MobWEL - Mobile Context-Aware Content-Centric Workflow Execution Language

Anna Kocurova, Samia Oussena, Peter Komisarczuk
School of Computing and Technology
University of West London
London, UK
Anna.Kocurova, Samia.Oussena, Peter.Komisarczuk@uwl.ac.uk

Tony Clark
School of Engineering and Information Sciences
Middlesex University
London, UK
Email: T.N.Clark@mdx.ac.uk

***Abstract*—Collaboration among people has been empowered by using mobile devices in daily life. However, increasing user demands for a better mobile collaboration experience require constant evolution and adaptation of existing mobile collaborative technologies. In this paper, the collaborative workflow technology is adapted to enhance mobile peer-to-peer collaboration. With context awareness integrated, workflows are adapted to collaborators' needs and circumstances. Extra management support for content behaviour is incorporated in order to increase content awareness in workflows and enable communicating progress among collaborators. This paper introduces MobWEL, a context-aware content-centric workflow language designed for mobile peer-to-peer collaboration. MobWEL extends BPEL, using constructs from existing workflow approaches, Context4BPEL and BPEL^{light}, and adopting elements from the BALSA workflow model.**

***Index Terms*—mobile; peer-to-peer; context-aware; content; workflows.**

I. Introduction

Mobile phones have become ubiquitous, opening new possibilities for collaboration, communication and sharing ideas in real-time. Using smartphones allows people to remain productive regardless of their geographical locations and collaborate while on the move. However, with the spread of mobile technologies, collaborators expect to have tools that support collaboration, content sharing and management. The raising user expectations for mobile collaboration include the demand for an intelligent mobile collaborative environment which responds to people's needs.

Collaborative workflow is a widely accepted technology which offers rich support for multi-party collaboration, coordination of distributed teamwork and content sharing. With the arrival of mobile computing on the scene, mobile devices have been integrated into workflow management scenarios and the technology has been adapted for various sets of requirements. Smanchat et al. [1] states that when workflows are utilised in ubiquitous environments, adaptability and context-awareness are the features that should be included in the workflow mechanism. Earlier workflow management approaches based on a server-client model required centralised workflow management systems deployed on servers. However, this centralised approach is impractical in various situations in which people prefer to collaborate solely by using mobile devices. A typical example of such a situation can be a meeting between business partners who share private information. Moreover, in the centralised architectural style, control and management of workflows is driven by a central management unit with mobile devices behaving only as thin clients. The central management unit makes all important decisions and drives the whole workflow execution without considering the current situation of users, and the context the devices reside in. In this work, Dey's definition of context is used: 'context is any information that can be used to characterize the situation of an entity' [2].

The need for a mobile device-centric workflow process and systems that manage the workflow processes in a completely distributed manner has been recognised in the work of Pajunen and Chande [3]. Mobile collaboration in the distributed manner, also labelled as mobile peer-to-peer collaboration, enables data and content distribution by direct exchange. Additionally, all decisions can be made by mobile devices and based on local, context information, thereby adapted to current collaborators' circumstances and needs. The work presented in this paper tackles the problem associated with representing context information available on a single mobile device, and using context information to shape peer-to-peer workflow execution. Context describes the current situation of a user, a device or an environment regarding a specific purpose. By expressing the context model explicitly for each workflow scenario, each mobile device can monitor, acquire and process only required context information. By adapting workflow logic to consume context information, the workflow execution can become more dynamic and responsive to an individual collaborator's situation.

Additionally, workflow processing often involves content manipulation such as reviewing and approving proposals or pictures. Mobile content such as picture, document or video/audio is usually user-generated or adapted for use on mobile devices. Mobile content is described by metadata and can be semantically enriched by context metadata such as location where it was created. Creating, processing and disposition of content are the basic stages of its lifecycle. However, content can flow through a more complex management process involving stages such as content editing, reviewing or approving. Workflow process activities modify content or its metadata. By way of illustration, if a proposal has been approved, its metadata specifying the state of approval has been changed, the proposal is considered as being in the 'Approved' state and often shortly referred as 'approved'. So at any given instant of time, content is in a specific state that is defined by values of its metadata. There are several benefits to increasing content awareness in mobile workflows and integrating advanced content management capabilities in the workflow management system. Firstly, content behaviour is visible and its lifecycle is expressed as explicitly as the process logic is. Secondly, especially in peer-to-peer collaboration, content state-related information can be used significantly in communicating progress among collaborators. Dissemination of such information enables easier content synchronisation over a number of devices. Furthermore, there can be relationships and dependencies between various content items. For example, the proposal can have more sections edited by co-workers on their devices. Receiving regular information about each section state would enable faster processing of the proposal.

This paper develops the ideas presented in our previous work [4] and presents our solution for increasing context and content awareness in the collaborative workflow technology. In particular, a mobile context-aware content-centric workflow execution language (MobWEL) is introduced and an architecture of a workflow management system that carries out MobWEL workflows is described. The remainder of the paper is structured as follows. Section 2 presents the usage scenario. An overview of related work is given in Section 3. Domain analysis and the constructs of the MobWEL language are described in Section 4. The MobWEL workflow language is presented in Section 5. Section 6 outlines the logical architecture of the MobWEL workflow management system. This paper is concluded in Section 7.

II. Mobile Peer-to-Peer Collaboration Scenario

The concept introduced in this paper is illustrated by using the following usage scenario. A team of ten designers work on interior design of buildings. Designers often work out in the field using smart phones to communicate and directly share pictures. Although each design project is assigned to a particular designer, design decisions are never done by a single person. The following work pattern is used to complete projects:

1) Designer Jane takes a picture of a new room design by using her smart phone.
2) A simple rating system is used to quickly assess design ideas. Jane adds her own rating to the picture.
3) The picture is sent to her fellow co-workers. They work out in the field so each of them can review the picture by using own mobile phone.
4) Reviewer's subjective opinion can be captured by adding a comment.
5) Reviews and comments are sent back to Jane. She finally reassesses her idea according to opinions of other designers.
6) If the idea is good, the picture is sent for final approval to customer.
7) Approved picture is added to Jane's completed work.

The real work pattern can be abstracted into a collaborative workflow. However, the workflow is not adapted to individual collaborator's needs. Context and content awareness could enrich the workflow, as shown in Fig. 1.

The simplified workflow process is illustrated from a user's point of view in the middle of the figure. It shows three roles that participate in this collaboration: Interior Designer, Reviewer and Customer. The lifecycle of picture is outlined on the left side. The picture can go through a number of states such as *Initial, Reviewed, Assessed, Approved or Final*. Tasks

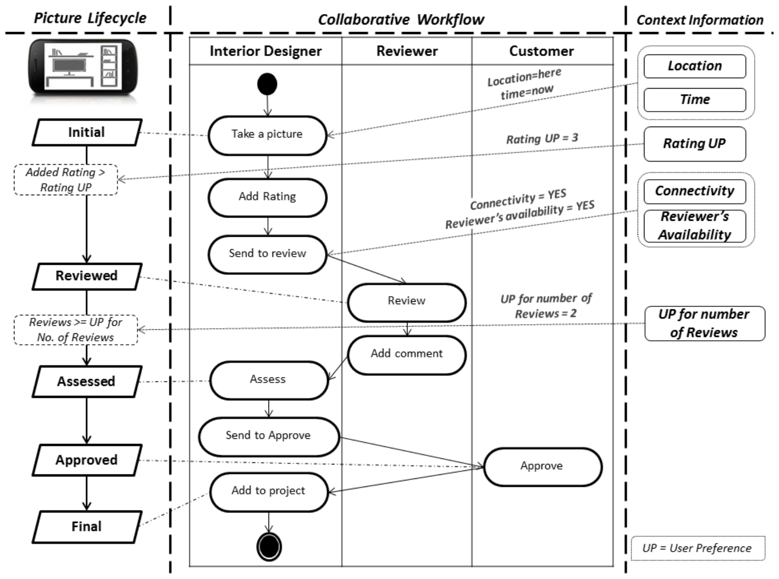

Fig. 1. Collaborative Workflow

or activities that can influence a change of the picture state are marked. For instance, if the *Review* task is performed, the picture can move to the *Reviewed* state or if the *Approve* task is performed, the picture may go to the *Approved* state. Relevant context data are shown on the right side of the figure. Context information can be used in several ways. Firstly, it can enrich picture such as *location* can specify the place where the picture is taken. Secondly, Jane can specify that only pictures with rating larger than her *Rating User Preference (RatingUP)* can move from the *Initial* state to the *Reviewed* state. So if the *RatingUP* is set to 3 and the rating added to the picture is 4, then the picture can be sent to be reviewed. Finally, knowing work context and current availability of fellow co-workers would enhance the mobile collaboration. For instance, the picture can be sent only to those co-workers who are currently available to review it (*Reviewer's Availability = YES*).

III. Related Work

A. Workflow Contextualisation

Workflow contextualisation has been addressed in a number of works [5][6][7], however, there are many research challenges to make context-aware workflow systems ready for practical use. Three main research challenges related to a context-aware process design have been recognised by Rosemann et al. [8]: context description, design for context and process adaptation. *Context description* refers to the identification and description of context variables relevant for a business process, respectively to a relevant context model. This includes the integration of the context model with an existing metamodel of a process modelling language. *Design for context* refers to the incorporation of contextual elements in the design of business process. In particular, how the knowledge can be embedded and utilised in the process design for context. Finally, *Process adaptation* considers the support for context-aware business processes.

Although the need for an explicit context description and identification of context variables that influence process design and execution has been highlighted, using context in applications is difficult because of the nature of context information. Numerous general context frameworks have been already developed to facilitate context modelling, recognition, reasoning and management. One of the earlier widely acknowledged works, the Context Toolkit introduced by Salber et al. [9], provides a framework for context modelling and management based on context widgets. Focus on mobile platforms aiming to simplify the development of context-aware mobile applications has been addressed in another context management framework proposed by Korpipaa et al. [10]. The framework employs the context manager to provide control between the acquisition and use of context information by applications. One of the context management approaches introduces a preference context model for representing context-dependent application requirements [11]. Based on the existing frameworks, a summary of the requirements, which should be met in the development of context modelling and reasoning techniques, has been elaborated by Bettini et al. [12]. An original and holistic view of the existing approaches for context data distribution for mobile ubiquitous systems has been undertaken by Bellavista et al. [13].

There have been various context management frameworks developed. However, the frameworks are either too general and designed to be used with a wide range of context-aware applications, or the frameworks are designed for a specific class of scenarios and a particular set of requirements. However, we believe that the context modelling and management approach need to be adapted for the problem it is applied to. Thereby, to provide different types of context information described in the usage scenario, a layered, self-contained approach to context acquisition has been designed and described in our previous work of Kramer [14]. In this paper, the use of context information is also considered, thereby, the approach is extended and the extended approach to a workflow-specific context representation and management is described.

Following the same vision that workflow meta-models should support context modelling and its use in workflows, a context-aware workflow process model has been developed by Wieland et al. [15]. The implemented process model based on Business Process Execution Language (BPEL) has been named Context4BPEL. Context4BPEL enables modelling of context-aware workflows directly without hiding the context usage in the invoked web services. The approach is based on the context-related concepts such as context event, context query and context decision. Context4BPEL workflow management system has been coupled with the Nexus context-provisioning platform, therefore, the use of the concepts in Context4BPEL follows the Nexus access pattern. The concepts have been adopted in our workflow language. However, the Nexus provisioning platform is not suitable for our scenario as it has been developed as a framework that supports a global context model, as described by Grossman et al. [16]. In our work, only local context situations of mobile devices need to be monitored, thus the context provider should reside on each device. It means that the Context4BPEL context-related concepts have been adapted in order to be used in alignment with our context provisioning platform and proposed workflow-specific context management approach.

B. Object Awareness in Workflows

So far, however, too little attention has been paid to content processed in process-centric workflows. Many workflow approaches are organised around process-centric and activity-centric workflow models, which are relatively flat with limited focus on artifacts processed in workflows. Some data-centric and artifact-centric workflow models have been developed, however, Hull et al. [17] states that the field of artifact-centric workflows is still in infancy. The need for more data-driven and object-aware workflow processes has been recognised in the research community and various approaches, presented in this section, have been developed to outline or address the associated challenges.

The belief that traditional process modelling approaches which focus on activities fail to capture informational structure relevant to the business contexture has been supported by Liu et al. [18]. Business artifacts, such as Purchase Order or Insurance Claim, are seen as an additional dimension with which business analyst can model their business. A business artifact has been described as *'identifiable, self-describing unit-of-information through which business stakeholders add value to the business'*. Therefore, the artifact has an id to be uniquely identified within the business and its attributes are so named that their use within the domain is apparent. This approach highlights the need for discovering and modelling of the artifact behaviour.

The issue that entity, such as content or business artifact, is integrated in workflows as an input or output of an activity and the effects of how performed activities influence entity's behaviour are not visible, motivated the development of the Business Artifacts with Lifecycles, Services and Associations (BALSA) workflow model [19]. The artifact-centric workflow model comprises four key elements or dimensions: business artifact information model, business artifact (macro-level) life-cycle, services, and associations. By varying the paradigms used to specify the information model, lifecycle, services and associations, numerous BALSA models can be obtained. For example, the information model might be specified as attributes with scalar values or XML; the lifecycle might be specified by using flowcharts or finite state machines; and the services might be specified in black box or BPEL activities. The choice among the various paradigms depends on the intended area of application. Therefore, the BALSA workflow model is flexible in its use and provides a guided framework how to develop a customised artifact-centric workflow approach. This motivated us to use some aspects of the BALSA model in building the MobWEL workflow approach for mobile platform.

Another framework for integrated process and object life cycle modelling has been developed by Wahler [20]. Business objects processed by business processes can be associated with distinct states abstracted from the details of the performed tasks. The states mark the milestones of the overall processing, and are useful in communicating progress to stakeholders who are unaware of the exact process logic. Unlike other approaches which base entity lifecycles on variants of finite state machines, Hull et al. [21] introduces the guard-stage-milestone lifecycle model. The model is an evolutionary work on previous approaches based on business entities with lifecycles, but is more declarative than finite machines and supports hierarchy within a single entity instance.

Object-awareness in process-centric workflows is still very limited, and a holistic approach to integrate data, processes and users is undertaken in the development of the PHILharmonicFlows framework by Künzle et al. [22].

Most of the presented studies support the use of a process model and an object model as two complementary assets in object-aware workflows. However, research to date has tended to focus on the lifecycle of the artifact without considering its adaptation to context changes and its evolution in peer-to-peer workflow execution. This motivates the need for MobWEL workflows to support context-awareness in both models and be designed for peer-to-peer collaboration.

IV. Domain Analysis

The objective of this work is to enhance mobile peer-to-peer collaboration by developing context-aware content-centric workflows. Such workflows can be described by a workflow language that possesses the following main characteristics: a) is executable in a mobile peer-to-peer environment; b) is context-aware; and c) is content-centric. Building the workflow language from scratch would have been inefficient and impractical. Moreover, there have been many suitable existing workflow constructs with well-defined semantics that could have been reused. This section describes the MobWEL constructs that have been produced by domain analysis.

A. Peer-to-peer collaboration

Peer-to-peer collaboration imposes requirements on functional aspects of workflows. Workflow partitions are executed on mobile devices, therefore, the workflow language must provide the support to describe control flows in the partitions. The second requirement is that messages and information need to be exchanged directly between peers. To satisfy this requirement, asynchronous conversation needs to be supported and coordinated.

BPEL has been used as a base for MobWEL because of its characteristics. BPEL is platform-agnostic; expressed entirely in XML; extensible and adaptable; a widely accepted and adopted workflow standard language with well-defined semantics; and provides robust interaction model that enables peer-to-peer conversation. Although BPEL has been designed for processes in a Web Service world, it can be adapted for mobile peer-to-peer workflow execution. Strong coupling of process logic with the web services technology in BPEL is not very convenient and flexible approach to be used for mobile peer-to-peer interactions. Workflow participants and their roles are known beforehand so a much lighter interaction model can be used.

BPEL^{light} is a WSDL-less BPEL extension which decouples process logic from interface definition [23]. The BPEL^{light} approach allows modelling of the workflow process independently of Web service technology by introducing a single type of interaction activity. The interaction activities ($<receive>, <reply>, <invoke>, <pick>$) defined in BPEL 2.0 are resumed by single $<interactionActivity>$. In addition, BPEL^{light} processes can be modelled without specifying interface definitions (port types), hence they can be used in non-WS environment. By discarding the static specification of port types, this approach enables direct message exchange between workflow partners. The BPEL^{light} approach offers better flexibility than BPEL and is more suitable for the description of peer-to-peer mobile workflows in which all participating and interacting parties are known beforehand. Hence some concepts defined in BPEL^{light} have been adopted in MobWEL. The $<interactionActivity>$ has been further adapted for the use of collaboration-related context information. An optional *collaborationContext* attribute has been added to determine whether the availability of collaborator(s) should be consider when a message is sent.

B. Context Management

To make workflows context-aware, workflows need to have the ability to react to context changes and adapt their behaviour to the changed environment. To achieve this, workflows need to be adapted for the use of context information in two ways:

- context information can be queried when needed,
- context changes and events need to be captured and handled at any time of the workflow execution.

The Context4BPEL context-related concepts have been adopted in the MobWEL workflow approach to support both ways of the use of context information.

In addition, to ensure that only relevant and meaningful context information is consumed, the abstraction level of the delivered context data should be raised. There are numerous reasons why the context raw data should be processed before its consumption in workflows. Firstly, the raw context data is heterogeneous and inconsistent. Secondly, if a context provisioning platform distributes the raw context data, workflow management system would need to deal with irrelevant context information or make context-related decisions very often. To prevent these situations, high-level context information is derived. For example, knowing when battery level is low might influence decisions whether certain operations should be performed or not. If a consuming workflow requires being informed only about the change when the battery level drops down to 5 %, constant notifications about every battery level change would be completely inefficient. By specifying a high-level context information such as *LOW* for battery level in a range between 0% and 5% and *HIGH* for a range between 5% and 100% would ensure that the context-aware workflow is notified only when the context value is changed from *HIGH* to *LOW*.

Further, relationships and dependencies between various context information can be indicated through the accommodation of context aggregation. An example of context aggregation is illustrated in Fig. 2.

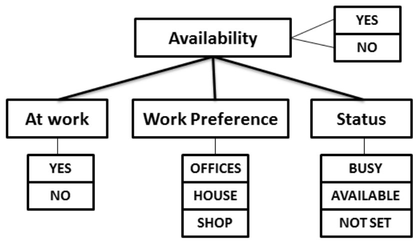

Fig. 2. Example of context aggregation

Work context of interior designers is expressed as a context aggregation of three user preferences: *At work, Work Preference* and *Status*. Context values for the *At work* context are *YES* and *NO* representing the designer's work status. Each designer can specify own work preference, for illustration purposes only *OFFICES*, *HOUSES* and *SHOPS* are shown. The *Status* is used to show whether collaborator is currently busy. *Availability* of the designer has an informative character to indicate whether the task can be taken by the person. The context value of *Availability* is determined from children context values by using associated rules. The aggregated context value can be: *(Availability-YES)* or *(Availability-NO)*.

C. Content Behaviour

Adding the content management support means that workflows should be adapted to use content state-related information similarly as they use context information:

- content state can be queried,
- content state-related changes and events are captured and handled at any time of the workflow execution.

To achieve this, some ideas developed in the BALSA workflow model seem to be practical and applicable for modelling of content-centric mobile workflows, despite the fact that context awareness has not been considered in this model. Two elements of the model, namely the Business Artifact Information Model and the Business Artifact Lifecycle have been adopted in our workflow approach. Firstly, the substance of the Business Artifact Information Model has inspired us to consider the content-related data. To gain control over processed content and its behaviour, a set of certain content-related metadata that is accessible and manageable by the workflow management system needs to be identified beforehand. This can be briefly illustrated by an example from usage scenario. If there is a rating system created for pictures, an attribute called *'ratingScore'* should be associated with each picture to hold an information about added rating. By adding the metadata to a picture, the information can be accessed at any time and can be used to make certain management decisions.

Secondly, the concept of the BALSA Business Artifact Lifecycle element is adopted in the construction process of our context-aware content lifecycle. Similarly, the context-aware content lifecycle is represented by a variant of finite state machines. Generally, a state machine contains a number of states, each state corresponding to a stage in the content lifecycle. Therefore, a *content state* is an essential construct in the content lifecycle. Content might move from one stage to another. The connections between two states are called *transitions*. When content is created, it is in an initial state with no incoming transition. At the end of its evolution, there is a final state that indicates the end of its lifecycle and has no outgoing transition. Between the initial and the final state, there are states with incoming and outgoing transitions. *Conditions* may be attached to these transitions. A condition can depend on a certain value of content attribute or external event.

A context change is an external event that can trigger a transition between two content states. Because of this, conditions placed on transitions need to be modified to deal with context. With context considered, two possible situations can happen.

- In the first situation, the further evolution of content depends on the current context of the execution environment. For example, if a rating has been added to a picture, the picture moves to the *Rated* state.
  After that, based on available context information, the picture can go to either the *Ready to Review* or *Archived* state, see Fig. 3a. So if the added picture *ratingScore* is greater than or equal to the value of the user preference, the picture goes to the *Ready to Review* state. Otherwise, it is moved to the *Archived* state. In this situation, context information has to be acquired at real time. When context information is obtained, the conditions are evaluated. Context information presented in this example is simplified, considering only *RatingUP* as a representant of the current context.
- In the second situation, the transition between two content states occurs only when specific context emerges. The situation is illustrated in Fig. 3b. Rating added to the picture is stored in the *ratingScore* attribute. For example, it can have a value of '2'. Current *RatingUP* is set to '3', therefore the picture cannot move to the *Ready to Review* state at this time. Instead, it waits whether any context change in the *RatingUP* happens. The picture remains in the *Rated* state until the *RatingUP* is changed and becomes less than or equal to '2'. In contrast to the first situation, this situation requires an awaiting, monitoring and filtering mechanism for context events.

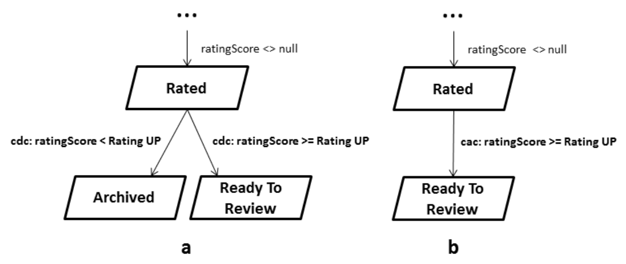

Fig. 3. Context conditions

Therefore, to distinguish between these situations, two types of context conditions are created: *context-driven condition (CDC)* and *context-aware condition (CAC)*.

D. Support for context and content awareness

Based on the requirements outlined for context and content awareness, the control flow has to be adapted for information querying and events handling. Again, BPEL has the potential to be extended to fulfil the requests. Firstly, BPEL provides event handlers to deal with occurring events. Event Handlers can be used if a context change or a content state-related change occurs. Secondly, using BPEL constructs for conditional branching, such as *switch* or *while*, enables to query information at decisions points. To query context information or content state-related information, two extension functions need to be supported:

(Object contextValue) mobwel:getContextValue(String contextName)

(String contentState) mobwel:getContentState(String contentVariable)

The use of the functions is illustrated in Fig. 4.

```
<switch>
  <case condition="mobwel:getContextValue(RatingUP)=3">
    ...
  </case>
  <case condition="mobwel:getContentState(picture)='Reviewed'">
    ...
  </case>
  <otherwise>
    ...
  </otherwise>
</switch>
```

Fig. 4. Example of the use of the extension functions

Constructs presented in this section have been accommodated in the design of the MobWEL workflow language. The language is defined next.

V. MobWEL Language Definition

This section introduces the MobWEL workflow language. Firstly, the anatomy of a MobWEL workflow is outlined (Fig. 5).

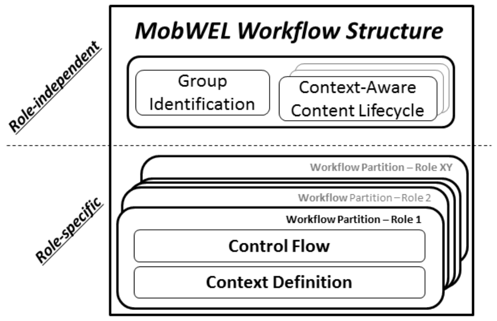
MobWEL Workflow Structure
Role-independent
Group Identification
Context-Aware Content Lifecycle
Role-specific
Workflow Partition – Role XY
Workflow Partition – Role 2
Workflow Partition – Role 1
Control Flow
Context Definition

Fig. 5. Anatomy of MobWEL workflow

MobWEL workflows are designated for mobile peer-to-peer collaboration, thereby a number of roles and workflow participants can be involved in the workflow execution. Workflow participants have assigned roles, each role performs only activities defined in an allocated workflow partition.

MobWEL workflows are composed of various constituents, which are either *role-specific*, explicitly defined for each participating role, or *role-independent*, defined same for all participating roles. The workflow constituents are described next.

A. Group Identification

In collaborative workflows, tasks are performed by collaborators. In this concept, workflow collaborator is a person who uses a mobile device to collaborate, share content and communicate with other team members in order to achieve a common goal. To execute workflows in peer-to-peer collaboration, each mobile device needs to be given the relevant details about fellow collaborators. The *Group Identification* workflow constituent specifies all *Collaborators* participating in a workflow and their associated *Roles* (Fig. 6).

Each *Collaborator* is described by personal name and the attributes which identify collaborator's mobile device such as

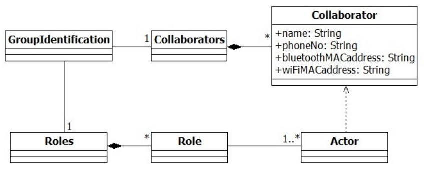

Fig. 6. Metamodel of Group Identification

phone number, Wi-Fi and Bluetooth MAC addresses (assuming that a collaborator uses only one device to collaborate). A collaborator can play more than one role in a workflow (not in same workflow instance). A good example is given in our usage scenario where Jane can play the role of designer but also the role of reviewer if asked to review a picture created by her co-worker. Therefore, the *Roles* element describes roles involved in workflow. Each *Role* can be played by a number of actors, each *Actor* refers to a *Collaborator* and is identified only by the *phoneNo* attribute of the collaborator. This enables collaborators to be associated with more roles.

B. Workflow-Specific Context Definition

Context Definition Metamodel, see Fig. 7, expresses all the constructs, concepts and relationships between the constructs needed to build workflow-specific context definition models.

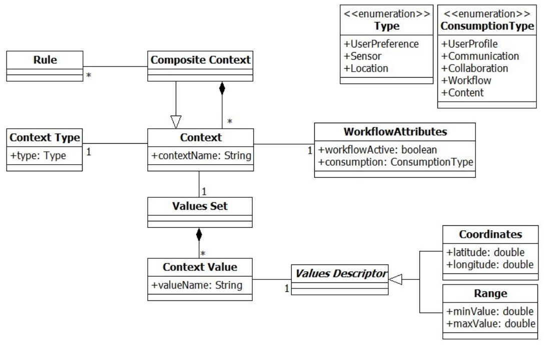

Fig. 7. Metamodel of Context Definition

The workflow specific context definition modelling approach allows building context models that are specific for each workflow partition. It enables to express context aggregations, sets of values and other attributes related to a workflow partition.

Context is described by its contextName, *Context Type* and *Values Set*. For example, context can have the name: *Status*, the context type of *User Preference*, and the context values set: $\{$*Busy, Available, Not Set*$\}$. A high-level *Context Value* can be derived from raw context data. We use the *Values Descriptor* construct to associate the high-level context values with raw data. Two examples: *Range* and *Coordinates* are outlined. For instance, *LOW* as a high-level context value for battery can be defined for a range with 'minValue set to 0' and 'maxValue set to 5' (%). Coordinates are used to specify a location or a place of interest.

Composite context, dedicated for context aggregation, is designed as a context container. It is a subtype of *Context* and inherits its attributes and behaviour. A composite context can be built as an aggregation of other composite and atomic contexts. Based on the context values of child contexts, *Rule* is used to determine a context value of the composite context. Therefore, the composite context values set can be fully defined by workflow designers. For example, the context value of *Availability* can be defined as *YES* if the current values of *At work* is *YES*, *Work Preference* is *HOUSE* and *Status* is *AVAILABLE*.

Workflow Attributes are the attributes required for better functioning of the workflow management system. Firstly, the workflow management system does not need to obtain context information of all defined contexts. For example, in order to execute a workflow step, it might need to obtain only context information: *Availability-YES* without knowing that the context value of *At work* is set to *YES* or the value of *Status* is set to *AVAILABLE*. Therefore, each context is associated with the *workflowActive* attribute which can be set to *true* if a particular workflow is interested in listening to changes of this context, respectively to *false* if the context is only auxiliary and its values do not directly influence the workflow execution. Secondly, the *consumption* attribute is used to identify the consumption of context information within the workflow management system.

C. Context-Aware Content Lifecycle Definition

A number of content items can be processed across multiple mobile devices in a MobWEL workflow case, each content item following its own content lifecycle. The evolution of content items is distributed across all collaborators, therefore a set of content lifecycles is defined once and remain same for all workflow partitions. Based on the design specifications presented in the previous section, the metamodel for a context-aware content lifecycle has been constructed. All constructs related to a content lifecycle and their relationships are shown on the metamodel in Fig. 8.

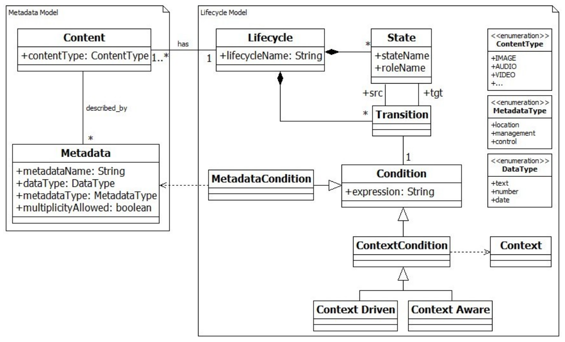

Fig. 8. Metamodel of Content Lifecycle

There are two parts presented, one representing content information model and another representing content lifecycle. *Lifecycle* is described by its lifecycleName. The lifecycle can describe the behaviour of several content items, however, in a particular definition it can be associated only with one *Content* item. The *Content* item is characterised by its contentType such as image, audio or video and a set of workflow-specific *Metadata*, each metadata of a specific *metadataType*. Furthermore, *Lifecycle* is composed of a number of *State*s and *Transition*s. *State* is characterised by stateName and roleName. The latter attribute determines the role associated with the ownership of the state. Each *Transition* has a source and target *State* and can be associated with a *Condition*. To trigger the transition of the content item between the source and target state, the associated condition must evaluate to true. There are two subtypes of conditions, *MetadataCondition* and *ContextCondition*. Fulfillment of *MetadataCondition* depends on the value of particular metadata. *ContextCondition* depends on certain context and can be *ContextAware* or *ContextDriven*.

D. Process Control Flow

The main part that integrates other constituents is the control flow of the MobWEL workflow. MobWEL extends BPEL, therefore, the major part of its syntax and semantics have been inherited from BPEL. The focus is put on constructs that have been introduced or existing constructs that have been conceptually modified. These constructs depicted in Fig. 9 are as follows:

- *MobWEL Workflow Process:* A workflow process identified by its unique *targetNamespace*. The *groupIdentification, contentLifecycleDescription and contextDefinition* attributes specify namespaces of definitions of group identification, content lifecycle and context.
- *Variable:* A variable is used in a standard BPEL process. The variables used for content items have the default value for the variable type set to 'content'.
- *Lifecycle:* This element refers to the description of an explicitly defined content lifecycle. Each lifecycle is specified by its *name* and *resource*, a path to the explicit definition. A *variable* of a content type must be also associated with the lifecycle. More lifecycles can be declared within one workflow process.
- *ContextDefinition:* This element refers to the explicit context model. The *contextDefinitionSource* attribute specifies the path to the context definition model. A workflow process can have zero or one context definition declared.
- *CollaboratorsGroup:* A group of collaborators is explicitly defined and imported in the workflow definition through this element. It is specified by its *groupName* and *groupSource*, a path to the group description. A workflow process has one group of collaborators declared.
- *Partner and Conversation:* A declaration of partners and conversations involved in collaboration. The constructs are adopted from the BPEL^{light} workflow definition. A workflow process can have one or more partners.
- *MobWEL Activity:* A base type for MobWEL activity.
- *InteractionActivity:* An interaction activity adopted from BPEL^{light} used for all interaction activities between partners and adapted for collaboration-related context.
- *ContentActivity:* An activity with an informative character. The activity is used to access content-related data stored in process variables and communicate them out to external parties.

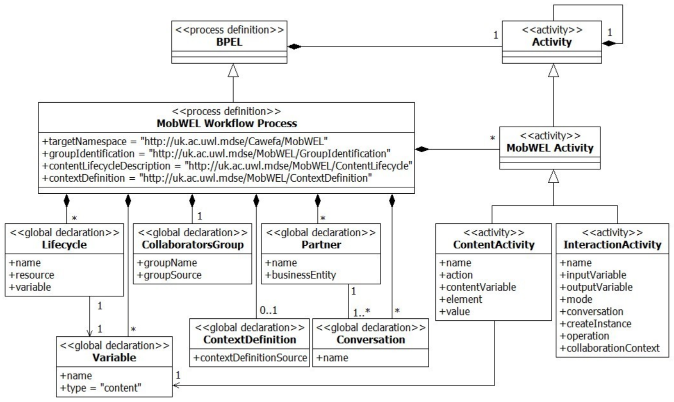

Fig. 9. MobWEL constructs

Most of the introduced constructs are global declarations in the process. Only *interactionActivity* and *contentActivity* are specific activities used in the description of the control flow.

VI. MobWEL Workflow Management System

MobWEL workflows are carried out on mobile devices. Management and execution of MobWEL workflows in a distributed manner across a number of devices require an adapted workflow management system that is deployed on each device. In this section, the architecture of such mobile workflow management system is described.

The high-level architecture of the MobWEL workflow management system is shown in Fig. 10.

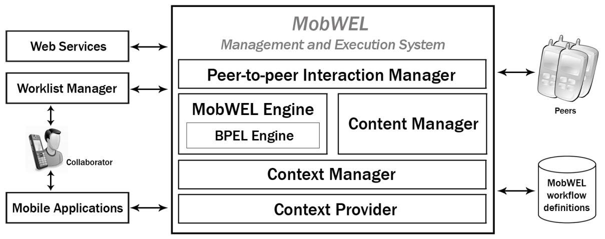

Fig. 10. High-Level Logical Architecture

The system is composed of the following components: Context Provider, Context Manager, Content Manager, MobWEL Engine, and Peer-to-peer Interaction Manager.

A. Context Provider

Context Provider is a component responsible for context monitoring, acquiring, processing, aggregating and disseminating. Context Provider interprets context models and creates corresponding internal data structures. Each particular context concern is managed through a self-contained context component. When a context change occurs, raw context data is processed, and context information in the *AppKey-ContextName-ContextValue-ContextDate* format is broadcasted to interested parties.

B. Context Manager

Context Manager acts as intermediator between Context Provider and other internal workflow management components. It manages the use and consumption of context information and drives context routing to other components of the MobWEL workflow management system at run-time.

It contains mechanisms to synchronously and asynchronously communicate with Context Provider. To communicate effectively, both components need to be aware about the same contexts. When the definition for a new workflow is deployed to Context Manager, Context Manager also passes it to Context Provider. Because of this, the same workflow-specific context definition is deployed to both components, but they extract different information from it. While Context Provider is more interested in extracting information related to context hierarchies and context reasoning, Context Manager focuses only on active contexts. Context Manager extracts *contextName, contextValuesSet, workflowID* and *workflowUse* for all contexts labelled as workflow-active. This ensures that Context Provider broadcasts context messages that are related to the workflows deployed in the workflow management system, and Context Manager can process them further. Context Manager has implemented a mechanism that monitors and filters messages broadcasted by Context Provider. As opposed to this asynchronous way of communication, Context Manager contains also a mechanism that enables synchronous binding to Context Provider and supports querying of context information at real time.

Context information is consumed within the workflow management system in several ways. Based on the workflow-specific context *consumption* value extracted from the context definition, Context Provider knows which component is interested in obtaining the context information. For example, if it is content-related, Content Manager is informed, if it is workflow-related, MobWEL engine gets the context notification. Context Manager also persists last context values.

C. Content Manager

Content Manager provides advanced content management functionalities and manages content lifecycles. This component parses the *Context-aware content lifecycle* elements of the MobWEL workflow definition. Each lifecycle element contains description of content-related metadata and description of content lifecycle. Content-related data is persisted and managed by using *Content Provider*. When workflow case is instantiated, multiple content items are created that follow the same lifecycle. *Content State Transition System* is used to manage and control content lifecycles.

D. MobWEL Engine

MobWEL Engine is the execution engine and the heart of the MobWEL workflow management system that parses and instantiates workflows, and provides an overall control over the management and execution of MobWEL workflow instances. The MobWEL engine is based on a BPEL engine. Sliver is an open source lightweight workflow engine that supports the execution of SOAP services and BPEL processes on a wide range of devices, and communication protocols [24]. This BPEL engine has been used as a base for the MobWEL engine.

The MobWEL engine contains a main parser and an execution unit. The parser interprets the whole MobWEL workflow definition and invokes corresponding parsers in other components for parsing of all workflow constituents.

The execution unit extends the BPEL execution unit, therefore, only the support for the execution of constructs introduced in the MobWEL language, namely *interactionActivity* and *contentActivity*, has to be added. The semantics for *interactionActivity* is inherited from BPEL^{light}. The *contentActivity* has only an informative character with a simple role to inform Content Manager and its execution does not change the data flow, thereby it is handled in the same manner as other BPEL activities and its internal execution depends on the action specified in the activity.

E. Peer-To-Peer Interaction Manager

Peer-to-peer Interaction Manager manages communication and messages transmissions with other mobile devices. *Event Handlers* handle incoming messages from other devices and requests coming from mobile applications or services. An event handler can be designed for each form of communication such as incoming binary data message, multimedia message, or messages coming via Bluetooth connection. Structured information and messages are sent between devices as a sequence of bytes, *Message Parser* is used to convert workflow-related data and objects into such message format that can be transmitted across the network. The parser also extracts workflow objects from incoming messages. *Identity Manager* is used to store and manage the details about participating collaborators. The details include collaborators' names, their roles, and other data needed for device-to-device communication.

F. Validation

To prove that the theory is functional, concrete research artefacts have been constructed. Firstly, the metamodel of MobWEL workflow language has been mapped into an XML schema. The MobWEL XML Schema is a tool that can be used by workflow designers who would be able to model and define the concrete workflows in an XML format. Secondly, formal MobWEL semantics has been defined. Finally, software prototypes have been implemented for the Android platform. Context provider operates on a mobile device and can be built as an internal component of the workflow management system, which provides services solely to the MobWEL workflow management system. However, there might be numerous context-aware applications running on the same mobile device which need to use the same context provisioning services. Thereby, Context Provider has been implemented as an external component, and named *ContextEngine*. The remaining components of the MobWEL workflow management system have been implemented as a system named *CAWEFA* (Context-Aware Workflow Engine For Android). A MobWEL workflow definition has been constructed for the usage scenario. The workflow definition has been deployed to CAWEFA and ContextEngine, and instantiated several times. The expected behaviour of each workflow instance expressed by using the MobWEL semantics has been compared with the actual behaviour of running workflow instances. The conducted experimentation and obtained results have shown that the MobWEL workflow language can be used to describe executable mobile context-aware content-centric workflows.

VII. Conclusion and Future Work

In this paper, we have presented MobWEL, a context-aware content-centric workflow language designed for mobile peer-to-peer collaboration, and the architecture of a MobWEL workflow management system. One of our main goals has been to build a language which is context-aware, visualises and integrates content behaviour, and supports peer-to-peer interaction. We have extended BPEL, considered the dimensions of the data-centric approach proposed in the BALSA model, and adopted some existing constructs from Context4BPEL and BPEL^{light}. In this paper, only a first milestone in the definition, management and execution of MobWEL workflows has been presented.

Integration of context and content awareness into the workflow technology offers numerous benefits. Firstly, by adding context definitions to the workflow description, the execution of workflow partition can be adapted to individual collaborator's needs and situation in the current run-time environment. Secondly, additional content management functionalities enable to monitor content behaviour and disseminate progress to other peers. However, there have been also some limitations discovered that should be addressed in future work. For example, the development of the MobWEL language has been tailor-made for a specific class of workflows. It would be interesting to use and validate the MobWEL workflow approach in other scenarios. Further, although integration of context awareness enables workflow adaptation to individual user needs and preferences, the proposed approach requires a prediction of possible context situations beforehand. However, context emerges in the moment and cannot be fully predicted. In addition, content sharing between devices might be a time consuming and costly operation, especially when one collaborative task may be accomplished by a number of actors with the same role but only few of them might be able to perform the task. So another valuable extension of this work would be in the development of an appropriate content sharing strategy.

A major limitation of this work is that it is up to workflow designers to construct MobWEL workflows in a way that ensures the consistency between the process-based workflow models and associated content lifecycle(s). Thereby, further work is required to establish a MobWEL workflow design methodology.

References

[1] S. Smanchat, S. Ling, and M. Indrawan, "A survey on context-aware workflow adaptations," in Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, 2008, pp. 414–417.

[2] A. K. Dey, "Understanding and using context," in Personal and Ubiquitous Computing, vol. 5, 2001, pp. 4–7.

[3] L. Pajunen and S. Chande, "Developing workflow engine for mobile devices," in Enterprise Distributed Object Computing Conference, 2007, pp. 279–286.

[4] A. Kocurova, S. Oussena, P. Komisarczuk, and T. Clark, "Context-aware content-centric collaborative workflow management for mobile devices," in Second International Conference on Advanced Collaborative Networks, Systems and Applications, 2012, pp. 54–57.

[5] M. Rosemann, J. Recker, and C. Flender, "Contextualisation of business processes," in International Journal of Business Process Integration and Management, vol. 3(1), 2008, pp. 47–60.

[6] O. Saidani and S. Nurcan, "Context-awareness for adequate business process modelling," in Third International Conference on Research Challenges in Information Science, 2009, pp. 177–186.

[7] M. Adams, A. Ter Hofstede, N. Russell, and W. Van der Aalst, "Dynamic and context-aware process adaptation," in Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments/Ed. MM Wang, Z. Sun, 2009, pp. 104–140.

[8] M. Rosemann and J. Recker, "Context-aware process design: Exploring the extrinsic drivers for process flexibility," in The 18th International Conference on Advanced Information Systems Engineering, Proceedings of Workshops and Doctoral Consortium, 2006, pp. 149–158.

[9] D. Salber, A. Dey, and G. Abowd, "The context toolkit," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: the CHI is the limit, 1999, pp. 434–441.

[10] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E. Malm, "Managing context information in mobile devices," in Pervasive Computing, vol. 2(3), 2003, pp. 42–51.

[11] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach," in Pervasive and Mobile Computing, vol. 2(1), 2006, pp. 37–64.

[12] C. Bettini et al., "A survey of context modelling and reasoning techniques," in Pervasive and Mobile Computing, 2010, pp. 161–180.

[13] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A Survey of Context Data Distribution for Mobile Ubiquitous Systems," in ACM Computing Surveys, vol. 45(1), 2013, pp. 1–49.

[14] D. Kramer, A. Kocurova, S. Oussena, T. Clark, and P. Komisarczuk, "An Extensible, Self Contained, Layered Approach to Context Acquisition" in Proceedings of the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing, 2011, pp. 1–6.

[15] M. Wieland, O. Kopp, D. Nicklas, and F. Leymann, "Towards Context-aware Workflows," in CAISE 07 Proceedings of the Workshops and Doctoral Consortium, 2007, pp. 11–15.

[16] M. Grossmann et al., "Efficiently managing context information for large-scale scenarios," in Pervasive Computing and Communications, 2005, pp. 331–340.

[17] R. Hull, "Artifact-centric business process models: Brief survey of research results and challenges," On the Move to Meaningful Internet Systems: OTM 2008, pp. 1152–1163.

[18] R. Liu, K. Bhattacharya, and F. Wu, "Modeling business contexture and behavior using business artifacts," in Advanced Information Systems Engineering, 2007, pp. 324–339.

[19] K. Bhattacharya, R. Hull, and J. Su, "A data-centric design methodology for business processes," in Handbook of Research on Business Process Management, 2009, pp. 503–531.

[20] K. Wahler, "A Framework for Integrated Process and Object Life Cycle Modeling," PhD Thesis, 2009.

[21] R. Hull et al., "Introducing the guard-stage-milestone approach for specifying business entity lifecycles," in Web Services and Formal Methods, 2011, pp. 1–24.

[22] V. Künzle and M. Reichert, "PHILharmonicFlows: towards a framework for object-aware process management," in Journal of Software Maintenance and Evolution: Research and Practice, vol. 23, 2011, pp. 205–244.

[23] J. Nitzsche, T. Van Lessen, D. Karastoyanova, and F. Leymann, "BPEL^{light}," in Proceedings of the 5th International Conference on Business Process Management, 2007, pp. 214–229.

[24] G. Hackmann, M. Haitjema, C. Gill, and G. Roman, "Sliver: A bpel workflow process execution engine for mobile devices," in Service-Oriented Computing, 2006, pp. 503–508.