Software Architecture for Mobile Distributed Computing

Valérie Issarny, Ferda Tartanoglu, Jinshan Liu, Françoise Sailhan

Software Architecture for Mobile Distributed Computing*

Valérie Issarny, Ferda Tartanoglu, Jinshan Liu, Françoise Sailhan
INRIA, UR Rocquencourt
Domaine de Voluceau, 78153 Le Chesnay, France
{Valerie.Issarny,Galip-Ferda.Tartanoglu,Jinshan.Liu,Francoise.Sailhan}@inria.fr

Abstract

Today's wireless networks and devices support the dynamic composition of mobile distributed systems, according to device connectivity. This has in particular led to the introduction of a number of supporting middleware. However, such solutions need to be complemented with adequate modeling and verification support towards enforcing the correctness of the dynamic mobile systems with respect to both functional and non-functional properties. Building on the elegant properties of software architecture modeling, this paper introduces base modeling of mobile software components, which integrates key features of the wireless infrastructure and allows for reasoning about the behavior of dynamically composed systems.

1. Introduction

Modeling software architectures is now recognized as a useful approach towards controlling the complexity of software systems, in particular due to the associated support for assessing the software systems' behavior with respect to provided functional and non-functional properties [12]. Architectural modeling further enables dealing with the dynamic evolution and adaptation of software systems, providing adequate abstractions to specify changes to the system's configuration and to further reason about the dynamic system's behavior [14, 17, 3]. Mobile distributed software systems constitute a typical example of dynamic systems that need to adapt according to the environment. Specifically, mobile systems need to be dynamically configured according to the networked resources that are reachable, which change over time due to the highly dynamic network topology. Such a requirement has led to revisiting traditional architectural styles of distributed systems so as to deal with the network's dynamics and resource constraints of the mobile, wireless devices. In the most general case, the mobile distributed system is dynamically composed out of the networked resources, which may be driven by adequate architecture modeling [26, 13]. However, such a modeling must not only support the dynamic integration and composition of the networked resources, it must also allow enforcing correctness of the mobile distributed systems with respect to functional and non-functional properties.

This paper addresses base architectural modeling for mobile distributed computing, towards enforcing systems correctness. We first provide an overview of mobile distributed systems, discussing their key features (§ 2), from which we derive base architectural style, i.e., architectural modeling so as to support dynamic integration and composition of networked resources for adaptive configuration of mobile distributed software systems (§ 3). We then concentrate on assessing dynamic systems using our model, which is further illustrated in the context of mobile e-services (§ 4). The actual dynamic composition of mobile distributed systems is tightly coupled with the supporting run-time environment (i.e., middleware), leading to the investigation of the integration of our architectural modeling with the WSAMI middleware based on the Web services architecture, which we introduced for distributed mobile computing (§ 5). Finally, we conclude by a summary of our contribution and our current and future work (§ 6).

2. Mobile Distributed Systems

Mobile distributed systems cover a broad spectrum of software systems, by considering all the forms of mobility, i.e., personal, computer, and computational [22]. In this paper, we focus on the mobility of devices, as enabled by today's wireless devices. Then, most specifics of mobile distributed systems compared to their stationary counterpart follow from the features of the wireless infrastructure (§ 2.1). Mobile software systems must in particular cope with the network's dynamics (§ 2.2) and quality of service management, as the mobile environment makes it particu-

* This work has received the support of the European Commission through the IST programme, in the context of the Ozone project (http://www.extra.research.philips.com/euprojects/ozone/).

Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)
0-7695-2172-X/04 $ 20.00 © 2004 IEEE

larly challenging due to resource constraints of the wireless devices and varying bandwidth (§ 2.3).

2.1. Wireless infrastructure

The wireless infrastructure is primarily characterized by the heterogeneity of networks and devices.

Today's wireless networks may be subdivided into two categories: (i) infrastructure-based and (ii) ad hoc [21]. With the former, base stations handle traffic to/from mobile terminals in their transmission range, using either licensed (e.g., GSM, GPRS, UMTS) or unlicensed frequency bands (e.g., IEEE 802.11 Wireless Local Area Network). With the latter, wireless nodes form a dynamic and temporary network according to their respective location, using unlicensed frequency bands (e.g., IEEE 802.11 in the ad hoc mode, Bluetooth Personal Area Network), and may further act as routers in the context of Mobile Ad hoc NETworks [7]. Hybrid networks then integrate the various wired and wireless technologies, actually leading to the mobile Internet.

The diversity of wireless devices keeps evolving and embodies devices from the ICT (e.g., PDAs, cell phones, wireless storage devices) and Consumer Electronics domains (e.g., TV, MP3 players, gaming devices, cameras), with most of them featuring a combination of wireless connectivities (e.g., PDAs equipped with GSM, WLAN and WPAN). Specialized devices are also being offered such as sensors/beacons, smartcards.

The above allows deployment of mobile distributed systems that provide mobile users with seamless access to a rich set of services and content. However, the wireless infrastructure poses specific challenges on the software systems compared to the wired environment. In particular, the dynamics of the network calls for dedicated support, as discussed in the next section.

2.2. Dynamic networking

Initial work on the management of the network's dynamics has focused on handling mobility-induced failures that result from the occurrence of network disconnections, which has led to significant research effort since the emergence of wireless infrastructures in the early 90s. The basic technique to handle such failures is optimistic replication, i.e., the content (and related service) is replicated on the wireless devices so as to allow continuing access when disconnected, and is later synchronized with peer copies according to network connectivity (e.g., see [27]). The hybrid network further allows for replication at various levels, from the wireless device to some area network, on either trusted or untrusted node, which may be wireless or stationary [25]. However, optimistic replication impacts upon the behavior of the system, which is dependent upon the specific replica that is accessed and cannot be made transparent to the application.

A more general approach to the management of the network's dynamics, following advances in wireless networks, lies in the automatic configuration and reconfiguration of networked devices and services. This is in particular supported by resource discovery protocols that provide proactive mechanisms for dynamically discovering, selecting and accessing reachable resources that meet a given specification [5]. This leads to building event-based systems, as (wireless) nodes advertise and consume networked resources according to their specific situation and requirements. This further leads to the design of mobile distributed systems as systems of systems, whose component systems are autonomous and hosted by networked nodes, either wireless or stationary. The systems' configuration then evolves and adapts according to the network connectivity of component systems. Specific solutions differ according to the way they deal with the integration of components, which is closely related to the system's interaction paradigm. Two approaches can be identified:

- Data-oriented systems that integrate component systems at the level of the underlying data repository, where the repositories of the networked nodes are dynamically and virtually coupled according to network connectivity of the nodes, as, e.g., in the LIME system [20].
- Service-oriented systems that integrate component systems through the dynamic binding of clients with services in a peer-to-peer way according to network connectivity of clients, as, e.g., in the WSAMI system [13].

Note that replication management to handle mobility-induced failures may be considered as a special case of the dynamic integration of component systems; replicas are component systems that in particular synchronize when connectivity allows.

Independent of the composition approach, the autonomous systems that get composed are in general loosely coupled due to the network's dynamics. However, the composition shall ensure the correctness of the system's behavior with respect to target functional and non-functional properties. With respect to the former, the composition must enforce selection of the appropriate component systems and coordination protocols that conform to the specification of the component systems. With respect to the latter, it is mandatory to account for the quality of service delivered by component systems and their integration.

2.3. Quality of service

The features of the wireless network make the management of Quality of Service (QoS) mandatory for mobile distributed systems. Specifically, the dynamic composition of mobile distributed systems must both minimize resource consumption on mobile nodes and satisfy the users' requirements with respect to perceived QoS. The resource limitation of wireless computing devices makes QoS-aware resource management even more important. QoS specification associated with component systems is then concerned with capturing the systems' QoS requirements and policies.

QoS specification should: (i) allow description of both quantitative (e.g., service latency) and qualitative (e.g., CPU scheduling mechanism) QoS attributes, as well as adaptation rules [24]; (ii) be declarative in nature, that is, specify only what is required, but not how the requirements are implemented [4]. In addition, although more QoS parameters yield more detailed description, the gain has to be put against the increased overhead. Usually, dominant QoS properties of systems may be captured with a small number of attributes [8]. We identify the following key QoS attributes for the mobile system [16]: (i) performance, (ii) reliability, (iii) security, and (iv) transactional properties. This is further complemented with related resource consumption (i.e., CPU load, memory, bandwidth, battery) and service adaptation.

3. Base Architectural Style

As discussed in the previous section, mobile distributed systems now tend to be built out of the composition of mobile, autonomous component systems. This may be conveniently modeled at the software architecture level: components abstract mobile component systems (§ 3.1) and connectors abstract interaction protocols above the wireless network (§ 3.2).

Architectural modeling must comprehensively address the key characteristics of mobile distributed systems, i.e., dynamic composition, in a way that ensures correctness of the system. The following introduces a base architectural style that models key aspects of mobile distributed systems. Our model is grounded in the service-oriented interaction paradigm, i.e., a component abstracts a networked service that invokes operations of peer components and dually executes operations that are invoked. However, it applies as well to interactions based on data sharing, since the distinction is apparent at the implementation level only.

3.1. Mobile components

As in traditional software architecture modeling, a system component specifies the operations that it provides to and requires from the environment. The dynamic composition of the mobile component with components of the environment further requires: (i) enriching the component's functional specification so as to ensure adherence to the coordination protocols to be satisfied for ensuring correct service delivery despite the dynamics of the networks, i.e., the interaction protocols that must be atomic, and (ii) specifying the component's non-functional behavior so as to enforce quality of service. In the following, we denote keywords using the type writer font, and zero or more occurrences of n by $\{n\}^*$.

Functional specification. The specification of coordination protocols among mobile components relates to the one of *conversation*, also known as *choreography*, in the context of Web services[1]. Such a specification also relates to the one of interaction protocols associated with component ports to ensure conformance with connector roles, as, e.g., supported by the Wright language [2]. Hence, from the standpoint of component's functional behavior, a mobile component is modeled as:

```
Mobile component name
  Provides: {operation name: (signature)}*
  Requires: {operation name: (signature)}*
  Coordinates:
    Input conversation: {conversation name: (process)}*
    Output conversation: {conversation name: (process)}*
```

In the above, an operation is abstracted by its signature, which provides the operation's name and the type of associated input and output content. Conversations are specified as processes in the π calculus [18], where we use the following notation:

$P, Q ::=$		Processes
	$P\|Q$	Parallel composition
	$P + Q$	Choice
	$!P$	Replication
	$v(x)$	Input communication
	$\bar{v}(x)$	Output communication
	$\mathbf{0}$	Null process

We recall that the input process $v(x).P$ is ready to input from channel v, then to run P with the formal parameter x replaced by the actual message, while the output process $\bar{v}(y).P$ is ready to output message y on channel v, then to run P. The $reduction$ relation, noted $\rightarrow$, is further defined over processes, with $P \rightarrow P_1$ expressing that P can evolve to process P_1 as a result of an action within P. For instance, we have: $((\bar{v}(x).P + P')|(v(y).Q + Q')) \rightarrow P|Q\{^x/_y\}$, with $Q\{^x/_y\}$ meaning that x replaces y in Q.

Specifically, an input conversation is modeled as a process P with the communications in P being restricted to input communications over provided (input) operations and to

1 http://www.w3.org/2002/ws/chor

output communications over required (output) operations of the mobile component, given that the process starts with an input communication. In addition, communication actions abstract the interaction protocol provided by the presentation layer, i.e., the middleware whose behavior is modeled at the connector level (see § 3.2). This is because, here, we are concerned with modeling the interactions handled by application-related components. For instance, assume a mobile component that is built on top of a middleware supporting RPC-based interactions, i.e., synchronous, two-way interactions. Further assume that the component offers the operation `browse(x)` that returns the set of goods matching the criterion `x`. An input communication (e.g., $browse(x)$) then abstracts the handling of operation invocation at the callee, i.e., reception of the request message followed by the emission of the corresponding result message. Dually, an output communication (e.g., $\overline{browse}(x)$) abstracts operation invocation at the caller, i.e., emission of the request message followed by the reception of the corresponding result message.

An input conversation then characterizes the set of operations that need to be invoked on the software component by the caller for the former to deliver a correct service, as specified by the input communications stated in the corresponding process. In the specific case where P amounts to a single input communication (e.g., $op(x)$), this means that the named operation (e.g., *op*) is self-contained. As an illustration, consider an e-commerce service that allows browsing a catalogue (with operation `browse`) and buying goods (with operation `buy`) after the customer logs in (with operation `login`) until the consumer logs out (with operation `logout`). The corresponding input conversation of the e-service is modeled as: $login(id).(!(browse(x) + buy(y))) + logout(s))$. An input conversation may additionally specify the output communications on which depend the realization of the conversation, i.e., related component integration/composition that is required for correct service provisioning. Such a specification is not mandatory. However, it may be conveniently exploited by the run-time system for anticipating interactions and thus realizing related dynamic bindings before the actual invocations, which is known as prefetching.

Similarly, an output conversation is specified as a process P, with communications in P being restricted to output communications over the component's output operations. The process P specifies the set of interactions that must take place between the caller and the peer components with which it integrates. Still considering our previous example, an output conversation that conforms to the above provided input conversation is: $\overline{login}(id).(!\overline{browse}(x)+\overline{logout}(s))$, assuming a customer that only browses the service catalogue.

A number of consistency checks may then be performed over the specification of a mobile component. In particular, any input conversation that specifies output communications should conform with the component's output conversations associated with related operations. Such a conformance checking lies in verifying observational equivalences between processes, restricted to output communications.

Non-functional specification. As discussed in Section 2.3, the minimal set of QoS attributes for mobile components relates to specifying the component's behavior with respect to performance, reliability, security, and transactional properties, which is further complemented with related resource consumption and service adaptation [16].

Performance and reliability properties are both quantitative. They are respectively specified with the service latency (i.e., response time) and the probability of service availability, for individual operations. From the perspective of the caller, the values of both attributes are further impacted by the available network bandwidth and related mobility of the involved hosts. Security and transactional properties are both qualitative. The former specifies the security protocol on which the mobile component relies for secure communication; it thus relates to the behavior of connectors (see § 3.2). The latter specifies the transactional properties that hold over individual operations and conversations. Resource consumption is further specified quantitatively, stating for each operation, the percentage of resource that is consumed relative to overall resource availability at the host. Finally, service adaptation specifies scaling actions that can be undertaken to accommodate resource constraints.

We get the following modeling for non-functional properties, as highlighted in bold face:

```
Mobile component name
  Provides: {operation name: (signature) (QoS)}*
  Requires: {operation name: (signature) (QoS)}*
  Coordinates:
    Input conversation:
      {conversation name: (process) (transactional) }*
    Output conversation:
      {conversation name: (process) (transactional)}*
```

The value of **QoS** associated with an input (resp. output) operation specifies the provided (resp. expected) QoS for that operation by the component, which decomposes into:

```
QoS:
  (performance:    mean service time in ms)
  (reliability:    mean probability of availability)
  (transactional:  boolean value that evaluates to true
                   if transactional)
  (resources:      CPU, memory, bandwidth, battery)
  (scaling:        scaling actions)
```

The quantitative QoS attributes (i.e., performance, reliability, resource) provide mean values at the provider, given

that they are assumed to be regularly updated. Scaling actions are specified with respect to the content type that is exchanged, providing a sequence of message types ordered according to decreasing level of quality of service with respect to the accuracy of the content (e.g., for an image type, we may have <colour, black&white, text >).

The **transactional** attribute associated with a conversation is a real value d, which equals 0 if the conversation is non-transactional, and is greater than 0 otherwise. The latter case specifies that under the occurrence of a disconnection that lasts longer than d seconds, during the execution of the conversation, the conversation's effect will be undone by the provider(s) according to the atomicity property of transactions.

3.2. Wireless connectors

In the mobile environment, connectors specify the interaction protocols that are implemented over the wireless network. In addition, the dynamic composition of mobile components leads to the dynamic instantiation of connectors. Hence, the specification of wireless connectors is integrated with the one of mobile components (actually specifying the behavior of connector roles), given that the connectors associated with two interacting mobile components must *compose*. The specification of any mobile component is then extended with the one of connector types, as follows:

```
Mobile component name
  Connectors
    {connector name: (protocol) (network) }*
  Provides: as above
  Requires: as above
  Coordinates:
    Input conversation:
      {conversation name: (process) (trans.) (connector) }*
    Output conversation:
      {conversation name: (process) (trans.) (connector) }*
```

`Connectors` specifies the set of connector types via which the mobile component communicates, and **connector** specifies for every conversation, the type of the connector that is used for the corresponding interactions. More precisely, a wireless connector specifies: (i) the base interaction protocol with peer components (**protocol**), and (ii) the underlying dynamic network (**network**).

Interaction protocol. Interaction protocols are specified as processes in the π calculus. However, it should be noted that this characterizes message exchanges at the transport level to realize the higher-level protocol offered by the middleware on top of which the mobile component executes.

Dynamic networking. Specification of the underlying network together with its dynamics abstracts the discovery process that is inherent to today's mobile distributed systems. Such a characterization is further needed due to the diversity of discovery protocols (e.g., wide-area, location-based in the local area and/or with respect to geographic positioning, centralized versus decentralized) together with the one of wireless networks. The dynamic network is then modeled as:

```
Network:
  (Dynamics:      boolean)
  (Area:          [Local | Internet])
  (Location:      area restriction)
  (Security:      protocol)
```

The `Dynamics` attribute specifies whether the peer mobile components accessed via the connector should be dynamically retrieved or not. In the latter case, the addresses of the components are provided as parameters of the connector. The `Area` attribute characterizes whether the peer mobile components should be sought within the local or the wide area (i.e., Internet). The `Location` attribute, which is optional, further specifies the area within which the components are to be retrieved, which may be a given domain identified by its IP address, or a location-dependant domain that is characterized by a geographical area and/or a number of hops, i.e.:

```
Location:
  (Domain:        list of domains' IP addresses)
  (Region:        geographical coordinates of reference,
                  max distance)
  (AdHoc:         Number of hops)
```

The `Security` attribute specifies the security protocol that is implemented for any interaction with components over the wireless connector. The protocol is specified using the π calculus, as introduced in [1] and further elaborated in [6]. Note that in the open, mobile environment, traditional security tokens based on public-key signatures may be too restrictive for the interacting mobile nodes. Instead, interactions may be undertaken on the basis of the mobile components' reputation, and related security protocols [15].

Quality of service. The system's QoS may be solely dependent upon the QoS properties of the interacting components and the underlying network. However, connector customization towards enforcing QoS has been recognized as a key technique since the emergence of mobile clients. Such a customization is realized by specialized networked components, which may be hosted either by stationary servers tightly coupled with the network's infrastructure in the context of infrastructure-based networks [9], or hosted by (possibly mobile) nodes on the network's path [10, 13].

Connector customization is very much dependent upon the non-functional behavior of the interacting components since it aims at adapting the interaction so that the respective provided and requested QoSs match. Hence, customization should be implicit, and inferred from the respective

behavior of interacting components and properties of the underlying network. This requires the specification of customization processes so as to be able to dynamically integrate them. Based on our previous work [13], this leads us to introduce *customizer* mobile components that filter exhanged messages. The specification of a customizer is then similar to that of a mobile component (§ 3.1).

4. Dynamic Composition

Given the above specification of mobile components and related wireless connectors, integration and composition of mobile components in a way that ensures correctness of the mobile distributed system may be addressed in terms of conformance of respective functional and non-functional specifications.

4.1. Specification conformance

We use the following notation to characterize the specification of a component M:

$\mathcal{I}$	Input/provided operations, with element denoted by in
$\mathcal{O}$	Output/required operations, with element denoted by out
$\mathcal{C}_\mathcal{I}$	Input conversations, with element denoted by $conv$
$\mathcal{C}_\mathcal{O}$	Output conversations, with element denoted by $conv$
$\mathcal{I}_{QoS}$	QoS of input operations, with element denoted by $qos(in) = (p, a, t, r, s)$
$\mathcal{O}_{QoS}$	QoS of output operations, with element denoted by $qos(out) = (p, a, t, r, s)$
$\mathcal{C}_{I_t}$	Transactional properties of input conversations, with element denoted by $t(conv)$
$\mathcal{C}_{O_t}$	Transactional properties of output conversations, with element denoted by $t(conv)$
$\mathcal{C}$	Set of connector types
$\mathcal{B}$	Connectors associated, with conversations, with element denoted by $connector(conv)$

Consider two mobile components:

$$M_1 =< \mathcal{I}_1, \mathcal{O}_1, \mathcal{C}_{\mathcal{I}1}, \mathcal{C}_{\mathcal{O}1}, \mathcal{I}_{QoS_1}, \mathcal{O}_{QoS_1}, \mathcal{C}_{I_{t_1}}, \mathcal{C}_{O_{t_1}}, \mathcal{C}_1, \mathcal{B}_1 >$$
$$M_2 =< \mathcal{I}_2, \mathcal{O}_2, \mathcal{C}_{\mathcal{I}2}, \mathcal{C}_{\mathcal{O}2}, \mathcal{I}_{QoS_2}, \mathcal{O}_{QoS_2}, \mathcal{C}_{I_{t_2}}, \mathcal{C}_{O_{t_2}}, \mathcal{C}_2, \mathcal{B}_2 >$$

Then, M_1 and M_2 may be composed if the conversations that bind according to respective input and output operations, conform to each other, i.e., if the following $Conform(M_p, M_c)$ (where p stands for provider and c for client) relation holds for $(p, c) = (1, 2)$ (i.e., M_1 being the service provider) and $(p, c) = (2, 1)$ (i.e., M_1 being the service client):

$$\begin{aligned} Conform(M_p, M_c) \equiv ((\forall(conv_p = P) \in \mathcal{C}_{\mathcal{I}_p}) : \\ (init(conv_p, out) \wedge (out \in \mathcal{O}_c)) \Rightarrow \exists(conv_c \in \mathcal{C}_{\mathcal{O}_c}) | \\ conv_c \hookrightarrow conv_p \wedge \\ t(conv_c) \leq t(conv_p) \wedge QoS_{conv_c} \leq_{qos} QoS_{conv_p} \wedge \\ connector(conv_c) \sim_{co} connector(conv_p)) \end{aligned}$$

where $\forall, \exists, \rightarrow, \wedge$ denote classical logical operators for quantification, implication and conjunction; and:

- $init(conv_p, out)$ holds if $conv_p$ starts by executing the out operation.
- $conv_c \hookrightarrow conv_p$ holds if the ouput conversation $conv_c$ conforms to the input conversation $conv_p$ for all the calls to M_p's input operations. Specifically, the parallel execution of the two processes when restricted to interactions with M_p's input operations (noted $conv'_c$ and $conv'_p$) reduces to the null process (i.e., $(conv'_c | conv'_p) \rightarrow^* \sim \mathbf{0}$ with $\rightarrow^*$ denoting a series of reduction and $\sim$ denoting observational equivalence) and the types of the exchanged messages match in the classical sense of type checking.
- $t(conv_c) \leq t(conv_p)$ holds if the transactional behavior of $conv_c$ is weaker than that of $conv_p$ (i.e., $conv_c$ is transactional if $conv_p$ is and the timeout of $conv_c$ is smaller).
- $QoS_{conv_c} \leq_{qos} QoS_{conv_p}$ holds if the QoS provided by the callee is at least the one expected by the caller for all the input operations of M_p that belong to $conv_p$. Specifically, the quantitative performance (resp. reliability) attributes of the provider must be smaller (resp. greater) or equal to the values requested by the client, the operation must be transactional if it is requested to be so, and the client and provider must agree on similar scaling actions, i.e.:
  $\forall in \in conv_p | qos_p(in) = (p_p, a_p, t_p, r_p, s_p) :$
  $in \in conv_c \wedge qos_c(in) = (p_c, a_c, t_c, r_c, s_c) \wedge$
  $(p_c \geq p_p) \wedge (a_c \leq a_p) \wedge (t_c \Rightarrow t_p) \wedge scalable(s_c, s_p)$
  where the $scalable(s, s')$ relation holds if for any type in s, there is a matching type in s'.
- $connector(conv_c) \sim_{co} connector(conv_p)$ holds if the connectors associated with the respective conversations *compose*, i.e., the following conditions are met for $connector(conv_p) = (I_p, net_p)$ and $connector(conv_c) = (I_c, net_c)$:
  - The connectors implement compatible interaction protocols. Specifically, the parallel execution of the corresponding processes reduces to the null process, i.e., $(I_c | I_p) \rightarrow^* \sim \mathbf{0}$
  - The respective dynamic networks of the components intersect. Specifically, when omitting keywords, for:
    $net_p = (dyn_p, (area_p, dom_p, region_p, adhoc_p), sec_p)$,
    $net_c = (dyn_c, (area_c, dom_c, region_c, adhoc_c), sec_c)$,
    we have:
    $$(dom_p \bigcap\nolimits_{IP} dom_c \neq \emptyset) \wedge$$
    $$(region_p \bigcap\nolimits_{space} region_c \neq \emptyset) \wedge$$
    $$(adhoc_p \bigcap\nolimits_{adhoc} adhoc_c \neq \emptyset)$$
    where $\bigcap_{IP,space,adhoc}$ define intersection on the

respective value domains.
- The connectors implement compatible security protocols. Specifically, the parallel execution of the corresponding processes reduces to the null process, i.e., $(sec_c|sec_p) \rightarrow^* \sim \mathbf{0}$).

4.2. Customization

Two mobile components may further be dynamically composed through connector customization, which adapts the components' behaviors so that they conform. We say that a customizer *composes* with a pair of interacting components if the interaction protocol implemented by its connectors for processing input messages (resp. output messages) is compatible with the ones implemented by the relevant connectors of the components, where compatibility of protocols is defined as above, i.e., in terms of reduction to the null process. We then weaken the above conformance relation as follows to support connector customization:

- The types of exchanged messages are not required to match, provided that there exists a customizer that *composes* and adapts the exchanged content in a way that the content type conforms to the one expected by the interacting components.
- The QoS of an input operation is not required to be at least the one of the output operation with which it binds; provided that there exists a customizer that *composes* and delivers the expected QoS.

4.3. Example

In order to illustrate the exploitation of our model, we consider access to a shopping e-service, for which there exist mobile replicated instances, noted me, on the sellers' wireless devices and a stationary instance, noted se, hosted by some server. Access to se and me is allowed for any authenticated client over the wireless Internet.

For simplicity, we focus on the interactions between the mobile and stationary service instances, and consider a subset of relevant operations, i.e., the service offers two transactional conversations that respectively allows browsing a catalogue and buying goods. Both conversations are carried out over an RPC-like connector with reception of the request message followed by the emission of the request result. The wireless connector implements a security protocol based on certificates, specified as:

$$CP \equiv \quad (authenticate().\overline{srv}(certificate).\\ shared(sk).login(encryptedId_{sk}))$$

Informally, the client requests the service provider to authenticate itself by issuing the $authenticate$ message. This leads the service provider to send back its certificate ($\overline{srv}(certificate)$), and then the client to send the shared secret key sk that it computed ($shared(sk)$) followed by its login that is encrypted using the shared key ($login(encryptedId_{sk})$). We further assume the dual security protocol $\overline{CP}$ at the client, i.e.:

$$\overline{CP} \equiv \quad (\overline{authenticate}().srv(certificate).\\ \overline{shared}(sk).\overline{login}(encryptedId_{sk}))$$

All the operations of se are further considered to be transactional and to provide very high performance and availability guarantees (denoted by ∞) with negligible resource consumption (denoted by ϵ); this leads se to not support content scaling. We use $optimal$ as a shorthand notation to specify that an operation offers maximal QoS guarantees (i.e., we have (`QoS:` (`performance:` ∞) (`reliability:` ∞) (`transactional:` `true`) (`resources:` $\epsilon, \epsilon, \epsilon, \epsilon$) (`scaling:` $\emptyset$))). Finally, there is no restriction on the location area, leading us to omit the optional `Location` attribute. We get the following specification for the stationary service se, for which we do not detail content types:

```
Mobile component se
  Connectors
    service:
      (req(x).res(y))
      (Network:
        (Dynamics: true)
        (Area: Internet)
        (Security: CP process, as specified above)
  Provides:
    browse: (x) (QoS: optimal)
    book: (y) (QoS: optimal)
    buy: (z) (QoS: optimal)
  Requires: ∅
  Coordinates:
    Input conversation:
      BROWSE: (browse(x)) (0) (service)
      BUY: (book(x).(!book(x) + buy(y))) (60) (service)
    Output conversation: ∅
```

The specification of me is similar to the one of se regarding the definition of input operations and conversations, since the functional behavior that it offers to its clients is equivalent to that of se. On the other hand, its non-functional behavior changes due to its mobile nature, and me further synchronizes with the stationary server when connection allows by forwarding the requests that it has logged. In addition, the conversations with the stationary server do not specify any QoS requirements and are carried out over a connector that restricts the service provider to be se, which is passed as parameter. We get:

```
Mobile component me
  Connectors
    service:
      (req(x).res(y))
      (Network:
```

(Dynamics: true)
(Area: Local)
(Security: *CP process, as specified above*))
client(s: se):
($\overline{req}(x).res(y)$)
(Network:
(Dynamics: false)
(Area: Internet)
(Security: $\overline{CP}$ *process, as specified above*))
Provides:
browse: (x)
(QoS:
(performance: 10)
(reliability: 0.99)
(transactional: false)
(resources: 2%, 10%, 10%, 0.2%)
(scaling: $\emptyset$))
book: (y)
(QoS:
(performance: 20)
(reliability: 0.99)
(transactional: true)
(resources: 3%, 10%, 5%, 0.1%)
(scaling: $\emptyset$))
buy: (z)
(QoS:
(performance: 60)
(reliability: 0.99)
(transactional: true)
(resources: 3%, 10%, 5%, 0.5%)
(scaling: $\emptyset$))
Requires:
browse: (x) (QoS: $optimal$)
book: (y) (QoS: $optimal$)
buy: (z) (QoS: $optimal$)
Coordinates:
Input conversation:
BROWSE: ($browse(x)$) (0) (service)
BUY: ($book(x).(!book(x) + buy(y))$) (30) (service)
Output conversation:
CACHE: ($\overline{browse}(x)$) (0) (client)
SyncBUY: ($\overline{book}(x).(!\overline{book}(x) + \overline{buy}(y))$) (50) (client)

From the above, it is quite trivial to show that se conforms to me according to the definition provided in Section 4.1. Hence, they can be dynamically composed, while ensuring correct system behavior with respect to provided functional and non-functional properties. However, this needs to be realized online and hence requires adequate middleware support.

5. Supporting Middleware

We have introduced base architectural modeling for mobile distributed systems, which enforces correctness of the distributed systems despite the high dynamics of the underlying network. However, related verification of the dynamically composed systems must be integrated with the runtime system, i.e., the middleware. The issue that arises then relates to the processing and communication costs associated with verification, which must be kept low on the wireless devices.

As part of our work, we have developed the WSAMI service-oriented middleware for mobile computing that is based on the Web services architecture [13]. Briefly stated, the WSAMI core middleware subdivides into: (i) the WSAMI SOAP-based core broker, including the CSOAP SOAP container for wireless, resource-constrained devices, and (ii) the Naming & Discovery (ND) service for the dynamic discovery of (possibly mobile) services that are available in the local and wide area, according to network connectivity and available resources. The ND service further includes support for connector customization, so as to enforce quality of service through the dynamic integration of middleware-related services over the network's path. The WSAMI middleware has been designed so as to minimize resource consumption on the wireless devices. In particular, mobile services that are integrated and composed using WSAMI are modeled by specifying the URIs of the XML documents that characterise their input and output operations, and their input and output conversations.

Using WSAMI, components composition with respect to QoS and wireless connectors is built-in in the middleware, i.e., QoS-awareness is managed by the ND service and the interaction protocol is that of CSOAP. It follows that the retrieval of peer components with which a mobile service dynamically composes is simply based on the comparison of URIs[2], which is much effective in terms of low resource consumption. However, such an approach leads to a much stronger conformance relation for dynamic composition than the one discussed in the previous section, i.e., the conformance relation that is implemented in WSAMI requires two interacting components to have structural equivalence over output and input operations, and over output and input conversations. As part of our current work, we are investigating weaker conformance relations for WSAMI based on our general definition, while still making the associated computation and communication costs low for wireless devices. A number of techniques need to be combined in this context, including effective tool for checking conformance relationship in the wireless environment, possibly leading to strengthening base conformance relation, and exploiting the capabilities of resource-rich devices in the area so as to effectively distribute the load associated with the dynamic composition of mobile components.

2 Note that this leads to model output conversations from the standpoint of the callee at the caller so that input and output conversations are structurally equivalent.

WSAMI further restricts the systems' openness in that it supports the dynamic composition of WSAMI-enabled components. Such a feature is inherent to middleware-based systems. However, high-level modeling of mobile components for composition, as addressed in this paper, paves the way for semantic-based integration and composition of mobile systems. Specifically, the behavioral specification of components may be exploited towards adapting the non-functional behavior of the mobile components so that they can integrate, regarding both offered QoS and interaction protocols. Such a concern is in particular exemplified by our approach to connector customization. It further relates to work undertaken within the Semantic Web for better reasoning about the Web content[3]. However, such a capability can only be supported if provided with effective tools for conformance checking and techniques for adapting the interaction protocols of components. Base approaches can be found in the literature to address the latter, introducing automated solutions to, e.g., dynamically change the discovery protocols [11] and to adapt to protocol evolution [23].

6. Conclusion

Supporting the development of applications for the wireless environment has led to tremendous research effort over the last ten years. In particular, much work is devoted to performance-related issues, investigating middleware solutions to deal with the key features of the wireless infrastructure, i.e., dynamics of the network and resource constraints of the wireless devices. Mobility has also deserved much attention from a more theoretical perspective, with primary focus on the mobility of computation (i.e., mobile code), as opposed to personal and computer mobility. In the context of computer mobility, distributed systems now tend to be dynamically composed according to the networking of mobile services, which is in particular supported by the integration of resource discovery protocols within the middleware (§ 2). However, such a composition must be addressed in a way that enforces correctness of the dynamic composite systems with respect to both functional and non-functional properties. Towards that goal and building upon the elegant properties of software architecture, this paper has introduced a base approach to the modeling of mobile software components (services) hosted by wireless hosts and of related wireless connectors. The proposed solution integrates key features of today's wireless infrastructure, in particular dealing with the network's dynamics and quality of service requirements that call for special care due to inherent resource limitation (§ 3). We have further introduced a conformance relation over component and connector models so as to be able to reason about the correctness of the composition of peer mobile components with respect to offered functional and non-functional properties (§ 4). Related verification needs to be carried out online and hence integrated within the middleware, possibly leading to stronger conformance relation to make the associated resource consumption acceptable for wireless devices (§ 5).

By undertaking an approach based on software architecture modeling, our work builds upon work in the area and in particular on the modeling of interaction protocols associated with connectors [2]. We more specifically use the π calculus as the base process algebra, which allows dealing with dynamic architectures [17]. In addition to the specification of the base interaction protocol that abstracts the behavior of the middleware with respect to communication, we introduce the specification of coordination (also referred to as conversation and choreography) and security protocols within components. The former is mandatory to reason about the functional composition of components and further about the impact of disconnection. The latter relates to one of the key QoS attributes in the wireless context. Our approach to the specification of coordination protocols relates to the definition of XML languages for specifying choreography associated with Web services[4], since it is quite direct to translate corresponding π processes in a choreography language and conversely. In this context, our modeling of components resembles work on the definition of a formal ontology framework for Web services to support the description, matching and composition through logic reasoning techniques [19]. Our contribution specifically relates to dealing with mobile components (which may be instantiated as mobile Web services), leading us to integrate the specifics of the wireless environment in the component modeling.

Architecture-based development of mobile distributed systems is in particular addressed by the AURA project [26]. Briefly stated, a user-task is modeled as a software architecture description that specifies the abstract application-related services and connectors to be composed, and the environment instantiates and adapts the architecture according to available component instances and resources. Architecture modeling in AURA is specifically targeted at the AURA middleware, providing input to the built-in discovery and composition process. Our work is thus complementary, addressing formal modeling towards reasoning about the correctness of composite mobile systems with respect to functional and non-functional behavior, and further addressing key characteristics of the wireless infrastructure.

As discussed in Section 5, openness of the composition process with respect to the mobile components that can actually be integrated is dependent upon the underlying middleware. Considering existing middleware (e.g., AURA and

3 http://www.w3.org/2001/sw

4 http://www.w3.org/2002/ws/chor

WSAMI), a direct approach is to strengthen the conformance relation over mobile components, which in particular allows minimizing resource consumption induced by the composition process. However, this impacts upon the degree of pervasiveness of mobile computing. We are thus currently investigating base online tools and techniques to support open, dynamic system composition according to the conformance relation introduced in this paper, while keeping the runtime overhead acceptable for wireless, resource-constrained devices.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, 2001.

[2] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.

[3] L. Andrade and J. Fiadeiro. Architecture-based evolution of software systems. In *Formal Methods for Software Architectures*, 2003. LNCS 2804.

[4] C. Aurrecoechea, A. Campbell, and L. Hauw. A survey of QoS architectures. *ACM/Springer Verlag Multimedia Systems Jounral, Special Issue on QoS Architectures*, 1998.

[5] C. Bettstetter and C. Renner. A comparison of service discovery protocols and implementation of the service location protocol. In *Proceedings of the 6th EUNICE Open European Summer School: Innovative Internet Applications*, 2000.

[6] K. Bhargavan, C. Fournet, and A. Gordon. A semantics for web services authentication. In *Proceedings of the 31st ACM Symposium on Principles of Programming Languages (POPL'01)*, 2004.

[7] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of Mobicom'98*, 1998.

[8] H. Dijk, K. Langendoen, and H. Sips. ARC: A bottom-up approach to negotiated QoS. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'00)*, 2000.

[9] A. Fox, S. D. Gribble, and Y. Chawathe. Adapting to network and client variation using active proxies: Lessons and perspectives. *Special Issue of IEEE Personal Communications on Adaptation*, 1998.

[10] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS: composable, adaptive network services infrastructure. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001.

[11] P. Grace, G. Blair, and S. Samuel. Middleware awareness in mobile computing. In *Proceedings of the 1st International ICDCS Workshop on Mobile Computing Middleware*, 2003. Available at http://www.cs.ucl.ac.uk/staff/c.mascolo/mcm03.

[12] V. Issarny, C. Kloukinas, and A. Zarras. Systematic aid for developing middleware architectures. *Communications of the ACM*, 45(6), 2002.

[13] V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona. Developing ambient intelligence systems: A solution based on web services. *Journal of Automated Software Engineering*, 2004. To appear.

[14] D. Le Métayer. Software architecture styles as graph grammars. In *Proceedings of the ACM SIGSOFT'96 Symposium on Foundations of Software Enineering*, pages 15–23, 1996.

[15] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In *Proceedings of the 2nd International Conference on Trust Management (iTrust'04)*, 2004.

[16] J. Liu and V. Issarny. QoS-aware service location in mobile ad hoc networks. In *Proceedings of the 5th IEEE International Conference on Mobile Data Management*, 2004.

[17] J. Magee and J. Kramer. Dynamic structure in software architecture. In *Proceedings of the ACM SIGSOFT'96 Symposium on Foundations of Software Engineering*, pages 3–14, 1996.

[18] R. Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[19] C. Pahl and M. Casey. Ontology support for Web service processes. In *Proc. of the Joint 9th European Software Engineering Conference (ESEC) and 11th SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, 2003.

[20] G. Picco, A. Murphy, and G.-C. Roman. Developing mobile computing applications with LIME. In *Proceedings of the 22nd International Conference on Software Engineering*, 2000.

[21] M. Ritter. Mobility network systems. *ACM Queue – Tomorrow's Computing Today*, 1(3), 2003.

[22] G.-C. Roman, G. Picco, and A. Murphy. Software engineering for mobility: A roadmap. In *Proceedings of the 22nd International Conference on Software Engineering*, 2000.

[23] N. Ryan and A. Wolf. Using Event-based Parsing to Support Dynamic Protocol Evolution. Technical Report TR CU-CS-947-03, Department of Computer Science, University of Colorado, Boulder, CO, USA, 2003.

[24] B. Sabata, S. Chatterjee, M. Davis, J. Sydir, and T. Lawrence. Taxonomy for QoS specification. In *Proceedings of the of the International Workshop on Object-oriented Real-time Dependable Systems (WORDS'97)*, 1997.

[25] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, 2003.

[26] J. P. Sousa and D. Garlan. Aura: an architectural framework for user mobility in ubiquitous computing environments. In *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, 2002.

[27] D. B. Terry, M. M. Theimer, K. P. A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in a weakly connected replicated storage system. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, 1995.