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 activities explicitly (cf. Fig. 1①). The meta-model, in turn, is based on an extensive literature review (e.g., [2], [4]) and denoted as *mobile process meta-model*. From a technical perspective, the execution of mobile activities mainly requires a sophisticated worklist management. Therefore, all entities in our *mobile process meta-model* concerned with worklist management are marked accordingly (cf. Fig. 1). 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. 1①,②) 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. 1①) 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. 1②). 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. 1④).

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

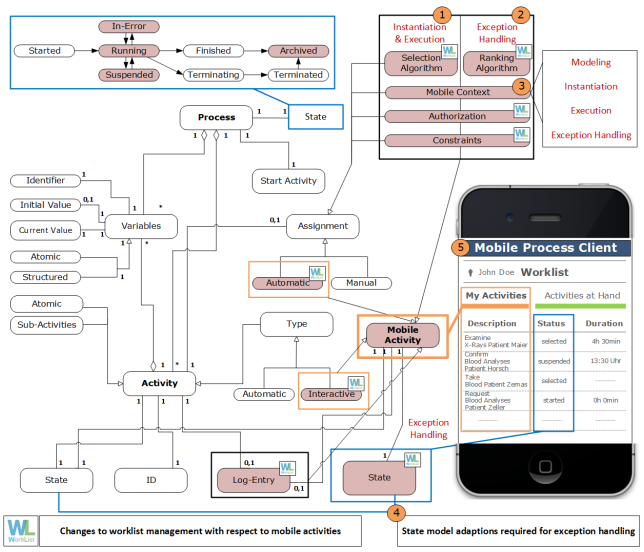

Figure 1: Mobile Process Meta-Model

CPM	Description	T	UA	EP	EH
Category I: Smart Mobile Device (SMD)					
SMD_{BS}	Battery Status	M	√	√	√
SMD_{FF}	Form Factor	S	√	√	√
SMD_{NT}	Network Type	M	√	√	√
SMD_{GC}	Geometric Coordinate	M	√	√	√
Category II: Mobile Activity (MA)					
MA_{SC}	Symbolic Coordinate(s)	S	√	√	√
MA_{GC}	Geometric Coordinate	M	√	√	√
MA_{LR}	Location Range	S	√	√	√
MA_{BS}	Battery Status	M	√	√	√
MA_{U}	Urgency Value	S	√	√	√
MA_{OFF}	Offline Mode	S	√	√	√
MA_{FF}	Form Factor	S	√	√	√
MA_{RF}	Response Frequency	S	√	√	√
MA_{UT}	User Threshold	S	√	√	√
Category III: Process (P)					
P_{IST}	Instant Shutdown Threshold	S	√	√	√
Category III IV: Mobile User (MU)					
MU_{SC}	Symbolic Coordinate(s)	S	√	√	√
MU_{IS}	Instant Shutdowns	S	√	√	√

T=Type ⇒ M=Measured, S=Symbolic, UA=User Assignment, EP=Exception Prevention EH=Exception Handling, √=holds, CPM=Context Parameter

Table I: Mobile Context Parameters

qualified for it. For this purpose, worklists are managed for all mobile users on their process clients, which are centrally synchronized by the process management system. More precisely, all mobile activities a user qualifies for are added to his *ActivitiesAtHand* list (cf. Fig. 1⑤). These activities can then be claimed by the user. In the latter case, the mobile activity is added to the *MyActivities* list (cf. Fig. 1⑤) and removed from the *ActivitiesAtHand* 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

	Selection Algorithm	Ranking Algorithm	AT	ET	ExT
Exception Prevention	Selection algorithm [5] determines those mobile users to perform a mobile activity that less likely cause an exception.	X	√	√	X
Exception Handling	X	Ranking algorithm evaluates resource status and exception behavior of all qualified mobile users and determines based on the evaluation those mobile users that are appropriate targets for handling an exception.	X	X	√

AT=Assignment Time, ET=Execution Time, ExT=Exception Time, √=holds, X=not holds

Table II: Prevention and Handling of Exceptions

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

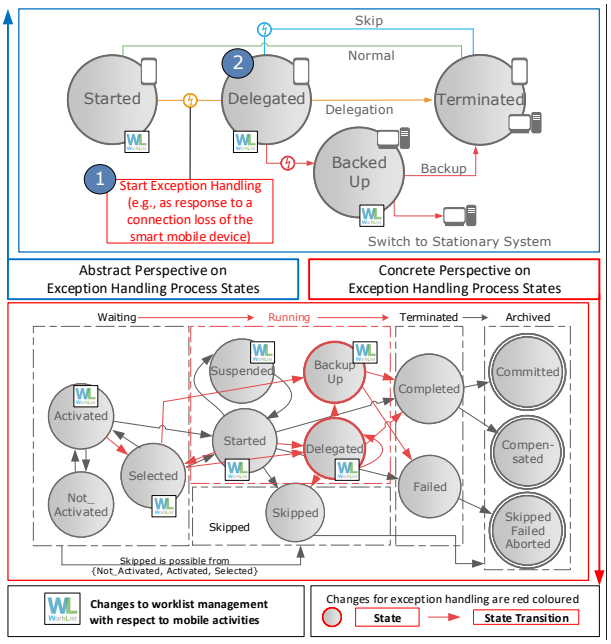

Figure 2: Exception Handling State Model

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

	Delegation Aspect (DA) Description	Requires
DA1	What kinds of exceptions trigger a delegation?	Evaluation of mobile context.
DA2	Which mobile users are appropriate delegation targets?	Evaluation of mobile context.
DA3	What changes must be applied to the assignment protocol?	Evaluation of assignment protocol.
DA4	Do we have to distinguish different practical delegation variants?	Evaluation of mobile context.

Table III: Delegation Aspects

Algorithm 1: Ranking Algorithm

Data: Relevant context parameters
$aMU(n)$: Set of all qualified mobile users of mobile activity n
For all users in aMU their current parameters MU_R, MU_{DB}, MU_{RB}
The location values LV1-LV3
Result: $coMU(n)$: Ranked list of mobile users being an appropriate delegation target

1 **begin**
2 $coMU(n) \longleftarrow \emptyset$; /* initialize rank list */
/* Determine MU_R for all mobile users in aMU */
3 **foreach** *mobile user* $mu \in aMU(n)$ **do**
/* Evaluate geometric coordinates */
4 **if** $(0 < nLR(mu) \leq 1)$ **then**
5 $MU_R(mu) \longleftarrow LV1 + MU_{DB}(mu) + MU_{RB}(mu) + MU_{IS}(mu)$;
6 $coMU(n) \longleftarrow coMU(n) \cup \{mu\}$;
7 **end**
/* Evaluate symbolic coordinates if needed */
8 **else if** $(MA_{SC}(n) = MU_{SC}(mu))$ **then**
9 $MU_R(mu) \longleftarrow LV2 + MU_{DB}(mu) + MU_{RB}(mu) + MU_{IS}(mu)$;
10 $coMU(n) \longleftarrow coMU(n) \cup \{mu\}$;
11 **end**
/* All other cases with no location match if needed */
12 **else**
13 $MU_R(mu) \longleftarrow LV3 + MU_{DB}(mu) + MU_{RB}(mu) + MU_{IS}(mu)$;
14 $coMU(n) \longleftarrow coMU(n) \cup \{mu\}$;
15 **end**
16 **end**
/* Call method to sort $coMU(n)$ based on MU_R in ascending order */
17 **end**

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

Relevant Context Parameters

MU_{CL} stores offline times during the execution of a mobile activity (CL: Connection Losses).
MU_{LBC} counts how often a mobile user claimed an activity though his device had an inappropriate battery status, i.e., $SMD_{BL} < MA_{BS}$ (LBC: Low Battery Counter).
MU_{RB} stores resource behavior of a mobile user based on MA_{CL}, MA_{LBC} (RB: Resource Behavior).
MU_{DB} 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).
MU_R stores the rank of a mobile user determined by the *Ranking Algorithm* (R: Rank).
$LV1-LV3$ store location values used by the *Ranking Algorithm* (LV: Location Value; $LV1 < LV2 < LV3$).

Applied Calculations

$$\text{nLR(MU)} = \left\{ \frac{MA_{GC} - SMD_{GC}(MU)}{MA_{LR}} \right. \quad (1) \qquad \text{MU}_{RB} = \left\{ MU_{CL} + MU_{LBC} \right. \quad (2)$$

$$\text{MU}_{DB} = \left\{ \frac{\#Activities\ from\ MU\ delegated}{\#Activities\ started\ by\ MU} \right. \quad (3)$$

Table IV: Parameters for Ranking Algorithm

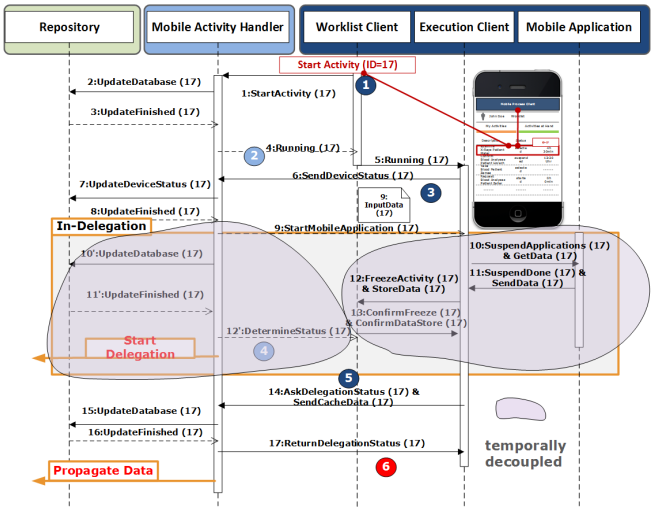

Figure 3: Delegation Protocol

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. 3⑦)). 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. 3⑥) after a reconnection. First, if a delegation has not been accomplished yet, the reconnecting smart mobile device gets

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 $coMU(n)$, 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 also not 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

[1] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer, 2012.

[2] G. Hackmann *et al.*, "Mobiwork: mobile workflow for manets," *Technical Report, Washington University in St. Louis, Report Number: WUCSE-2006-18*, 2006.

[3] N. Chen *et al.*, "Goal-Driven Service Composition in Mobile and Pervasive Computing," *IEEE Transactions on Services Computing*, no. 99, pp. 1–1, 2016.

[4] S. Zaplata *et al.*, "Flexible Execution of Distributed Business Processes Based on Process Instance Migration," *Journal of Systems Integration*, vol. 1, no. 3, pp. 3–16, 2010.

[5] R. Pryss *et al.*, "Context-Based Assignment and Execution of Human-Centric Mobile Services," in *IEEE 5th Int'l Conf on Mobile Services*. IEEE, 2016.

[6] C. Becker and F. Dürr, "On Location Models for Ubiquitous Computing," *Personal Ubiquitous Computing*, vol. 9, no. 1, pp. 20–31, 2005.

[7] M. Kloppmann *et al.*, "WS-BPEL Extension for People–BPEL4People," *IBM and SAP White Paper*, vol. 183, p. 184, 2005.

[8] N. Russell and W. van der Aalst, "Work Distribution and Resource Management in BPEL4People: Capabilities and Opportunities," in *Advanced Information Systems Engineering*. Springer, 2008, pp. 94–108.

[9] C. Kunze *et al.*, "Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments," *Journal of Computers*, vol. 2, no. 1, pp. 1–11, 2007.

[10] N. Cardozo and S. Clarke, "Language design for developing smart adaptive services," in *1st Int'l WS on Smart Cities Conference*. IEEE, 2015, pp. 1–2.

[11] "IBM Mobile Business Process Management," http://www.redbooks.ibm.com/abstracts/sg248240.html?Open, 2014, [Online; accessed 10-May-2017].