Towards QoS Prediction Based on Composition Structure Analysis and Probabilistic Environment Models

D. Ivanović¹, P. Kaowichakorn¹, M. Carro²

¹ Universidad Politécnica de Madrid, Spain
² IMDEA Software Institute, Madrid, Spain

PESOS 2013, San Francisco, CA
26 May 2013
Introduction

- Quality of Service (QoS) critical for real-world usability
  - performance, cost, user experience...
  - design-time analysis ⇒ evolving high quality services
  - run-time prediction ⇒ proactive adaptation
  - simulation modeling ⇒ configuration, SLA offering
Introduction

▶ Quality of Service (QoS) critical for real-world usability
  ⊃ performance, cost, user experience...
  ▶ design-time analysis ⇒ evolving high quality services
  ▶ run-time prediction ⇒ proactive adaptation
  ▶ simulation modeling ⇒ configuration, SLA offering

▶ QoS analysis for service compositions: uncertainty
  ▶ component services: 3rd party components
  ▶ limited information on implementation
  ▶ many actors ⇒ many factors ⇒ uncertain data / QoS

\[ \Pr[a \leq QoS \leq b \mid \text{struct, inputs, env, comp} \ldots] = ? \]
Motivation: Uncertainty

Simple aggregation: averages

\[ \bar{q}_i \quad \text{structure} \quad \bar{Q} \]

component services \( i = 0..n \)

service composition

Constraint modeling: intervals

\[ [q_{\text{min}}, q_{\text{max}}] \]

structure

completeness?

level of confidence?

granularity?

Both cases: effects of data?
Motivation: Uncertainty

Simple aggregation: averages

\[
\bar{q}_i = \frac{1}{n} \sum_{i=0}^{n} q_i
\]

- How representative?
- Spread?
- Distribution shape?
Motivation: Uncertainty

**Simple aggregation: averages**

\[
\bar{q}_i \quad \text{structure} \quad \bar{Q}
\]

- component services \( i = 0..n \)
- service composition

- how representative?
- spread?
- distribution shape?

**Constraint modeling: intervals**

\[
[q_{\min}, q_{\max}]_i \quad \text{structure} \quad [Q_{\min}, Q_{\max}]
\]

- completeness?
- level of confidence?
- granularity?

Both cases: effects of data?
Motivation: Uncertainty

**Simple aggregation: averages**

\[
\bar{q}_i \rightarrow QoS \quad \text{structure} \quad QoS \rightarrow \bar{Q}
\]

- component services \( i = 0 \ldots n \)
- service composition

- how representative?
- spread?
- distribution shape?

**Constraint modeling: intervals**

\[
[q_{\text{min}}, q_{\text{max}}]_i \rightarrow QoS \quad \text{structure} \quad [Q_{\text{min}}, Q_{\text{max}}]
\]

- completeness?
- level of confidence?
- granularity?
Motivation: Uncertainty

**Simple aggregation: averages**

$$\bar{q}_i \rightarrow QoS \quad structure \quad \bar{Q}$$

- component services: $$i = 0..n$$
- service composition

- how representative?
- spread?
- distribution shape?

**Constraint modeling: intervals**

$$[q_{min}, q_{max}] \quad structure \quad [Q_{min}, Q_{max}]$$

- completeness?
- level of confidence?
- granularity?

Both cases: effects of data?
Motivation: Uncertainty

Cost analysis: QoS bounds

- safe bounds: data + cost
- functions of input data (size)
- complex control
Motivation: Uncertainty

Cost analysis: QoS bounds

- safe bounds: data + cost
- functions of input data (size)
- complex control
Motivation: Uncertainty

Cost analysis: QoS bounds

Safe bounds: data + cost

Functions of input data (size)

Complex control

Loss of precision?

Difficult maths?

Data uncertainty?
Proposed Approach

- Unifying analysis: interpreting in a *probabilistic domain*

\[
data + \text{QoS} \rightarrow \text{discrete random variables}
\]
Proposed Approach

- Unifying analysis: interpreting in a *probabilistic domain*

data + QoS $\rightarrow$ *discrete random variables*

$X_1, X_2, \ldots, X_n$

inputs & state

$S$

control constructs
data operations

$Q$

processing steps

component

service QoS

**Joint probability distribution**: a finite approximation

$v : \mathbb{Z}^N \rightarrow [0,1]$ $N = n + m + 1$ $v(q, x, s) = \Pr[Q = q_1, X = x_1, S = s] = \sum q xs v = 1$ $\Pr[Q \leq a] = \sum q \leq a \sum xs v(q, x, s)$
Proposed Approach

- Unifying analysis: interpreting in a *probabilistic domain*

\[
data + \text{QoS} \rightarrow \text{discrete random variables}
\]

\[
X_1, X_2, \ldots, X_n \\
\text{inputs & state}
\]

\[
S_1, S_2, \ldots, S_m \\
\text{component QoS}
\]

\[
Q \quad \text{composition QoS}
\]

\[
\text{control constructs} \\
\text{data operations}
\]

\[
\text{processing steps}
\]

\[
\Pr[Q \leq a] = \sum_{q \leq a} \sum_{x, s} v(q, x, s) \quad \text{Joint probability distribution: a finite approximation}
\]

\[
v: \mathbb{Z}^n \rightarrow [0, 1] \\
N = n + m + 1
\]
Proposed Approach

- Unifying analysis: interpreting in a **probabilistic domain**
  
  data + QoS → *discrete random variables*

\[ \begin{aligned}
X_1, X_2, \ldots, X_n & \quad \text{inputs} \quad \text{control constructs} \\
\text{inputs & state} & \quad \text{data operations} \\
S_1, S_2, \ldots, S_m & \quad \text{component} \\
\text{component QoS} & \quad \text{service QoS} \\
\text{processing steps} & \quad \text{composition QoS}
\end{aligned} \]

- **Joint probability distribution**: a finite approximation

\[
\rho : \mathbb{Z}^N \rightarrow [0, 1] \quad N = n + m + 1
\]

\[
\rho(q, x, s) = \Pr[Q = q \land X = x \land S = s] \quad \sum_{qxs} \rho = 1
\]
Proposed Approach

- Unifying analysis: interpreting in a probabilistic domain

\[ \text{data + QoS } \rightarrow \text{discrete random variables} \]

\[ X_1, X_2, \ldots, X_n \]
\[ S_1, S_2, \ldots, S_m \]

- Joint probability distribution: a finite approximation

\[ \rho : \mathbb{Z}^N \rightarrow [0, 1] \]
\[ N = n + m + 1 \]
\[ \rho(q, x, s) = \Pr[Q = q \land X = x \land S = s] \]
\[ \sum_{q,x,s} \rho = 1 \]
\[ \Pr[Q \leq a] = \sum_{q \leq a} \sum_{x,s} \rho(q, x, s) \]
Example

if $X > 3$ then call $S_1$ else call $S_2$
Example

if $X > 3$ then call $S_1$ else call $S_2$
if $X > 3$ then call $S_1$ else call $S_2$
Example

if $X > 3$ then call $S_1$ else call $S_2$
if $X > 3$ then call $S_1$ else call $S_2$
if $X > 3$ then call $S_1$ else call $S_2$
Sample Language

\[ C ::= \]
\[ \langle\text{variable}\rangle := E \]
\[ \mid\text{call }\langle\text{service}\rangle \]
\[ \mid\text{if }B\text{ then }C\text{ else }C \]
\[ \mid\text{while }B\text{ do }C \]
\[ \mid\text{begin }C[;\ C]\ast\text{ end} \]
\[ \mid\text{skip} \]

(composition construct)

.assignment)

(service invocation)

(conditional)

(while loop)

(sequence)

(do nothing)
Sample Language

\[ C ::= \]
\[ (\text{variable}) := E \]
\[ | \text{call} (\text{service}) \]
\[ | \text{if } B \text{ then } C \text{ else } C \]
\[ | \text{while } B \text{ do } C \]
\[ | \text{begin } C[; C]^* \text{ end} \]
\[ | \text{skip} \]

\[ E ::= \]
\[ (\text{numeral})|(\text{variable})|E \circ E \]

(composition construct) (assignment) (service invocation) (conditional) (while loop) (sequence) (do nothing) (integer expressions) \( \circ \in \{+, -, \ast, \text{div, mod}\} \)
Sample Language

\[ C ::= \]
\[ \langle \text{variable} \rangle := E \]
\[ \mid \text{call} \langle \text{service} \rangle \]
\[ \mid \text{if} \ B \ \text{then} \ C \ \text{else} \ C \]
\[ \mid \text{while} \ B \ \text{do} \ C \]
\[ \mid \text{begin} \ C[; \ C][\text{* end} \]
\[ \mid \text{skip} \]

\[ E ::= \]
\[ \langle \text{numeral} \rangle|\langle \text{variable} \rangle|E \circ E \]

\[ B ::= \]
\[ E \ \rho \ E|B \land B|B \lor B|\neg B \]

(composition construct)
(assignment)
(service invocation)
(conditional)
(while loop)
(sequence)
(do nothing)
(integer expressions)
(\( \circ \in \{+,-,\ast,\text{div},\text{mod}\} \))
(Boolean conditions)
(\( \rho \in \{>,\geq,=,\neq,<,\leq\} \))
Sample Language

\[ C ::= \]
\[ \langle \text{variable} \rangle := E \]
\[ \mid \text{call} \langle \text{service} \rangle \]
\[ \mid \text{if} \ B \ \text{then} \ C \ \text{else} \ C \]
\[ \mid \text{while} \ B \ \text{do} \ C \]
\[ \mid \text{begin} \ C[; \ C]^* \ \text{end} \]
\[ \mid \text{skip} \]

\[ E ::= \]
\[ \langle \text{numeral} \rangle | \langle \text{variable} \rangle | E \circ E \]

\[ B ::= \]
\[ E \ \rho \ E | B \land B | B \lor B | \neg B \]

\text{Interpretation step: } \rho \text{ before } \rightarrow \rho' \text{ after}
Initial Conditions

- **Initial X**: default [singular] distros + input

\[ X_1 \quad X_2 \quad \ldots \quad X_n \]
\[ S_1 \quad S_2 \quad \ldots \quad S_m \]
\[ Q \]

\[ v(q, x, s) = v_Q(q) \times v_{X_1}(x_1) \times \cdots \times v_{X_n}(x_n) \times v_{S_1}(s_1) \times \cdots \times v_{S_m}(s_m) \]
Initial Conditions

- **Initial $X$:** default [singular] distros + input
- **Initial $Q$:** “natural” starting QoS value [singular distro]
Initial Conditions

- **Initial X**: default [singular] distros + input
  - $X_1, X_2, \ldots, X_n$

- **Initial Q**: “natural” starting QoS value [singular distro]
  - $Q$

- **Initial S**:
  - collected from *empirical observations* [static]
  - jointly analyzed using *prob. interpretation* [dynamic]
  - $S_1, S_2, \ldots, S_m$
Initial Conditions

- **Initial X**: default [singular] distros + input

- **Initial Q**: “natural” starting QoS value [singular distro]

- **Initial S**: collected from *empirical observations* [static]
  - jointly analyzed using *prob. interpretation* [dynamic]

- **Pairwise independence** between all random variables.

\[
\rho(q, x, s) = \rho_Q(q) \times \rho_X(x_1) \times \cdots \times \rho_X(x_n) \times \rho_S(s_1) \times \cdots \times \rho_S(s_m)
\]

- Each \(\rho_V : \mathbb{Z} \rightarrow [0, 1]\) a finite table [\(V\) from \(QXS\)]
Evolving Dependencies

- After $X_1 := X_2 + X_n$
  - $X_1$ dependent on $X_2, X_n$
    - remove factor $\rho_{X_1}(x_1)$
    - compute & insert $\rho_{X_1|X_2,X_n}(x_1|x_2, x_n)$
Evolving Dependencies

- After $X_1 := X_2 + X_n$
  $\Rightarrow X_1$ dependent on $X_2, X_n$
  - remove factor $\rho_{X_1}(x_1)$
  - compute & insert $\rho_{X_1|X_2,X_n}(x_1|x_2,x_n)$

- After **call** $S_1$ and **call** $S_2$
  $\Rightarrow Q$ dependent on $S_1, S_2$
  - remove factor $\rho_Q(q)$
  - compute & insert $\rho_{Q|S_1,S_2}(q|s_1,s_2)$
Evolving Dependencies

- After $X_1 := X_2 + X_n$
  $\Rightarrow$ $X_1$ dependent on $X_2, X_n$
  - remove factor $\rho_{X_1}(x_1)$
  - compute & insert $\rho_{X_1|X_2,X_n}(x_1|x_2, x_n)$

- After `call S_1` and `call S_2`
  $\Rightarrow$ $Q$ dependent on $S_1, S_2$
  - remove factor $\rho_Q(q)$
  - compute & insert $\rho_{Q|S_1,S_2}(q|s_1, s_2)$

- After a test $X_1 < X_2$
  $\Rightarrow$ $X_1, X_2$ co-dependent
  - remove factors $\rho_{X_1|X_2,X_n}(x_1|x_2, x_n)$ and $\rho_{X_2}(x_2)$
  - compute & insert $\rho_{X_1,X_2|X_n}(x_1, x_2|x_n)$
Assignments and Arithmetics

\[ X := E \]

- \( X \) becomes dependent on all variables in \( E \)
- **Update procedure**: (re-)computing factor tables
Assignments and Arithmetics

$X := E$

- $X$ becomes dependent on all variables in $E$
- **Update procedure**: (re-)computing factor tables
- **Example**: $X_1 := X_2 + X_n$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$\rho_{x_1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$\rho_{x_2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_n$</th>
<th>$\rho_{x_n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Factor tables**

| $x_1$ | $x_2$ | $x_n$ | $\rho'_{x_1|x_2,x_n}$ |
|-------|-------|-------|-----------------------|
| 1     | 1     | 0     | 1.0                   |
| 2     | 1     | 1     | 1.0                   |
| 2     | 2     | 0     | 1.0                   |
| 3     | 2     | 1     | 1.0                   |
| 4     | 4     | 0     | 1.0                   |
| 5     | 4     | 1     | 1.0                   |

**Joint distribution (reconstructed)**

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_n$</th>
<th>$\rho'_{x_1,x_2,x_n}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0.20</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0.30</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0.12</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0.20</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.30</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Service Invocations

\[ \textbf{call} \langle service_i \rangle \]

- Composition QoS $Q$ becomes dependent on $S_i$. 

- For monotonic, cumulative QoS metrics $Q := Q + S_i$

- \(\text{n}ative: \) execution time, cost

- \(\text{transformed}: \) availability

\(q = -\log p\)

- Homogeneous data / QoS treatment

- Data passing: enriched model

- Call arguments: component analysis inputs

- Return values: fixed distros or component analysis results
Service Invocations

\[
\text{call } \langle \text{service}_i \rangle
\]

- Composition QoS \( Q \) becomes dependent on \( S_i \).
- For \emph{monotonic, cumulative} QoS metrics \( \equiv Q := Q + S_i \):
  \begin{itemize}
    \item native: \emph{execution time, cost}
    \item transformed: \emph{availability}: \( \lambda = -\log p \)
  \end{itemize}

\( \Rightarrow \) \emph{Homogeneous data / QoS treatment}
Service Invocations

\[
\text{call } \langle \text{service}_i \rangle
\]

- Composition QoS \( Q \) becomes dependent on \( S_i \).
- For \textit{monotonic, cumulative} QoS metrics \( Q := Q + S_i \)
  - native: \textit{execution time, cost}
  - transformed: \textit{availability}: \( \lambda = -\log p \)

\( \Rightarrow \) \textit{Homogeneous data / QoS treatment}

- Data passing: enriched model
  - \textit{call arguments}: component analysis inputs
  - \textit{return values}: fixed distros or component analysis results
Branching

**if** \( B \) **then** \( C_1 \) **else** \( C_2 \)

- Variables in \( B \) become *co-dependent*
- Condition \( B \rightarrow \) probability \( p \) [of being true]
- The \( \rho \) before \( \rightarrow \rho_1 \) (\( B \) true, for \( C_1 \)), \( \rho_2 \) (\( B \) false, for \( C_2 \))
- Resulting \( \rho' = p \times \rho_1' + (1 - p) \times \rho_2' \) [linear combination on \( p \)]
Branching

if $B$ then $C_1$ else $C_2$

- Variables in $B$ become co-dependent
- Condition $B \rightarrow$ probability $p$ [of being true]
- The $\rho$ before $\rightarrow \rho_1$ ($B$ true, for $C_1$), $\rho_2$ ($B$ false, for $C_2$)
- Resulting $\rho' = p \times \rho_1' + (1 - p) \times \rho_2'$ [linear combination on $p$]
- Example: $B \equiv X_1 < X_2$

\begin{tabular}{|c|c|c|c|c|}
\hline
\multicolumn{1}{|c|}{\multirow{2}{*}{$x_1$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_2$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_n$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,2|x_n}$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,n|x_n}$}} \\
\hline
1 & 1 & 0 & 0.3 & $\overline{B}$ \\
2 & 1 & 1 & 0.3 & $B$ \\
2 & 2 & 0 & 0.5 & $\overline{B}$ \\
3 & 2 & 1 & 0.5 & $B$ \\
4 & 4 & 0 & 0.2 & $\overline{B}$ \\
5 & 4 & 1 & 0.2 & $B$ \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|c|c|}
\hline
\multicolumn{1}{|c|}{\multirow{2}{*}{$x_1$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_2$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_n$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,2|x_n}$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,n|x_n}$}} \\
\hline
2 & 1 & 1 & 0.18 & \\
3 & 2 & 1 & 0.30 & \\
5 & 4 & 1 & 0.12 & \\
\hline
\end{tabular}

$\Sigma : 0.60 \leftarrow p$

\begin{tabular}{|c|c|c|c|c|}
\hline
\multicolumn{1}{|c|}{\multirow{2}{*}{$x_1$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_2$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$x_n$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,2|x_n}$}} & \multicolumn{1}{|c|}{\multirow{2}{*}{$\rho x_{1,n|x_n}$}} \\
\hline
1 & 1 & 0 & 0.12 & \\
2 & 2 & 0 & 0.20 & \\
4 & 4 & 0 & 0.08 & \\
\hline
\end{tabular}

$\Sigma : 0.40 \leftarrow 1 - p$
Loops

**while** \( B \) **do** \( C \)

- Interpretation *unfolds it* into:

```plaintext
if \( B \) then begin
    \( C \);
    while \( B \) do \( C \)
end
else
    skip
```

- Assuming (non-probabilistic) termination
- Termination analysis: well studied field
Loops

\[ \textbf{while } B \textbf{ do } C \]

- Interpretation *unfolds it* into:
  
  \[
  \begin{aligned}
  \text{if } B \text{ then begin} \\
  & C; \\
  & \text{while } B \text{ do } C \\
  \text{end}
  \end{aligned}
  \]

  \[
  \text{else}
  \]

  \[
  \text{skip}
  \]

- Assuming (non-probabilistic) *termination*
  - *termination analysis*: well studied field
    - [part of functional correctness]
Implementation Notes

- **Analyzer prototype** implemented in Prolog
  - ease of symbolic manipulation [ASTs, distros]
Implementation Notes

- **Analyzer prototype** implemented in Prolog
  - ease of symbolic manipulation [ASTs, distros]

- **Inputs**:
  - initial distros for composition inputs and state vars [X]
  - composition code describing its structure
  - QoS distros for component services [empirical, S]

- **Outputs**:
  - final distribution for composition QoS [Q]
  - final distros for composition state vars [X – optional]
Implementation Notes

- **Analyzer prototype** implemented in Prolog
  - ease of symbolic manipulation [ASTs, distros]

- **Inputs**:
  - initial distros for composition inputs and state vars [X]
  - composition code describing its structure
  - QoS distros for component services [empirical, S]

- **Outputs**:
  - final distribution for composition QoS [Q]
  - final distros for composition state vars [X – optional]

- **Example**:

```prolog
?- analyze(if(x>3, call(s1), call(s2)),
            [x=[0-0.2,2-0.4,5-0.4]],
             [s1=[2-0.3,4-0.5,10-0.2],
              s2=[1-0.3,3-0.4,9-0.3]],
             QoS).
QoS = [1-0.18, 2-0.12, 3-0.24, 4-0.2, 9-0.18, 10-0.08]
```
Validation: Execution Time

Java App Client

Internet

Java Servlet Service

GoogleApp Engine

Total Execution Time

Net Execution Time

Probabilistic interpreter structure

Reusable for other clients

X

Q

actual

1000 ×

S

actual

1000 ×

\( \delta \)

\( \Delta \)

\( ? \)

network latency
Validation: Execution Time

- **Total Execution Time**
- **Net Execution Time**
- **δ**

- **Java App Client**
- **Internet**
- **Google App Engine**
- **Java Servlet Service**

Probabilistic interpreter structure is reusable for other clients.

Network latency.
Validation: Execution Time

- **Total Execution Time**
- **Net Execution Time**
- **δ**

**Java App Client** to **Java Servlet Service** via the **Internet**

- **Probabilistic interpreter**

- **GoogleApp Engine**

- **network latency**

- **structure**

- **1000×**

- **actual S**
Validation: Execution Time

Total Execution Time
Net Execution Time
δ

Java App Client
Internet
Java Servlet Service
GoogleApp Engine

probabilistic interpreter
structure
network latency

predicted Q
1000 ×
actual S

1 ×
Validation: Execution Time

Total Execution Time

Net Execution Time

GoogleApp Engine

Java Servlet Service

Internet

network latency

Probabilistic interpreter

Q

actual

predicted

1000×

structure

δ

1×

reusable for other clients
Validation: Execution Time

Java App

Client

GoogleApp

Engine

Java

Servlet

Service

Internet

Probabilistic interpreter

network latency

Total Execution Time

Net Execution Time

actual

predicted

1000×

1×

δ

Structural

1000×

actual

reusable for other clients
Validation Results: Fitting

- Very small mean square error $\approx 0.07$ [total execution time]
  - order-of-magnitude better than uniform probability
  - two orders better than a fixed average
Multi-modal distributions: inflection points
- escape regularity and normality patterns
- difficult to analyze using standard measures of dispersion
Conclusions

- Modeling uncertainty with probability distributions ⇒ finer grained, more detailed QoS predictions
- Basic ingredients: structure + empirical data on component QoS → probabilistic interpretation

Some ideas for future work:
- reverse: composition QoS → component QoS
- tradeoffs: complexity ↔ precision
- internal representation, algorithms
- handling dynamic updates for component QoS

Thank you!
Conclusions

- Modeling uncertainty with probability distributions ⇒ finer grained, more detailed QoS predictions
- Basic ingredients: structure + empirical data on component QoS → probabilistic interpretation

Some ideas for future work:
- reverse: composition QoS → component QoS
- tradeoffs: complexity ↔ precision
- internal representation, algorithms
- handling dynamic updates for component QoS
Conclusions

- Modeling uncertainty with probability distributions ⇒ finer grained, more detailed QoS predictions
- Basic ingredients: structure + empirical data on component QoS → probabilistic interpretation

Some ideas for future work:
- reverse: composition QoS → component QoS
- tradeoffs: complexity ↔ precision
- internal representation, algorithms
- handling dynamic updates for component QoS

Thank you!