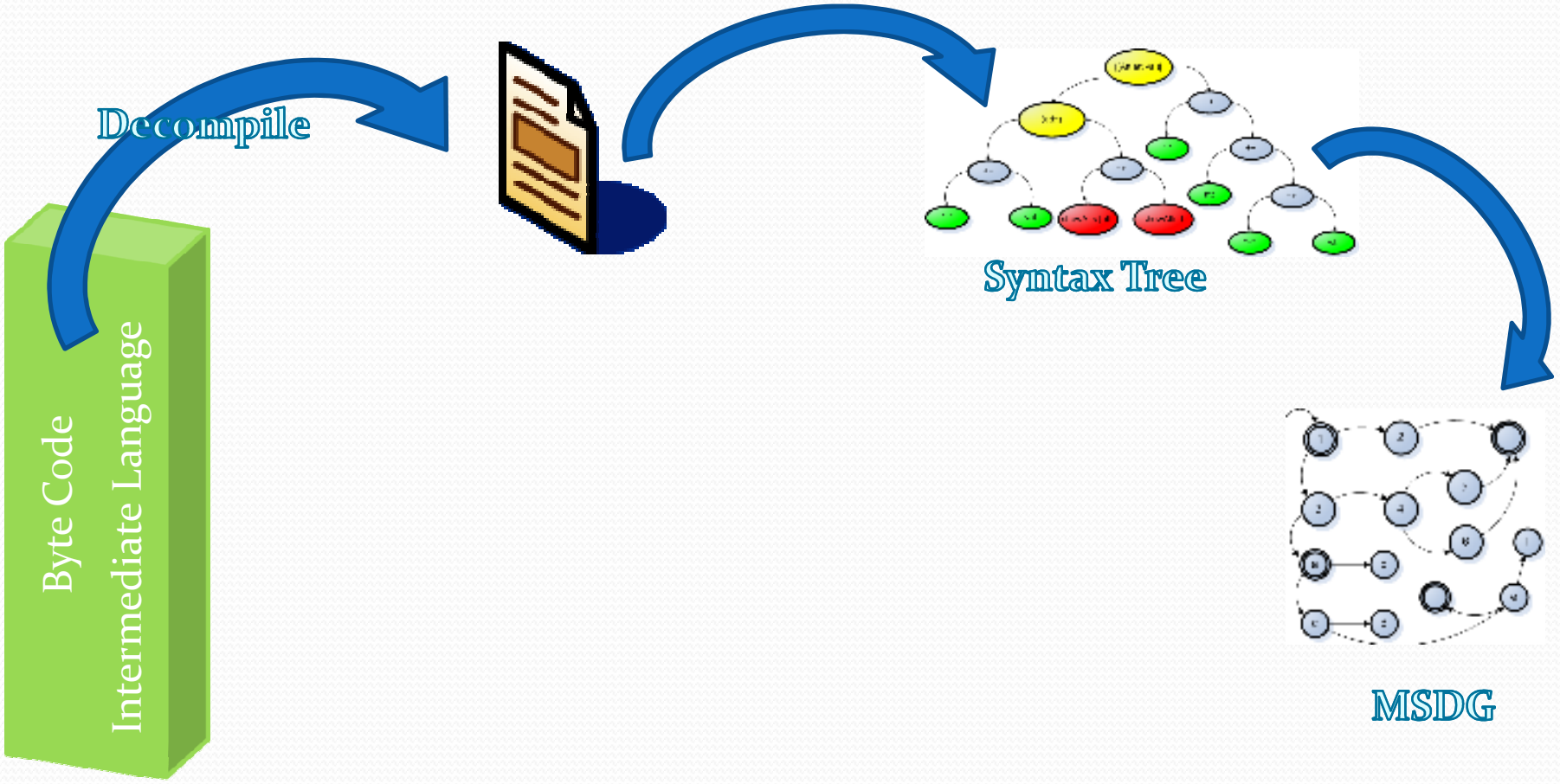
- What are the building blocks of coordination?
  - Communication Primitives
    - Web Services
    - CORBA
    - RMI
    - .net Remoting
  - Programming Logic
    - Multithreading
    - Control flow statements
    - Exception handling
  - CP+ PL = System Coordination

Coordination in Practice

- Communication Primitives
  - Synchronous
  - Asynchronous
- But we also need to keep the code that regulates the execution of the communication primitive calls

```
flights findFlight(string dest) {
        if(dest.equals("Amsterdam"))
                return KlmReservationWebService("AMS");
        else
                return TapReservationWebService(dest);
}
```

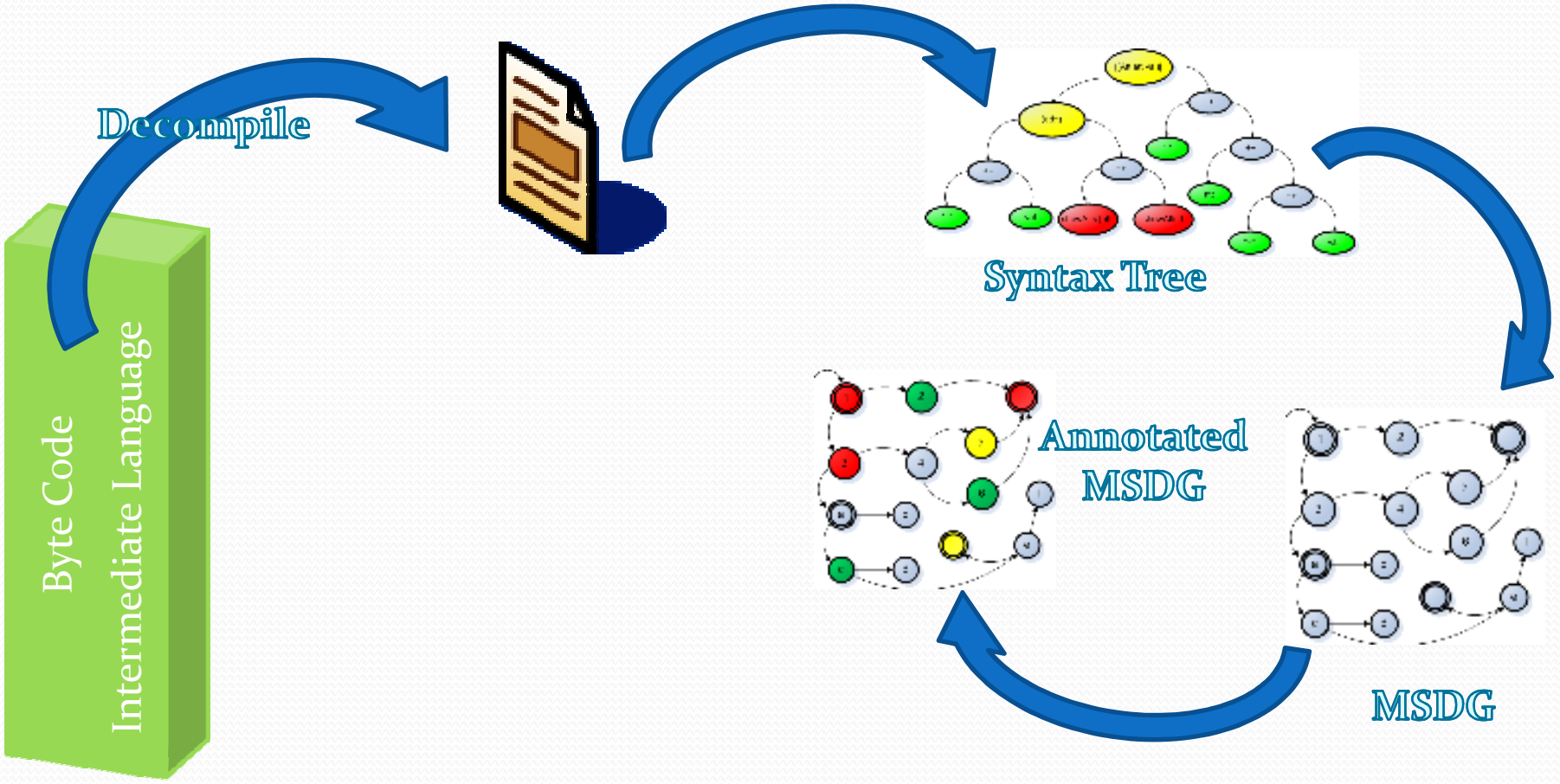
How do we extract coordination?



Syntax Tree -> MSDG

- Create a node for each statement
- Create flow edges between control flow dependent statements
- Create data dependency
  - Edges for data dependent statements (*def* and *use* variables)
- Control Dependencies
- Spatial Dependencies
- Triangular vertices for forks and joins
- Interference Dependencies
- If statement contains function call, then create the MSDG for the called function and connect it to the analyzed statement, creating new nodes for the actual parameters and return values
  - Heap structure for keeping track of the already visited functions

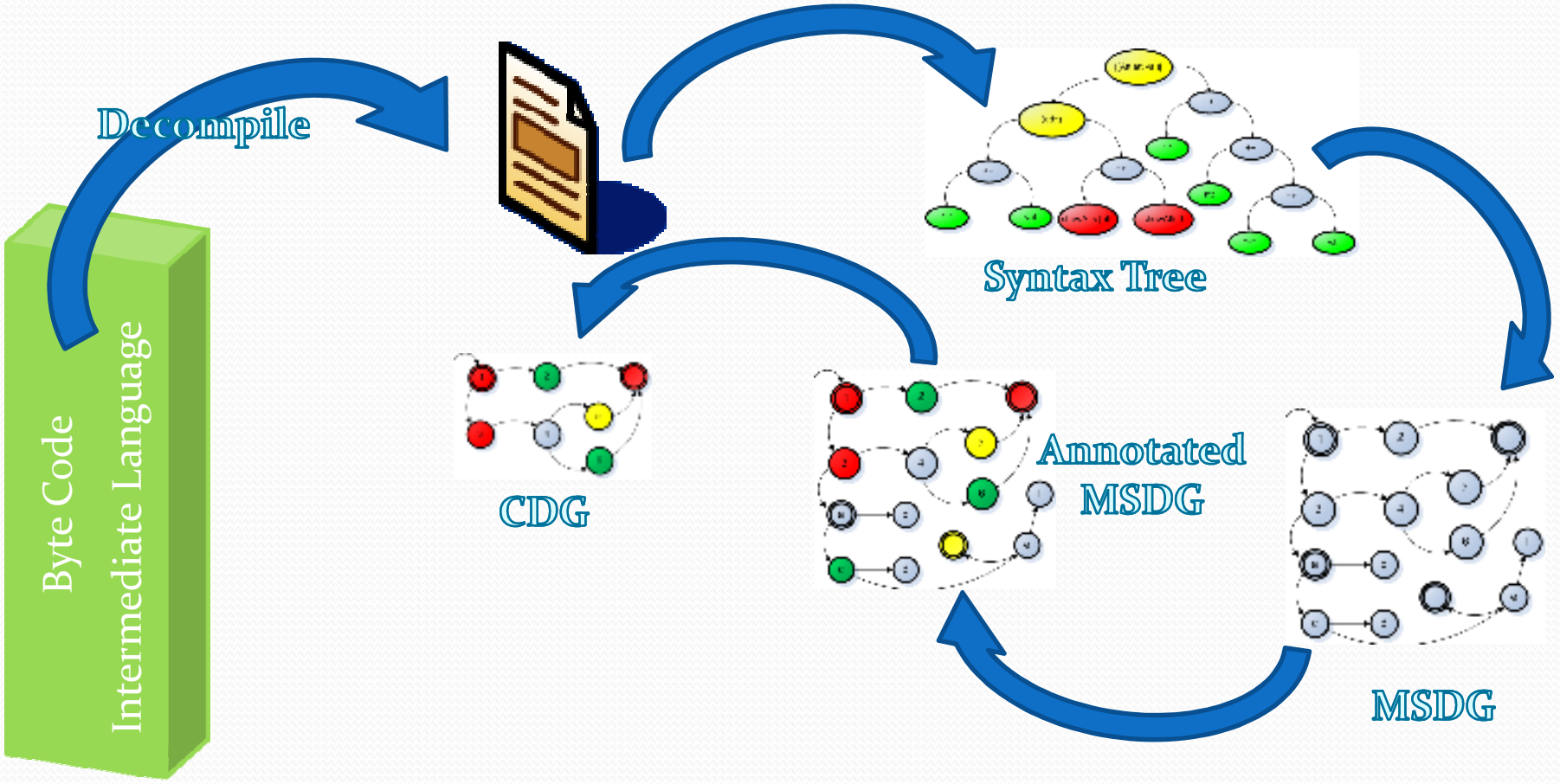
How can we look it for?



MDG -> Annotated MSDG

- Parametric Annotations (based on regular expressions)
  - Annotate Direct Web Service Calls (SOAP)
    - Synchronous
    - Asynchronous
  - Annotate COM and CORBA object calls
    - Synchronous
    - Asynchronous
  - Annotate Object Remoting calls (RMI, .net Remoting)
    - Synchronous
    - Asynchronous
  - Annotate Inter Thread Calls
    - Synchronous
    - Asynchronous
  - …

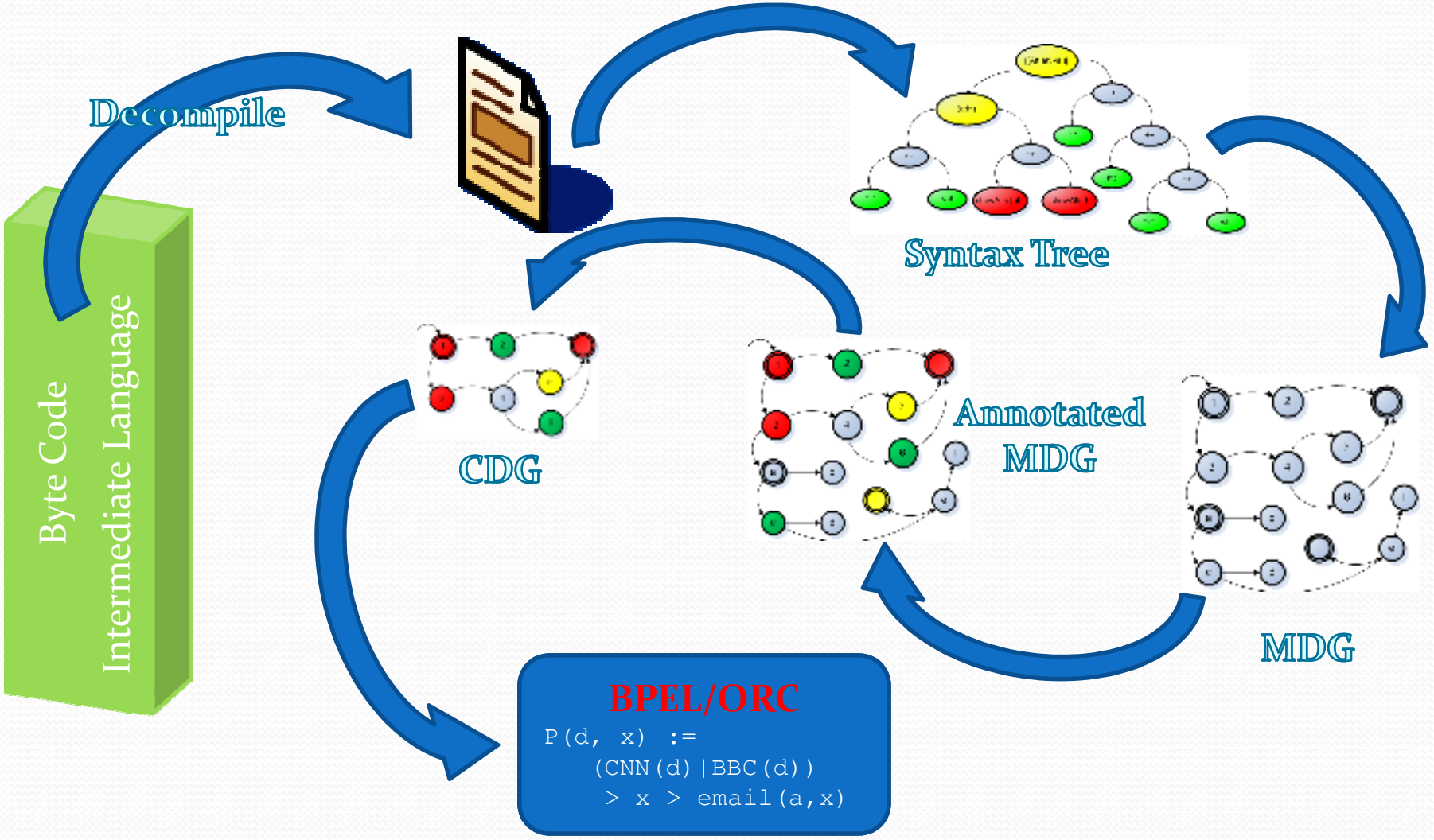
How can we look it for?



Annotated MDG -> CDG

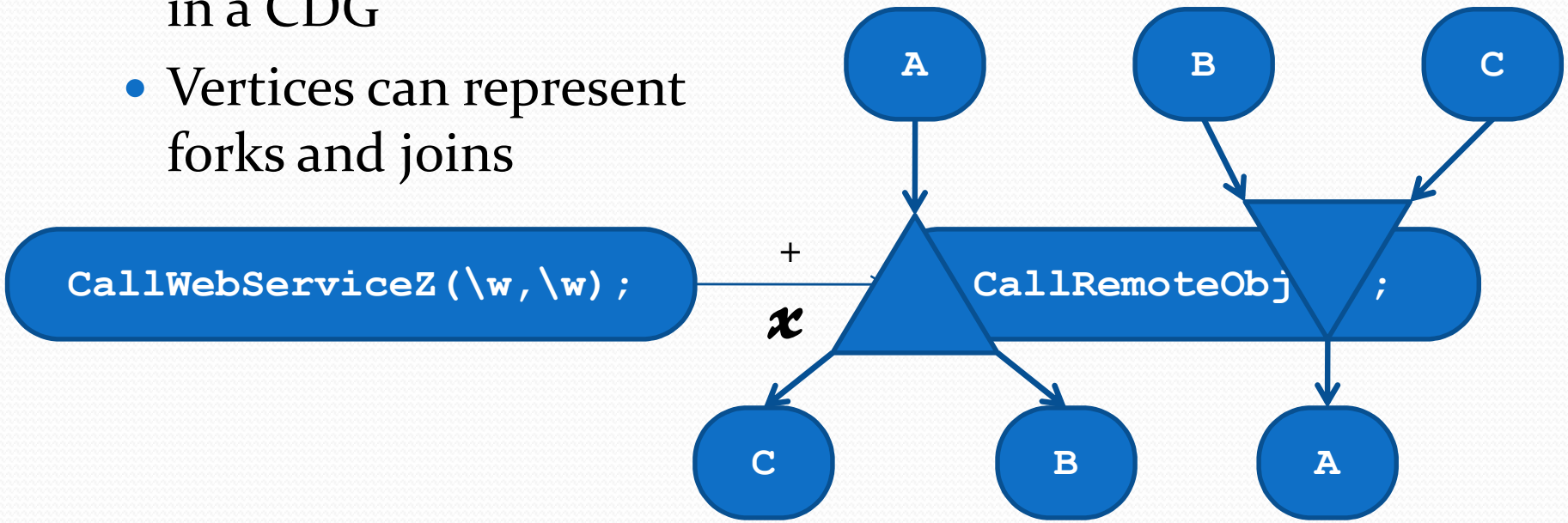
- Remove all nodes that are not annotated except:
  - 1) Nodes in the union of the backward slice of any annotated node
    - The backward slice is calculated based on the MSDG
      - Data Dependencies
      - Control Dependencies
      - Spatial Dependencies
      - Interference Dependencies

How can we look it for?



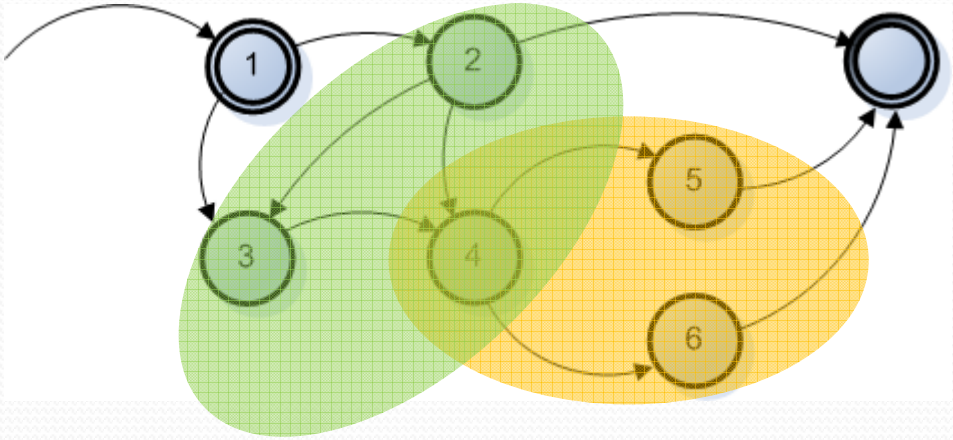
A Coordination Pattern (Implementation) Language

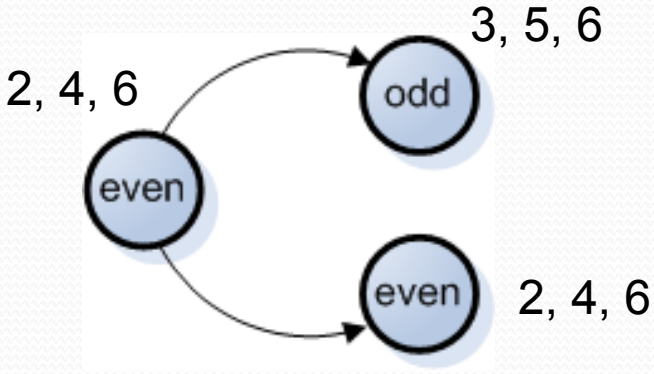
- A graph query language where
  - Vertices contain regular expressions
  - Edges are labelled with the thread Id's
  - Edges define how many sequential edges they represent in a CDG
  - Vertices can represent forks and joins



Graph “Isomorphism's” Algorithm

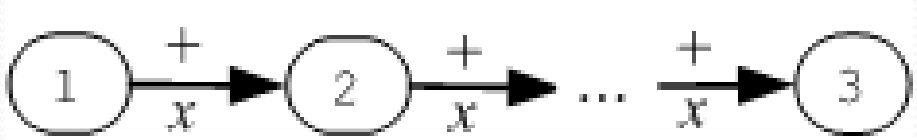
- Calculate the candidates for each vertex
- Calculate the candidates for each edge
- Try to build every combination with the discovered candidates
  - Taking into consideration already visited candidates
  - Loops



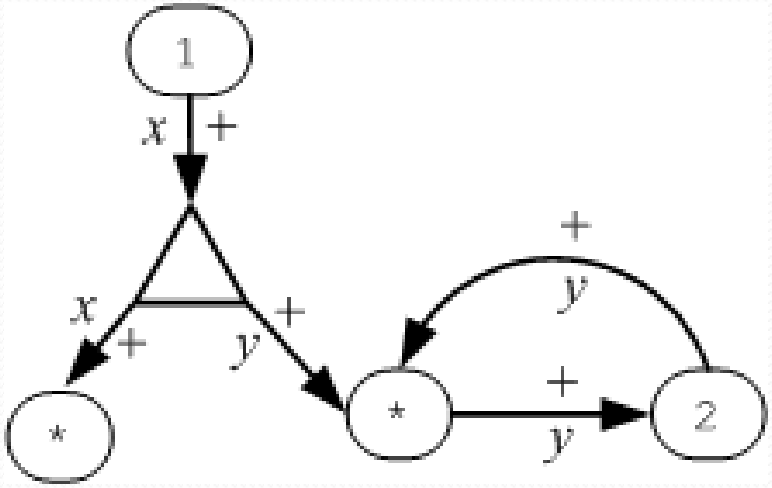


Coordination Patterns

- Identify implementations of specific coordination patterns

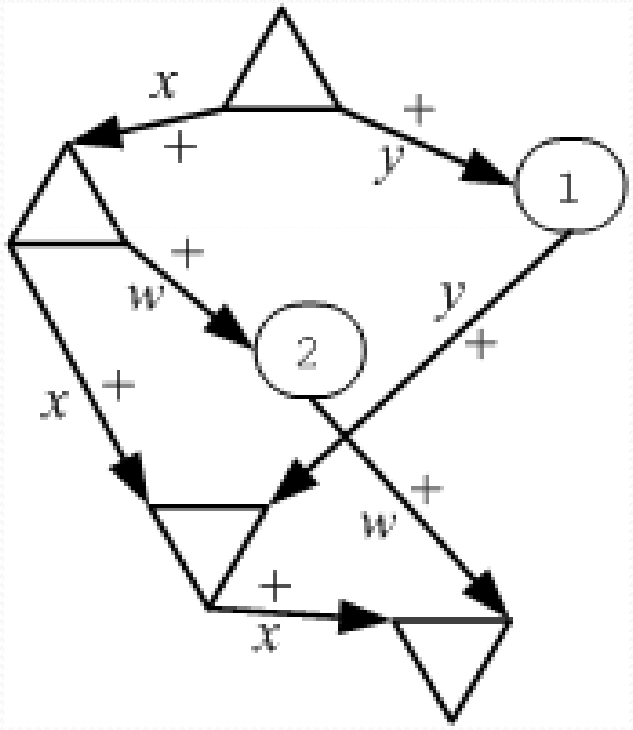


Sequential Query Pattern



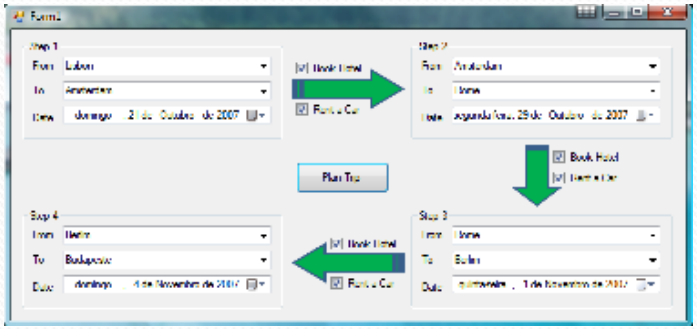
Asynchronous Query Pattern with Client Multithreading

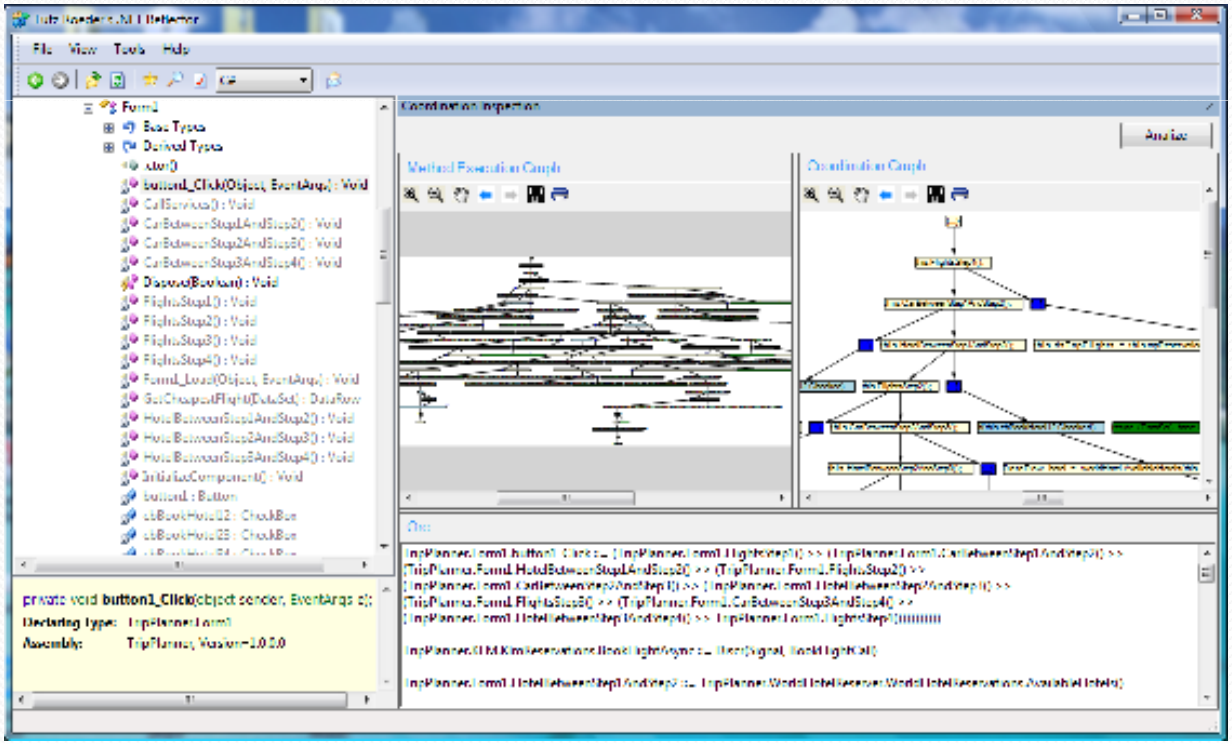
Coordination Patterns



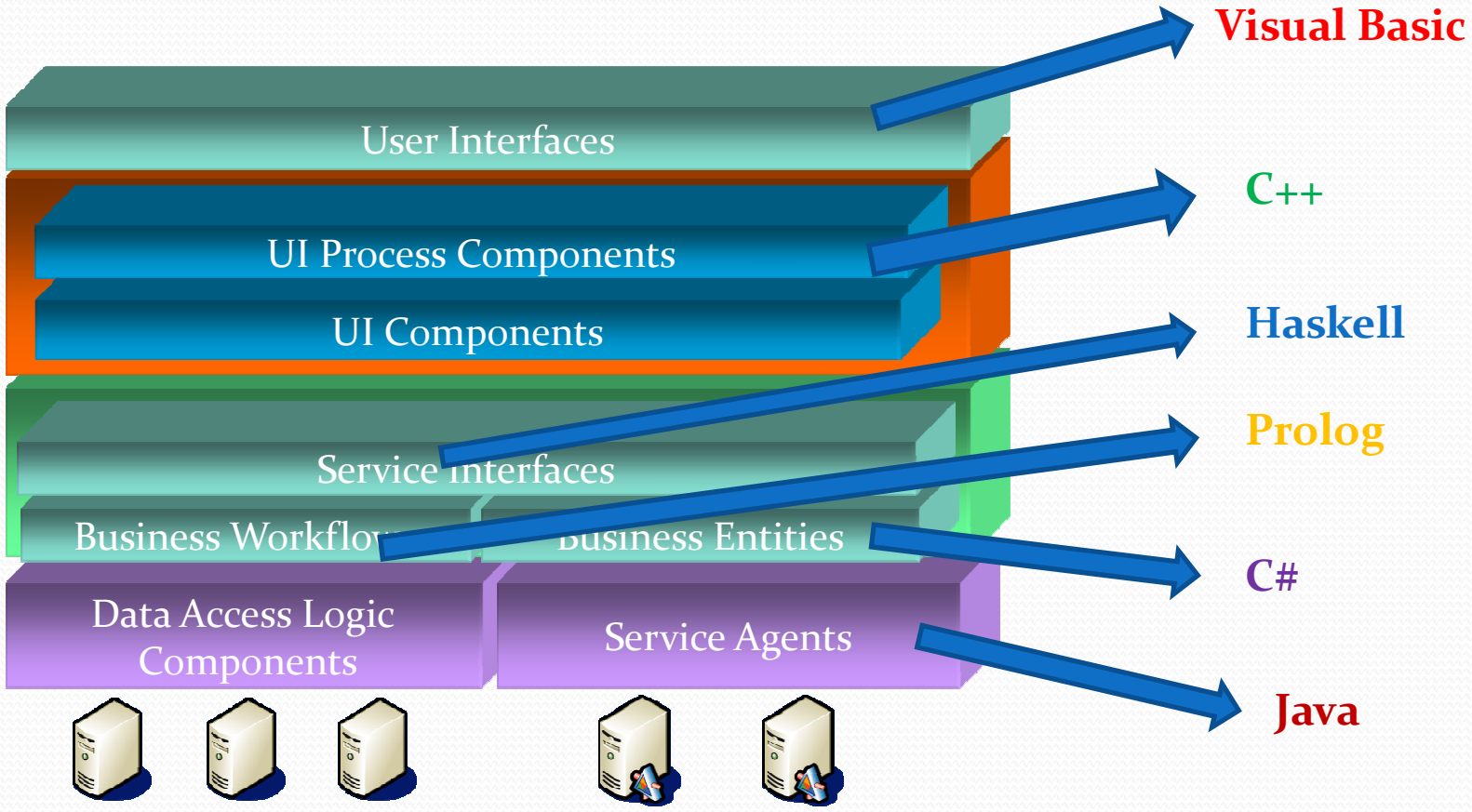
Joined Asynchronous Query Pattern

CoordInspector

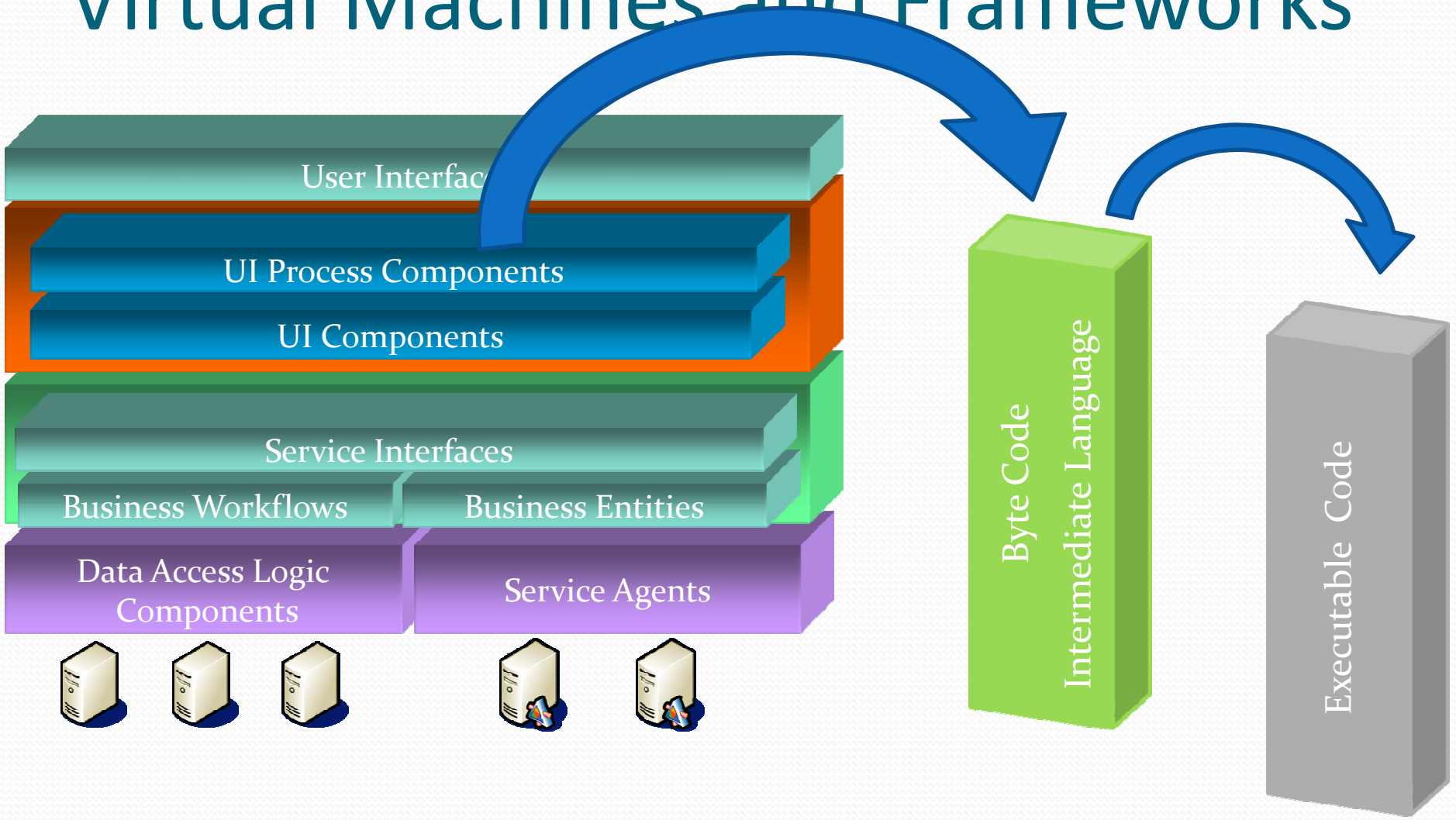




Where to look it for?



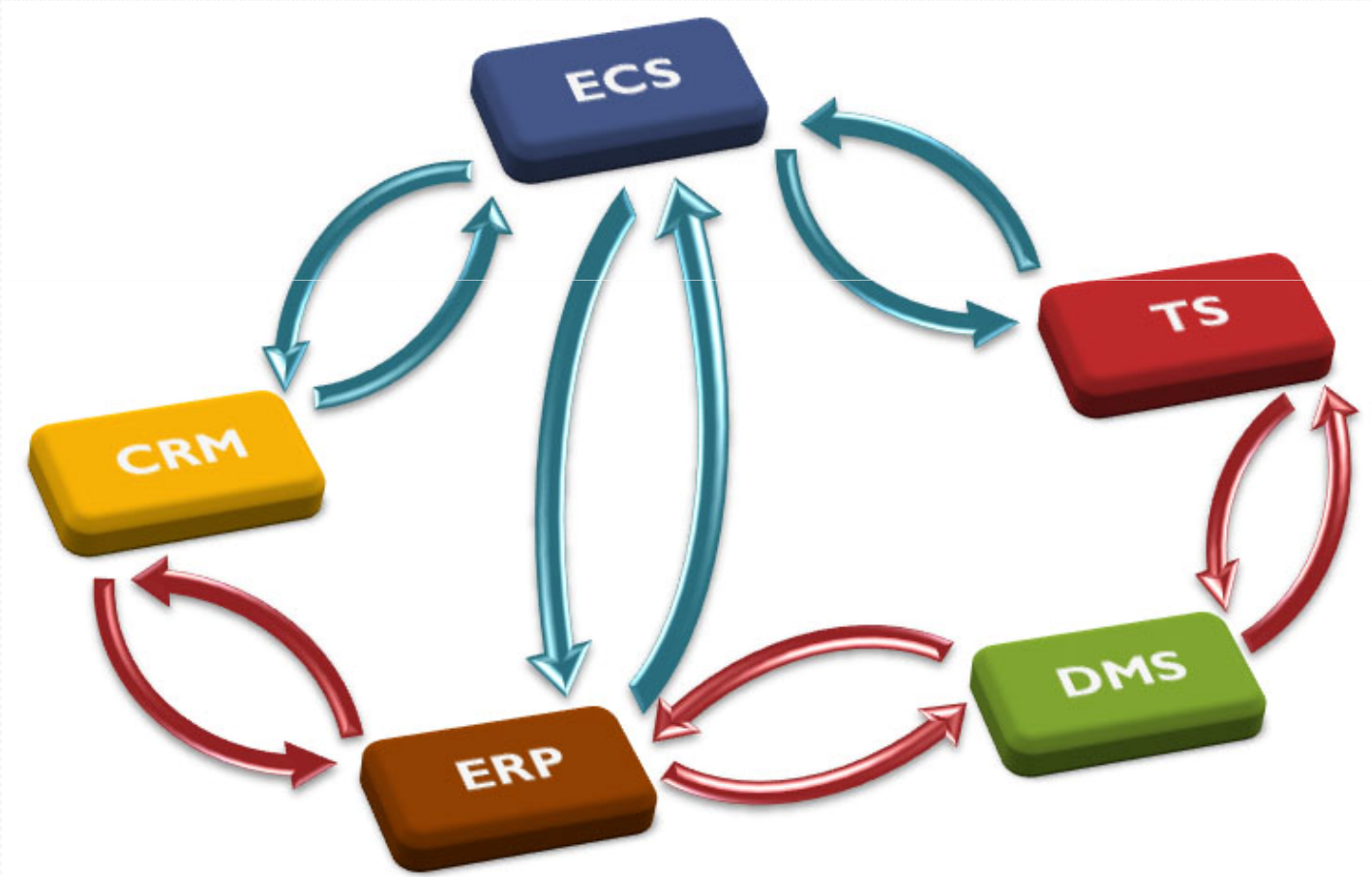
Virtual Machines and Frameworks



Case Study

- Integration of 4 applications
  - CRM
  - ERP
  - Web Portal
  - Training Management Software (TS)
  - Document Management System (DMS)

Integration Architecture



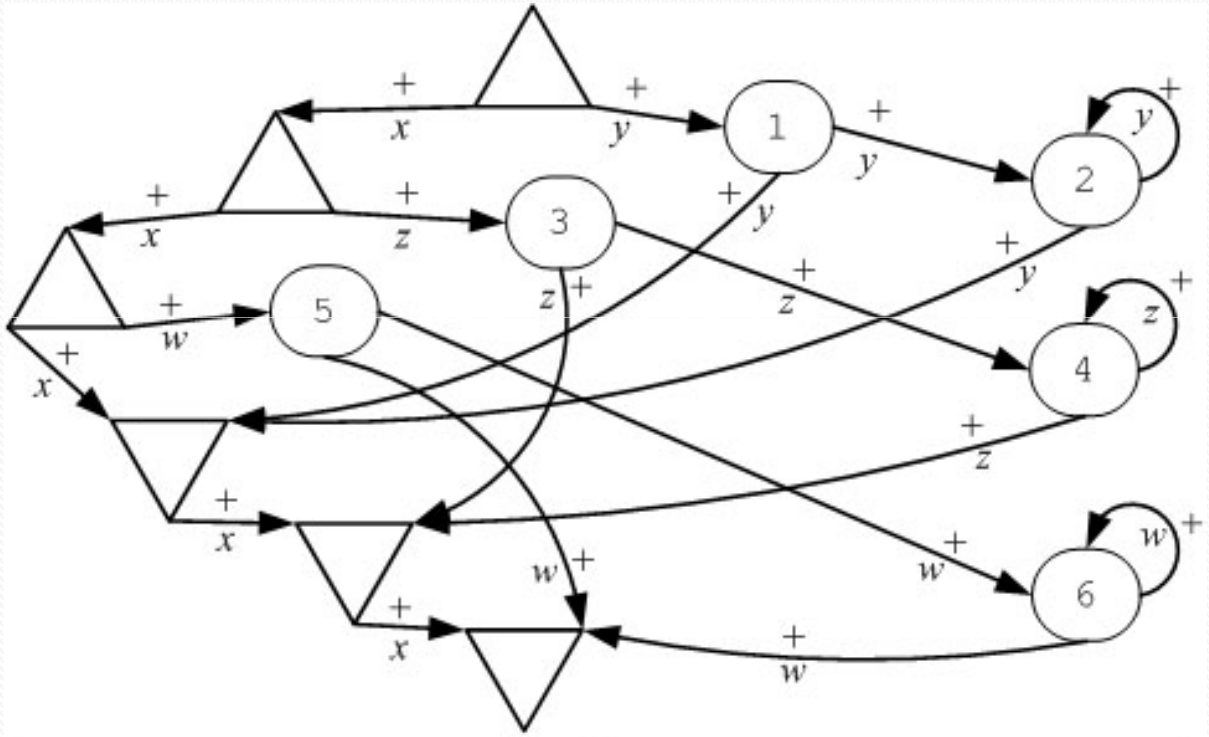
Methodology

1. Interview the EAI team in order to capture each integration action logic
2. Build the CDGPL pattern corresponding to the transmitted integration action logic defined in 1.
3. Search for the pattern defined in 2.
    1. When not found, relax the pattern by removing vertices and edges
4. Analyse the actual found coordination logic, by direct analysis or transforming it into ORC
5. Transmit to the EAI team potential problems in the EAI implementation

Create a user from the Web Portal

- The user has to be create on the TS, CRM, ERP
- The user might already be inserted in one of these systems
  - Check first if the user exists
  - If user doesn't exist, try to create it on the corresponding system
- In case of permanent failure, the error logs are kept in a error log table to be processed later.

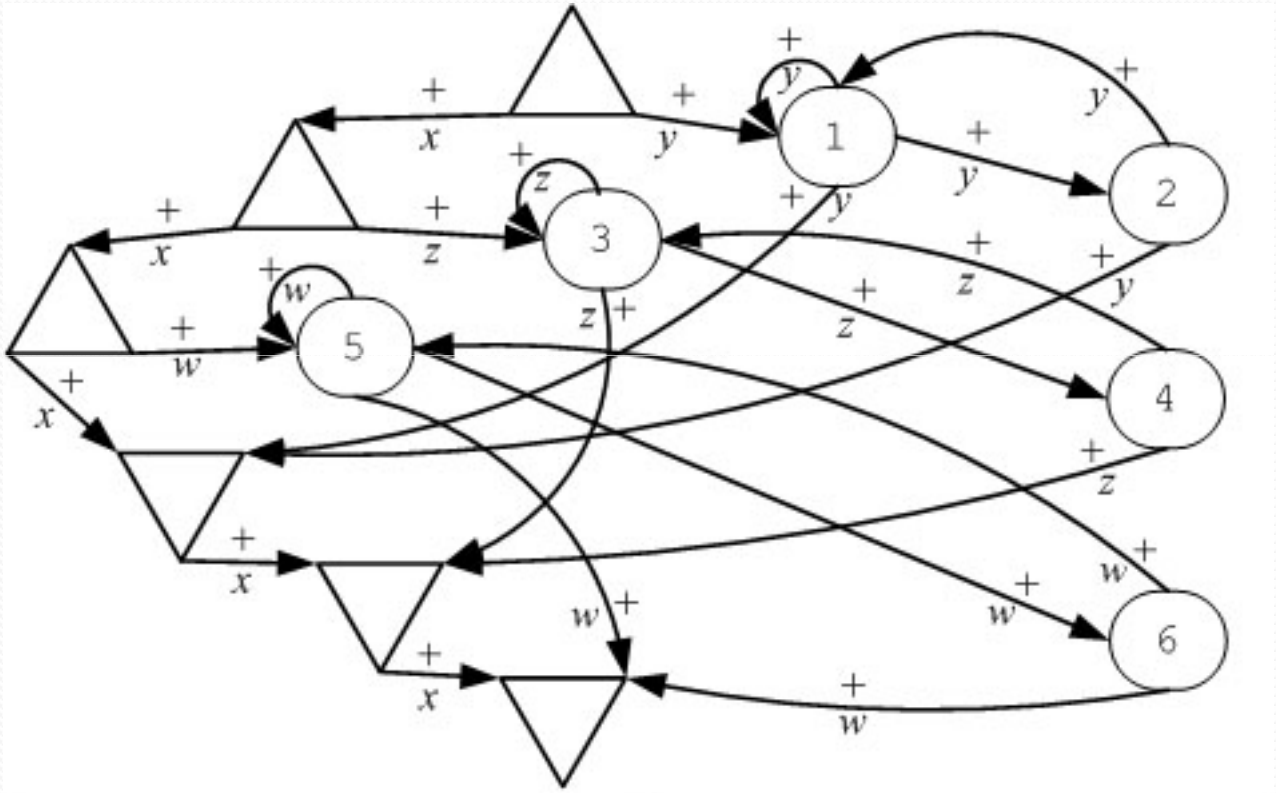
Create a new User



1 - CallWs ("CrmExistsUser", 2)
2 - CallWs ("CrmCreateUser", 2)
3 - CallWs ("ErpExistsUser", 2)

4 - CallWs ("ErpCreateUser", 2)
5 - CallWs ("TsExistsUser", 2)
6 - CallWs ("TsCreateUser", 2)

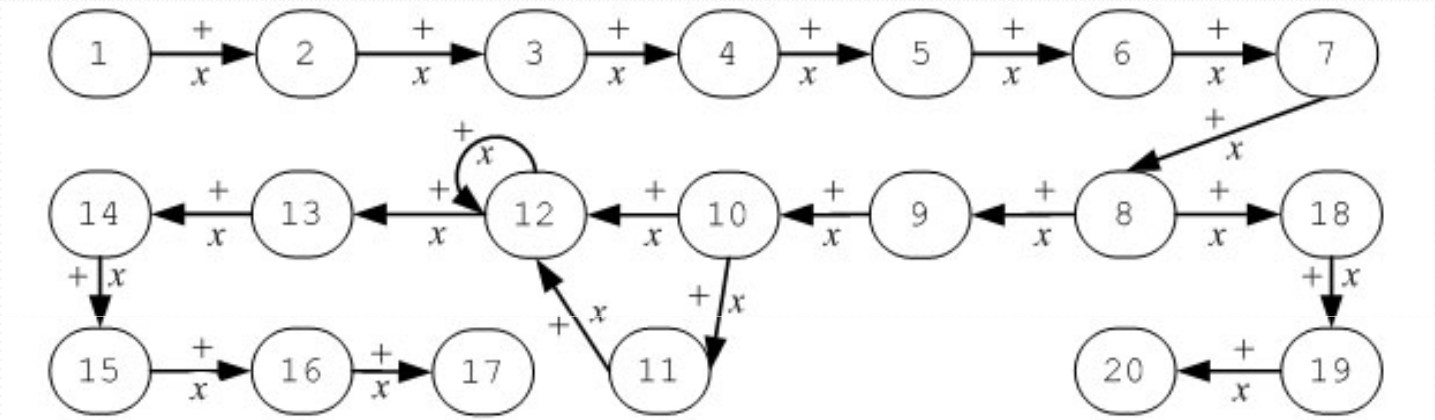
Create a new User (fixed)



1 - CallWs ("CrmExistsUser", 2)
2 - CallWs ("CrmCreateUser", 2)
3 - CallWs ("ErpExistsUser", 2)

4 - CallWs ("ErpCreateUser", 2)
5 - CallWs ("TsExistsUser", 2)
6 - CallWs ("TsCreateUser", 2)

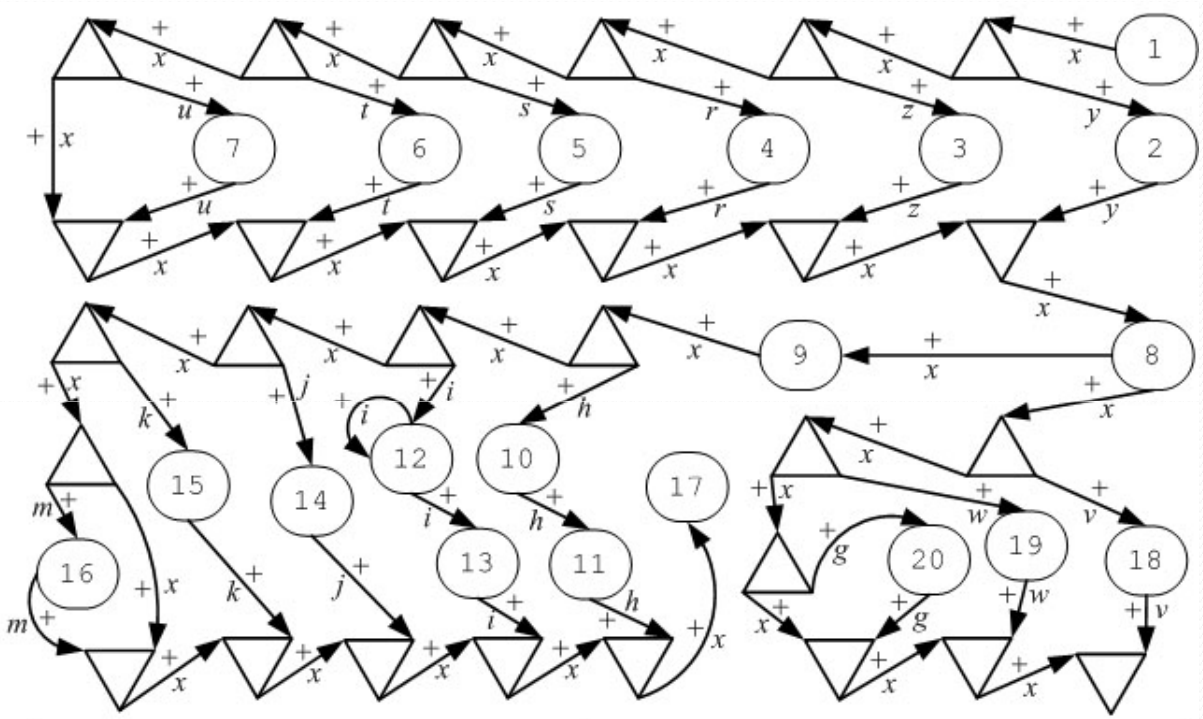
Sell a course online



1 - CallWs ("TsGetCourseAvailability", 2)
2 - CallWs ("TsReserveCoursePos", 2)
3 - CallWs ("CrmReserveCoursePos", 2)
4 - CallWs ("ErpReserveCoursePos", 2)
5 - CallWs ("CrmGetClientCampaigns", 2)
6 - CallWs ("CrmGetQtVolDiscounts", 2)
7 - CallWs ("CrmGetUserCreditNotes", 2)
8 - CallWs ("Unicre3DSecureDebit", 2)
9 - CDGPL_UpdateUser
10 - CallWs ("CrmGetUserPromotions", 2)

11 - SendMail\s\(\s*arg*\)\s*;*
12 - CDGPL_CreateUser
13 - SendMail\s\(\s*arg*\)\s*;*
14 - CallWs ("TsBookCoursePos", 2)
15 - CallWs ("CrmBookCoursePos", 2)
16 - CallWs ("ErpBookCoursePos", 2)
17 - SendMail\s\(\s*arg*\)\s*;*
18 - CallWs ("TsAnulateReservationPos", 2)
19 - CallWs ("CrmAnulateReservationPos", 2)
20 - CallWs ("ErpAnulateReservationPos", 2)

Sell a course online (fixed)



1 - CallWs ("TsGetCourseAvailability", 2)
2 - CallWs ("TsReserveCoursePos", 2)
3 - CallWs ("CrmReserveCoursePos", 2)
4 - CallWs ("ErpReserveCoursePos", 2)
5 - CallWs ("CrmGetClientCampaigns", 2)
6 - CallWs ("CrmGetQtVolDiscounts", 2)
7 - CallWs ("CrmGetUserCreditNotes", 2)
8 - CallWs ("Unicre3DSecureDebit", 2)
9 - CDGPL_UpdateUser
10 - CallWs ("CrmGetUserPromotions", 2)

11 - SendMail\s*\(\s*arg*\)\s*;
12 - CDGPL_CreateUser
13 - SendMail\s*\(\s*arg*\)\s*;
14 - CallWs ("TsBookCoursePos", 2)
15 - CallWs ("CrmBookCoursePos", 2)
16 - CallWs ("ErpBookCoursePos", 2)
17 - SendMail\s*\(\s*arg*\)\s*;
18 - CallWs ("TsAnulateReservationPos", 2)
19 - CallWs ("CrmAnulateReservationPos", 2)
20 - CallWs ("ErpAnulateReservationPos", 2)

Conclusions and Future Work

- It is possible to retrieve a system coordination logic in an almost automatic way
  - The extracted logic as to be further treated with special purpose tools and language semantics
- No need for source code, one can have a multi-language approach by analysing virtual machine code.
- Coordination is really the biggest problem in EAI
- Extend the Pattern Language
  - Negations
  - Variable bindings between vertex expressions
- Improve the MSDG construction
- Transform the implementation according to the discovered and transformed patterns

Thank you

Universidade do Minho

DI-CCTC

Braga, Portugal

Nuno F. Rodrigues – nfr@di.uminho.pt

Luís S. Barbosa – lsb@di.uminho.pt