Self-Adaptation of Service based Systems based on Cost/Quality Attributes Tradeoffs

Raffaela Mirandola ‡  Pasqualina Potena §

‡ Politecnico di Milano, Italy
§ Universita’ dell’Aquila, Italy
An application should be self-adaptive in order to automatically and autonomously adapt its behavior for several reasons, such as

- service evolution (e.g. a new version may be available)

- hardware volatility (e.g. network quality changes)

- varying users demands with new requirements (e.g. a new functionality or a different level of quality of service)
Objective of the paper: A Framework, ...

..., based on an optimization model (generated and solved by the Generator and Evaluator module), dynamically adapts a system (by changing software and hardware features by means of the Executor module) while minimizing the adaptation costs and guaranteeing a required level of the system qualities.
Objective of the paper: A Framework

Adaptation actions can be triggered both by an user (using the User Requests Manager module that can also interact with the Monitor) and/or automatically by the framework itself (after it receives alerts from the Monitor monitoring the system and interacting with services repositories by means of the Provider Info module).
Presentation roadmap

• Overview of Optimization Model

• Zoom-in into state of the art

• Conclusion
Presentation roadmap

- Overview of Optimization Model
- Zoom-in into state of the art
- Conclusion
Overview of Optimization Model

Let us assume to dispose of a deployed system composed by software services and that for each (not) active service offered by the system we dispose of an UML SD.

**Getting User Requests** – An user adaptation scenario is a set of new requirements…

(i) a **new functional requirement**, i.e. introducing a new external service/modifying the dynamics of an existing external service

(ii) a **new non-functional requirement**, i.e. requiring a value threshold for a new system quality/modifying the value threshold for an already required system quality.

**Getting Alerts by the Monitor** -

(i) the **violation of a non-functional requirement**

(ii) the **appearing/disappearing of a service** in the environment.
To satisfy either the new requirements required by the user or exploit the alerts raised by the Monitor some adaptation actions have to be performed.

The user defines adaptation plans for each requirement, whereas the framework itself defines adaptation plans when it gets alerts about the violation of non-functional constraints.

An adaptation plan is a set of actions modifying the static and dynamic structure of the software architecture and the hardware architecture to (exploit a certain alert) address a certain requirement.
Software adaptation actions:

• Introducing new software services

• Replacing existing service instances with functionally equivalent ones

• Modifying the interactions between software services in a certain external service
Hardware adaptation actions:

- Defining the deployment of software service on hardware nodes
- Introducing hardware resources
- Modifying hardware resources
- Modifying the interactions between hardware nodes

In addition, we leave to the solver the possibility to choose additional service replacement actions and hardware actions that have not been embedded in any selected plan.
<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Adaptation Plan ID</th>
<th>Adaptation Plan Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req₁</td>
<td>ap₁₁</td>
<td>Replacing 𝑠₃ with its first instance AND Replacing 𝑠₄ with its second instance</td>
</tr>
<tr>
<td></td>
<td>ap₁₂</td>
<td>Adding a new service news₂</td>
</tr>
<tr>
<td>req₂</td>
<td>ap₂₁</td>
<td>Replacing 𝑠₂ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₂</td>
<td>Adding a new service news₁ AND Replacing 𝑠₅ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₃</td>
<td>Adding a new service news₂</td>
</tr>
</tbody>
</table>

Example of Output of the Model
<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Adaptation Plan ID</th>
<th>Adaptation Plan Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req₁</td>
<td>ap₁₁</td>
<td>Replacing $s_3$ with its first instance AND Replacing $s_4$ with its second instance</td>
</tr>
<tr>
<td></td>
<td>ap₁₂</td>
<td>Adding a new service news₂</td>
</tr>
<tr>
<td>req₂</td>
<td>ap₂₁</td>
<td>Replacing $s_2$ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₂</td>
<td>Adding a new service news₁ AND Replacing $s_5$ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₃</td>
<td>Adding a new service news₂</td>
</tr>
</tbody>
</table>

Example of Output of the Model
<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Adaptation Plan ID</th>
<th>Adaptation Plan Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>req₁</td>
<td>ap₁₁</td>
<td>Replacing $s_3$ with its first instance AND Replacing $s_4$ with its second instance</td>
</tr>
<tr>
<td></td>
<td>ap₁₂</td>
<td>Adding a new service $news_2$</td>
</tr>
<tr>
<td>req₂</td>
<td>ap₂₁</td>
<td>Replacing $s_2$ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₂</td>
<td>Adding a new service $news_1$ AND Replacing $s_5$ with its first instance</td>
</tr>
<tr>
<td></td>
<td>ap₂₃</td>
<td>Adding a new service $news_2$</td>
</tr>
</tbody>
</table>

Example of Output of the Model
where:

- $n$ number of existing services
- $\text{NewS}$ number of new services
- $p_{\text{exec}_k}$ probability that the $k$-th system functionality will be invoked
- $\Theta^q$ threshold value required for the $q$-th quality attribute
- $\theta^0_i (\overline{\theta}^q_i)$ cost of the existing/new service
- $\theta^q_{ki} (\overline{\theta}^q_{ki})$ value of the $q$-th quality attribute of the $i$-th existing (new) service
**Optimization Model**

\[
\begin{align*}
\text{min } & \text{Cost}(\theta_1^0, \ldots, \theta_n^0, \bar{\theta}_1^0, \ldots, \bar{\theta}_{|\text{NewS}|}^0) \\
\sum_{k \in K} p_{\text{exec}} k \cdot G_k q (\theta_k^q, \ldots, \theta_k^n, \bar{\theta}_k^q, \ldots, \bar{\theta}_k^{|\text{NewS}|}) & \geq \Theta^q
\end{align*}
\]

**An example of Cost function**

\[
\text{Cost} = \sum_{i=1}^{n} \sum_{j=1}^{|\text{Aval}_i|} c_{ij} x_{ij} + \sum_{h=1}^{|\text{NewS}|} c_h z_h
\]

- \(c_{ij}\) cost of the \(j\)-th instance available for the service \(i\)
- \(\bar{c}_h\) cost to adopt the \(h\)-th new service into the system

\(n\) number of existing services

\(\text{NewS}\) number of new services

\(p_{\text{exec}}\) probability that the \(k\)-th system functionality will be invoked

\(\Theta\) threshold value required for the \(q\)-th quality attribute
OPTIMIZATION MODEL

$$\min Cost(\theta_1^0, \ldots, \theta_n^0, \bar{\theta}_1^0, \ldots, \bar{\theta}_{|\text{NewS}|}^0)$$

$$\sum_{k \in K} p_{\text{exec}_k} \cdot G_{kq}(\theta_{k1}^q, \ldots, \theta_{kn}^q, \bar{\theta}_{k1}^q, \ldots, \bar{\theta}_{k|\text{NewS}|}^q) \geq \Theta^q$$

$q$-th quality attribute of the $i$-th existing service

$$\theta_{ki}^q = \sum_{j=1}^{\left|\text{Aval}_i\right|} x_{ij} \Gamma_{kq}(\lambda_{i1}^q, \ldots, \lambda_{iu}^q, \lambda'_{i1}^q, \ldots, \lambda'_{iv}^q, \Lambda_{ij1}^q, \ldots, \Lambda_{ijw}^q)$$

- $n$ number of existing services
- $\text{NewS}$ number of new services
- $p_{\text{exec}_k}$ probability that the $k$-th system functionality will be invoked
- $\Theta^q$ threshold value required for the $q$-th quality attribute

$\Gamma_{kq}$ function that predicts the $q$-th quality attribute
$u$ number of software architecture observable parameters
$v$ number of hardware observable parameters
$w$ number of parameters expressing the specific features of the service implementation
$bp_{ki}$ number of busy periods that the service $i$ shows in the SD $k$
$VBP_{p(i,k)}$ variation of the number of busy periods of the $i$-th existing service in the $k$-th external service.

For example, the number of busy periods...

$$\lambda_{i1}^q = bp_{ki} + \sum_{r=1}^{m} \sum_{p \in AP_r} VBP_{p(i,k)} \cdot y_{rp}$$
**Optimization Model**

\[
\min \text{Cost}(\theta_1^0, ..., \theta_n^0, \bar{\theta}_1^0, ..., \bar{\theta}_{|\text{NewS}|}^0)
\]

\[
\sum_{k \in K} p_{\text{exec}_k} \cdot G_{kq}(\theta_{k1}^q, ..., \theta_{kn}^q, \bar{\theta}_{k1}^q, ..., \bar{\theta}_{k|\text{NewS}|}^q) \geq \Theta^q
\]

\[\text{q-th quality attribute of the h-th new service}\]

\[
\bar{\theta}_{kh}^q = z_h \Gamma_{kq} (\lambda_{h1}^q, ..., \lambda_{hu}^q, \lambda_{hv}^1, ..., \lambda_{hv}^q, \Lambda_{h1}^q, ..., \Lambda_{hw}^q)
\]

- \(n\) number of existing services
- \(\text{NewS}\) number of new services
- \(p_{\text{exec}_k}\) probability that the \(k\)-th system functionality will be invoked
- \(\Theta^q\) threshold value required for the \(q\)-th quality attribute

**Additional Information**

- \(\Gamma_{kq}\) function that predicts the \(q\)-th quality attribute
- \(u\) number of software architecture observable parameters
- \(v\) number of hardware observable parameters
- \(w\) number of parameters expressing the specific features of the service implementation
- \(\text{BP}_p(h, k)\) number of busy periods that the chosen adaptation plans suggest for the \(h\)-th new service within the \(k\)-th external service

**Example**

\[
\lambda_{h1}^q = \sum_{r=1}^m \sum_{p \in \text{AP}_r} \text{BP}_p(h, k) \cdot y_{rp}
\]
**Optimization Model**

\[
\begin{align*}
\text{min} & \quad \text{Cost}(\theta_1^0, \ldots, \theta_n^0, \overline{\theta}_1^0, \ldots, \overline{\theta}_{|\text{NewS}|}^0) \\
\sum_{k \in K} p_{\text{exec}_k} \cdot G_{kq} (\theta_k^q, \ldots, \overline{\theta}_k^q, \ldots, \overline{\theta}_{|\text{NewS}|}^q) & \geq \Theta^q
\end{align*}
\]

**An example of system quality: the System Reliability**

\[
G_{kq} = \prod_{i=1}^{n} \left( \sum_{j=1}^{|\text{Aval}_i|} x_{ij} (1 - \Lambda_{ij}^q )^{\lambda_{ij}^q} \right) \cdot \prod_{h=1}^{|\text{NewS}|} (1 - \overline{\Lambda}_h^q z_h)^{\lambda_{h1}^q}
\]

- \(\Lambda_{ij}^q\) is the probability of failure on demand of the j-th instance available for the service i.
- \(\overline{\Lambda}_h^q\) is the probability of failure of the new service h.

\[
\lambda_{ij}^q = b_{p_{ki}} + \sum_{r=1}^{m} \sum_{p \in A'_{P_r}} VBP_p (i, k) \cdot y_{rp}
\]

\[
\lambda_{h1}^q = \sum_{r=1}^{m} \sum_{p \in A'_{P_r}} BP_p (h, k) \cdot y_{rp}
\]
Presentation roadmap

• Overview of Optimization Model

  • Zoom-in into state of the art

• Conclusion
Existing Approaches:

✓ Usually the frameworks adapt a system only after a user triggers a request.
  ➢ They support service selection with respect to a composition defined by an user (e.g. the VRESCo runtime environment [Rosenberg09])
  ➢ They choose the service composition, which they have generated together with a finite set of other candidates, that better fulfill the required quality (e.g. [Chiu09], [Ibrahim09])

✓ The spontaneous service composition is typically not supported.
   In [Ibrahim1-09] the spontaneous service selection and the extension of system functionality is supported.

---


With respect to existing approaches, our approach...

- is the first one (to the best of our knowledge) introducing a dynamic self-adaptive framework supporting both the software (including both static and dynamic models) and hardware architecture adaptation using an optimization model
- is general and does not rely on specific architectural style, development process or service application domain
- can facilitate the work of a maintainer that does not have to insert as input value architectures satisfying all changes required
Presentation roadmap

• Overview of Optimization Model

• Zoom-in into state of the art

• Conclusion
We have introduced a **framework**, based on an optimization model, that dynamically adapts both the software and hardware features of a service based system while minimizing the **adaptation costs** and guaranteeing a required level of the **system qualities**.

Adaptation actions can be triggered both by an user request and/or automatically after the runtime violation of system quality constraints, or the appearing/disappearing of services into the environment.
We intend to...

• specialize our framework by enhancing it for guaranteeing specific properties of a service application domain. We are implementing a prototype to apply our approach on realistic examples

• investigate the evaluation of dependencies among requirements

• investigate the use of other quality constraints

• take into account dependencies between quality attributes and dependencies between services

• solve problems due, for example, to the model solution too large computation time using the meta-heuristic techniques combined with the simulation techniques

• introducing dependencies between adaptation plans

....