Context-Based Prevention and Handling of Exceptions for Human-Centric Mobile Services

Rüdiger Pryss and Manfred Reichert
Institute of Databases and Information Systems, Ulm University, Germany
{ruediger.pryss, manfred.reichert}@uni-ulm.de

Abstract—Using smart mobile devices to support human-centric services is a frequent demand in business scenarios. As a particular challenge, tasks performed in a paper-driven way shall be digitally transformed with the use of mobile devices. With the goal to enable business applications supporting human-centric mobile services in mind, we developed a framework that extends existing process management technology with mobile activities running on smart mobile devices. Note that when considering the frequently changing conditions of mobile environments, the prevention and the proper handling of exceptions (e.g., lost connections) become crucial. The developed framework, therefore, aims to prevent exceptions and to provide a sophisticated exception handling service not supported by existing process management technology so far.

Keywords—mobile service, mobile process, exception handling

I. INTRODUCTION

Using process management technology in the context of business scenarios is a fundamental trend in enterprise computing [1]. However, approaches integrating smart mobile devices with process management technology are rather premature. To remedy this drawback, we developed a framework that enables support for mobile activities. The latter constitute process activities, i.e., single process steps, to be executed on mobile devices. The developed framework, in turn, uses a mobile context to tackle the challenges raised by the use of mobile devices (e.g., connection losses). To elaborate fundamental requirements for a framework enabling human-centric mobile services, we analyzed a multitude of real-world scenarios and other works [2]. Based on insights gained from the various scenarios, we were able to elicit fundamental requirements for human-centric mobile services. This work focuses on the two important requirements how to prevent and handle exceptions. The paper is organized as follows. Section II introduces human-centric mobile services. In Section III, the mobile context is defined, whereas Section IV presents the exception handling service. Finally, Section V discusses related work and Section VI concludes the paper with a summary and outlook.

II. HUMAN-CENTRIC MOBILE SERVICES

First of all, we develop our framework to enable human-centric mobile services in a process context. Thereby, a process management system assigns and manages (i.e., starts and stops) process activities based on a given process schema [1]. In this context, two basic types of activities need to be distinguished: the ones automatically executed (e.g., Web Service calls) on one hand and activities performed by users on the other. The latter are denoted as human activities. Based on this, we denote context-aware human activities running on smart mobile devices and being controlled by a process management system as human-centric mobile services. In this context, a match-making model [3] becomes necessary that determines which human-centric mobile service will be assigned to which mobile user. For this purpose, we developed a process meta-model that considers mobile services in mind, we developed a framework that enables support for mobile activities. The latter are denoted as human activities. As can be seen, many interdependencies must be tackled to enable the required worklist management for mobile activities.

In the context of worklist management, two algorithms were developed (cf. Fig. 1(1),2) that are responsible for the match-making model. These algorithms are the basis for our exception handling service. The first algorithm manages user assignments and the execution of mobile activities (cf. Fig. 1(1)) with the goal to prevent exceptions. It is denoted as Selection Algorithm. The second algorithm, the Ranking Algorithm, handles exceptions (e.g., smart mobile device crashes) during the execution of mobile activities (cf. Fig. 1(2)). To enable a proper exception handling, and as a prerequisite for the worklist management, a complex state model of mobile activities became necessary (cf. Fig. 1(3)).

Since worklist management plays an important role for the exception handling service, we conceive the basic technical aspects behind the management of worklists [5]. To properly assign mobile activities to mobile users, a process client must be provided, which needs to be deployed to the users’ mobile devices. In turn, the communication between the process client and the process management system requires protocols and the worklist management. The latter is fundamental to determine which mobile user actually performs a mobile activity as more than one user may be
ActivitiesAtHand for all other users whose and removed from the ActivitiesAtHand mobile activity is added to the MyActivities list. In addition, for all other users whose ActivitiesAtHand list contains this mobile activity, the latter will be removed. Finally, if the user declines a mobile activity, it is solely removed from this particular ActivitiesAtHand user list.

III. Mobile Context

The handling of mobile activities is governed by a mobile context to enhance worklist management. We learned from the scenarios we analyzed that users base their decision whether or not they actually execute a mobile activity on recurrent patterns. For example, if a mobile activity is often performed at the same location, users keep that in mind when making this decision. Accordingly, a location pattern can be derived. Based on these insights as well as a comprehensive literature study (e.g., [2], [4]), for each identified pattern we elaborated an appropriate value (called parameter). These values are used to assign mobile users to mobile activities, to prevent exceptions, and to handle exceptions (cf. Table I). Consider Table I. Note that Column $T$ indicates whether a parameter is of type symbolic or measured. From the considered application scenarios we revealed that such differentiation is useful. Symbolic parameters are used in related work to define parameters on an abstract level [6]. For example, regarding the location of a mobile activity, the symbolic parameter emergency room might be used. Symbolic parameters are considered as they can already be evaluated before starting a process. For example, if a symbolic parameter emergency room is assigned to a mobile activity, it can be further determined how many mobile users hold value emergency room as their symbolic parameter. Conversely, measured parameters are automatically determined by the process management system after starting a process instance. For example, if a mobile activity shall be executed, the battery status of all mobile users will be gathered. A detailed discussion of all parameters can be found in [5].

Regarding the use of the identified parameters, we do not claim that they cover all relevant patterns, i.e., they rather reflect empirical insights we gathered from the analyzed scenarios. Future analyses might reveal additional parameters or invalidate existing ones. Furthermore, the efficacy of parameters must be evaluated. Therefore, we are currently running a case study with the goal to measure efficacy in more detail. However, in the context of worklist management enhancement, their use has been promising in all practical scenarios. In particular, domain experts were able to determine useful parameter values.

IV. Exception Prevention and Handling

Our exception handling is based on two measures (cf. Table II). As the first measure, the selection algorithm presented in [5] assigns mobile activities only to those users who less likely cause an exception. For example, the selection algorithm evaluates whether or not a mobile user is closely located to a mobile activity. The assignment based on a close location revealed quicker execution times and hence less errors. Second, if errors occur, the ranking algorithm determines appropriate mobile users to handle an exception. Note that many existing approaches do not generally consider exception prevention/handling for human
activities in a process context. Although WS-BPEL 2.0 extensions like BPEL4People and WS-HumanTask [7], [8] focus on human activities, they solely address exception handling through mechanisms on an abstract level. Our approach deals with this drawback and handles exceptions in a new way. Thereby, we mainly focus on mobile activities. More precisely, we developed the delegation mechanism. The delegation constitutes the primary exception handling concept for mobile activities. As scenarios exist in which a delegation cannot be performed, additionally, we perform a skip or backup of mobile activities as follow-up strategy.

To decide whether a skip or backup will be performed, the mobile context is evaluated. Note that a backup means transferring the execution from the mobile device executing the mobile activity to a stationary system. Further note that two activity states were added to the presented mobile process meta-model (i.e., States Delegated and Backed Up; cf. Fig. 2) to realize the overall exception handling service.

This work focuses on the delegation concept. Based on the mobile context, a delegation identifies appropriate mobile users being able to perform the mobile activity (as alternative to the mobile user who caused the exception). The process management system then considers these users as possible targets for handling the exception. If no such user can be identified based on the mobile context, the mobile activity will be either skipped or its execution will be migrated to a stationary system. For realizing the delegation concept, four delegation aspects (DA1–DA4) have to be covered (cf. Table III). Regarding DA1, parameter response frequency is evaluated. It determines the frequency with which the mobile device of a particular user must report its online status to the process management system. If the smart mobile device does not obey this reporting frequency, an exception handling will be triggered.

Regarding DA2, the Ranking Algorithm was developed to determine appropriate mobile users for a delegation. Based on the mobile context, it determines all mobile users being an appropriate target for the delegation. In addition, it determines a rank, again based on the mobile context, for all appropriate users. Then, the process management system solely considers the user ranked highest with respect to a delegation. If the user does not accept the request, all other users will be subsequently requested in descending rank order until one of them accepts the request or all users will have declined it. In this context, tracking the number of delegations, required in the context of a mobile user, is useful for determining the rank value of a user. To manage the number of delegations, we use parameter $MU_{DB}$ (Mobile User Delegation Behavior; cf. Table IV). In addition, parameter $MU_{RB}$ can be used to manage the resource behavior (cf. Table IV). More precisely, with $MU_{RB}$ we evaluate how users behave with respect to the resources of their mobile devices when executing mobile activities. The rank value for mobile users is determined through Algorithm 1, which is triggered when delegating a mobile activity. First, all mobile users are ranked according to their resource and delegation behavior (cf. Lines 4, 8, and 12). Second, the location of a user is considered through the location values (cf. Table IV). Best case, the user is inside the location range of the mobile activity (cf. Line 4), i.e., the user gets the highest possible rank. Those users only matching the symbolic coordinates (cf. Line 8) are still ranked high, but not as good as in

### Table II: Prevention and Handling of Exceptions

<table>
<thead>
<tr>
<th>Exception Handling</th>
<th>Prevention</th>
<th>Exception Handling</th>
<th>Selection Algorithm</th>
<th>Ranking Algorithm</th>
<th>AT</th>
<th>ET</th>
<th>ExT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

### Table III: Delegation Aspects

<table>
<thead>
<tr>
<th>Delegation Aspect (DA) Description</th>
<th>Requires</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA1</td>
<td>Evaluation of mobile context.</td>
</tr>
<tr>
<td>DA2</td>
<td>Evaluation of mobile context.</td>
</tr>
<tr>
<td>DA3</td>
<td>Evaluation of assignment protocol.</td>
</tr>
<tr>
<td>DA4</td>
<td>Evaluation of mobile context.</td>
</tr>
</tbody>
</table>

Figure 2: Exception Handling State Model
the best case. All other users get lower ranks (cf. Line 12). After calculating the rank list coMU(n) of mobile users, the worklist of the user ranked highest in coMU(n) (i.e., the first list entry) is updated by adding the mobile activity to the DelegationRequests list. As soon as this list is updated on the smart mobile device, the user gets informed about the delegation request. If the user declines it, the delegation is requested from the next user in coMU(n). This may be repeated until all users have decided about the request. If all users decline, the respective mobile activity will be skipped or the backup be performed (cf. Fig. 2).

Regarding DA3, the protocol coordinating the interactions between the mobile process client and the process management system is presented (cf. Fig. 3). The delegation concept presented in this work is realized by the mobile activity handler. Therefore, we realized a mobile process client consisting of a worklist client and an execution client to manage the entire communication between the mobile process client and a process management system. Thereby, the worklist client manages the worklist, whereas the execution client manages the communication between the worklist client, an invoked mobile application, and the process management system. The invoked mobile applications, in turn, actually perform the mobile activity (e.g., invoking Mobile Microsoft Excel). Based on this, the delegation protocol was realized. It governs the interactions between the mobile process client and the process management system in case of a delegation.

The protocol steps are depicted in Fig. 3. Thereby, steps within the In-Delegation box are crucial for handling the steps performed after a delegation.

Two scenarios must be basically distinguished. First, the smart mobile device might no longer work after the occurrence of the exception. Second, it might still work, but no longer be connected to the process management system. In the first case, all steps shown for the mobile process client are not performed. In the second case, all shown steps are performed. After starting the delegation, the process management system performs the following steps. First, it withdraws the mobile activity running on the smart mobile device by updating its status (cf. Steps 10′-11′; (cf. Fig. 3(7))). Second, after updating the status it determines whether the smart mobile device has reconnected in case the connection loss was only a short-term problem (cf. Step 12′). Third, depending on the result of Step 12′, it may start the delegation, i.e., Algorithm 1 will be invoked. The mobile process client, in turn, applies the following steps. First, the running activity is stopped and the data created is locally cached (cf. Steps 10-13). Second, after the smart mobile device reconnects to the process management system, it requests the status of the delegation (cf. Step 14) and sends its cached data to the process management system.

Two additional scenarios need to be distinguished (cf. Fig. 3(6)) after a reconnection. First, if a delegation has not been accomplished yet, the reconnecting smart mobile device gets

---

**Algorithm 1: Ranking Algorithm**

**Data:** Relevant context parameters
- a MU(n): Set of all qualified mobile users of mobile activity n
- For all users in a MU, their current parameters MURB, MURB, MULB, the location values LV1-LV3

**Result:** coMU(n): Ranked list of mobile users being an appropriate delegation target

```
begin
1. foreach mobile user mu ∈ aMU(n) do
2.   if (0 < nLR(mu) ≤ 1) then
3.     LV1 = MURB(mu) + MULB(mu) + MU1G(mu);
4.     MULB(mu) ←- MULB(mu) ∪ {mu};
5.   else if MASC(n) = MASC(mu) then
6.     LV2 = MURB(mu) + MULB(mu) + MU1G(mu);
7.     MULB(mu) ←- MULB(mu) ∪ {mu};
8.   else if MASC(n) = MASC(mu) then
9.     LV3 = MURB(mu) + MULB(mu) + MU1G(mu);
10.    MULB(mu) ←- MULB(mu) ∪ {mu};
11. end
12. end
13. /* Call method to sort coMU(n) based on MULB in ascending order */
```

---

**Table IV: Parameters for Ranking Algorithm**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULB</td>
<td>Stores resource behavior of a mobile user based on MACL (LB: Low Battery).</td>
</tr>
<tr>
<td>MURB</td>
<td>Stores the delegation behavior of a mobile user. It is determined by the number of activities a user has claimed in relation to the # of activities that must be delegated for this user (DB: Delegation behavior).</td>
</tr>
<tr>
<td>MU1G</td>
<td>Stores the rank of a mobile user determined by the Ranking Algorithm (Rank).</td>
</tr>
<tr>
<td>LV1-LV3</td>
<td>Stores location values used by the Ranking Algorithm (LV: Location Value). LV1 &lt; LV2 &lt; LV3.</td>
</tr>
</tbody>
</table>

---

**Algorithm 2: Mobile Activity Handler**

**Data:** Relevant context parameters
- a MA(C): Set of all mobile activities executable by a mobile client
- For all mobile activities, their current parameters MA1C, MA2C, MA3C, the location values LV1-LV3

**Result:** coMA(C): Ranking list of mobile activities executable by a mobile client

```
begin
1. foreach mobile activity ma ∈ aMA(C) do
2.   if (0 < nLR(ma) ≤ 1) then
3.     LV1 = MA1C(ma) + MA2C(ma) + MA3C(ma);
4.     MA1C(ma) ←- MA1C(ma) ∪ {ma};
5.   else if MASC(n) = MASC(ma) then
6.     LV2 = MA1C(ma) + MA2C(ma) + MA3C(ma);
7.     MA1C(ma) ←- MA1C(ma) ∪ {ma};
8.   else if MASC(n) = MASC(ma) then
9.     LV3 = MA1C(ma) + MA2C(ma) + MA3C(ma);
10.    MA1C(ma) ←- MA1C(ma) ∪ {ma};
11. end
12. end
13. /* Call method to sort coMA(C) based on MA1C in ascending order */
```

---

**Figure 3: Delegation Protocol**

---

**Table IV: Parameters for Ranking Algorithm**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULB</td>
<td>Stores resource behavior of a mobile user based on MACL (LB: Low Battery).</td>
</tr>
<tr>
<td>MURB</td>
<td>Stores the delegation behavior of a mobile user. It is determined by the number of activities a user has claimed in relation to the # of activities that must be delegated for this user (DB: Delegation behavior).</td>
</tr>
<tr>
<td>MU1G</td>
<td>Stores the rank of a mobile user determined by the Ranking Algorithm (Rank).</td>
</tr>
<tr>
<td>LV1-LV3</td>
<td>Stores location values used by the Ranking Algorithm (LV: Location Value). LV1 &lt; LV2 &lt; LV3.</td>
</tr>
</tbody>
</table>

---

**Algorithm 1 (cont.):**

```
begin
/* Call method to sort coMU(n) based on MULB in ascending order */
```

---

**Algorithm 2 (cont.):**

```
begin
/* Call method to sort coMA(C) based on MA1C in ascending order */
```
the activity execution back. Second, if the delegation is still running, the cached data is transferred to the smart mobile device currently performing the delegation. This way it can be ensured that no data is lost. In addition, a feature was realized that enables recipients to manually decide whether or not to use cached data before it will be actually transferred. Furthermore, we identified the protocol points at which the mobile context parameters shall be exchanged between the mobile process client and the process management system. Fig. 3 ②,④ depicts protocol points at which the process management system requests parameter values from the mobile process client, whereas Fig. 3 ①,③,⑤ depicts protocol points at which the mobile process client sends parameter values to the process management system.

Regarding DA4, we distinguish two delegation types based on urgency parameter $MA_U$. If this parameter is set to true, the final mobile user in rank list $coM(n)$, who gets the delegation request, cannot decline it. This ensures that a delegation is certainly performed if $MA_U$ is set to true. Note that if for a delegation the result of Algorithm 1 corresponds to $coMU(n) = \emptyset$, the mobile context is again evaluated to decide whether a skip or backup will be performed.

Finally, although many aspects are covered by the delegation service to handle exceptions (including the assignment concept to prevent exceptions [5]), more research questions must be addressed. First, the exception handling service must be practically evaluated with respect to system performance and user acceptance. In this context, also psychological aspects must be addressed. Second, the concept must be technically evaluated towards a multitude of existing process management systems. Third, the protocol between the mobile process client and the process management system must be evaluated in more detail. Finally, with respect to practical demands, constraints of mobile activities (e.g., two mobile activities must be performed by the same mobile user) should be addressed. To conclude, more aspects have to be addressed for a feasible technical solution that can be easily used in everyday practice.

V. RELATED WORK

In general, the dynamic assignment of mobile activities has proven its usefulness [2]. Interestingly, only few approaches deal with exception prevention and exception handling in the context of human-centric mobile services. In general, there are only a few approaches dealing with the support of mobile activities [2], [4], [9] as we do. The use of a mobile context for executing mobile activities as well as for handling exceptions has been also less considered so far [2], [10]. Recent related works address exception handling in the context of mobile services [3]. However, they do not particularly focus on human-centric mobile services. Finally, commercial process management systems supporting the integration of mobile activities do not also provide an exception handling concept [11].

VI. SUMMARY AND OUTLOOK

We introduced an approach that uses a context model for preventing and handling exceptions in the context of human-centric mobile services. We showed that the latter can be realized as mobile activities integrated with process management technology. We further showed that a mobile context is a proper basis to cope with exceptions for mobile activities. The components developed in this context were elaborated in cases studies as well as a comprehensive literature study. First results indicate that the developed approach for preventing and handling exceptions is useful for the support of human-centric mobile services. Currently, we run a case study that evaluates the efficacy of the used parameters and the proposed exception handling techniques in more detail. Altogether, the presented approach constitutes a major step towards the widespread use of human-centric mobile services in a variety of application domains.

REFERENCES


