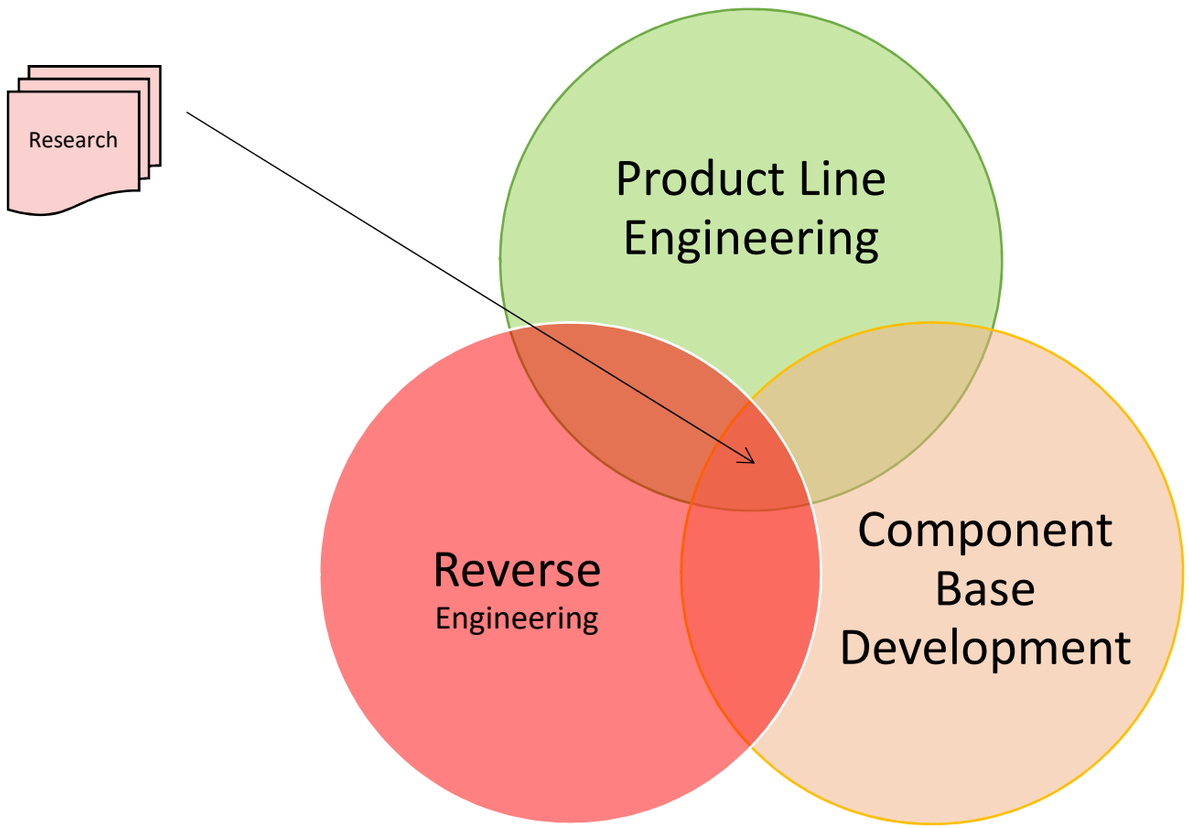
MANCHESTER 1824
The University of Manchester

Extracting Executable Architecture from Legacy Code using Static Reverse Engineering

REHMAN ARSHAD

The University of Manchester, UK

Research Context



MANCHESTER 1824
The University of Manchester

Reverse Engineering

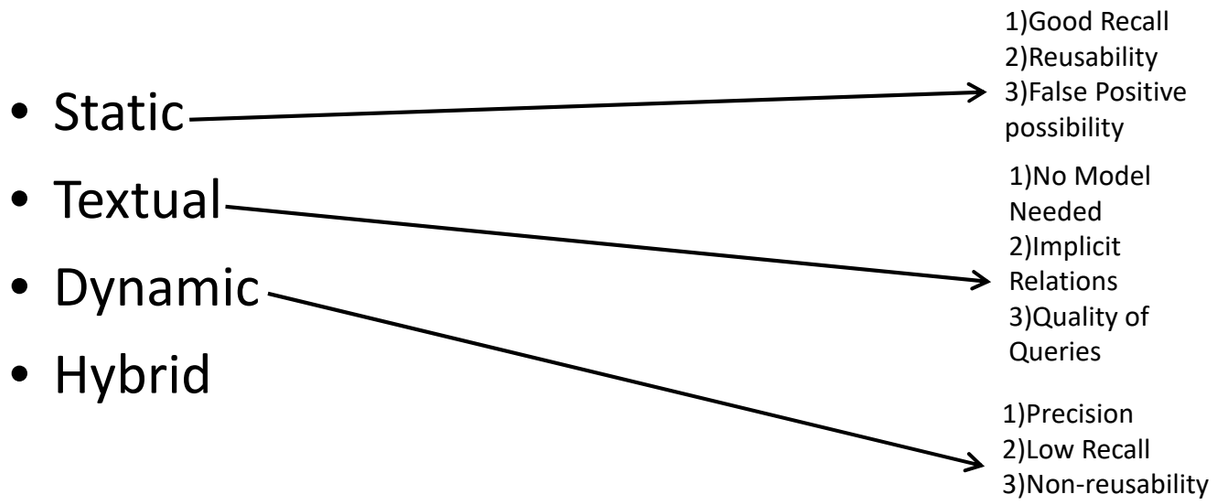
- Reverse engineering can be viewed as the process of analysing a system to identify the system's components and their interrelationships, create representations of the system in another form or a higher level of abstraction or to create the physical representation of that system.

MANCHESTER 1824
The University of Manchester

Current Reverse Engineering Approaches

- General
  - Semantics extraction
  - Bug localisation
  - Model extraction
- Product Line
  - Formation of feature model
  - Feature locations in terms of code

Reverse Engineering Analysis Techniques



MANCHESTER 1824
The University of Manchester

Executable Architecture

- An executable architecture is a dynamic simulation of an architecture model. It captures both structural and behavioral aspects of the architecture in a form that can be visualized and analyzed in a time dependent manner.

MANCHESTER 1824

The University of Manchester

Classification of Component Models

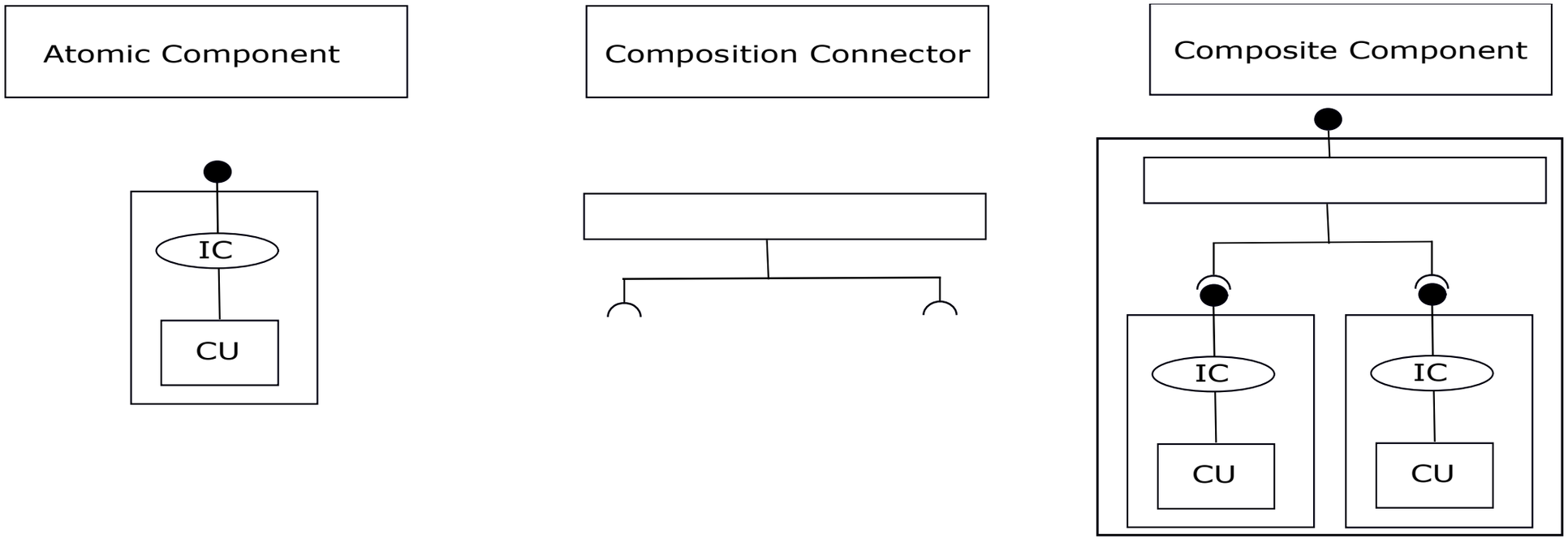
Category	Models	Design				Deploy
		Deposit–N	Retrieve	Compose	Deposit–C	Compose
Design without Repository	Acme–like ADLs, UML2.0, PECOS, Fractal	×	×	✓	×	×
Design with Deposit–only Repository	EJB, COM, .NET, CCM	✓	×	✓	×	×
Deployment with Repository	JavaBeans, Web Services	✓	×	×	×	✓
Design with Repository	Koala, SOFA, KobrA	✓	✓	✓	✓	×
Design and Deployment with Repository	Exogenous Composition	✓	✓	✓	✓	✓

MANCHESTER 1824
The University of Manchester

Research Problems

- Why extraction of components is not widely considered by the reverse engineering researchers, specially in the field of product line engineering?
- Why static reverse engineering is the best option to extract architectural notation from legacy systems?
- How extraction of architecture can reduce coupling?
- How reverse engineering in general is different from component based reverse engineering?

X-MAN



IC: Invocation Connector
CU: Computation Unit

MANCHESTER 1824

The University of Manchester

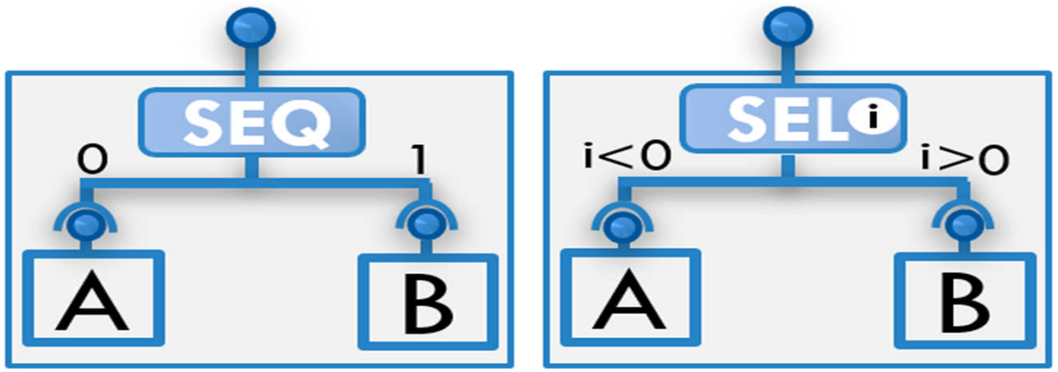
X-MAN

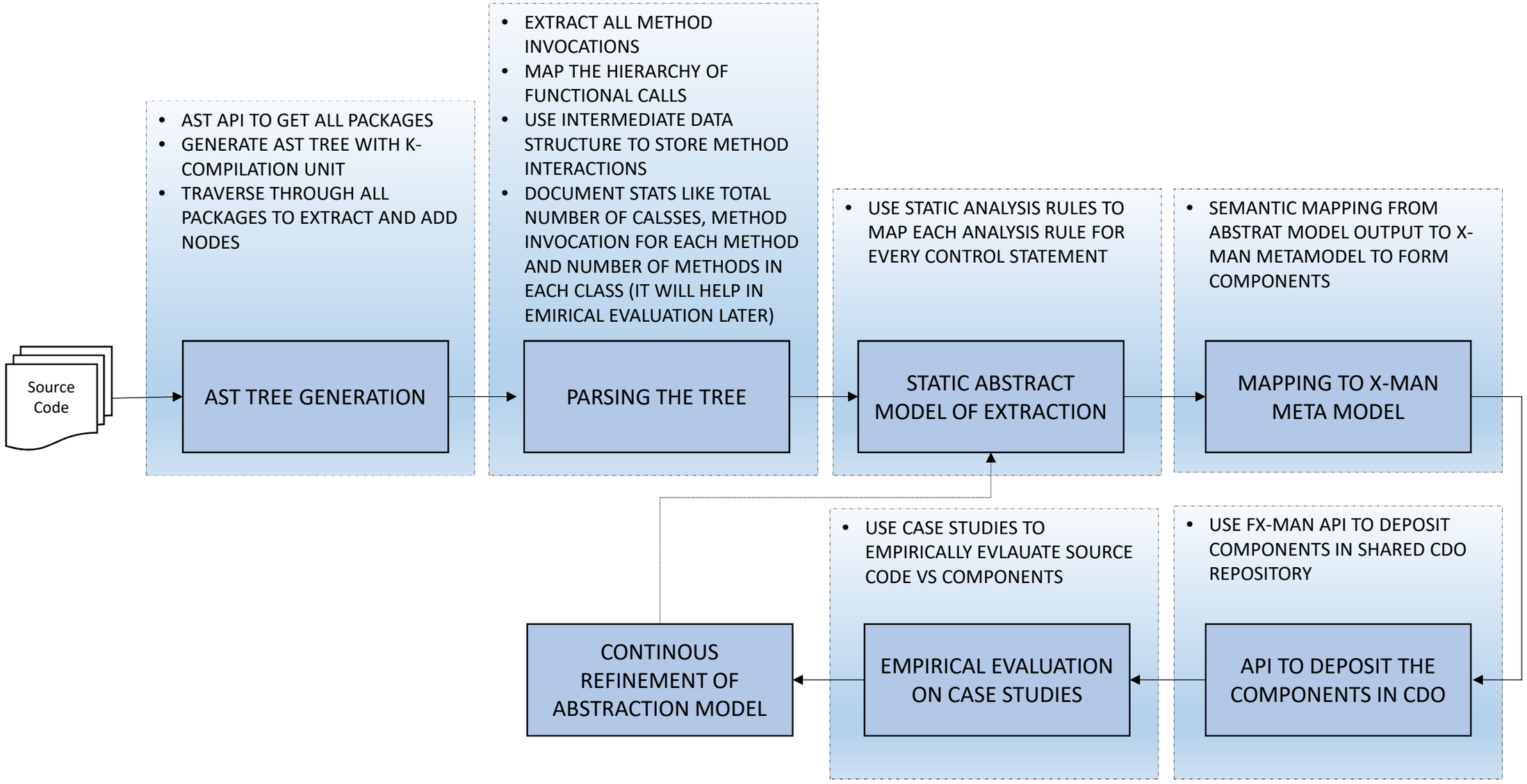
1. Exogenous composition
2. Exogenous connectors for composition that define control coordination
3. Invocation connectors: connected to computational units and access their methods
4. Composition connectors: define and coordinate control for set of components

MANCHESTER 1824

The University of Manchester

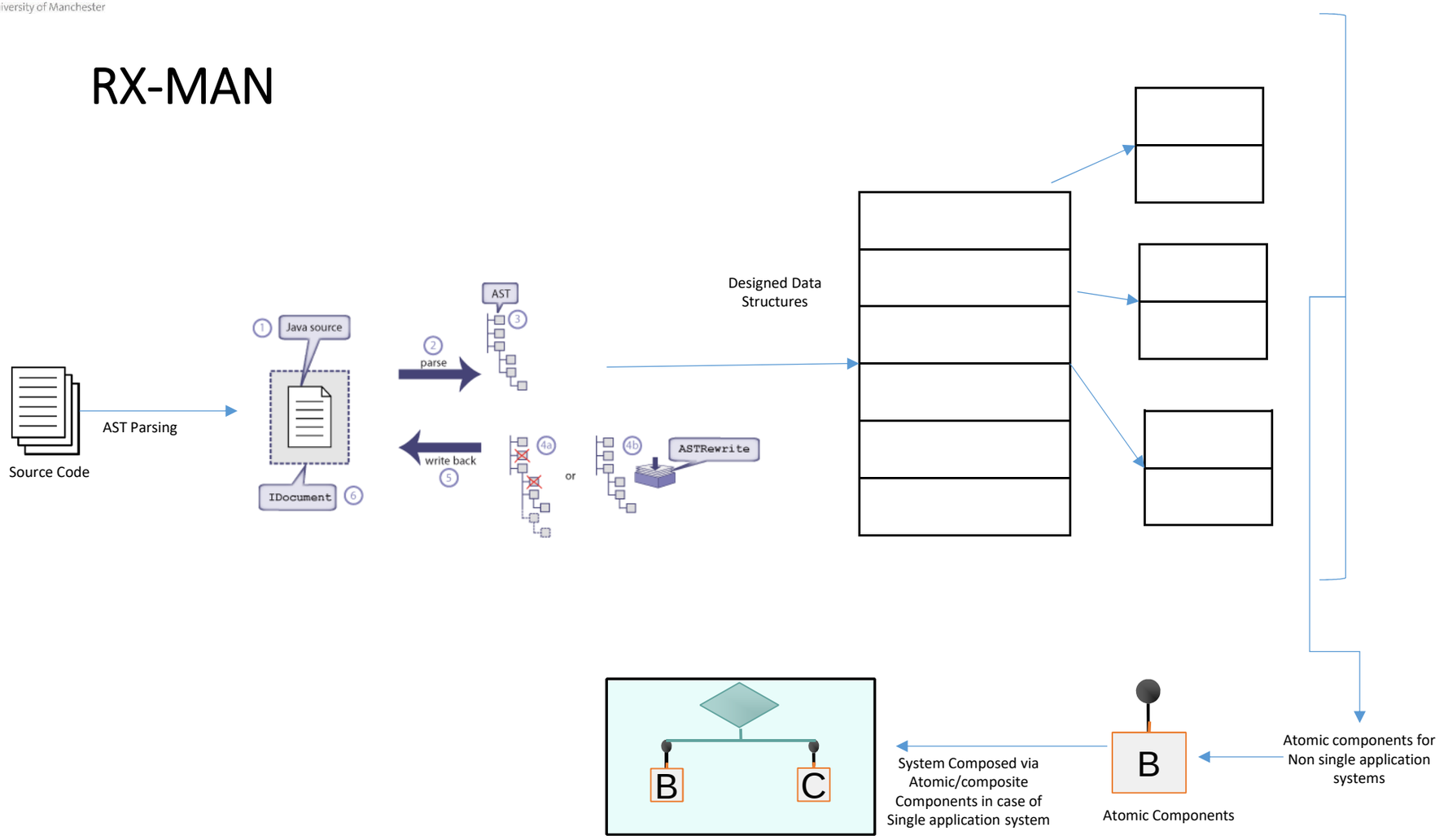
Composition Connectors in X-MAN




Source Code
AST TREE GENERATION
• AST API TO GET ALL PACKAGES
• GENERATE AST TREE WITH K-COMPILATION UNIT
• TRAVERSE THROUGH ALL PACKAGES TO EXTRACT AND ADD NODES
PARSING THE TREE
• EXTRACT ALL METHOD INVOCATIONS
• MAP THE HIERARCHY OF FUNCTIONAL CALLS
• USE INTERMEDIATE DATA STRUCTURE TO STORE METHOD INTERACTIONS
• DOCUMENT STATS LIKE TOTAL NUMBER OF CALSSES, METHOD INVOCATION FOR EACH METHOD AND NUMBER OF METHODS IN EACH CLASS (IT WILL HELP IN EMIRICAL EVALUATION LATER)
STATIC ABSTRACT MODEL OF EXTRACTION
• USE STATIC ANALYSIS RULES TO MAP EACH ANALYSIS RULE FOR EVERY CONTROL STATEMENT
MAPPING TO X-MAN META MODEL
• SEMANTIC MAPPING FROM ABSTRAT MODEL OUTPUT TO X-MAN METAMODEL TO FORM COMPONENTS
API TO DEPOSIT THE COMPONENTS IN CDO
• USE FX-MAN API TO DEPOSIT COMPONENTS IN SHARED CDO REPOSITORY
EMPIRICAL EVALUATION ON CASE STUDIES
• USE CASE STUDIES TO EMPIRICALLY EVLAUATE SOURCE CODE VS COMPONENTS
CONTINOUS REFINEMENT OF ABSTRACTION MODEL

MANCHESTER 1824
The University of Manchester

RX-MAN



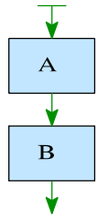
MANCHESTER 1824
The University of Manchester

Control Statements to be considered

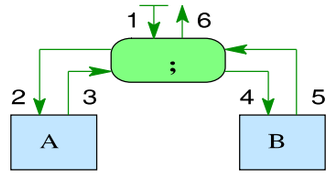
- If else Statements
- For Statements
- While Statements
- Do Statements
- Switch Statements

State Charts VS X-MAN Control Structure

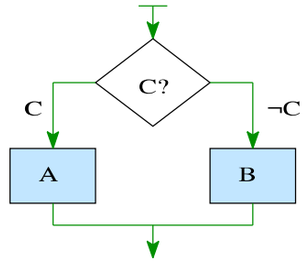
A ; B



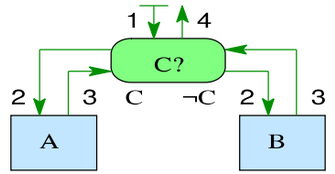
Sequencer



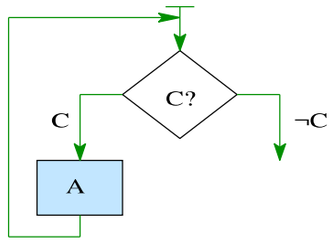
if C then A else B



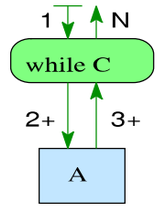
Selector



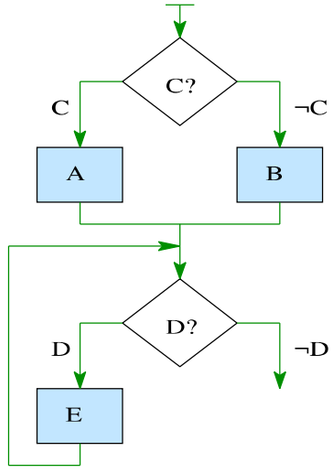
while C do A

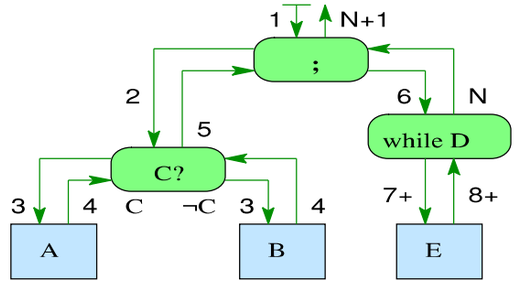


Iterator



if C then A else B; while D do E





Key

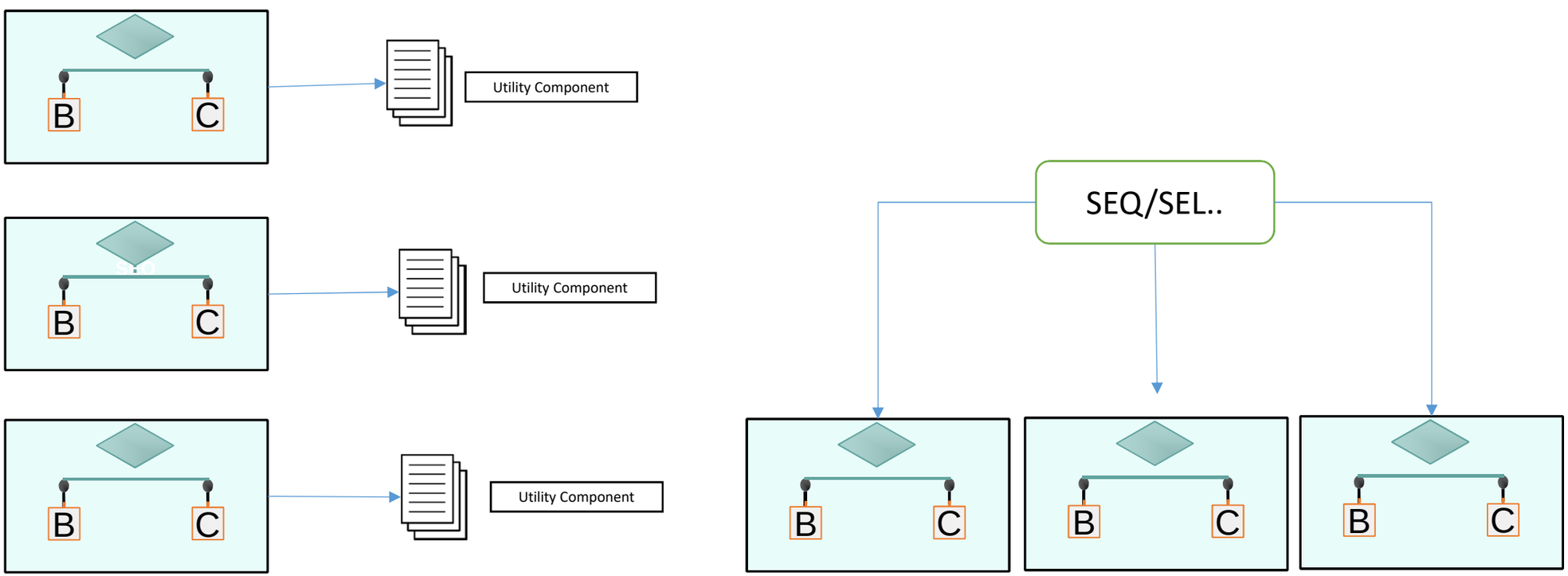
Control flow

Computation

Control entry point

Control connector

Single Application System VS non single code bases



MANCHESTER 1824
The University of Manchester

Approach applied on

- JabRef Model packages (35 Classes, 8 Atomic Components), 40 Secs
- EverNote API SDK (27 Classes, 4 Components) 30 Secs
- TeamMates API (51 Classes, 4 Components), 55 Secs
- JabRef Full Code from Quality Corpus (903 Classes, 37 Components), 22 Minutes

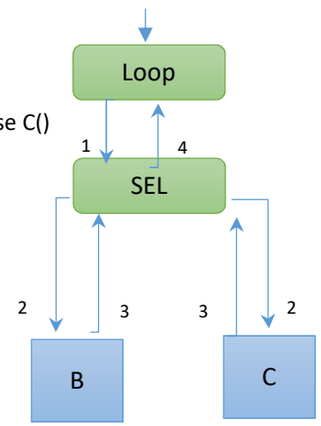
MANCHESTER 1824
The University of Manchester

Classes to Components Ratio So Far

- JabRef Model packages (35 Classes, 8 Atomic Components), 4.3%
- EverNote API SDK (27 Classes, 4 Components), 6.75%
- TeamMates API (51 Classes, 4 Components), 12.7%

MANCHESTER 1824
The University of Manchester

While A
{
If B then B() else C()
}



MANCHESTER 1824
The University of Manchester

5)Switch (i)

{

Case (1):

A();

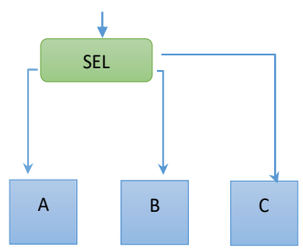
Case (2):

B();

Default:

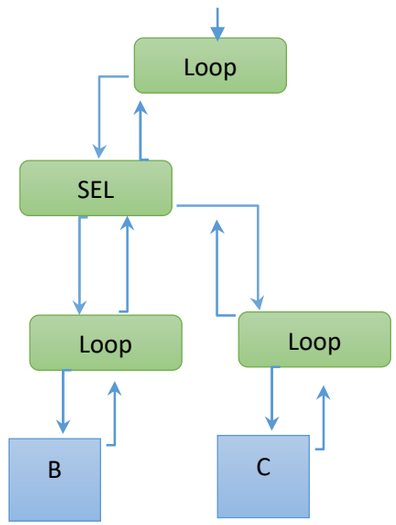
C();

}

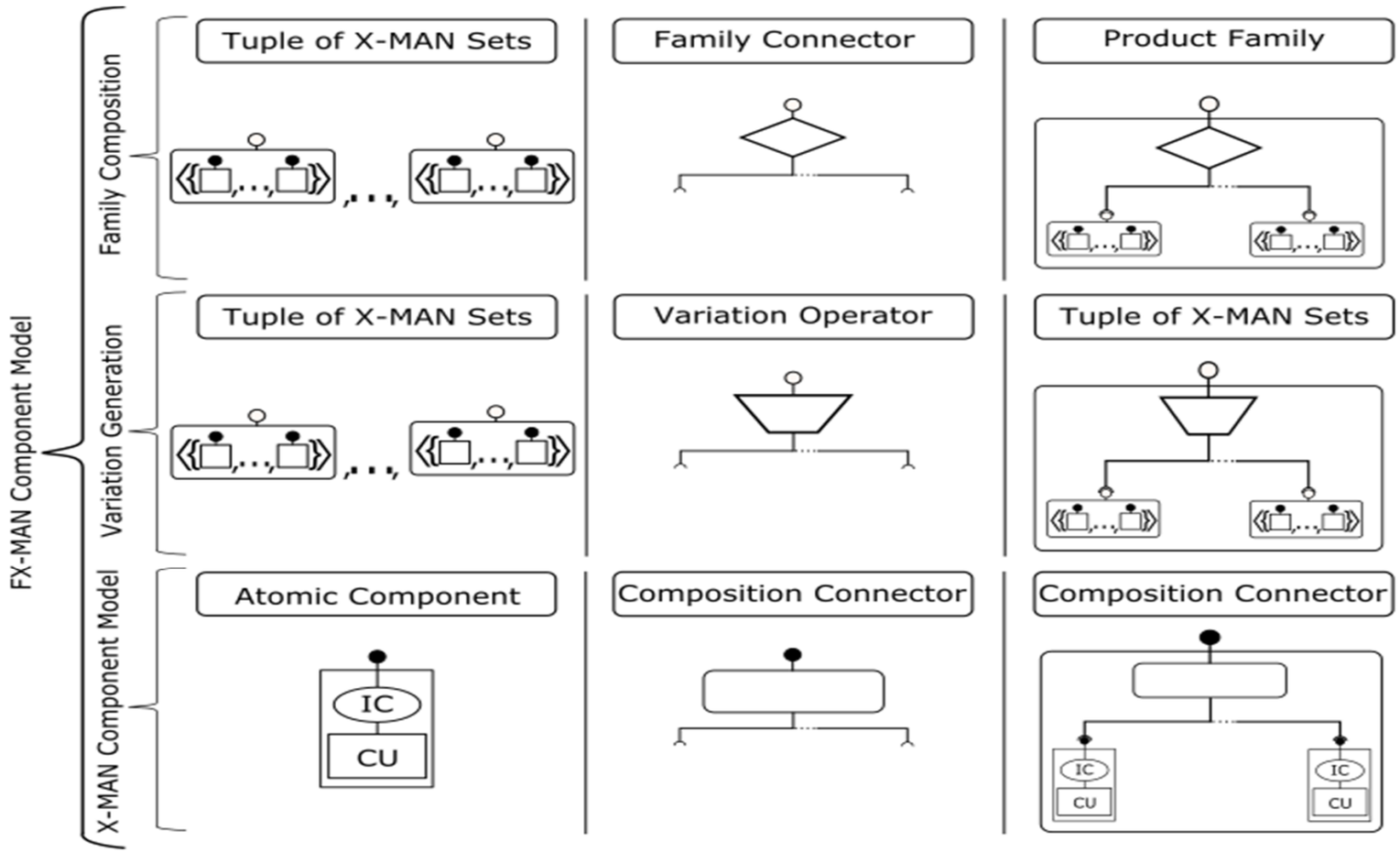


```
6) for (A <n)
{
While (C<m)
{
C();
}
do B();

}
```



FX-MAN


FX-MAN Component Model
Family Composition
Variation Generation
X-MAN Component Model
Tuple of X-MAN Sets
Family Connector
Product Family
Tuple of X-MAN Sets
Variation Operator
Tuple of X-MAN Sets
Atomic Component
Composition Connector
Composition Connector
IC
CU
IC
CU
IC
CU

Q&A

Questions?