Leveraging State to Facilitate Separation of Concerns in Reuse-oriented Performance Models

Dominik Werle, Stephan Seifermann, Sebastian D. Krach
8th Symposium on Software Performance 2017
Contribution Overview

Argument:
Performance models need to express state for separating concerns/roles (in specific cases)

Approach (conceptual):
Extension: stochastic expressions that preserve separation. Reduction to ‘stateless’ model.
©call» IService.get

System Architect

Component Developer

Domain Expert

Werle et al. – Leveraging State to Facilitate Separation of Concerns in Reuse-oriented Performance Models (SSP’17)
Desired behavior: after $k$ requests by a user in a session, deep package inspection for all subsequent requests.
Desired behavior: after $k$ requests by a user in a session, deep package inspection for all subsequent requests.
Desired behavior: after $k$ requests by a user in a session, deep package inspection for all subsequent requests.
Component
Developer

\[ n \cdot \mathcal{U}(0,1) \geq k \]

«CPU demand»

\[ d \]

«call» IService.get

\[ \text{get} \]
\[ \text{get} \]
\[ \text{get} \]
\[ \ldots \]
\[ \text{get} \]
\[ \text{get} \]
\[ \text{get}' \]
\[ \ldots \]
\[ \text{get}' \]

1
2
3
\[ \ldots \]
k - 1
k
k + 1

n

n - k
Component
Developer

\[ 3 \cdot \mathcal{U}(0,1) \geq k \]

«CPU demand»
\(d\)

«call» IService.get

normal User

3

get

15s
\[ 100 \cdot \mathcal{U}(0,1) \geq k \]
**Approach: Session Count Expressions**

- **Express number of previous calls to a method in the current session**
- **Transformation to approximation**

```
[ \mathcal{SC}_{IDS}.get \geq k ]
```

```
 Component Developer

 «call» IService.get

 «CPU demand»

 [ else ]
```
Approach: Session Count Expressions

Random variable for "number of previous calls to \texttt{IDS.get} in the current session" in behavior descriptions

Express number of previous calls to a method in the current session
Transformation to approximation
Approach: Session Count Expressions

\[ SC_M = ? \]
Approach: Session Count Expressions

\[ \text{number of previous calls: } p_M^{in} \]

\[ SC_M = p_M^{in} + c_M^g \]
Approach: Session Count Expressions

\[ SC_M = p_{M}^{in} + c_{M}^{g} \]

number of previous calls: \( p_{M}^{in} \)

\( \text{«call» A} \)

\( \text{in} \)

\( g \)
Approach: Session Count Expressions

\[ \mathcal{U}(0, X - 1) \]

\[ SC_M = p_M^{in} + c_M^g \]
Approach: Session Count Expressions

\[ SC_M = p_M^{in} + c_M^g + \mathcal{L} \]
Approach: Session Count Expressions

Transformation sketch:
1. for each used $SC_M$:
   add a parameter $p_M^{in}$ to all signatures of all methods
2. replace each reference to $SC_M$ by $SC_M = p_M^{in} + c_M^g + \mathcal{L}$
   ($g$: subgraph of all preceding actions)
   loop approximation $\mathcal{L} = 0$, except for references to $SC_m$ inside loops with calls to $M$. Inside a loop ($X$ iterations): $\mathcal{L} = \mathcal{U}(0, X - 1)$
3. for each service call:
   pass $SC_M = p_M^{in} + c_M^g + \mathcal{L}$ as the parameter $p_M^{in}$.
Example application

1. for each used $\mathcal{SC}_M$:
   add a parameter $p^\text{in}_M$ to all signatures of all methods
   $\Rightarrow$ add the parameter $p^\text{in}_{\text{IDS}.\text{get}}$ to the signature of IService.get
Example application

2. replace each reference to \( SC_M \) by \( SC_M = p^{in}_M + c^g_M + \mathcal{L} \) 
   
   \((g): \) subgraph of all preceding actions

\[ \Rightarrow \] no other calls to get \((c^g_M = 0)\), not inside loop \((\mathcal{L} = 0)\), therefore

\[ SC_{IDS.get} = p^{in}_{IDS.get} \]
Example application

1. for each service call:
   pass $SC_M = p_M^{in} + c_M^{g} + \mathcal{L}$ as the parameter $p_M^{in}$.

   $p_M^{in} = 0, c_M^{g} = 0, \mathcal{L} = 2 \cdot \mathcal{U}(0,1)$, resp. $\mathcal{L} = 99 \cdot \mathcal{U}(0,1)$
Stateful Palladio

- Happe et al. (2013): Stateful Palladio
- allow the definition and manipulation of state of:
  - components,
  - systems,
  - users,
  - session
- Simulator is extended to support stateful models

Conclusion

- Specific modeling scenarios break separation of concerns between roles for performance modeling
- Session Count Expressions help in specific cases (number of calls in a session)

Context: Method for modeling behavior which depends on sets of requests while preserving separation of concerns (≠ call & return)

Future work:
- Implementation and evaluation, approximation “good enough?”
- Other state approximations (component, system, user), if possible