Software Component Synthesis: Theory and Supporting Tools

Dick Hamlet
Portland State University
E.T.S. Walton Fellow
Software Component Synthesis: Theory and Supporting Tools

Dick Hamlet
Portland State University
E.T.S. Walton Fellow

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

fonduireacht eolaiochta éireann

NDP
Components in Engineering

Mechanical component:

- An independently specified and produced (small) part of a machine. Example: A rivet
Components in Engineering

Mechanical component:

- An independently specified and produced (small) part of a machine. Example: A rivet

Component specification:

| Functional: | rivet size, material, etc. |
| Quality:    | tolerance, hardness        |
Components in Engineering

Mechanical component:

- An independently specified and produced (small) part of a machine. Example: A rivet

Component specification:
Components in Engineering

Mechanical component:

▶ An independently specified and produced (small) part of a machine. Example: A rivet

Component specification:

Engineering design is combining components and calculating properties of the resulting system

Component-based design spectacularly successful in electrical, mechanical engineering
Software Components? (Me too!)

Definition (Szyperski): An executable, independently-deployable software unit with a black-box interface

In practice: Anything from a small utility program (e.g., UNIX ‘cp’) to a large system (e.g., Adobe acrobat)

In this research: An executable program with one floating-point input and output, no persistent internal state
Component-based Software

Component developer:

► Specify components: catalogue
► Complete access to all component details
► No knowledge of subsequent application
Component-based Software

Component developer:
► Specify components: catalogue
► Complete access to all component details
► *No* knowledge of subsequent application

System designer:
► Devise system structure (architecture)
► Select components from catalogue
► Calculate system properties
► *No* access to actual components
Specify function $f$ and run time $T(x)$ on input $x$

Mathematical specifications:

- Program logic (Hoare)
- Algebraic equations (Gougen)
- Functional set theory (Mills)

Two components $C_1$ and $C_2$ in series:

$$f(x) = f_2(f_1(x))$$
$$T(x) = T_1(x) + T_2(f_1(x))$$

Practicality: Deriving $T$ and $f$ is difficult
Sample the input space to estimate $f$ and $T$.

**Operational-profile** problem:

- Input domain is on the order of size $2^{64} \approx 10^{19}$ but only $10^5$ samples are practical
- Valid sampling requires a distribution
- The distribution a component sees in a system depends on the *system* input distribution and the *system* structure, **not available to the component developer**
Solving the Operational-profile Problem

- Partition component input space into subdomains $D = \bigcup_{i=1}^{n} S_i$

- Approximate component behavior on each subdomain: $<v_1, v_2, ..., v_n> \equiv <v_i>_{i=1}^{n}$

- Profile $<w_1, w_2, ..., w_n>$ as a weighting over subdomains
  - Component developer measures $<v_i>_{i=1}^{n}$
  - System designer later applies $<w_i>_{i=1}^{n}$

  to get a system average: $\frac{1}{n} \sum_{i=1}^{n} w_i v_i$
Algebra of Equivalent Components

‘Equivalent component’ construction rules:
- sequence
- conditional
- iteration

\[ \{\text{component specs}\} \xrightarrow{\text{rule}} \text{equiv-component spec} \]

Repeatedly apply the rules to analyze a system of arbitrary complexity
Subdomain, Vector Notation

Component subdomains: \{S^B_1, S^B_2, \ldots, S^B_n\}
\{S^C_1, S^C_2, \ldots, S^C_m\}

Functional vectors: 
\langle v^B_1, v^B_2, \ldots, v^B_n \rangle \Rightarrow \langle v^U_1, v^U_2, \ldots v^U_n \rangle 

Run-time vectors: 
\langle t^B_1, t^B_2, \ldots, t^B_n \rangle \Rightarrow \langle t^U_1, t^U_2, \ldots t^U_n \rangle
Sequence rule: \( C_1; C_2 \)

Subdomains: \( S^U_i = S^C_1_i \)

Suppose \( v^C_1_i \in S^C_2_j \). Then:

Functional values: \( v^U_i = v^C_2_j \)

Run-time values: \( t^U_i = t^C_1_i + t^C_2_j \)
Conditional: \( \text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi} \)

Let:

\[
D_T = \{ x \mid B(x) \}  \\
D_F = \{ x \mid \neg B(x) \}
\]

On subdomains \( D_T \cap S_{i}^{C_1} \):

\[
\begin{align*}
  v_{k}^{U} &= v_{i}^{C_1}  \\
  t_{k}^{U} &= t_{p}^{B} + t_{i}^{C_1}
\end{align*}
\]

On subdomains \( D_F \cap S_{j}^{C_2} \):

\[
\begin{align*}
  v_{k}^{U} &= v_{j}^{C_2}  \\
  t_{k}^{U} &= t_{p}^{B} + t_{j}^{C_2}
\end{align*}
\]
**Iteration rule:** while $B$ do $C$ od

- **Unwind to:**
  
  ```
  if $B$ then $C$ else $I$ fi; while $B$ do $C$ od
  ```

  where $I$ is the component computing identity with zero run time

- **Repeat until residual loop disappears in each subdomain**

- **Use conditional and sequence rules**
Piecewise-linear Approximation

Replace the vector constants $v_i, t_i$ with pairs $(slope, intercept)_i$

of the best-fit line over the subdomain
Piecewise-linear Approximation

Replace the vector constants $v_i, t_i$ with pairs $(slope, intercept)_i$ of the best-fit line over the subdomain

► Advantages:
  ▶ Better subdomains for sequences
  ▶ Exact representation of the identity component $I$
Piecewise-linear Approximation

Replace the vector constants $v_i, t_i$ with pairs $(slope, intercept)_i$ of the best-fit line over the subdomain.

- **Advantages:**
  - Better subdomains for sequences
  - Exact representation of the identity component $I$

- **Disadvantage:** Can’t be used with non-numeric data types
Ideal Supporting Tools

Components are measured to drive a computer-aided design (CAD) tool that does system-design calculations.

Systems are designed using a components catalogue, without any access to component code and without constructing or executing any system code.
Tools for Component Developers

Create catalog entry (specification) from:

- Subdomain list
- Component code

```perl
#!/usr/bin/perl
$x = <STDIN>;
$y = ((x**2)/10)*exp(-(x**2)/1500) + 35;
if ($x < 61) {
  $t = 5 + 2*sin(x/5);
}
elsif ($x >= 61 && $x < 83) {
  $t = 12 + 3*sin(x/3);
}
else {
  $t = 13;
}
$t = $t*.2;
print STDERR $t,"\n";
print $y,"\n";
```

1.bin
0 20 13
20 40 13
40 61 13
61 70 13
70 83 13
83 100 13
Tools for Component Developers

Create catalog entry (specification) from:

- Subdomain list
- Component code

![Graph showing functional and run-time values versus input.](image)
Approximating Component Behavior

Step-function approximation

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Function</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 20)</td>
<td>21.28</td>
<td>18.38</td>
</tr>
<tr>
<td>[20, 40)</td>
<td>9.52</td>
<td>32.97</td>
</tr>
<tr>
<td>[40, 61)</td>
<td>9.49</td>
<td>31.37</td>
</tr>
<tr>
<td>[61, 70)</td>
<td>6.28</td>
<td>17.32</td>
</tr>
<tr>
<td>[70, 83)</td>
<td>7.46</td>
<td>18.86</td>
</tr>
<tr>
<td>[83, 100)</td>
<td>4.40</td>
<td>0.00</td>
</tr>
<tr>
<td>Weighted errors</td>
<td>10.4</td>
<td>20.9</td>
</tr>
</tbody>
</table>
Approximating Component Behavior

Step-function approximation

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Root-mean-square % errors</th>
<th>Function Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 10)</td>
<td>7.65</td>
<td>9.87</td>
</tr>
<tr>
<td>[10, 20)</td>
<td>11.84</td>
<td>20.53</td>
</tr>
<tr>
<td>[20, 30)</td>
<td>7.50</td>
<td>12.16</td>
</tr>
<tr>
<td>[30, 40)</td>
<td>2.03</td>
<td>13.74</td>
</tr>
<tr>
<td>[40, 50)</td>
<td>3.03</td>
<td>18.84</td>
</tr>
<tr>
<td>[50, 61)</td>
<td>6.31</td>
<td>10.51</td>
</tr>
<tr>
<td>[61, 65)</td>
<td>3.14</td>
<td>5.72</td>
</tr>
<tr>
<td>[65, 70)</td>
<td>3.12</td>
<td>10.18</td>
</tr>
<tr>
<td>[70, 76)</td>
<td>4.11</td>
<td>12.97</td>
</tr>
<tr>
<td>[76, 83)</td>
<td>3.29</td>
<td>3.70</td>
</tr>
<tr>
<td>[83, 91)</td>
<td>2.87</td>
<td>0.00</td>
</tr>
<tr>
<td>[91, 100)</td>
<td>1.51</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Weighted errors 5.0 10.5
Approximating Component Behavior

Piecewise-linear approximation

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Root-mean-square % errors</th>
<th>Function Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 20)</td>
<td>3.87</td>
<td>13.97</td>
</tr>
<tr>
<td>(20, 40)</td>
<td>2.37</td>
<td>8.17</td>
</tr>
<tr>
<td>(40, 61)</td>
<td>1.30</td>
<td>14.55</td>
</tr>
<tr>
<td>(61, 70)</td>
<td>0.17</td>
<td>2.05</td>
</tr>
<tr>
<td>(70, 83)</td>
<td>0.63</td>
<td>6.07</td>
</tr>
<tr>
<td>(83, 100)</td>
<td>0.74</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Weighted errors 1.7 8.5
Tools for System Designers

(CAD) Synthesize Equivalent component from:

- System structure description
- Component-specification list (catalogue)
Tools for System Designers

(CAD) Synthesize Equivalent component from:
- System structure description
- Component-specification list (catalogue)
Tools for System Designers

(CAD) Synthesize Equivalent component from:

- System structure description
- Component-specification list (catalogue)

```
1 2 3 L S 4 5 6 C S
1.ccf
2.ccf
3.ccf
4.ccf
5.ccf
6.ccf
```
‘Validation’ Experiments

- Tool/theory debugging
- Creation of artificial components
- To be investigated:
  - Accuracy vs. subdomain size
  - Step-function vs. piecewise-linear approximation
  - Choice of subdomains
Artificial Components

Monitoring behavior in experiments (UNIX)

1. Time component externally

```bash
#!/bin/tcsh shell script

# get time in $Elapsed0 variable

perl ComPonent

# execute code, output to stdout

# get time in $Elapsed1
# compute $Elapsed = $Elapsed1 - $Elapsed0

echo $Elapsed > /dev/stderr # send time to stderr
```
Monitoring behavior in experiments (UNIX)

2. Wrap component to time itself

```perl
#! /usr/bin/perl

# call UNIX timer function

# code for Component here:
while ($LongTime) { ... } # runs 200 ms
print "99\n"; # output to stdout

# call UNIX timer function
$Diff = ...; # subtract times
print STDERR "$Diff\n"; # time to stderr
```
Artificial Components

Monitoring behavior in experiments (UNIX)

3. Fake the actual behavior

#!/usr/bin/perl
print "99\n"; #output to stdout
print STDERR "200\n"; # time to stderr
Artificial Components

Monitoring behavior in experiments (UNIX)

- Can’t tell the difference!
- Fake is *much* faster
- No operating-system variability
- Measure and replace real components with fakes
Case-study System and Components
Case-study System and Components

<table>
<thead>
<tr>
<th>Comp</th>
<th>Function $y = f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$y = \frac{1}{16}x^2e^{-x^2/1500} + 35$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{false } 30 \leq x &lt; 70$</td>
</tr>
<tr>
<td>3</td>
<td>(see graph)</td>
</tr>
<tr>
<td>4</td>
<td>$\text{true } x \geq 50$</td>
</tr>
<tr>
<td>5</td>
<td>$y = \begin{cases} 90 - 2.5x, &amp; x &lt; 30 \ 15 + 1.1(x - 30), &amp; x \geq 30 \end{cases}$</td>
</tr>
<tr>
<td>6</td>
<td>(same as 1)</td>
</tr>
</tbody>
</table>
Beginning system: Polish 1 2 3 L S 4 5 6 C S
Using piecewise-linear approximation

Loop: WHILE 2.bin DO 3.bin OD -> theory1
   Conditional: IF 2.bin THEN 3.bin ELSE FI ->
                once.ccf -> again.ccf
(again.ccf stripped of false subdomains -- 2 true)
Series: again.ccf; once.ccf -> another.ccf ->
        again.ccf (2 still true)
Series: again.ccf; once.ccf -> another.ccf ->
        again.ccf (2 still true)
Series: again.ccf; once.ccf -> another.ccf -> again.ccf

Series: 1.bin;theory1 -> theory2

Conditional: IF 4.bin THEN 5.bin ELSE 1.bin FI -> theory3

Series: theory2;theory3 -> theory4
## CAD Tool Execution

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Function</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0.000,12.239)</td>
<td>26.11</td>
<td>11.25</td>
</tr>
<tr>
<td>[12.239,19.039)</td>
<td>74.43</td>
<td>6.74</td>
</tr>
<tr>
<td>[19.039,20.000)</td>
<td>64.85</td>
<td>9.01</td>
</tr>
<tr>
<td>[20.000,20.351)</td>
<td>53.74</td>
<td>7.24</td>
</tr>
<tr>
<td>[20.351,23.525)</td>
<td>70.23</td>
<td>11.11</td>
</tr>
<tr>
<td>[23.525,30.666)</td>
<td>59.68</td>
<td>13.27</td>
</tr>
<tr>
<td>[30.666,33.840)</td>
<td>70.23</td>
<td>16.63</td>
</tr>
<tr>
<td>[33.840,40.000)</td>
<td>60.54</td>
<td>18.86</td>
</tr>
<tr>
<td>[40.000,45.026)</td>
<td>61.74</td>
<td>18.91</td>
</tr>
<tr>
<td>[45.026,48.359)</td>
<td>70.23</td>
<td>16.61</td>
</tr>
<tr>
<td>[48.359,55.857)</td>
<td>59.68</td>
<td>13.30</td>
</tr>
<tr>
<td>[55.857,59.189)</td>
<td>70.23</td>
<td>11.19</td>
</tr>
<tr>
<td>[59.189,61.000)</td>
<td>54.94</td>
<td>7.45</td>
</tr>
<tr>
<td>[61.000,64.551)</td>
<td>62.51</td>
<td>10.67</td>
</tr>
<tr>
<td>[64.551,70.000)</td>
<td>72.37</td>
<td>7.82</td>
</tr>
<tr>
<td>[70.000,73.345)</td>
<td>79.22</td>
<td>7.79</td>
</tr>
<tr>
<td>[73.345,83.000)</td>
<td>32.22</td>
<td>12.67</td>
</tr>
<tr>
<td>[83.000,100.000)</td>
<td>24.32</td>
<td>12.60</td>
</tr>
</tbody>
</table>
CAD Tool Execution

Functional and Run-time values

Input

- p.23
## Subdomains and Accuracy

Successively halve subdomains:

**Step-function Approximation**

<table>
<thead>
<tr>
<th>Subd count</th>
<th>Functional % error</th>
<th>Run-time % error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall Max Mean &gt; 5</td>
<td>Overall Max Mean &gt; 5</td>
</tr>
<tr>
<td>6</td>
<td>– ABORTED –</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5.16 12.96 3.85 6</td>
<td>3.56 18.87 4.49 7</td>
</tr>
<tr>
<td>24</td>
<td>9.69 24.61 7.86 13</td>
<td>8.05 40.37 6.27 8</td>
</tr>
<tr>
<td>32</td>
<td>0.70 21.35 1.65 3</td>
<td>0.40 15.71 1.49 3</td>
</tr>
<tr>
<td>48</td>
<td>8.47 19.01 6.13 24</td>
<td>3.22 17.03 2.22 7</td>
</tr>
<tr>
<td>76</td>
<td>0.52 55.01 0.97 2</td>
<td>0.55 53.55 1.00 2</td>
</tr>
<tr>
<td>96</td>
<td>4.42 28.85 2.82 20</td>
<td>1.20 18.42 0.93 5</td>
</tr>
<tr>
<td>152</td>
<td>0.04 55.00 0.40 1</td>
<td>0.02 54.02 0.41 1</td>
</tr>
<tr>
<td>192</td>
<td>1.40 25.01 1.55 6</td>
<td>0.50 15.56 0.58 9</td>
</tr>
<tr>
<td>384</td>
<td>0.64 34.43 0.79 5</td>
<td>0.09 21.73 0.32 8</td>
</tr>
<tr>
<td>768</td>
<td>0.69 30.46 0.42 5</td>
<td>0.50 27.24 0.19 8</td>
</tr>
</tbody>
</table>

**Piecewise-linear Approximation**

<table>
<thead>
<tr>
<th>Subd count</th>
<th>Functional % error</th>
<th>Run-time % error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall Max Mean &gt; 5</td>
<td>Overall Max Mean &gt; 5</td>
</tr>
<tr>
<td>6</td>
<td>– ABORTED –</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5.16 12.96 3.85 6</td>
<td>3.56 18.87 4.49 7</td>
</tr>
<tr>
<td>24</td>
<td>9.69 24.61 7.86 13</td>
<td>8.05 40.37 6.27 8</td>
</tr>
<tr>
<td>32</td>
<td>0.70 21.35 1.65 3</td>
<td>0.40 15.71 1.49 3</td>
</tr>
<tr>
<td>48</td>
<td>8.47 19.01 6.13 24</td>
<td>3.22 17.03 2.22 7</td>
</tr>
<tr>
<td>76</td>
<td>0.52 55.01 0.97 2</td>
<td>0.55 53.55 1.00 2</td>
</tr>
<tr>
<td>96</td>
<td>4.42 28.85 2.82 20</td>
<td>1.20 18.42 0.93 5</td>
</tr>
<tr>
<td>152</td>
<td>0.04 55.00 0.40 1</td>
<td>0.02 54.02 0.41 1</td>
</tr>
<tr>
<td>192</td>
<td>1.40 25.01 1.55 6</td>
<td>0.50 15.56 0.58 9</td>
</tr>
<tr>
<td>384</td>
<td>0.64 34.43 0.79 5</td>
<td>0.09 21.73 0.32 8</td>
</tr>
<tr>
<td>768</td>
<td>0.69 30.46 0.42 5</td>
<td>0.50 27.24 0.19 8</td>
</tr>
</tbody>
</table>
Subdomain Boundary Problems

Even small subdomains miss discontinuities
Subdomain Boundary Problems

Even small subdomains miss discontinuities
Subdomain Boundary Problems

Even small subdomains miss discontinuities
Open Problems

- Error prediction
  - Rms approximation error predictive?
  - Functions always badly behaved
- Components retaining state
  - Most real components have persistent state
  - State is not just an extra input variable
QUESTIONS? COMMENTS?