Automated Software Transplantation

Alexandru Marginean
Why Autotransplantation?
Why Autotransplantation?

Alexandru Marginean - Automated Software Transplantation
Why Autotransplantation?

Why not handle H.264?
Why Autotransplantation?

Why not handle H.264?
Why Autotransplantation?

Why not handle H.264?
Why Autotransplantation?

Why not handle H.264?

Video Player

Start from scratch

change picture
Why Autotransplantation?

- Why not handle H.264?

- Start from scratch

- Change picture

Alexandru Marginean - Automated Software Transplantation
Why Autotransplantation?

Why not handle H.264?

Check open source repositories

Video Player

Start from scratch

change picture

Alexandru Marginean - Automated Software Transplantation
Why Autotransplantation?

Why not handle H.264?

Start from scratch

Check open source repositories

Video Player

change picture

Start from scratch

Alexandru Marginean - Automated Software Transplantation
Open Source Repositories

Github Popularity (repositories)

Alexandru Marginean - Automated Software Transplantation
Motivation

• A lot of time is waisted in extending the functionality of an existing software system.

• Clone detection, code migration, code salvaging, reuse, dependency analyse, feature extraction techniques.

• The overall process is still manual, tedious and error prone.

• A lot of functionalities required for a new software, is already available on source code repositories.
Human Organ Transplantation
Automated Software Transplantation

Donor
Host
Organ
Vein
Ip
Automated Software Transplantation

Donor

Host

Organ

Vein

I_P

ENTRY

D

V

O

H

I_P

O
Automated Software Transplantation

Donor

Host

Organ

Vein

Ip

Ip
Video Encoder

~100 players

WMP
Real Player
VLC

H.264

x264
Video Encoder

~100 players

H.264

WMP
Real Player
VLC

x264
Video Encoder

- ~100 players
- WMP
- Real Player
- VLC
- H.264
- x264

Alexandru Marginean - Automated Software Transplantation
Video Encoder

~100 players

WMP
Real Player

VLC

μSCALPEL

H.264

x264

Alexandru Marginean - Automated Software Transplantation
Kate

μScalpel

Alexandru Marginean - Automated Software Transplantation
Kate

C Layout Feature?

C Call Graphs?

μSCALPEL

Alexandru Marginean - Automated Software Transplantation
Kate

Alexandru Marginean - Automated Software Transplantation
Kate

Alexandru Marginean - Automated Software Transplantation
Kate
Kate

μSCALPEL

Alexandru Marginean - Automated Software Transplantation
Approach

Stage 1
- Donor
  - Entry points
  - Backward Slicing
  - Forward Slicing
- Vein and Organ
- A Vein
- Organ
- SDG
- Matching Table
  - Donor --> Host
  - Semantic Analysis

Stage 2
- Ice Box Test Suite
- GP
- Transplantable Organ

Stage 3
- Transplantable Organ
- Host'

Manual process
Automated process

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- Insertion points
- Host Entry points
- Donor

- Matching Table: Donor <-> Host
- Backward Slicing
- Forward Slicing

- A Vein
- Organ
- SDG

- Fix slicing

- Semantics analysis

- Transplantable Organ

- OE

- ENTRY

- H

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- Host Entry points
- Donor Entry points
- Matching Table
  - Donor <---> Host
- Semantic Analysis
- Fix slicing
- Backward Slicing
- Forward Slicing
- Vein and Organ
- A Vein
- Organ
- SDG
- Ice Box Test Suite
- Manual process
- Automated process

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- Insertion points
- Host Entry points
- Donor
- Matching Table
  Donor <-> Host
  - Semantic Analysis
  - Backward Slicing
  - Forward Slicing
  - A Vein
  - Organ
  - SDG

- Fix slicing

- Ice Box Test Suite
  - Manual process
  - Automated process

Transplantable Organ

SDG

OE

ENTRY

Donor

H

IP

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- Insertion points
- Host Entry points
- Donor
- Matching Table
- Backward Slicing
- Forward Slicing
- Semantic Analysis
- Fix slicing
- A Vein
- Organ
- SDG
-Matching Table
- Donor <-> Host

- Ice Box Test Suite
- Manual process
- Automated process

- Transplantable Organ Host

- Vein and Organ

- Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- **Insertion points**
- **Host Entry points**
- **Donor**
  - Backward Slicing
  - A Vein
  - Organ
  - SDG
- **Forward Slicing**
- **Matching Table**
  - Donor <---> Host

**Semantic Analysis**

**Transplantable Organ Host**

**H**

**Axelndru Marginean - Automated Software Transplantation**
Stage 1 - Static Analysis

Matching Table

Donor <---> Host

Insertion points

Semantic Analysis

Backward Slicing

Forward Slicing

A Vein

Organ

SDG

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- **Insertion points**
- **Host Entry points**
- **Donor Entry points**

**Matching Table**

- Donor: X -> Host: A, B, C

**Semantic Analysis**

- Backward Slicing
- Forward Slicing
- SDG

**Fix slicing**

- Vein and Organ
- A Vein
- Organ
- SDG

**Manual process**

- Automated process

**Vein and Organ Backward Slicing**

**Vein and Organ Forward Slicing**

**A Vein**

**Organ**

**SDG**

**MATCHING TABLE**

- Donor <-> Host

**Vein**

**ENTRY**

**Organ**

**OE**

**H**

**IP**

**Matching Table**

- Donor: X -> Host: A, B, C

**Alexandru Marginean - Automated Software Transplantation**
Stage 1 - Static Analysis

Semantic Analysis

Matching Table

Vein and Organ

A Vein

Organ

SDG

Donor --> Host

Insertion points

Entry points

Donor

Backward Slicing

Forward Slicing

Manual process

Automated process

A Vein

Organ

SDG

Matching Table

Vein and Organ

H

Ip

Matching Table

Donor

ENTRY

VEIN

OE

Organ

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- **Insertion points**
- **Host**
- **Donor**

**Matching Table**
- Donor <---> Host
- A Vein
- Organ
- SDG

**Semantic Analysis**
- Fix slicing
- Backward Slicing
- Forward Slicing
- Vein and Organ

**SDG**

**H**

**IP**

**UCL**

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

- Fix slicing
- Matching Table
- SDG
- Manual process
- Organ
- Vein
- Entry points
- Semantics Analysis
- Backward Slicing
- Forward Slicing
- A Vein
- Organ
- Donor <---> Host

Stm: x = 10; -> Decl int x;

Alexandru Marginean - Automated Software Transplantation
Stage 1 - Static Analysis

Stage 2

Matching Table

SDG

A Vein
Organ
SDG

Vein and Organ

Semantic Analysis

Backward Slicing
Forward Slicing

Fix Slicing

 ENTRY

OE

SDG

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp

Matching Table

Donor Variable ID

$V_1^D$  $V_2^D$

Host Variable ID (set)

$V_1^H$  $V_2^H$  $V_3^H$  $V_4^H$  $V_5^H$

... 

Stage 2

Ice Box Test Suite

GP

Matching Table

Individual

Var Matching

M1: $V_1^D$  $V_1^H$

M2: $V_2^D$  $V_4^H$

Statements

$S_1$  $S_7$  $S_{73}$  ...

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp

Matching Table

Donor Variable ID

<table>
<thead>
<tr>
<th>(V_1^D)</th>
<th>(V_2^D)</th>
<th>...</th>
</tr>
</thead>
</table>

Host Variable ID (set)

<table>
<thead>
<tr>
<th>(V_1^H)</th>
<th>(V_2^H)</th>
<th>(V_3^H)</th>
<th>(V_4^H)</th>
<th>(V_5^H)</th>
<th>...</th>
</tr>
</thead>
</table>

Stage 2

Ice Box Test Suite

Individual Fix-Up

Individual Matching

Var Matching

\[M_1: \quad \begin{align*} V_1^D & \rightarrow V_1^H \\ V_2^D & \rightarrow V_4^H \end{align*} \]

\[M_2: \quad \begin{align*} V_2^D & \rightarrow V_4^H \end{align*} \]

Statements

<table>
<thead>
<tr>
<th>(S_1)</th>
<th>(S_7)</th>
<th>(S_{73})</th>
<th>...</th>
</tr>
</thead>
</table>

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Fitness Evaluation

- **Fixed Individual**
  - **M₁**: $V^D_1 \rightarrow V^H_1$
  - **M₂**: $V^D_2 \rightarrow V^H_4$

- **Statements**
  - $S_1$, $S_7$, $S_73$, ...

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Fitness Evaluation

\[ \text{fitness}(i) = \begin{cases} \frac{1}{3} (1 + \frac{|TX_i|}{|T|} + \frac{|TP_i|}{|T|}) & i \in I_C \\ 0 & i \notin I_C \end{cases} \]
Algorithm 1

with a probability of 0.5. We define two custom, crossover operators: fixed-two-points and uniform crossover. The fixed-two-points operator selects it, the resulting individual will be more fit. The uniform crossover operator produces new individuals. At the end of evolution, an organ's parameters. We then uniformly select one statement with the selected parameters and the statement vector, restricted to each vector.

To meet these constraints, we perused GitHub, SourceForge, and others. We found a wide range of sizes, from 0.4–63k SLoC and the hosts are large, all greater than 50k SLoC. The donors vary in size, including chat, static analysis, sound processing, audio streaming, encryption utility TuxCrypt, and the H.264 codec x264. Our goal is to transplant the functionality of Scalpel to these hosts.

Fifth, the host is the system we seek to augment, so it had to be diverse and operate on C programs. Second, they had to be real-world programs people use. Third, they had to be operable only on C programs. First, it uniformly picks a new variable from among the \( N_{init} \) variables.

For the autotransplantation goal, a viable candidate must, at each generation, we select top 10% most fit individuals (elitism) and insert them into the new generation. We use tournament selection to select 60% of the population for reproduction. Parents must be compilable; if the proportion of possible parents is less than 60% of the population, we replace them.

<table>
<thead>
<tr>
<th>Var Matching</th>
<th>Fixed Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1:</td>
<td>( V^D_1 \rightarrow V^H_1 )</td>
</tr>
<tr>
<td>M2:</td>
<td>( V^D_2 \rightarrow V^H_4 )</td>
</tr>
</tbody>
</table>

Does it compile?

Stage 2 - Fitness Evaluation

\[
\text{fitness}(i) = \begin{cases} 
\frac{1}{3}(1 + \frac{|TX_i|}{|T|} + \frac{|TP_i|}{|T|}) & \text{if } i \in I_C \\
0 & \text{if } i \notin I_C 
\end{cases}
\]

This section explains the subjects, test suites, and research problems for this transplantation.

**Fig. 5** Stage 2 - Fitness Evaluation
Stage 2 - Fitness Evaluation

\[
\text{fitness}(i) = \begin{cases} 
\frac{1}{3} (1 + \frac{|TX_i|}{|T|} + \frac{|TP_i|}{|T|}) & i \in I_C \\
0 & i \notin I_C 
\end{cases}
\]

Week Proxies: Does it executes test cases without crashing?

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Fitness Evaluation

\[
\text{fitness}(i) = \begin{cases} 
\frac{1}{3}(1 + \frac{|TX_i|}{|T|} + \frac{|TP_i|}{|T|}) & i \in I_C \\
0 & i \notin I_C 
\end{cases}
\]

Strong Proxies: Does it produce the correct output?

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp Operators

Replace Mapping

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp Operators

Replace Statement

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp Operators

Variable Matching Statements

\[
\begin{array}{c}
\text{M}_1: \quad V_1^D & \rightarrow & V_1^H \\
\text{M}_2: \quad V_2^D & \rightarrow & V_4^H \\
\end{array}
\]

\[
\begin{array}{cccc}
S_1 & S_7 & S_{73} & \ldots \\
\end{array}
\]

Remove Statement

\[
\begin{array}{c}
\text{M}_1: \quad V_1^D & \rightarrow & V_1^H \\
\text{M}_2: \quad V_2^D & \rightarrow & V_4^H \\
\end{array}
\]

\[
\begin{array}{cccc}
S_1 & \times & S_{73} & \ldots \\
\end{array}
\]
Stage 2 - Gp Operators

Individual

Var Matching

M1: $V_1^D \rightarrow V_1^H$
M2: $V_2^D \rightarrow V_4^H$

Statements

S1 S7 S73 ...

Add Statement

M1: $V_1^D \rightarrow V_1^H$
M2: $V_2^D \rightarrow V_4^H$

Statements

S1 S7 S73 S92 ...

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp Operators

Individual 1
- M₁
- M₂
- S₁
- S₇

Individual 2
- M₃
- M₄
- S₃
- S₉

Crossover Operator

Offspring 1
- M₁
- M₄
- S₁
- S₉

Offspring 2
- M₃
- M₂
- S₃
- S₇

Offspring 3
- M₃
- M₂
- S₁
- S₇
- S₁
- S₇

Alexandru Marginean - Automated Software Transplantation
Stage 2 - Gp Operators

Individual 1

\[
\begin{array}{cc}
M_1 & M_2 \\
S_1 & S_7 \\
\end{array}
\]

Individual 2

\[
\begin{array}{cc}
M_3 & M_4 \\
S_3 & S_9 \\
\end{array}
\]

Crossover Operator

Offspring 1

\[
\begin{array}{cc}
M_1 & M_4 \\
S_1 & S_9 \\
\end{array}
\]

Offspring 2

\[
\begin{array}{cc}
M_3 & M_2 \\
S_3 & S_7 \\
\end{array}
\]

Offspring 3

\[
\begin{array}{cc}
M_3 & M_2 \\
S_1 & S_7 & S_1 & S_7 \\
\end{array}
\]

Random Mapping Selection
Stage 3 - Organ Insertion

Transplantable Organ

**ORGAN_RET entry(host ST){}**

Alexandru Marginean - Automated Software Transplantation
Stage 3 - Organ Insertion

Transplantable Organ

ORGAN_RET entry(host ST){}

H

entry(ST)

Transplantable Organ

Host'

Transplantable Organ

Alexandru Marginean - Automated Software Transplantation
Stage 3 - Organ Insertion

Postoperative Host

Transplantable Organ

ORGAN_RET entry(host ST){}

entry(ST)

Transplantable Organ

Host'

Manual process

Automated process

A Vein

Organ

Vein and Organ

Backward Slicing

Forward Slicing

GP

Semantic Analysis
Demo
Validation

4 validation steps:

- Regression Tests;
- Augmented Regression Tests;
- Acceptance Tests;
- Manual Validation;

Alexandru Marginean - Automated Software Transplantation
# Subject Programs

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Type</th>
<th>Size</th>
<th>Regr.</th>
<th>Isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idct</td>
<td>Donor</td>
<td>2.3k</td>
<td>-</td>
<td>3-5</td>
</tr>
<tr>
<td>Mytar</td>
<td>Donor</td>
<td>0.4k</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Cflow</td>
<td>Donor</td>
<td>25k</td>
<td>-</td>
<td>6-20</td>
</tr>
<tr>
<td>Webserver</td>
<td>Donor</td>
<td>1.7k</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>TuxCrypt</td>
<td>Donor</td>
<td>2.7k</td>
<td>-</td>
<td>4-5</td>
</tr>
<tr>
<td>Pidgin</td>
<td>Host</td>
<td>363k</td>
<td>88</td>
<td>-</td>
</tr>
<tr>
<td>Cflow</td>
<td>Host</td>
<td>25k</td>
<td>21</td>
<td>-</td>
</tr>
<tr>
<td>SoX</td>
<td>Host</td>
<td>43k</td>
<td>157</td>
<td>-</td>
</tr>
<tr>
<td>Case Study</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td>Donor</td>
<td>63k</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>GNU Indent</td>
<td>Donor</td>
<td>26k</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>GNU cflow</td>
<td>Donor</td>
<td>25k</td>
<td>-</td>
<td>8</td>
</tr>
<tr>
<td>Kate</td>
<td>Host</td>
<td>43k</td>
<td>42</td>
<td>-</td>
</tr>
<tr>
<td>VLC</td>
<td>Host</td>
<td>422k</td>
<td>27</td>
<td>-</td>
</tr>
</tbody>
</table>
## Results

### Case Studies

<table>
<thead>
<tr>
<th>Donor</th>
<th>Host</th>
<th>Passed</th>
<th>Regression</th>
<th>Regression++</th>
<th>Acceptance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>x264</td>
<td>VLC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>26 (hours)</td>
</tr>
<tr>
<td>cflow</td>
<td>Kate</td>
<td>16</td>
<td>20*</td>
<td>17</td>
<td>18</td>
<td>101</td>
</tr>
<tr>
<td>Indent</td>
<td>Kate</td>
<td>18</td>
<td>20*</td>
<td>18</td>
<td>19</td>
<td>31</td>
</tr>
</tbody>
</table>

---

*Alexandru Marginean - Automated Software Transplantation*
### Empirical Study

<table>
<thead>
<tr>
<th>Donor</th>
<th>Host</th>
<th>All Passed</th>
<th>Regression</th>
<th>Regression++</th>
<th>Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idct</td>
<td>Pidgin</td>
<td>16</td>
<td>20</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>Mytar</td>
<td>Pidgin</td>
<td>16</td>
<td>20</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>Web</td>
<td>Pidgin</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>Cflow</td>
<td>Pidgin</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>Tux</td>
<td>Pidgin</td>
<td>15</td>
<td>20</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>188/300</strong></td>
<td><strong>233/300</strong></td>
<td><strong>196/300</strong></td>
<td><strong>256/300</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Donor</th>
<th>Host</th>
<th>All Passed</th>
<th>Regression</th>
<th>Regression++</th>
<th>Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idct</td>
<td>Cflow</td>
<td>16</td>
<td>17</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Mytar</td>
<td>Cflow</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>Web</td>
<td>Cflow</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Cflow</td>
<td>Cflow</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Tux</td>
<td>Cflow</td>
<td>14</td>
<td>15</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>188/300</strong></td>
<td><strong>233/300</strong></td>
<td><strong>196/300</strong></td>
<td><strong>256/300</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Donor</th>
<th>Host</th>
<th>All Passed</th>
<th>Regression</th>
<th>Regression++</th>
<th>Acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idct</td>
<td>SoX</td>
<td>15</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>Mytar</td>
<td>SoX</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>Web</td>
<td>SoX</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>Cflow</td>
<td>SoX</td>
<td>14</td>
<td>16</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>Tux</td>
<td>SoX</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>188/300</strong></td>
<td><strong>233/300</strong></td>
<td><strong>196/300</strong></td>
<td><strong>256/300</strong></td>
</tr>
</tbody>
</table>

*Alexandru Marginean - Automated Software Transplantation*
Publication


Automated Transplantation of Call Graph and Layout Features into Kate
Alexandru Marginean, Earl T. Barr, Mark Harman, and Yue Jia
UCL, Department of Computer Science, CREST Centre

Abstract. We report the automated transplantation of two features currently missing from Kate: call graph presentation and automatic layout for C programs, which have been requested by users on the Kate development forum. Our approach uses a lightweight annotation system with Search Based techniques augmented by static analysis for automated transplantation. The results are promising: on average, our tool requires 111 minutes of standard desktop machine time to transplant the call graphs features, and 31 minutes to transplant the layout feature. We repeated each experiment 20 times and validated the resulting transplants using unit, regression and acceptance test suites. In 34 of 40 experiments conducted our search-based autotransplantation tool, µScalpel, was able to successfully transplant the new functionality, passing all tests.

1 Introduction
We recently introduced a search based technique for automated software transplantation [22]. Guided by dependence analysis and testing, our approach uses a variant of genetic programming to identify and extract useful functionality from a donor program, and transplant it into a (possibly unrelated) host program. We implemented our approach as a tool called µScalpel, which is publicly available [14].

In this challenge paper, we illustrate the way in which realistic, scalable, and useful real-world transplantation can be achieved using µScalpel. We apply our tool to the NSISSE 2015 challenge project KATE, a popular text editor based on KDE. Its rich feature set and available plugins make it a popular, lightweight IDE for C developers. We perform two automated transplantations using µScalpel. In the first one, we transplant call graph drawing ability from the GNU utility program of line, to augment Kate with the ability to construct and display call graphs. This is a useful feature for a lightweight IDE, like Kate, and would clearly be nontrivial to implement from scratch. Using our search based autotransplantation, µScalpel, the developer merely needs to identify the entry point of the source code in the donor program (callw in this case) and the tool will do the rest; extracting the relevant code, matching names spaces between host and donor and executing regressions, unit and acceptance tests. Like much previous work on genetic programming [1-2], our approach relies critically on the availability of high quality test suites. We do not directly address this issue in the present paper, but believe
Automated Software Transplantation

**Donor**

**Host**

**Organ**

**Vein**

**IP**

---

**Approach**

Stage 1

- Donor
- Host

Stage 2

- Vein and Organ
- Ice Box Test Suite

Stage 3

- Transplantable Organ

**Matching Table**

**Insertion points**

**Manual process**

**Automated process**

**Validation**

4 validation steps:

- Regression Tests;
- Augmented Regression Tests;
- Acceptance Tests;
- Manual Validation;

**Results**

**Case Studies**

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Test Suites</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Donor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passed</td>
<td>1</td>
<td>1</td>
<td>26 (hours)</td>
</tr>
<tr>
<td>Regression</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Regression++</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Acceptance</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cflow</td>
<td>16</td>
<td>20*</td>
<td>101</td>
</tr>
<tr>
<td>Kate</td>
<td>17</td>
<td>18</td>
<td>31</td>
</tr>
<tr>
<td>Indent</td>
<td>18</td>
<td>20*</td>
<td>19</td>
</tr>
</tbody>
</table>