The methods of Component-based software synthesis

Dr. Vaidas GIEDRIMAS
Siauliai University
Lithuania

vaigie@mi.su.lt
CBS Life-cycle

1. Requirements analysis
2. Component selection and evaluation
   - Component search
   - Component adaptation
   - Component testing
3. System architecture design
4. System implementation
5. System integration
6. Verification and validation
7. Maintenance
The formal component model
Deductive synthesis in the CBSE context
Inductive synthesis in the CBSE context
Transformational synthesis in the CBSE context
Levels of abstraction of Software components

- Component specification
- Component implementation
- Component deployment
- Component object
### The concepts of component models

<table>
<thead>
<tr>
<th></th>
<th>Struktural entities</th>
<th>Descriptive entities</th>
<th>Connecting entities</th>
<th>Processing entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Components</td>
<td>Interfaces</td>
<td>Contract</td>
<td>NFP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Control attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Port</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Binding</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Glue-code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Membrane</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Gates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Script</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Process</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sandbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Transaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Coordination</td>
</tr>
<tr>
<td></td>
<td>Basic</td>
<td>Source</td>
<td>Sink</td>
<td>Component</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Complex</td>
<td>Configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Provided</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Design</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Basic c.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>System c.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>User c.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impl c.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aguirre &amp; Malba</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Berger</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Cervantes</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Cox &amp; Song</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yoshi &amp; Honiden</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Lau group</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Mahmood</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Moschoyiannis</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Nierstrasz group</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>ObjectWeb</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>PECOS</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Poemomo</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Salzmann</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Szyptersky</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>UML</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>UniFrame</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>UNU/IIST</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Whitehead</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

- +: Feature is present
- -: Feature is absent
The formal component model
The formal component model

\[ C = \langle \mathcal{S}_{\text{out}}, \mathcal{S}_{\text{in}}, \Delta, \Omega^c \rangle \], where

- \( \mathcal{S}_{\text{out}} = \{ I_{\text{out}}^1, I_{\text{out}}^2, \ldots, I_{\text{out}}^n \} \)
- \( \mathcal{S}_{\text{in}} = \{ I_{\text{in}}^1, I_{\text{in}}^2, \ldots, I_{\text{in}}^m \} \)
- \( I_{\text{out}}^i, I_{\text{in}}^j \) – provided and required interfaces

- \( \Delta \) - the constraints of component
- \( \Omega^c \) - the credentials of component (performance, security etc.)
Interface in component model

\[ I_i = \langle O_i, \Xi, \Omega_i \rangle, \text{ where} \]

- \[ O_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,k}\} \]
  - \[ o_{i,1} = \langle P, R, PreC, PostC, \Omega_{i,1}^0 \rangle \text{ — operation} \]
  - \( P = \{p_1, p_2, \ldots, p_{l}\} \) and \( R = \{r_1, r_2, \ldots, r_s\} \) — sets of operation’s arguments and results respectively
  - \( PreC \) and \( PostC \) — \textit{pre- and post- conditions}
  - \( \Omega_{i,j}^0 \) — credentials of operation

- \( \Xi \) — the set of interface invariants,
- \( \Omega_i \) — the credentials of interface
Outline

- The formal component model
- Deductive synthesis in the CBSE context
- Inductive synthesis in the CBSE context
- Transformational synthesis in the CBSE context
Taxonomy of software synthesis methods

- Deductive synthesis
  - Constructive synthesis
    - Structural synthesis of programs
  - Transformational synthesis
    - Partial evaluation
- Inductive synthesis
SSP Application to Development of Component-based systems

- Reusable assets
  - Components

- Interface-level problem:
  \[ \text{compute } l_{\text{out}_1}, l_{\text{out}_2}, \ldots, l_{\text{out}_n}, \Omega^C \]
  \[ \text{from } l_{\text{in}_1}, l_{\text{in}_2}, \ldots, l_{\text{in}_m} \text{ knowing } M. \]

- Parameters-level problems:
  \[ \text{compute } R, \text{PostC}, \Omega^o_{i,j} \]
  \[ \text{from } P, \text{PreC} \text{ knowing } M, \]
Structural synthesis of programs (SSP) method

- SSP is deductive synthesis method.
- Four stages:
  - problem specification;
  - problem represented in the formal theory;
  - proof of the problem;
  - program.
- Typical problem specification:
  - compute $y_1, \ldots, y_n$ from $x_1, \ldots, x_m$ knowing $M$

SSP method (2)

- PSPACE-complete for *propositional intuitionistic logic*
- Used in *Priz, NUT* and two more systems.
- Can be applied to development of component-based systems because in essence method deals with the “black-box” entities.
Curry-Howard protocol implementation for CBSE

- Logical calculus –
  - Intuitionistic proposition calculus (IPC)

- Logical type theory –
  - IPC-based theory to represent component model (LTT)

- Computational type theory –
  - CTT

### ITS inference rules

<table>
<thead>
<tr>
<th>İvedimo taisyklės</th>
<th>Eliminavimo taisyklės</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \land B} ) ((\land I))</td>
<td>( \frac{\Gamma \vdash A \land B}{\Gamma \vdash A} ) ((\land E_L)) ( \frac{\Gamma \vdash A \land B}{\Gamma \vdash B} ) ((\land E_R))</td>
</tr>
<tr>
<td>( \frac{\Gamma, u : A \vdash B}{\Gamma \vdash A \rightarrow B} ) ((\rightarrow I^u))</td>
<td>( \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} ) ((\rightarrow E))</td>
</tr>
</tbody>
</table>
The axioms of LTT

\[ A_1 \land A_2 \land D \land R \rightarrow I_1 \land I_2 \land E \]

\[ A_1 \land D \land R \rightarrow I_1 \]
\[ A_2 \rightarrow E \]
\[ R \rightarrow I_2 \]
Simple Architecture SoCoSyS system

- Problem theory
- Specification
- Reusable components
- Reusable architectures
- Program

Synthesis system:
- Parser
- Planner
- First Optimizer
- Architecture generator
- Program generator
Outline

- The formal component model
- Deductive synthesis in the CBSE context
- **Inductive synthesis** in the CBSE context
- Transformational synthesis in the CBSE context
Inductive method

- Base theory $B$
- Positive ($E^+$) and negative ($E^-$) examples
- Hypothesis $H$

\[
\begin{align*}
B & \not\models E \\
B \land E^- & \not\models \bot \\
B \land H & \models E^+ \\
B \land H \land E^- & \not\models \bot
\end{align*}
\]
The advantages of inductive method in software engineering

- IM allows to perform the synthesis of software from incomplete specifications.
- IM is suitable in the search of high-level regularities.
- IM helps to identify the relations between the concepts.
- Using the same examples the generated software can be different each time. This allows to chose right design ideas.
- IM can be used to generate the optimal set of test cases.

K.-K. Lau, Component-Based Software Development and Logic Programming 2003
The problems of SSP and their solutions using induction

- The incompleteness of formal specification
  - IM can help to complete specifications.
  - But this is very dangerous!!

- The problem of undefined components
  - IM helps to identify the relations between the concepts and can suggest what component is missing.

- The extrafunctional requirements
  - IM is suitable in the search of high-level interfaces, components.
  - IM can be useful to generate ideas for the alternative components.
Outline

- The formal component model
- Deductive synthesis in the CBSE context
- Inductive synthesis in the CBSE context
- Transformational synthesis in the CBSE context
The transformation-based methods enable to shrink the gap between the concepts of business domain for which software is developing and the concepts of the implementation domain (component models, frameworks, operating systems etc.).

TS is used in the SoCoSyS as an additional method.
The experiments with the SoCoSys

- The experiment of the performance and extrafunctional properties of the SoCOSyS itself
- The experiment of the reliability of the synthesis
# Related work

<table>
<thead>
<tr>
<th>Component model</th>
<th>NORA/HAMMR</th>
<th>QUASAR</th>
<th>CWB</th>
<th>CoSMIC</th>
<th>SoCoSyS</th>
</tr>
</thead>
<tbody>
<tr>
<td>–</td>
<td>–</td>
<td>Gray-box abstractions</td>
<td>CCM, EJB, DCOM</td>
<td>CCM (CIAO)</td>
<td>PCM</td>
</tr>
<tr>
<td>Levels of abstraction</td>
<td>–</td>
<td>SC</td>
<td>CO</td>
<td>SC, DC, CO</td>
<td>SC and IC</td>
</tr>
<tr>
<td>Synthesis method</td>
<td>Deductive synthesis</td>
<td>Transformational synthesis</td>
<td>Deductive synthesis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goals of the synthesis process</td>
<td>To find particular components</td>
<td>To make relations between requirements and the design patterns.</td>
<td>To create the wrappers in orderer to prepare the workplace of the software designer.</td>
<td>To create distributed real-time software.</td>
<td>To assemble the software in the chosen abstraction level</td>
</tr>
</tbody>
</table>
Deductive methods operate with "black-box" abstractions. The usage of deductive method of structural synthesis of programs ensures consistency of component-based system specification.

Main disadvantages of deduction-based method: the problem of incomplete specifications and the problem of missing components can be reduced using inductive software generation methods.

The transformation-based methods enable to shrink the gap between the concepts of business domain for which software is developing and the concepts of the implementation domain (component models, frameworks, operating systems etc.).
Main references

Main references (2)

- K. Czarnecki
  Generative Programming: Methods, Tools, and Applications, 2000
The methods of Component-based software synthesis

Dr. Vaidas GIEDRIMAS
Siauliai University
Lithuania

vaigie@mi_su.lt