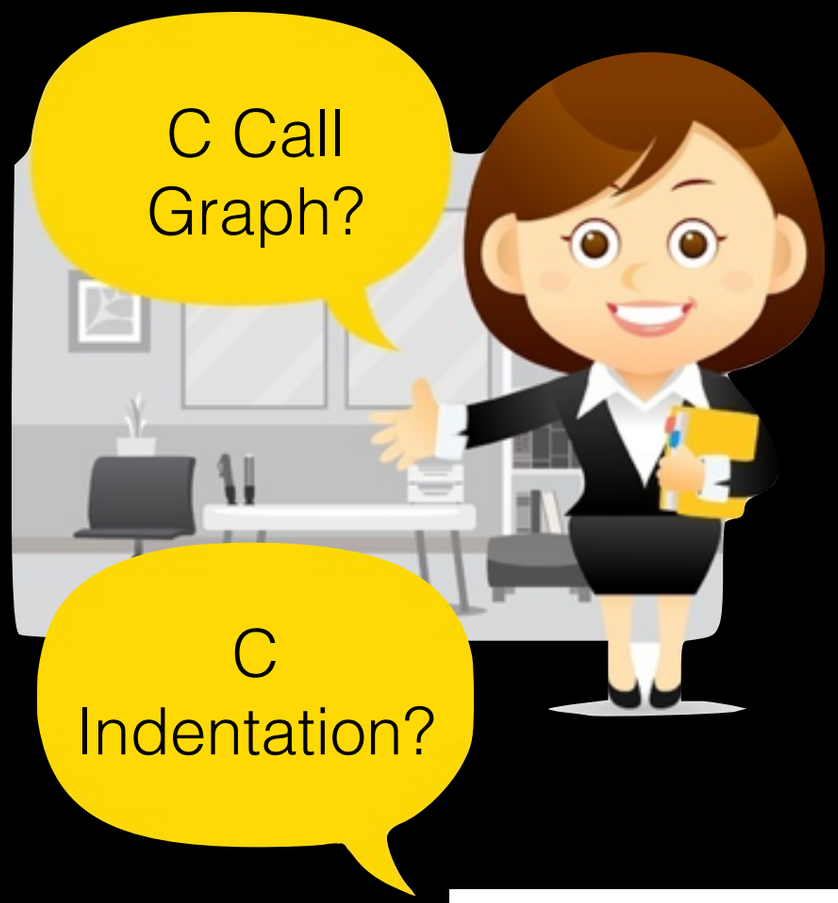
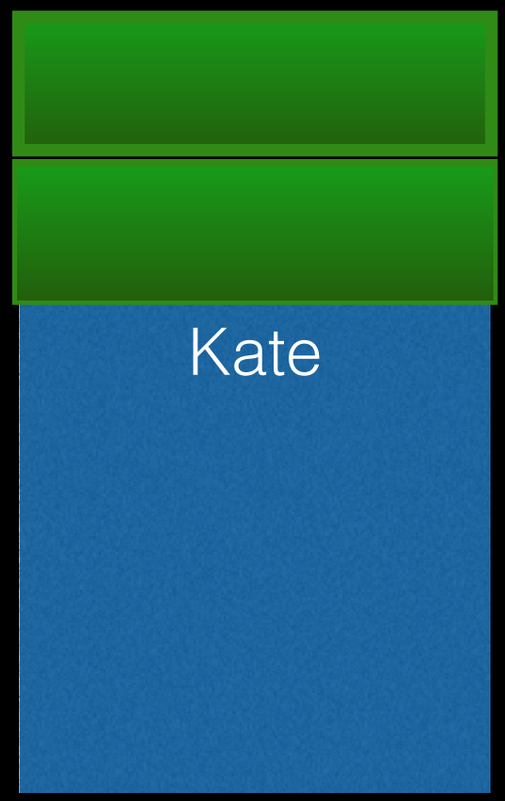
Automated Transplantation of Call Graph and Layout Features into Kate

Alexandru Marginean Earl T. Barr Mark Harman Yue Jia

CREST, University College London

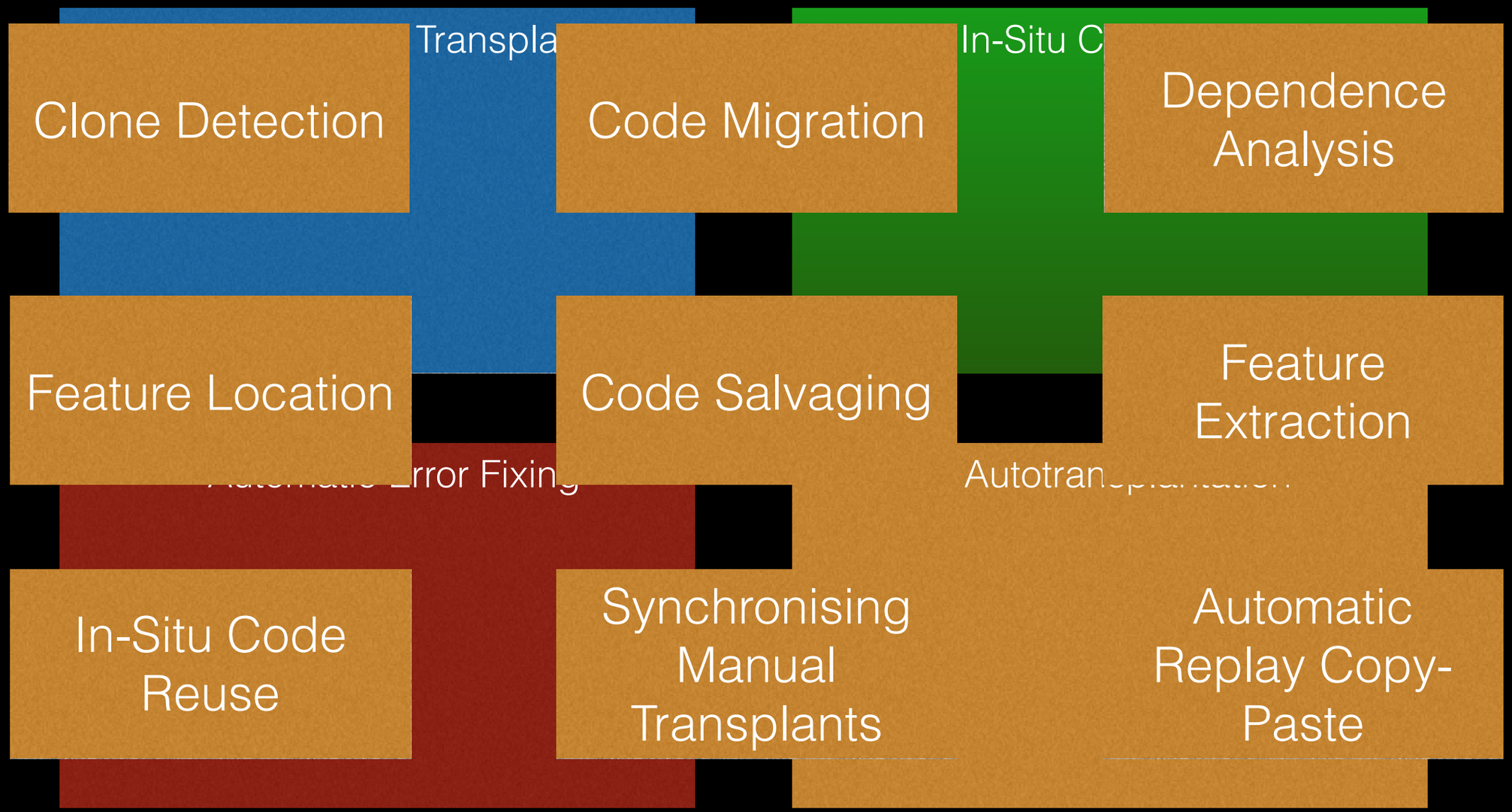
Why Autotransplantation?





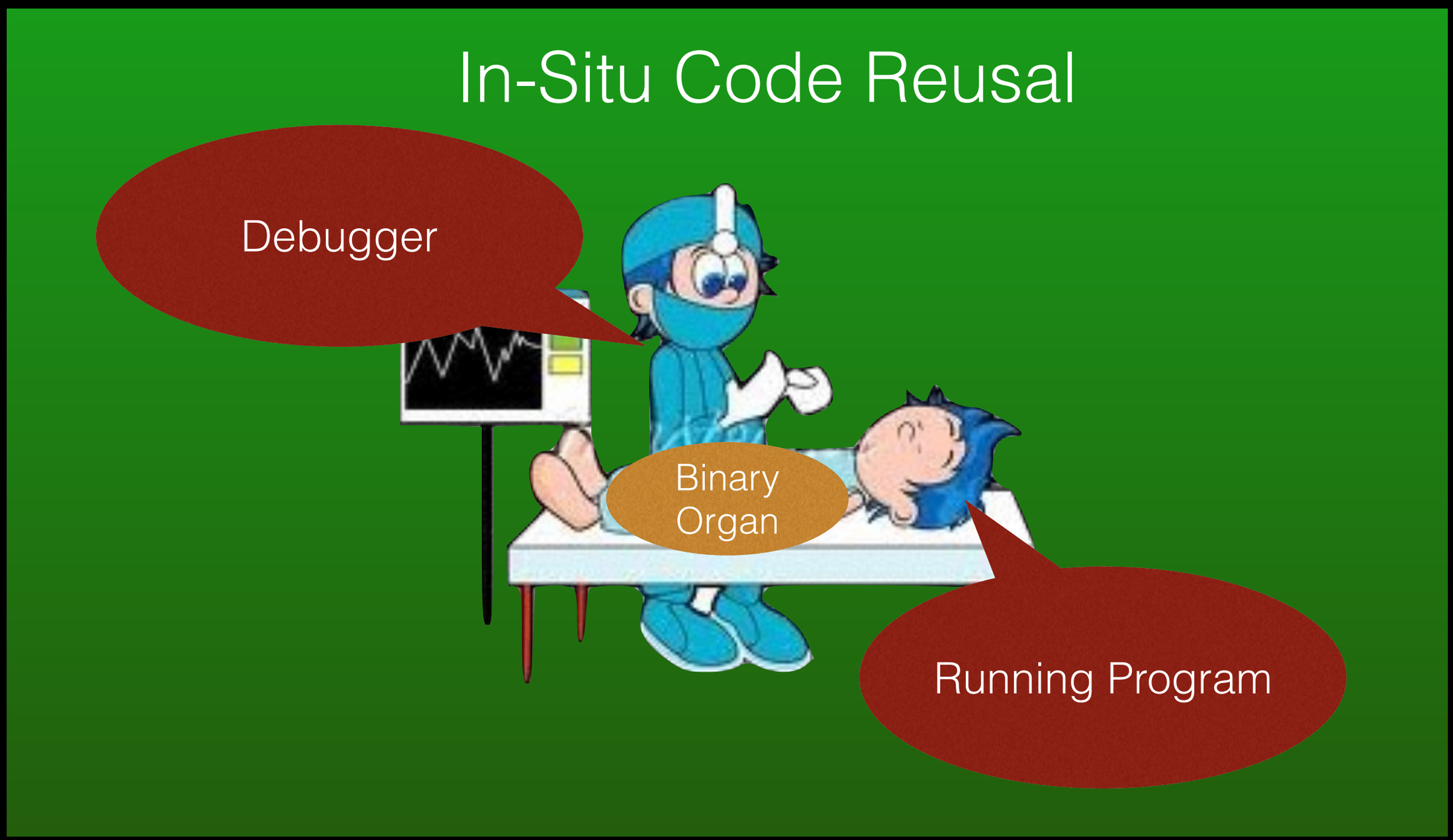
Start from scratch

Related Work



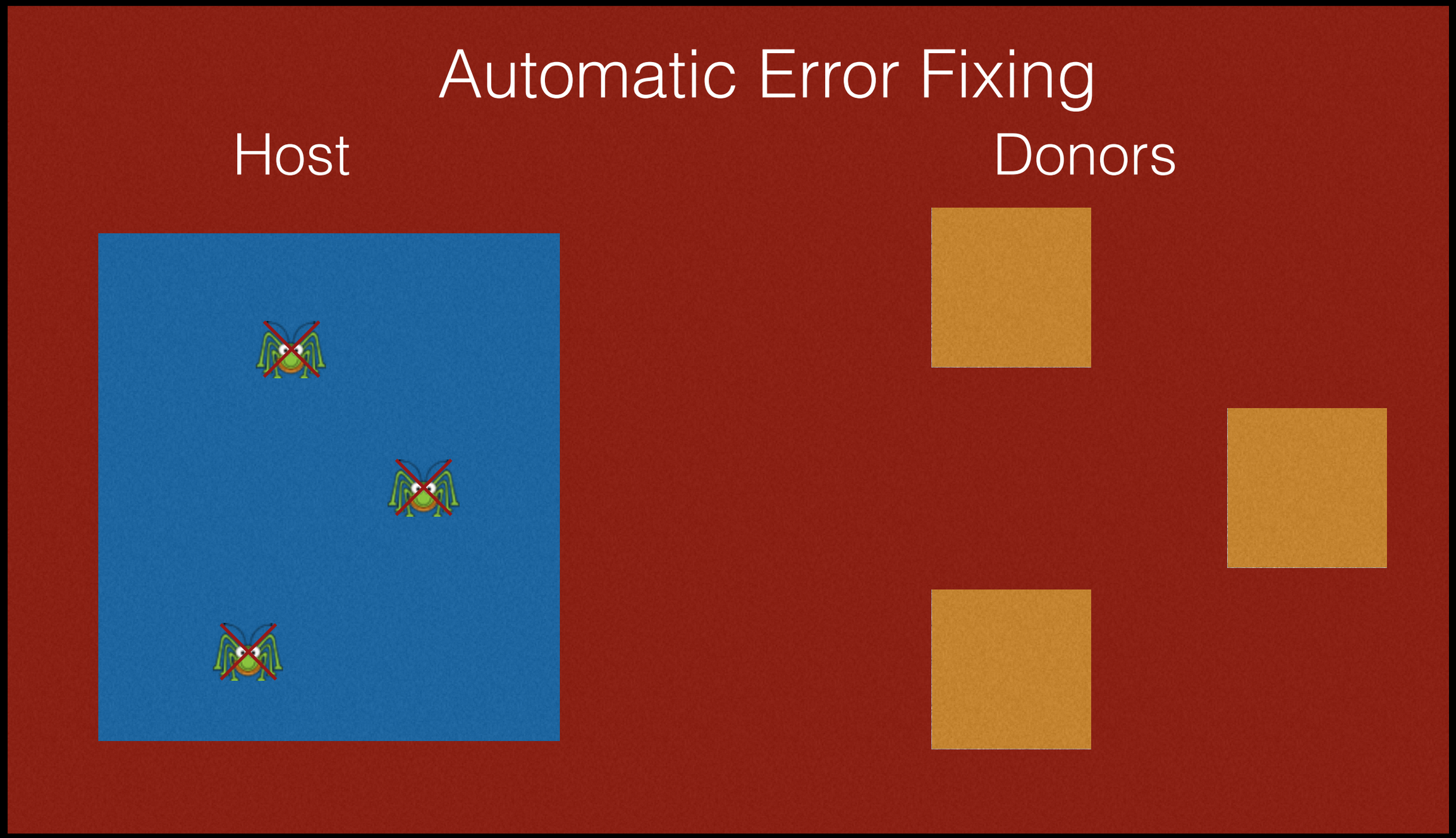
Related Work

Miles *et al.:* In situ reuse of logically extracted functional components



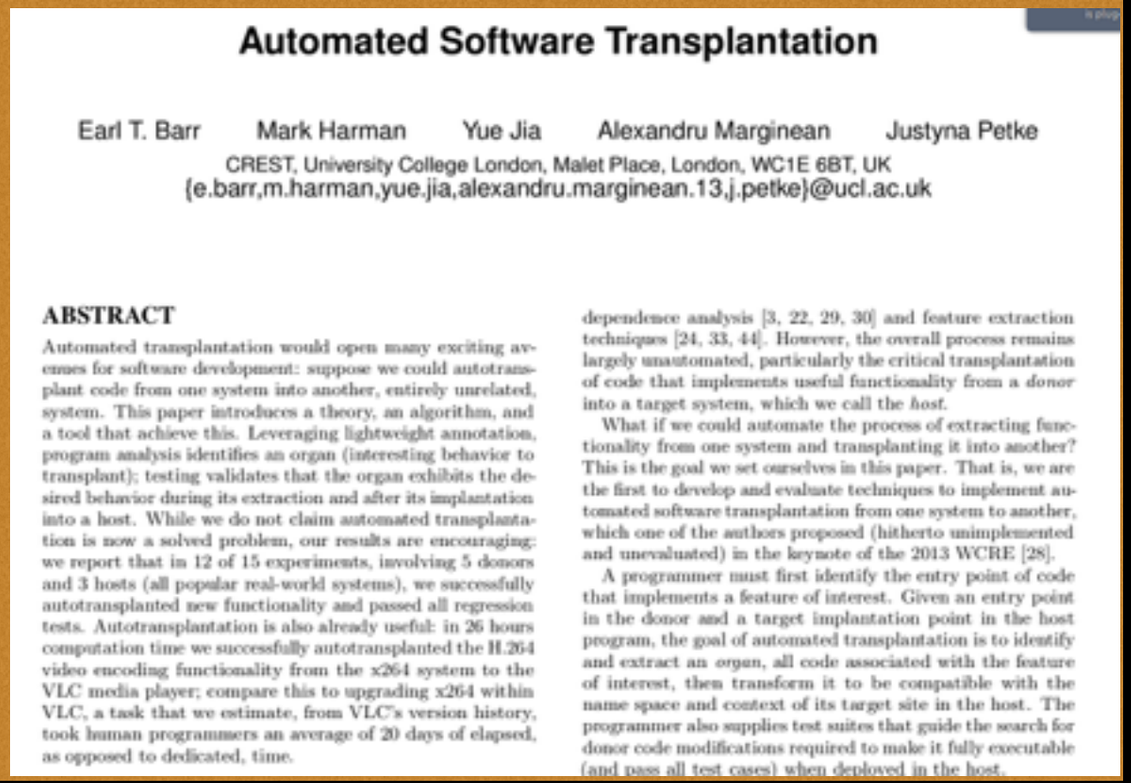
Related Work

Sidiroglou-Douskos *et al.:* Automatic Error Elimination by Multi-Application Code Transfer



Recognition for Our Tool MuScalpel

BBC World Service Interview
WIRED article
Many shares on Social Media
ACM Distinguished Paper Award at ISSTA ‘15



Automated Software Transplantation

Earl T. Barr Mark Harman Yue Jia Alexandru Marginean Justyna Petke
CREST, University College London, Malet Place, London, WC1E 6BT, UK
{e.barr,m.harman,yue.jia,alexandru.marginean.13,j.petke}@ucl.ac.uk

ABSTRACT

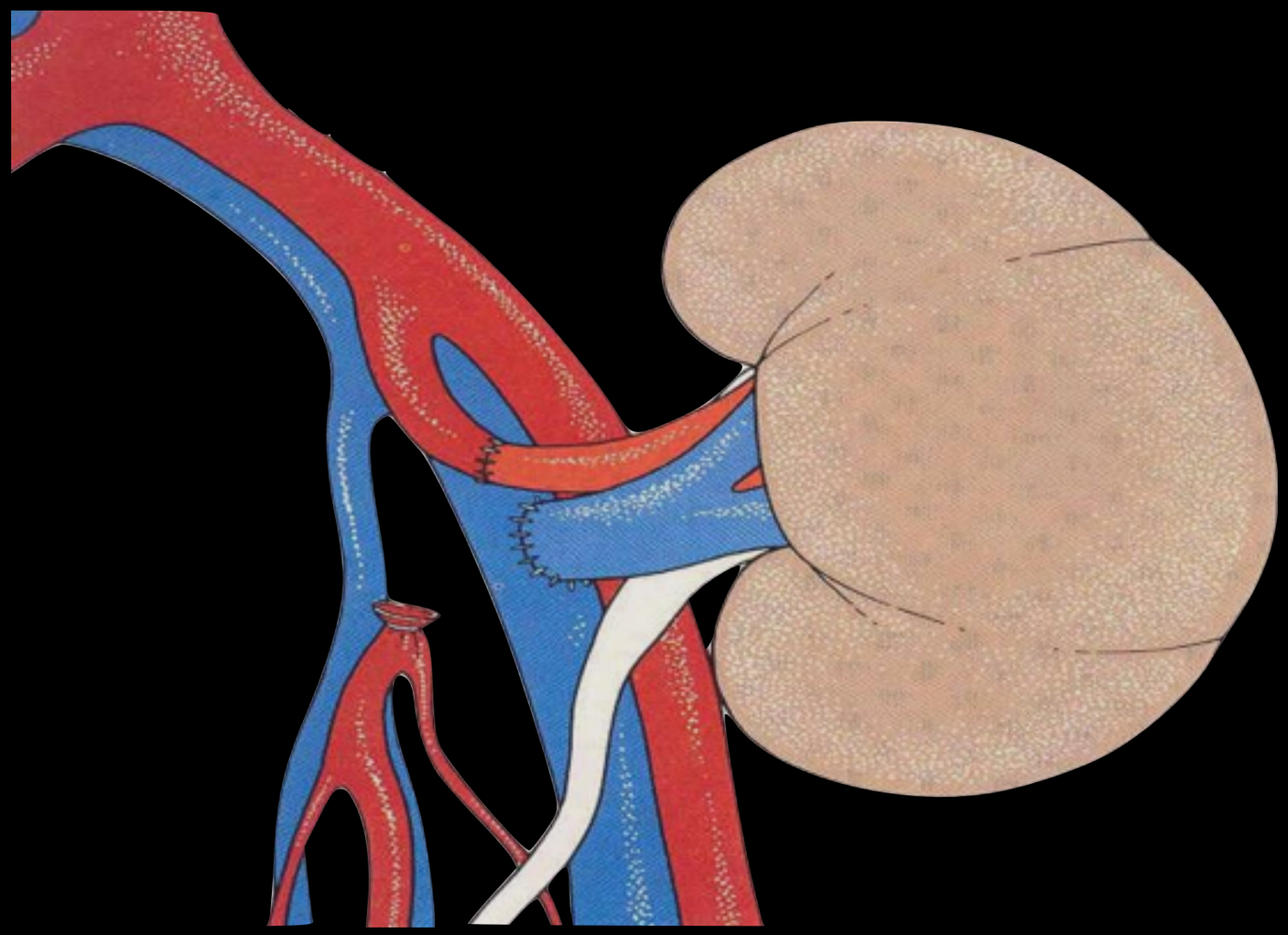
Automated transplantation would open many exciting avenues for software development: suppose we could autotransplant code from one system into another, entirely unrelated, system. This paper introduces a theory, an algorithm, and a tool that achieve this. Leveraging lightweight annotation, program analysis identifies an organ (interesting behavior to transplant); testing validates that the organ exhibits the desired behavior during its extraction and after its implantation into a host. While we do not claim automated transplantation is now a solved problem, our results are encouraging: we report that in 12 of 15 experiments, involving 5 donors and 3 hosts (all popular real-world systems), we successfully autotransplanted new functionality and passed all regression tests. Autotransplantation is also already useful: in 26 hours computation time we successfully autotransplanted the H.264 video encoding functionality from the x264 system to the VLC media player; compare this to upgrading x264 within VLC, a task that we estimate, from VLC's version history, took human programmers an average of 20 days of elapsed, as opposed to dedicated, time.

dependence analysis [3, 22, 29, 30] and feature extraction techniques [24, 33, 44]. However, the overall process remains largely unautomated, particularly the critical transplantation of code that implements useful functionality from a *donor* into a target system, which we call the *host*.

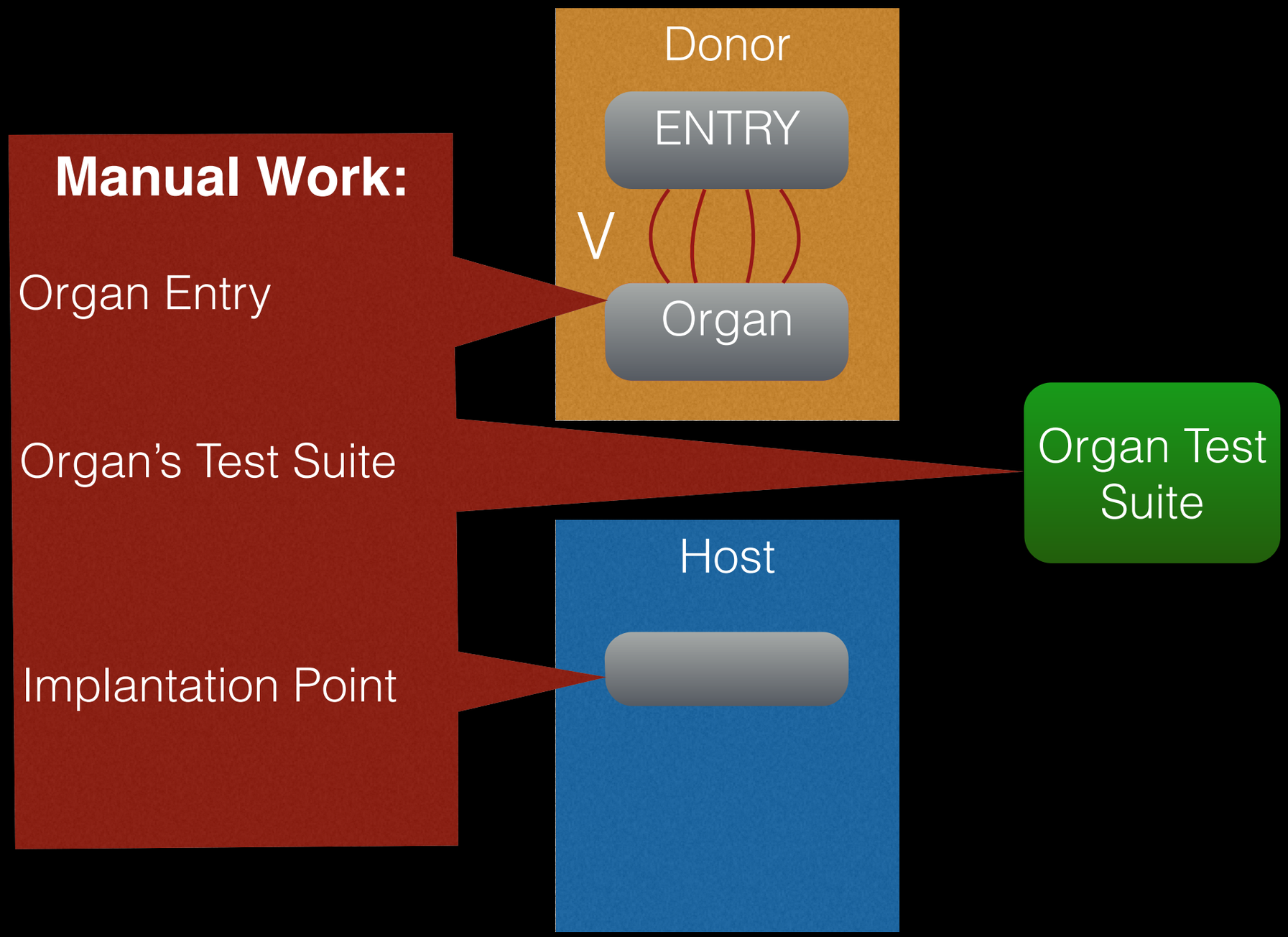
What if we could automate the process of extracting functionality from one system and transplanting it into another? This is the goal we set ourselves in this paper. That is, we are the first to develop and evaluate techniques to implement automated software transplantation from one system to another, which one of the authors proposed (hitherto unimplemented and unevaluated) in the keynote of the 2013 WCRE [28].

A programmer must first identify the entry point of code that implements a feature of interest. Given an entry point in the donor and a target implantation point in the host program, the goal of automated transplantation is to identify and extract an *organ*, all code associated with the feature of interest, then transform it to be compatible with the name space and context of its target site in the host. The programmer also supplies test suites that guide the search for donor code modifications required to make it fully executable (and pass all test cases) when deployed in the host.

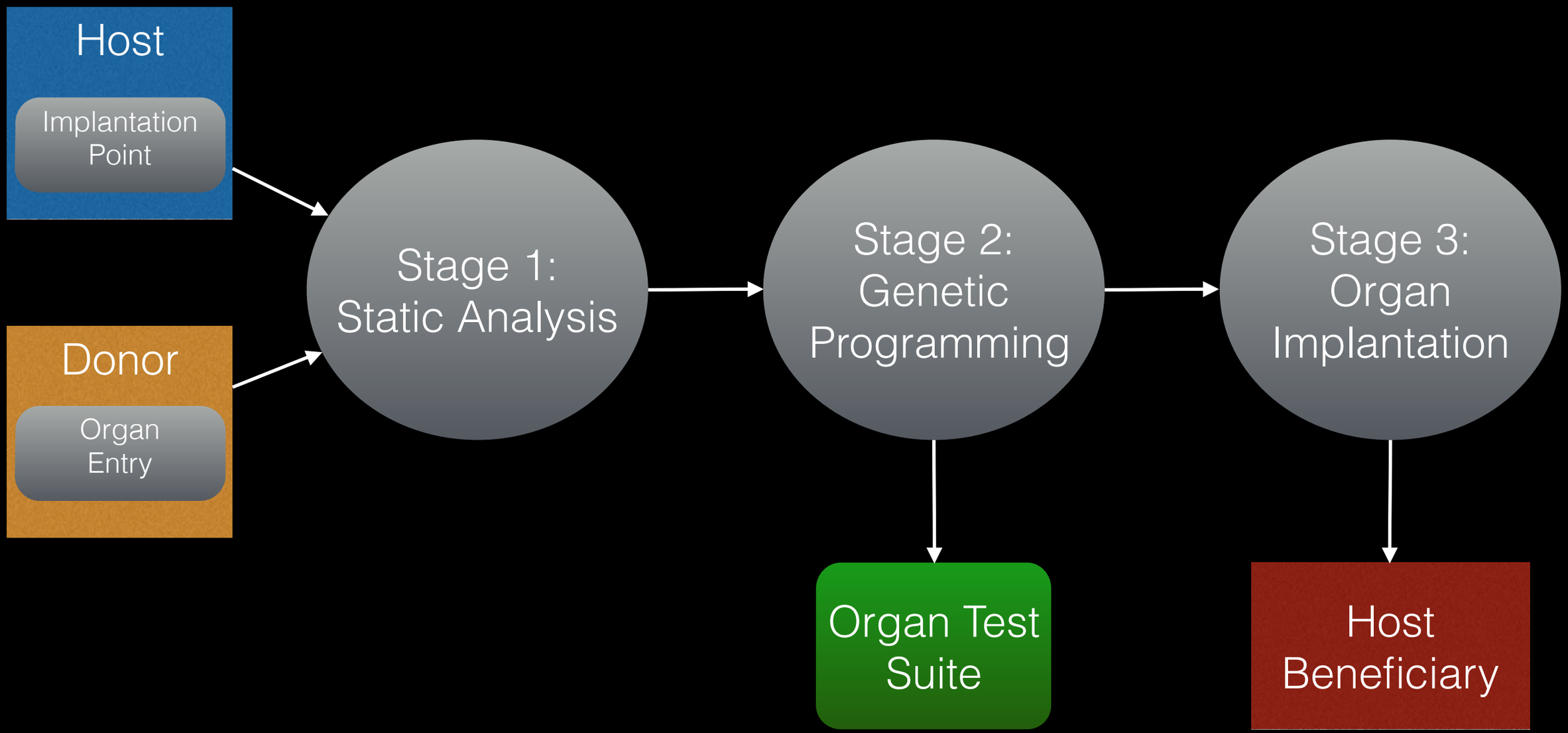
Human Organ Transplantation



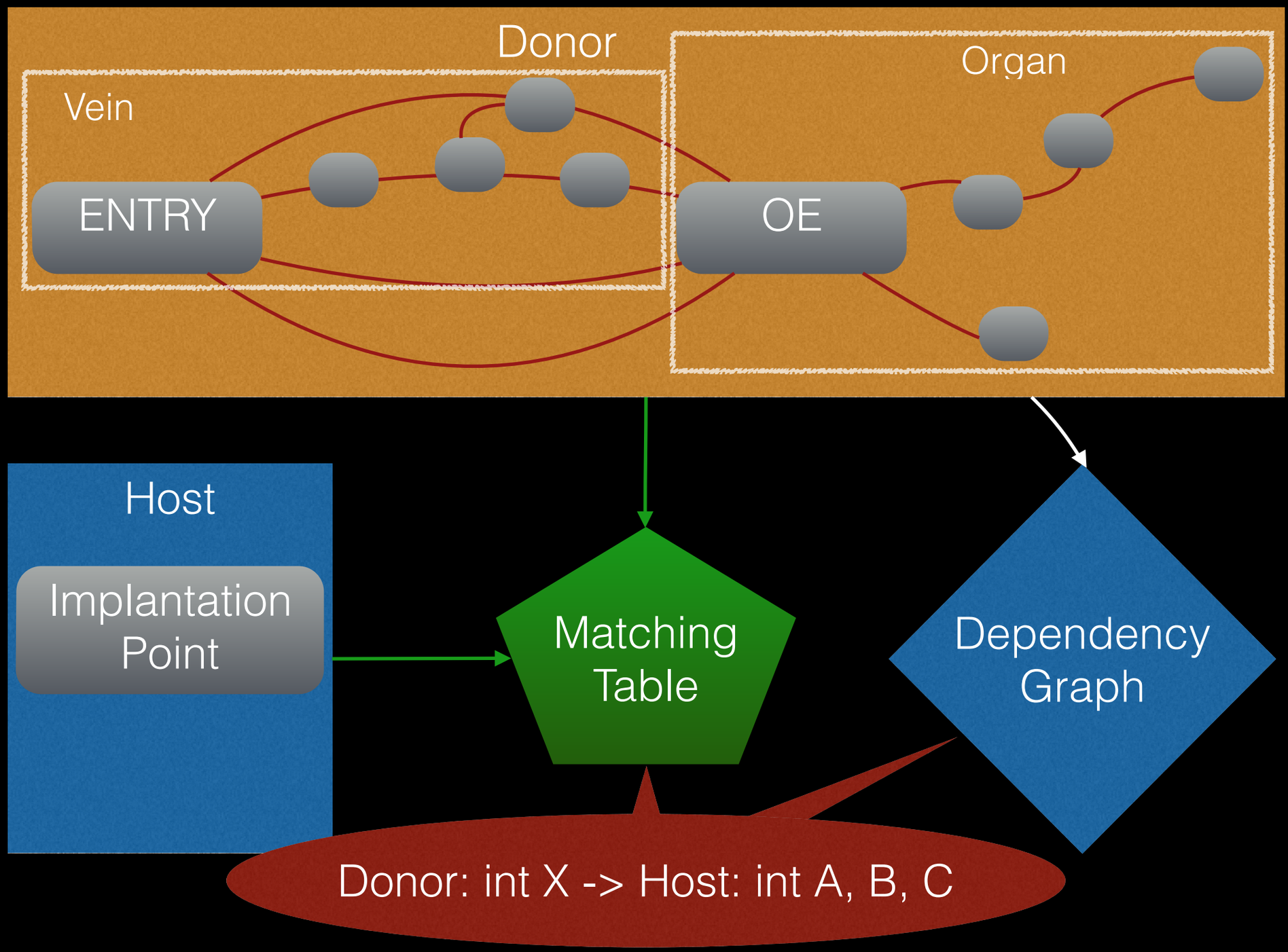
Automated Software Transplantation



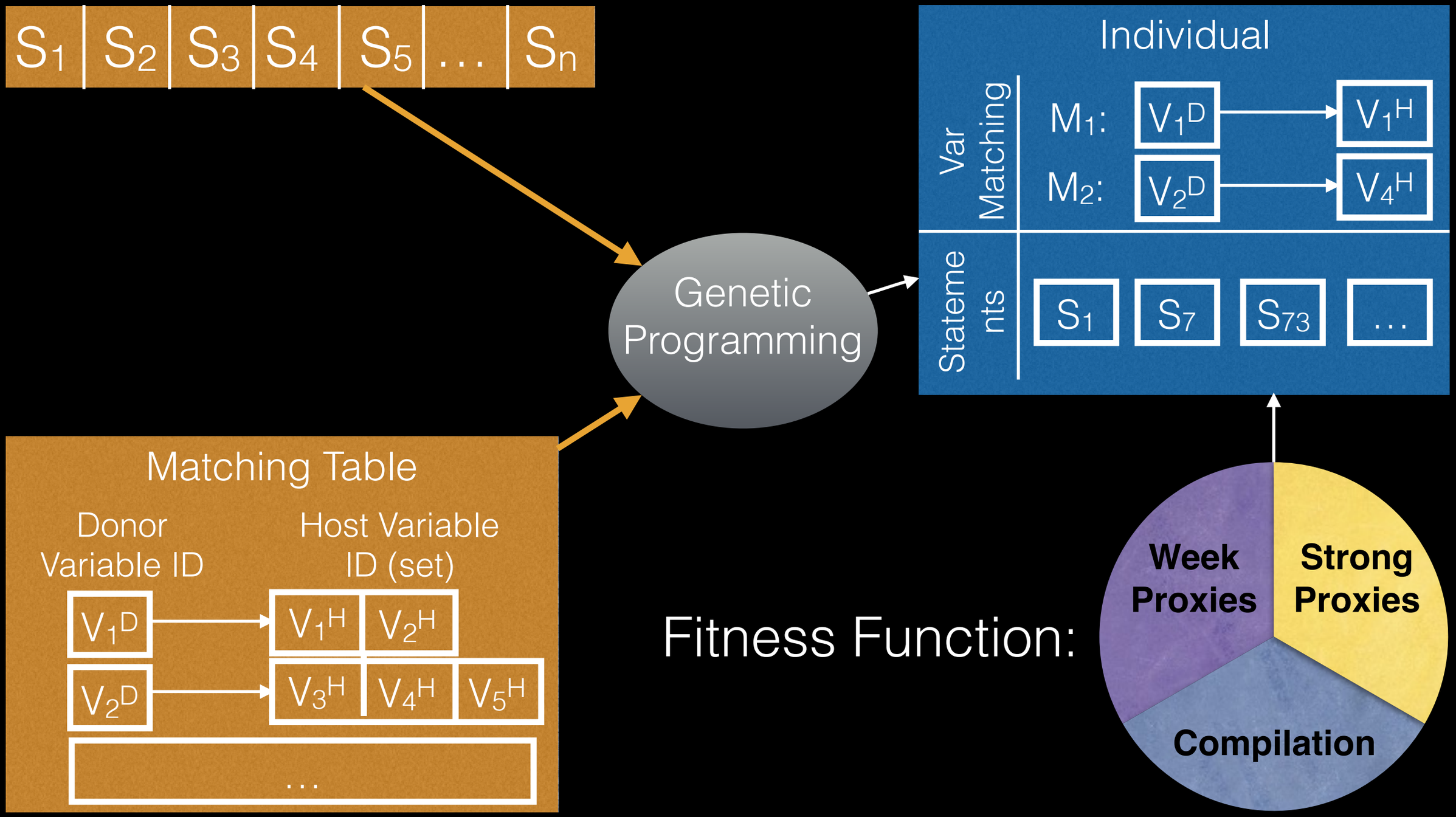
μTrans


Host
Implantation Point
Donor
Organ Entry
Stage 1: Static Analysis
Stage 2: Genetic Programming
Stage 3: Organ Implantation
Organ Test Suite
Host Beneficiary

Stage 1 — Static Analysis



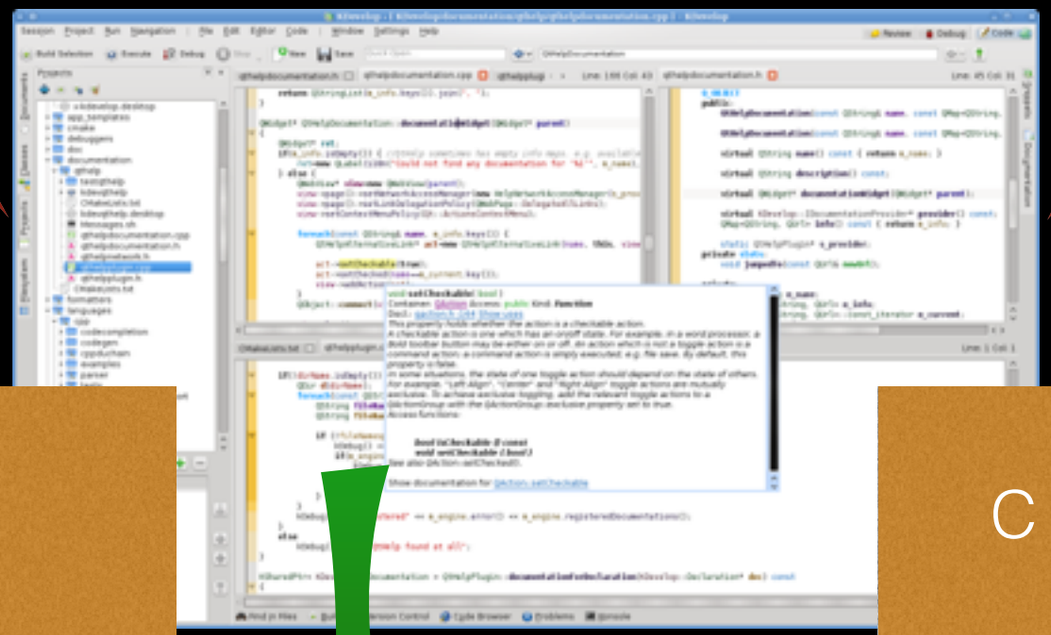
Stage 2 — GP


S1 S2 S3 S4 S5 … Sn
Genetic Programming
Individual
Var Matching
M1: V1D V1H
M2: V2D V4H
Statements
S1 S7 S73 …
Matching Table
Donor Variable ID
Host Variable ID (set)
V1D V1H V2H
V2D V3H V4H V5H
…
Fitness Function:
Week Proxies
Strong Proxies
Compilation

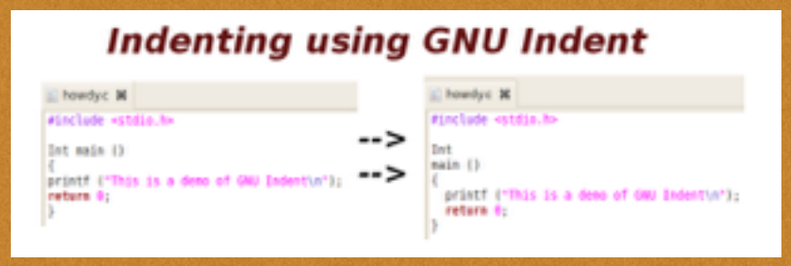
Kate

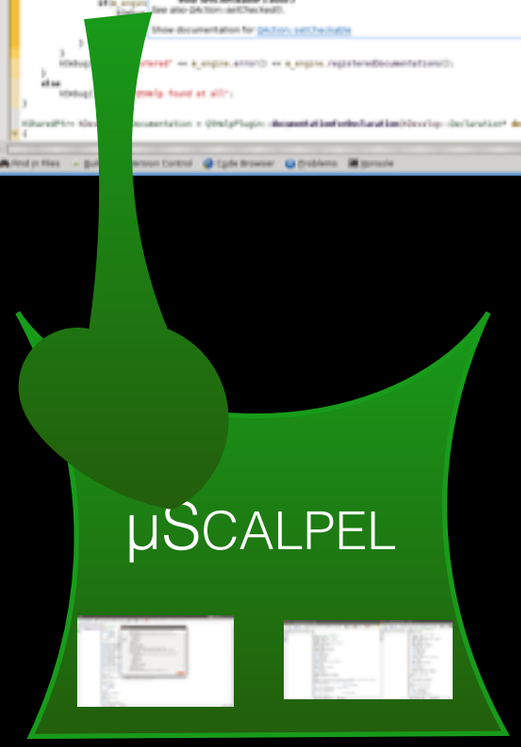
C Layout Feature?

C Call Graphs?



GNU Indent
C Code Indentation
26K LoCs
7 Organ Tests





GNU Cflow
C Call Graph Generator
22k LoCs
13 Organ Tests

Experimental Methodology and Setup

1 x 20

GNU Time

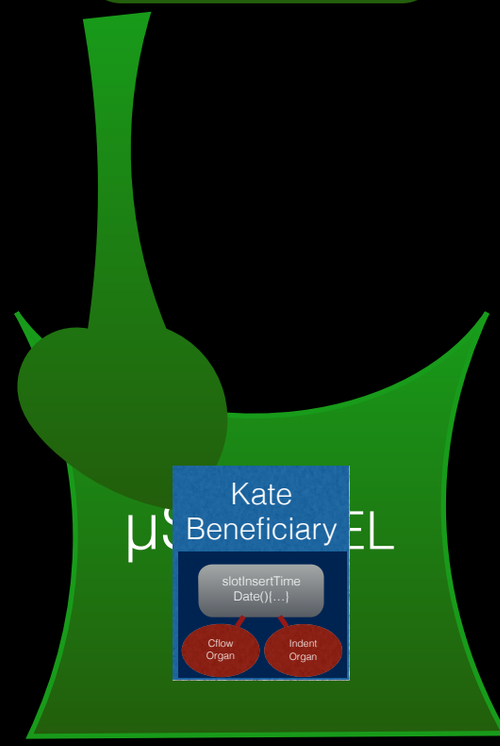
64 bit Ubuntu 14.10
16 GB RAM
8 threads

Kate

Time Date Plugin

slotInsertTime Date(){…}

Organ Test Suite



Donor

OE

Cflow

Indent

tree_output()

indent_single_file()

Validation Test Suites

Coverage Information: Gcov

Research Questions

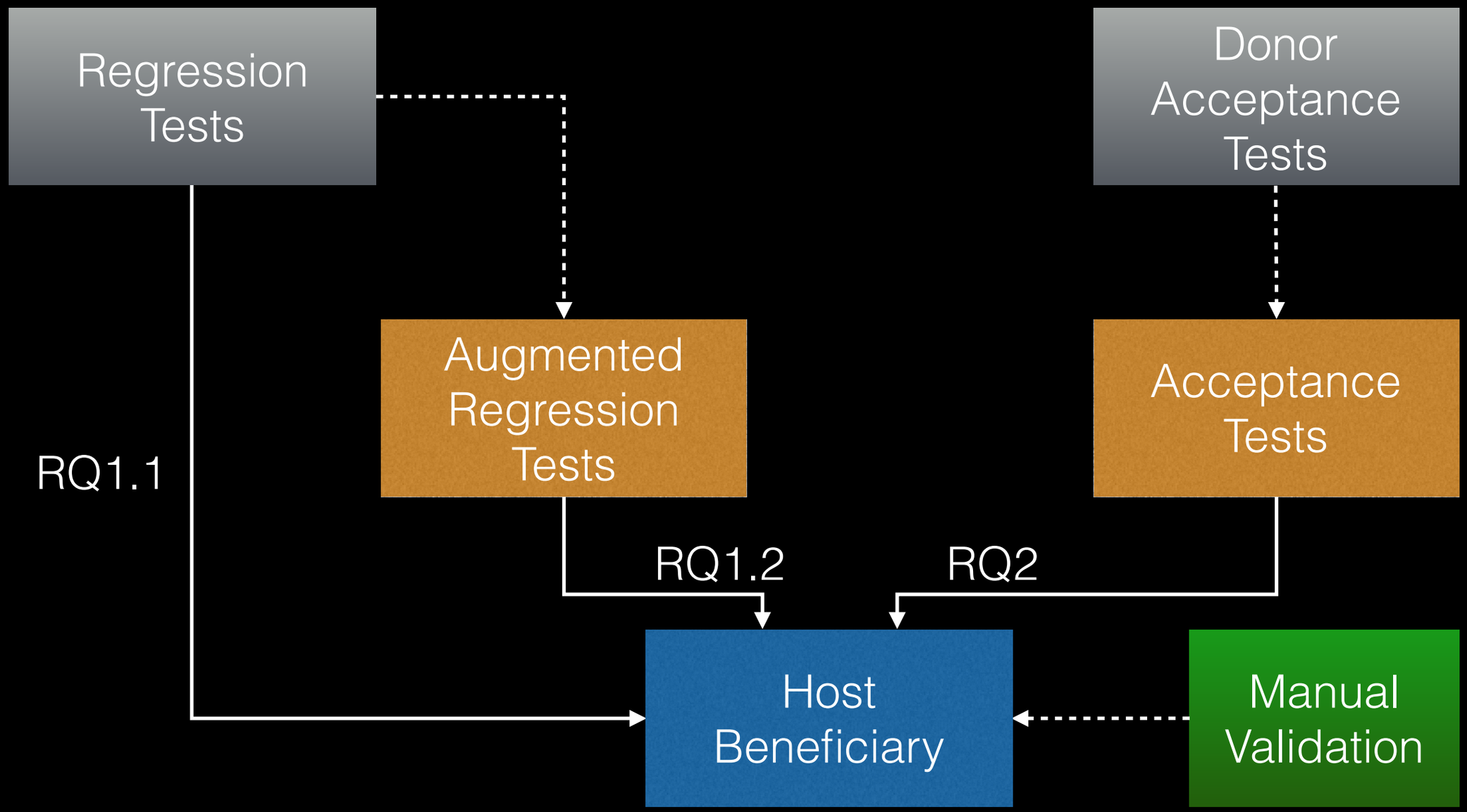
Acceptance Tests

Host

Donor

RQ3: How about the computational effort?

Validation


Regression Tests
Donor Acceptance Tests
Augmented Regression Tests
Acceptance Tests
RQ1.1
RQ1.2
RQ2
Host Beneficiary
Manual Validation

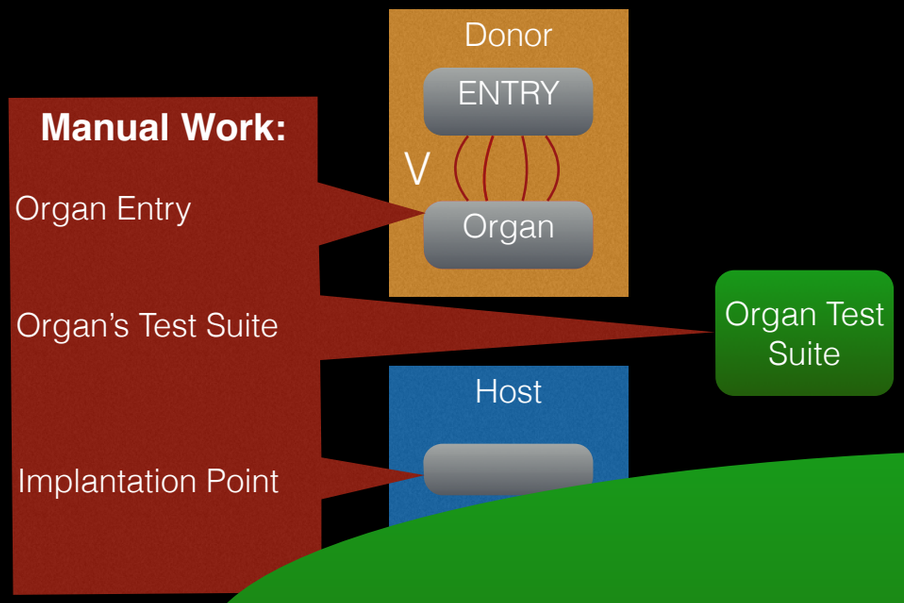
RQ1, RQ2

Donor	Host	All Passed	Organ Test Suite	Regression	Regression++	Acceptance
Cflow	Kate	**16**	**18**	**20**	**17**	**18**
Indent	Kate	**18**	**19**	**20**	**18**	**19**
TOTAL		34/40	37/40	40/40	35/40	37/40
		All Passed	—	RQ1.1	RQ1.2	RQ2

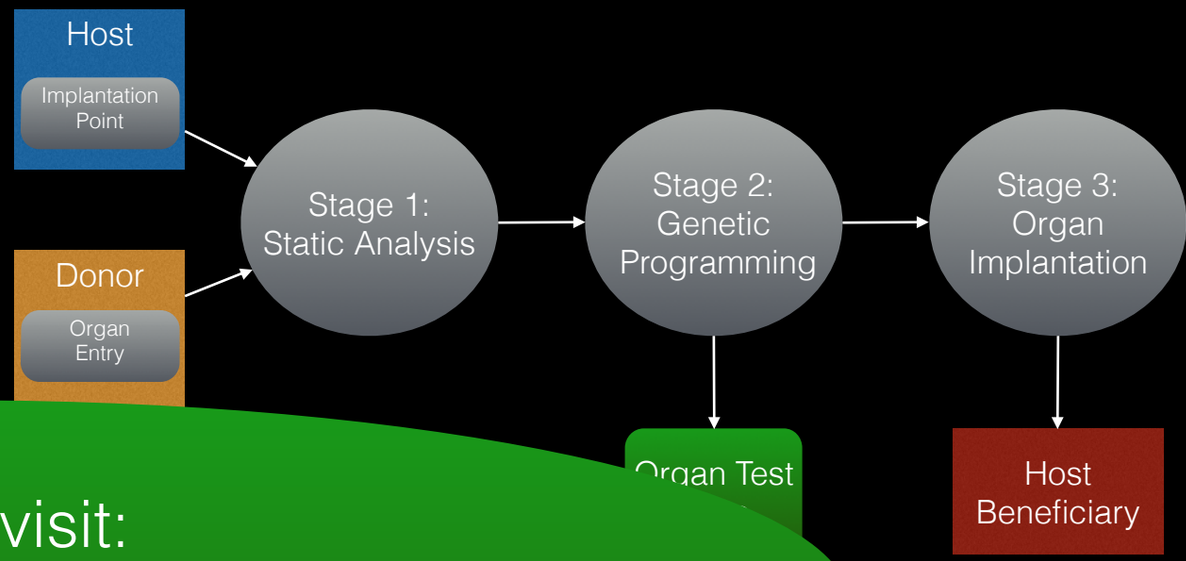
RQ3

		Execution Time (minutes)		
Donor	Host	Average (min)	Std. Dev. (min)	Total (hours)
Cflow	Kate	**101**	**31**	**33**
Indent	Kate	**31**	**6**	**11**
Total		132	18.5	44

Automated Software Transplantation

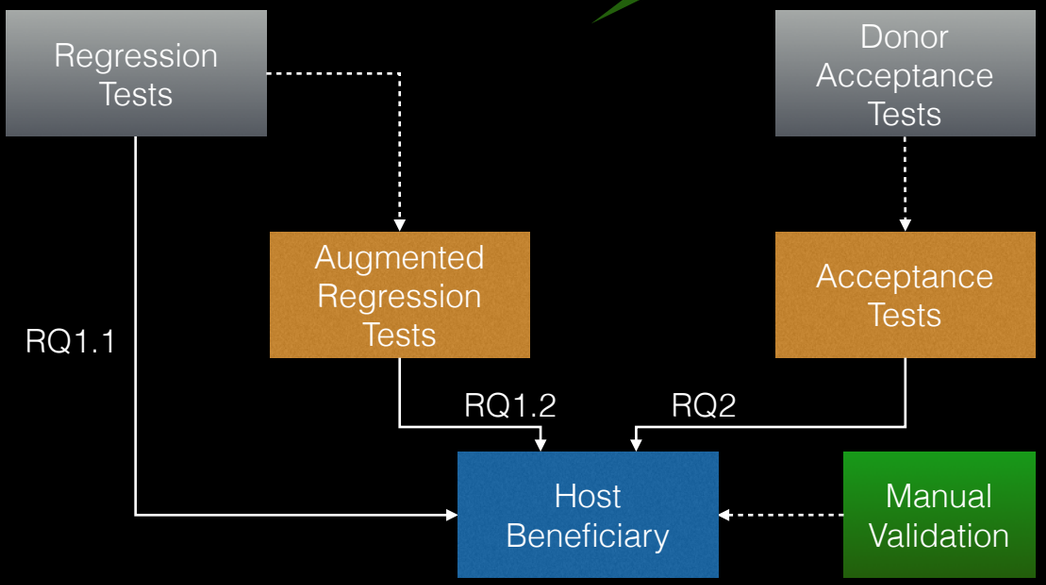


μTrans



Please visit:
http://crest.cs.ucl.ac.uk/autotransplantation
— tool, experiments, data sets available

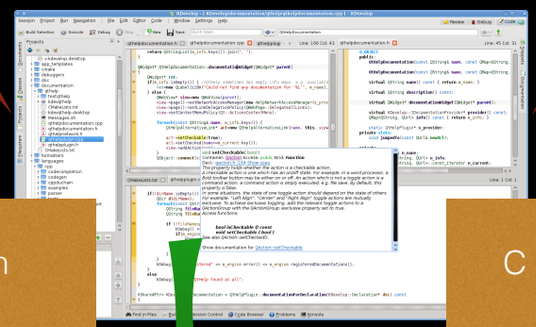
Validation



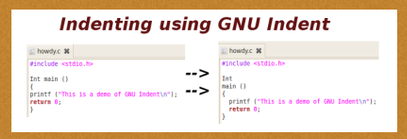
Kate

C Layout Feature?

C Call Graphs?



GNU Indent
C Code Indentation
26K LoCs
7 Organ Tests



μSCALPEL

GNU Cflow
C Call Graph Generator
22k LoCs
13 Organ Tests