PATTERN-BASED AND REUSE-DRIVEN ARCHITECTING OF MOBILE CLOUD SOFTWARE

Aakash Ahmad[1], Ahmed B. Altamimi[1], Abdulrahman Alreshidi[1], Mohammad T. Alshammari[1], Numra Saeed[2], Jamal M. Aqib[1]

[1]College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia
[2]School of Electrical Engineering and Computer Science, NUST, Pakistan

ABSTRACT

Context: *Mobile Cloud Computing (MCC) represents the state-of-the-art technology that unifies mobile computing and cloud computing to develop systems that are portable yet resource sufficient. Mobile computing allows portable communication and context-aware computation, however, due to the energy and resource constraints mobile computing lacks performance for computationally intensive tasks. Cloud computing model uses the 'as a service' model - providing hardware and software services - to offer virtually unlimited storage and processing resources. The integration of mobile and cloud computing has given rise to the MCC systems that are portable, context-aware and resource sufficient.*

Challenges and Solution: *To develop the MCC systems, some recurring challenges such as connectivity, context-awareness, portability and security must be addressed during the system design and architecting process. One way to address these challenges is to use the best practices and repeatable solutions to design and architect the MCC systems. In this research, we aim to utilise the empirically discovered patterns that support reusable design knowledge for architecture-driven development of the MCC systems. We follow a three-step process to empirically discover, document and apply patterns for architecting mobile cloud systems. Specifically, we have discovered three patterns as generic and reusable solutions for MCC systems. We demonstrate the applicability of the patterns based on a case study for architecture-centric development of the MCC patterns. The propose research aims to advance the state-of-the-art on reusable and knowledge-driven architecting of the MCC systems.*

KEYWORDS

Software Engineering, Software Architecture, Software Patterns, Mobile Cloud Computing, Software Reuse

1. INTRODUCTION

Mobile Cloud Computing (MCC) has recently emerged as an innovative technology that unifies mobile computing and cloud computing systems to provide portable and resource sufficient solutions [1]. Mobile computing as a pervasive technology allows its users to exploit portability, connectivity and context-aware computation to perform as variety of tasks such as conducting mobile commerce, acquiring location services, and monitoring personal health [2]. However, due to its portable nature, mobile computing lacks the energy, computation and memory resources to perform computationally intensive tasks [2, 3]. Cloud computing technology has successfully promoted the 'as a service' model that offers pay-per-use and virtually unlimited processing and storage hardware and software services [4]. The unification of mobile and cloud computing as MCC systems can integrate the context-aware and portable computation (from mobile computing) with processing and storage services (of cloud computing) to support systems that are portable, yet resource sufficient [3]. Specifically, in the MCC systems, a mobile device can act as a portable and context-sensitive user interface that relies on the backend cloud server to perform complex tasks such as real-time image processing, context sensing, and human decision support [5]. Despite these benefits, MCC systems entail a number of challenges that relate to the scalability, performance, and availability of the system in a dynamic environment.

To engineer and develop the MCC systems, there is a need to replace the ad-hoc and once-off solutions with reuse-driven and knowledge-based practices for system design. Also, with the growing demands for the adoption of the MCC systems, knowledgeable and experienced systems' designers/architects of MCC systems are needed who are not widely available as MCC has just started to emerge as an innovative technology [1, 5]. In this situation, software patterns and styles can provide reusable solutions and best practices to develop MCC systems. Specifically, software patterns focused on designing or architecting MCC can provide documented and well understood software design solutions to both the experienced and novice architects [6, 7, 8]. Software patterns have been proven successful in providing reusable packages of generic and repeatable solutions to recurring problems of software and system design [9]. This means that patterns capture the concentrated wisdom of practitioners and consolidated design rationale from multiple systems to develop software effectively and efficiently [10]. However, pattern-based architecting is faced with the challenge of empirically discovered patterns that must be systematically documented and recurrently applied to the systems under design [11].

In this research, we aim to exploit patterns to design and architect mobile cloud systems by incorporating pattern-driven reusability and efficiency in the software design process. To do so, we have followed a three-step process that includes (i) pattern discovery, (ii) pattern documentation, and (iii) pattern application to architect a MCC system. An overview of the proposed solution is illustrated in Figure 1. In the proposed solution, as in Figure 1, first the pattern sources are investigated to empirically discover patterns and maintain a repository that acts as a collection of the documented patterns. Finally, the discovered patterns can be selected and applied to support pattern-based software architecture for the MCC system. This means that patterns as generic and reusable solutions create the software architecture – providing the system blueprint – that acts as a bridge between system requirements and system implementation as in Figure 1. It is vital to mention that, while architecting the MCC systems, one exploits dynamically composed services to develop systems that are portable, context-sensitive and efficient [7, 8]. In comparison to the more traditional (object-oriented, component-based and service-driven) systems [10], patterns for mobile MCC architectures are characterized by specific requirements such as mobility, context-sensitivity for (front-end) mobile computing with service composition, and scalability of (back-end) cloud services [12].

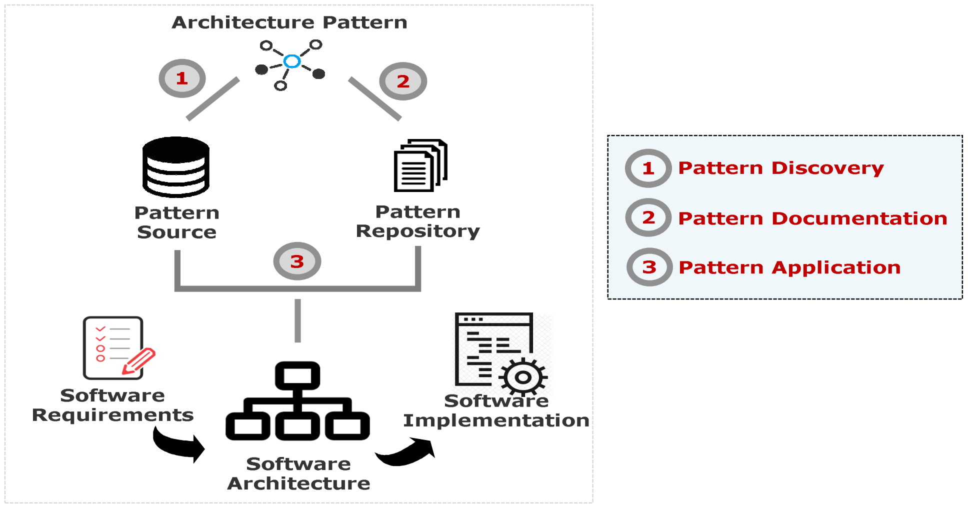


Figure 1. An Overview of the Proposed Solution.

Assumptions and Contributions The proposed research aims to exploit the traditional concepts of software patterns and apply them to architect innovative solutions such as MCC software. We aim to support pattern-based architecting that can accelerate the process of gaining design or architectural knowledge and applying it for architecture-centric software development. The research aims to advance the research state-of-the-art on MCC systems by focusing on empirically discovered patterns and their applications. The research assumes the existence of the pattern sources that can be investigated to discover new patterns. We also assume pattern discovery and document as a continuous process that involves frequent discovery of existing and new sources to discover innovative patterns. This paper provides a significant extension to our research in [6]. In contrast to the catalogue of MCC architectural patterns in [6], in this research we provide case study based demonstration of the applicability of the patterns. Moreover, we also evaluate pattern-based reusability and efficiency of architectural design process. The primary contributions of this research are to:

– Support a three-step process to discover, document, and apply patterns as reusable solutions to guide the system design and architecting phases.

– Apply the discovered patterns to recurring problems of software design and architecture. Pattern applicability demonstrates reusability and efficiency of tasks involved in the system designing and architecting phases.

The rest of the paper is organised as follows. Section 2 presents background details. Section 3 presents the related research. Section 4 discusses the research methodology. Section 5 presents the discovered patterns. Section 6 demonstrates case study based application of the patterns. Section 7 concludes the paper.

2. MCC ARCHITECTURES AND THEIR QUALITY CHARACTERISTICS

In this section, we introduce the software architecture for the MCC systems in Section 2.1. We also present the characteristics for the quality of the architectures in Section 2.2. The terms and concepts introduced in this section are used throughout the paper.

2.1. A LAYERED REFERENCE ARCHITECTURE FOR THE MCC SOFTWARE SYSTEMS

In Figure 2 we have presented a generic architectural view of the MCC systems. Specifically, Figure 2 a) illustrate a reference architecture for the MCC systems that comprises of two layers, (i) Cloud Computing Layer (Back-end), and (ii) Mobile Computing Layer (Front-end) [13]. Figure 2 b) presents the component and connector architectural view of the system [14]. We discuss about the layered architecture view (Figure 2 a)) and the component and connector architectural view (Figure 2 b)) detailed as below. Both the architectural views presented in Figure 2 a) and Figure 2 b) are complementary to each other. The reference architecture in Figure 2 a) provides a generic reference to the system under consideration, whereas, the component and connector view highlights the required components and their interactions in developed system [13].

A) Reference Architecture for the Layering of the MCC Systems

A reference architecture represents the documentation or the structure of a system - as a collection of best practices - that can be referred to during system design to derive advanced architectural solutions [13]. The extended details about the reference architectures and their usage are provided in [7, 9, 11]. In Figure 2 a), the reference architecture illustrates two layers of the system each of which deals with a specific aspect/functionality of the system. The presentation of the reference architecture helps us illustrate pattern-based architecting process, later in the paper.

- **Front-end - Mobile Computing Layer** provides the users with an interface to utilize and interact with the data and exploit the context-based and location information of the mobile device. In addition to the context sensitivity, mobile computing layer enables mobility and computation on the go. However, due to the increased mobility, an inherent issue with this layers is that mobile device(s) lacks computation and storage-intensive resources. Moreover, issues like performance and data security and privacy must also be addressed.

- **Back-end - Cloud Computing Layer** helps a mobile device to off-load the computationally intensive tasks and data to scalable and virtually unlimited storage and processing resources offered by the server [1, 5]. Specifically, the cloud-based servers utilise the pay-per-use and virtually unlimited 'as a service' model that provides hardware and software services to be used by other systems/entities. In Figure 2, the unification of the mobile and cloud computing as mobile-cloud computing enables the users to utilize the features of a mobile device such as context-sensitivity, location-awareness and mobility, while enjoying the virtually unlimited computation and storage resources of cloud computing. In order to support such unification, i.e., integration of the mobile and cloud computing, a continuous network connectivity is required. Network connectivity may involve latency along with security and privacy of data that communicates between mobile and cloud.

A) Component and Connector Architectural View

In Figure 2 b), we present the component and connector architectural view for the system [14]. The component and connector architectural view is composed of architectural components that act as the elements of computation and data stores for the system. The connectors enable the interaction between various components.

The component and connector architectural view is beneficial as it helps to decomposes a system into its building blocks, i.e., architectural components. For example, Figure 2 b) that represents a simplified component and connector-based architecture of the system that invokes location-based services. The *getLocationService* component (running on a mobile device) invokes the location providing service. The location-based service computes the current location coordinates of the user with *calculateLocationCoord* service. Assuming that *calculateLocationCoord* is computation intensive, therefore it is executed on the cloud-based server. Both the components are interconnected using *locSrv* connector.

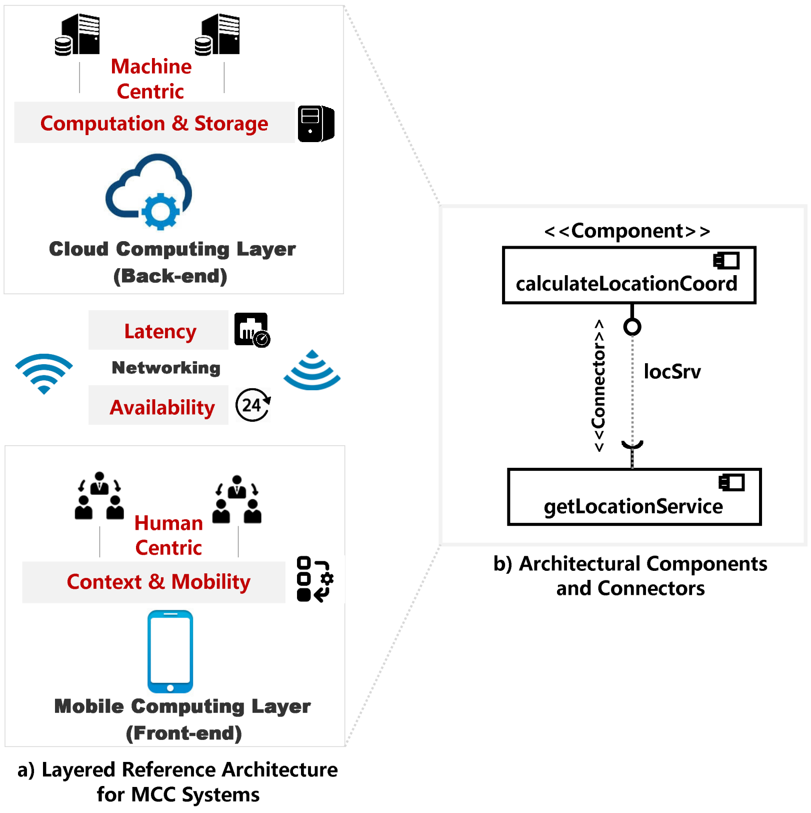


Figure 2. Overview of a Generic Architecture for MCC Systems.

2.2. Quality Characteristics for MCC Architectures

The quality characteristics or non-functional properties (NFRs) represents the attributes of extra functional qualities that a system needs to operate effectively and efficiently. In the MCC systems, in addition to supporting the core functionality, the characteristics of software quality are vital for both the mobile computing and cloud computing layers. For the mobile computing layer of the MCC systems, the main characteristics of the quality include but not limited to:

- *Context-awareness* that refers to the system's ability to exploit the contextual information to support context-aware computing.
- *Mobility* aims to support the computation and system level operations in a portable and on the go fashion.

- *Efficiency* means that system is able to operate and produce results efficiently and effectively.

In contrast to mobile computing layer, cloud based systems rely on the principle of service-orientation that allows service composition to implement cloud-based applications [4, 7]. Therefore, the quality of services being composed or provided are central to the performance of the cloud architectures. This means that cloud-based services must satisfy the desired quality characteristics. These characteristics include but not limited to

- *Elasticity* refers to the systems that are elastic (i.e., addition or removal of system resources) to accommodate the variations in system's operational environments.
- *Multi-tenancy* means that the system must support multiple tenants for the same services provided by cloud-based systems to the potential requesters.
- *Virtualization* means that systems' resources (i.e., hardware and software services) must be virtualised for their effective usage.

The quality attributes for the MCC architectures are distinct from traditional software architectures. This means that the patterns for traditional software development and architecture [9, 10, 11] cannot easily be applied to mobile cloud systems unless they support the above-mentioned characteristics specific to mobile cloud architectures [7, 8]. For example, a pattern that exploits context sensing for recommendations and decision support is only applicable to context-aware computing architectures. It is vital to mention that an individual pattern may not ensure all these characteristics highlighted above, however, a collection of patterns should ensure to address them all. For example, unlike the traditional architectures, mobile cloud based architectures are expected to serve context-aware multiple-tenants with each tenant having its own specific context and QoS requirement that can vary from performance and reliability to security aspects. Context-aware multi-tenant capabilities of MCC systems need to be considered not only at service but also at the platform and infrastructure [7, 8] level not addressed in existing patterns.

3. RELATED RESEARCH

We now present the related research that overviews the research state-of-the-art for the architecting of MCC systems. First, we discuss the existing patterns for mobile and cloud computing architectures in Section 3.1. We then present the related reference architecture and patterns for mobile cloud based systems. The presentation of the related research and solutions helps us to define the scope and contributions of the proposed solution.

3.1. PATTERNS-BASED ARCHITECTING OF MOBILE AND CLOUD COMPUTING SYSTEMS

- *Patterns for Cloud-based Architectures:* In [15], the authors have presented a comprehensive catalogue of architectural patterns for cloud-based systems. The patterns presented in this work promote recurring solutions and best practices for cloud-based Platform as a Service (PaaS). The patterns address the architectural issues relating to the scalability, big data, fault handling and distributed services on the Windows Azure platform. Patterns in [13] provide guidelines and practical solutions to address the scalability and elasticity in cloud-native applications for Windows Azure platform. The work presented in [16] provides a collection of patterns for the development as well as the deployment of the services for cloud-based systems. Specifically, the patterns provide solutions for the (i) development of service models for cloud computing (i.e., *Software as a Service*, *Platform as a Service*, and *Infrastructure as a Service*), and (ii) deployment of the services (using private, public, hybrid, or community clouds). In [16], the authors have organized the presented patterns as a pattern language that allows the architects/developers to select and apply these patterns.

It is vital to mention the role of the research communities to provide a common platform or repositories of newly discovered patterns that are published and made available publicly. Pattern communities can provide concentrated wisdom and knowledge for system architecting that can be reused to tackle the emerging challenges of system design. In the context of community-driven pattern repositories, the work presented in [17, 18] represents repositories of architectural patterns for cloud-based systems. The *Cloud Computing Design Patterns* in [17] presents a catalogue of patterns that address the challenges of scalability, reliability, security and monitoring based solutions for cloud applications. In a similar solution [18], referred to as *Cloud Design Patterns* [16] present a catalogue of patterns that is created by architects based on the type of system design challenges, and their generic solution(s) as design patterns.

- *Patterns for Mobile Computing Systems:* In contrast to the research and development on cloud computing patterns, there is a lack of research patterns and pattern-based architecting of the mobile computing systems. In a recent research in [8], some architectural patterns are presented to support the operations of resource constrained mobile devices. Specifically, three architectural patterns namely *data source integration*, *group-context awareness*, and cyber f*oraging* are presented for mobile computing systems in the context of tactical-edge resource-constrained environments. These pattern-based solutions support first responders and military personnel that operating in edge environments that are driven by flexibility, resource efficiency, and usability, which are key quality attributes for systems at the tactical edge. The patterns enable the architects to reuse best practices for system design and architecture while considering both the functional and quality requirements of the system. In [19], the authors have presented a collection of architectural solutions that are referred to as *mobility patterns*. Mobility patterns have been discovered by analysing the successful development of various mobile applications. Mobility patterns can empower the designers/architects to reuse design rationale and elements as building blocks to engineer and develop new mobile applications.

In comparison to the research state-of-the-art on pattern-based architecting of the cloud [13, 15, 16, 17, 18] and mobile computing systems [8, 19], the proposed patterns support architecture-based engineering and development of the MCC systems. Our research aims to establish a catalogue of architectural patterns - as a continuously evolving repository of patterns - based on discovery and specification of new architectural patterns guided by [16, 20]. We also demonstrate the usefulness of patterns in terms of reusability and efficiency of the architectural design process.

3.2. PATTERNS FOR MOBILE CLOUD COMPUTING SYSTEMS

In recent years, some reference architectures and pattern-based solution have emerged to support service oriented architecture (SOA) for mobile computing [7, 21, 22]. In the following, we discuss both the reference architectures and patterns for MCC systems, whereas the technical details and the distinction between the two can be found in [23].

- *Reference Architectures for MCC Systems:* In the MCC systems, off-loading computationally intensive tasks from a mobile device to the cloud-based server is a challenging task. To support the off-loading, [7] presents a reference architecture that supports the offloading of computational/memory-intensive tasks to cloud-based servers for the mobile devices that operate in hostile environments. The proposed reference architecture in [7] is used to demonstrate some implementations scenarios as well as the architectural trade-offs for the development of MCC systems. Another reference architecture is proposed in [21] that aims to support the first responder teams and their coordination. The reference architecture aims to support the development of group-context-aware mobile applications that exploits contextual information from individuals and their team members to enable context-aware computing and team coordination.

- *Patterns for Mobile SOAs:* To support software services or specifically SOAs for mobile computing six architectural patterns are presented in [12]. These patterns are named as, *standalone*, *full offloading*, *partial offloading*, *SaaS-based*, *CaaS-based*, and *offloaded CaaS-based* is presented. These patterns support both the functional and quality requirements for the mobile-based SOAs. The patterns aims to support the quality attributes that include but are not limited to the performance, efficiency and energy consumption for mobile applications.

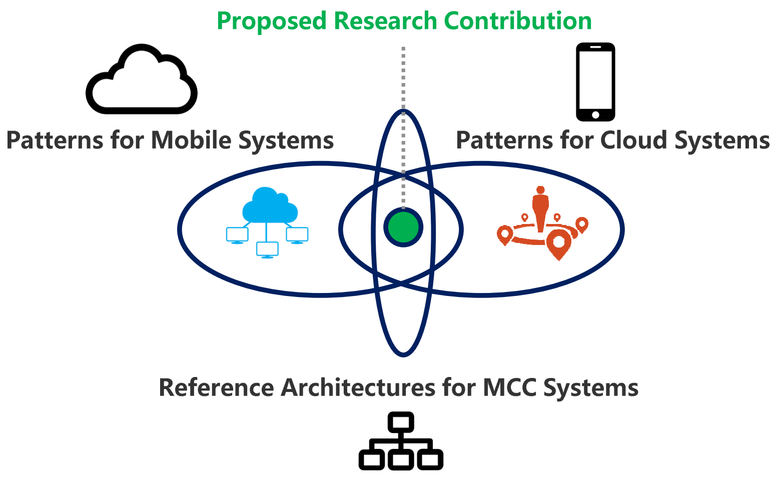

Proposed Research Contribution
Patterns for Mobile Systems
Patterns for Cloud Systems
Reference Architectures for MCC Systems

Figure 3. Overview of the Research State-of-the-Art and Proposed Contributions.

Figure 3 illustrates a high-level overview of different research domains and their overlaps. Based on Figure 3, we can conclude that our research proposes to contribute towards promoting pattern-based architecting for the MCC systems that currently lacks in the existing research. The research and solution presented in [7, 22, 21] on reference architecture and patterns [23] is relevant to our proposal. However, by considering the scope of existing research and the recent challenges for mobile computing [1, 2, 5, 12], we claim that the proposed solution aims to advance the research state-of-the-art on patterns and architectures that can be applied to MCC systems.

4. Research Method and Proposed Solution

We now present the research method the details the methodological steps to conduct this research based on the proposed solution in Figure 4. We use Figure 4 to detail the methodology and proposed solution.

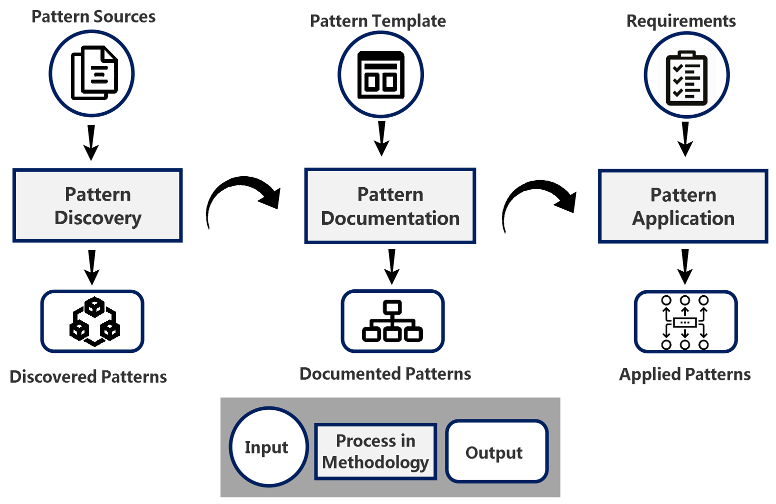


Figure 4. Research Methodology and Solution Overview.

As illustrated in Figure 4, the methodology for pattern-based architectural development comprises of three main processes namely pattern discovery, pattern documentation and pattern application. We also highlight the required inputs and outputs for each of the process. We discuss the first process of pattern discovery here, while the other two processes are detailed in subsequent sections of the paper.

Pattern Discovery Process – As presented in Figure 4, pattern discovery is the initial process that involves investigating the sources of patterns (i.e., logs and pattern files) to discover the recurring solutions as architectural patterns [20]. Specifically, we applied the design review method [24] to discover patterns, i.e., by reviewing recurring design solutions to frequent problems of architecting MCC systems. Our design review team was comprised of 3 members with an extensive experience of (a) conducting the systematic review, (b) pattern mining, and (c) development of mobile and cloud systems. The design review team followed the following steps to discover the patterns.

Step I – *Review of Architecture Based Challenges and Solutions for MCC Systems*: The design review was conducted to investigate the recurring challenges, design problems and existing solutions to develop mobile cloud architectures [25]. A systematic review of the existing solutions follows evidence based software engineering method [22] to minimize the bias and threats in the review. To conduct the design review objectively, we outlined a number of Research Questions (RQs), detailed below. Based on the research questions and available data we selected 85 solutions (problem-solution map) as the primary sources of pattern discovery.

RQ1 – What methods/techniques/frameworks/solutions are provided in existing (research and practices) to model/develop/evolve MCC system architectures?

Objective(s): The objectives of this RQ was to identify and understand the recurring challenges and their architectural solutions for the design and development of the MCC systems.

RQ2 – What are the patterns/styles/frameworks to support reusable design knowledge for architecting MCC systems?

Objective(s): The objectives of this RQ goes beyond analysis of architectural solution to focus primarily on the discovery of reusable design knowledge that can be used frequently as architectural patterns.

Step II *– Identification of Pattern Data Sets:* Once we have identified and reviewed the architectural solution based on the RQs, we extracted the relevant data from the solutions as in Table 1. The datasets here refer to existing architectural challenges, solutions, and their validations. Table 1 acts as a structured template to capture and maintain pattern related data in a systematic manner. Based on the design review process [24], we derived 7 data items that are listed in Table 1 (I1 to I7 – the collection of data items is referred to as datasets) by following the guidelines in [24] and our experience with the discovery of architectural pattern mining [18] The data items in Table 1 helped the team to objectively review and systematically extract the problem (P) and solution (S) mapping, the attributes (A) that affect the solution and the occurrence frequency (T) of the repeatable solution by analyzing the pattern datasets. Once a decision (D) is reached, the results are documented as pattern elements (E) for a peer-review before finalization.

Table 1. Dataset Items for Pattern Discovery Process

ID	**Items**	**Description**
I1	Design Space (C)	All the available architectural design (C = 85 studies).
I2	Recurring Problem (P)	Repeatable problems existing in the design space ($P \in C$).
I3	Frequent Solution (S)	Solutions to repeatable problems in the design space ($S \in C$).
I4	Frequency Threshold (T)	Threshold for occurrence of S to be discovered as a pattern
I5	Design Attributes (A)	Attributes affecting S (A = 08) *Context Awareness, Mobility, Computational Efficiency, Energy Efficiency, Service Reliability, Service Availability, Data Storage, Data Processing*
I6	Discovered Pattern (N)	2 = *Yes*, 1 = *Not sure* (consensus required), 0 = *No*
I7	Pattern Elements (E)	Elements of Pattern Description (E = 9) *Name, Intent, Problem (P), Solution (S), Impact, Origin, Uses, Reference Diagram, Architecture Elements, Constraints*

Step III - *Thematic Analysis based Investigation of the Datasets:* After identification of the datasets, and extraction of the data items, finally thematic analysis is performed. Thematic analysis aims to 'identify, analyse and report' recurring solutions as potential patterns from datasets. A theme is a frequent solution or a method that aims to address recurring issues.

(A) Data Analysis process comprises of (a) analysing datasets, (b) to extract design attributes from problem-solution mapping (I5 in Table 1).

(B) Pattern Discovery process involves (a) searching of the recurring themes based on data analysis, and (b) reviewing the identified themes. To discover patterns, we reviewed studies and aimed at discovering design problems (**I2**) and their relate solutions (**I3**). We consider a recurring theme as a discovered pattern (**I6**).

(C) Pattern Documentation is the last process that includes (a) classification of related themes based on design attributes (**I5**) and documented them in a template (**I7**).

5. TEMPLATE-BASED DOCUMENTATION OF ARCHITECTURE PATTERNS

After the discovery of the patterns, pattern documented is required to maintain the details of the patterns that can be looked-up whenever required to understand and utilise a pattern. Pattern template represents a structured document that provides the necessary elements to capture and represent the individual patterns in a systematic way to support pattern understanding and usage [9, 11]. In this section, first we introduce the pattern template and then demonstrate template-based documentation of the discovered patterns. In the following, we present the necessary elements of pattern template as per the guidelines from [9, 11] to document software patterns in a template. The pattern template and its individual elements are presented in Table 2.

Table 2. Overview of the Elements of Pattern Template

Template Item	Description
Pattern Name	It provides a unique and self-explanatory name for the pattern.
Pattern Intent	It describes the motivation or the known uses of the given pattern
Design Problem and Solution	These provides a mapping of the problem-solution view that the patterns aim to address.
Architecture Elements	Represent the component and connectors or the artefacts of the system to which a pattern can be applied.
Reuse Design Knowledge	The reusable design rationale that is supported by the pattern and can be frequently reused.
Quality Characteristics	These are the attributes of the quality (non-functional properties) that are affected by the pattern.
Reference Diagram	It provides overview of the pattern also known as *pattern thumbnail.*

Based on the details in Section 4, we now discuss three patterns that we discovered. These patterns are named as (i) *Adaptive Mobile-Cloud Offloading*, (ii) *Mobile Cloudlets*, (iii) *Mobile Sensing and Cloud Analytics.* We detail the patterns based on the elements of the pattern template. Before presenting the pattern details, we must distinguish between two important concepts of pattern abstraction and pattern instantiation.

5.1. PATTERN ABSTRACTION AND PATTERN INSTANTIATION

We use an example of one of the discovered patterns named *Adaptive Mobile-Cloud Offloading* to distinguish *abstraction* and *instantiation of the pattern as* in Figure 3.

(1). Pattern Abstraction – Modeling the Pattern

Abstraction aims to promote patterns as generic and high-level solution by abstracting out the complex and implementation specific details of the pattern as illustrated in Figure 5 (a). In Figure 5 (a), an abstract representation of the *Adaptive Mobile-Cloud Offloading* helps the pattern users' (designers/architects) to view and analyze a high-level solution to support the off-loading of computational and storage intensive data to the cloud-based servers. Pattern abstraction can be helpful to analyze the impacts of a pattern on the architecture model before pattern application (preconditions), architectural view when pattern is applied (post-conditions). The pre and post version of pattern-based architectural views also promote patterns as design rationale to support an incremental process of pattern-based architecting.

(2). Pattern Instantiation – Applying the Pattern

In contrast to the abstraction, pattern instantiation provides details with concrete architectural elements to instantiate a pattern as illustrated in Figure 5 (b). Pattern instantiation is also known as pattern application that refines the abstract elements of a pattern with concrete architectural elements, i.e., extending the abstract box and arrows with architectural components and connectors from Figure 5 (a) – to pattern abstraction in Figure 5 (b) [8, 10]. For example, as detailed in Figure 5 (b), the instantiated pattern of *Adaptive Mobile-Cloud Offloading* utilizes the *Data Access Bus* to bind services (in *Service Pool*) to data collection (in *Data Source*) component.

Table 3. Pattern 1 – Adaptive Mobile-Cloud Offloading

Pattern I - Adaptive Mobile-Cloud Offloading
A) Pattern Intent: To enable a mobile device to dynamically determine *what*, *how* and *where* to off-load its data to enhance efficiency and of mobile computing.
B) Design Problem: How to enable a (resource-constrained) mobile device to delegate its memory and computational-intensive data and tasks to (resource-sufficient) computers? **C) Solution:** Integrate the off-loading logic between Mobile Computing and Cloud Computing Layers. Such an integrated logic enables a mobile device to exploit dynamic parameters – such as energy efficiency, computational overhead and storage requirements – to determine and off-load data and tasks to cloud computing servers.
D) Architecture Elements: Mobile Computing Layer with (resource constrained) mobile devices, Cloud Computing Layer (resource Sufficient) server are integrated with off-loading knowledge. **E) Reuse Design Knowledge:** Integration of Off-loading logic/knowledge to delegate mobile computing data and tasks to cloud-based servers. **F) Quality Characteristics:** - *Elasticity* of cloud services (acquiring and releasing resources) based on dynamically determined off-loading. - *QoS-driven Offloading* to ensure that dynamic parameters such as energy, storage and computational efficiency.
G) Reference Diagram: Figure 3 b) represents a concrete instance of the abstract pattern representation in 3 a). Specifically, Figure 3 b) illustrates a scenario of pattern application where mobiles devices are used as portable computers to capture contextual

images that needs analytics and processing to gather information. The image processing must be delegated to the cloud servers that have image databases to match and process the image. The offloading logic is integrated between the mobile device and cloud server to enable a mobile device to selectively and dynamically off-load the images to the appropriate cloud-based server based on energy, computational or storage efficiency.

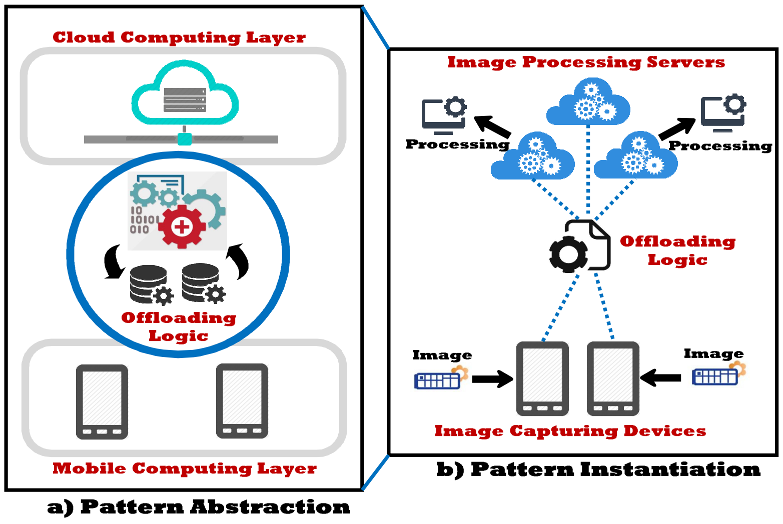


Figure 5. Adaptive Mobile Cloud Offloading Pattern
(Pattern Abstraction and Pattern Instantiation)

Table 4. Pattern 2 – Mobile Cloudlets

Pattern II – Mobile Cloudlets
A) Pattern Intent: To enable a mobile device in a hostile environment to frequently off-load data to servers that are in close proximity of mobile devices that they serve.
B) Design Problem: How to minimise the off-loading latency (on remote server) to a single-hop network while maximising the performance and QoS for mobile computing tasks?
C) Solution: The proposed architecture integrates an intermediate layer (based on localised cloud known as cloudlets) between the enterprise cloud and the mobile device. The solution assumes that connectivity to the main cloud (enterprise) cloud is either not reliable or commonly un-available.
D) Reference Diagram: This architecture inserts an intermediate layer between the central core (i.e., enterprise cloud) and the mobile devices. At the heart of this architecture is a large centralized core that could be implemented as one of Amazon's data centers or a private enterprise cloud. At the edges of this architecture are offload elements for mobile devices. These elements, or cloudlets, are dispersed and located close to the mobile devices they serve [23]. This architecture decreases latency by using a single-hop network and potentially lowers battery consumption by using WiFi or short-range radio instead of broadband wireless which typically consumes more energy [24] [25].

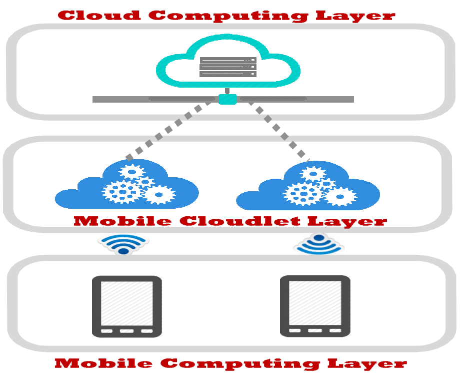


Figure 6. Mobile Cloudlets Pattern

Table 5. Pattern 3 – Mobile Sensing and Cloud Analytics

Pattern III – Mobile Sensing and Cloud Analytics
A) Pattern Intent To exploit the context-sensitive mobile device to capture contextual data that is sent to the cloud-server for data processing and analytics.
B) Design Problem How to gather the contextual information that can be processed and analysed to perform decisions?
C) Solution The proposed architecture presents a two layered architecture namely the mobile sensing and cloud analytics layer. Specifically, the mobile layers enables the capturing of the contextual information (e.g.; environmental condition, traffic congestion) and send it to the cloud based server. The cloud server runs the algorithms to analyse data and provide decision support based on processed data.
D) Reference Diagram: This architecture inserts an intermediate layer between the central core (i.e., enterprise cloud) and the mobile devices. At the heart of this architecture is a large centralized core that could be implemented as one of Amazon's data centers or a private enterprise cloud. At the edges of this architecture are offload elements for mobile devices. These elements, or cloudlets, are dispersed and located close to the mobile devices they serve [23]. This architecture decreases latency by using a single-hop network and potentially lowers battery consumption by using WiFi or short-range radio instead of broadband wireless which typically consumes more energy [24] [25].

E) Reference Diagram

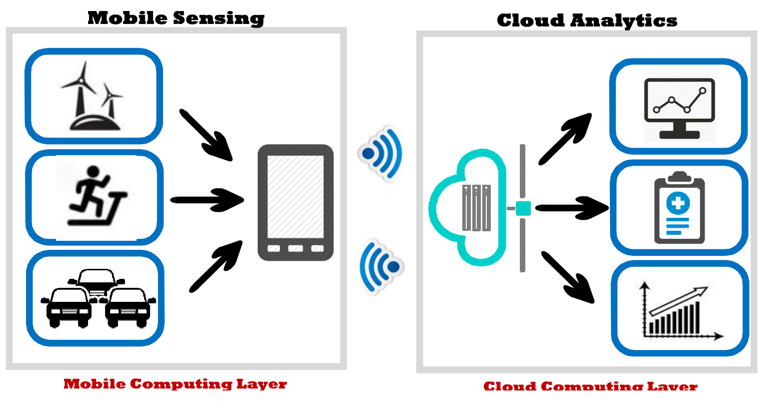


Figure 7. Mobile Sensing and Cloud Analytics.

6. Case Study on Pattern Application

In this section, first we demonstrate the case study based architecting in Section 6.1. We also discuss some threats to the validity of the research in Section 6.2.

6.1 Architecting Safe Campus System

After presenting the discovered patterns, we now demonstrate the pattern applicability to an architectural case study. The case study is based on an ongoing project names 'Safe Campus', that aims to provide student and staff safety at the campus.

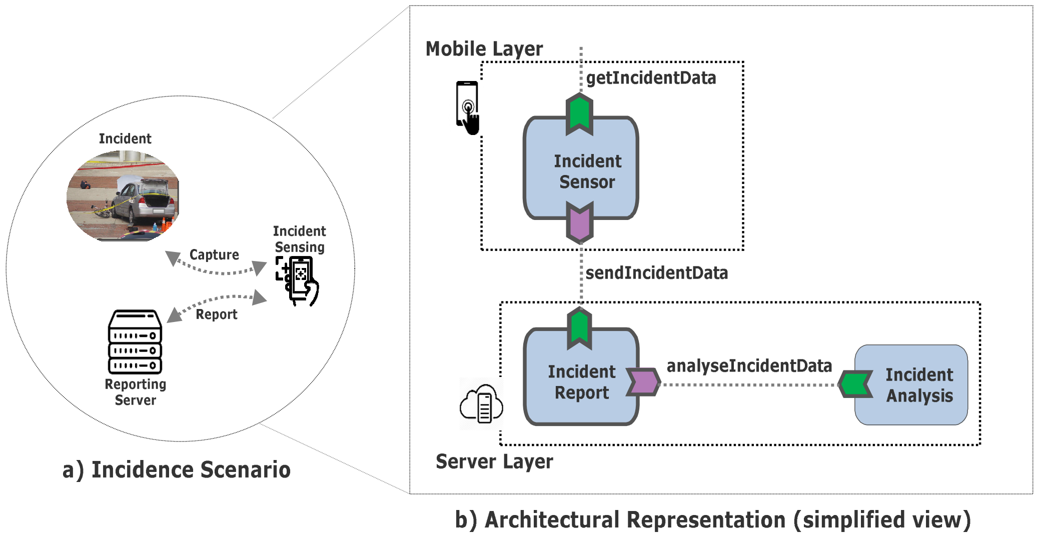


Figure 8. A Partial Architectural Overview of the Safe Campus System

A partial architectural view of the system is presented in Figure 8, that illustrates an incident on the campus. In Figure 8, we have presented the component and connector architectural view of the system. For example, the component named *IncidentSensor* at the mobile computing layer senses an incident and reports it to the component *IncidetReport* using the connector *sendIncidentData* on the server layer. The incident can be captured (normally an image, textual details, location of the incident etc.) with a mobile device that runs the Safe Campus. The incident captured by any individual using their mobile device can be communicated with the incident reporting server that manages the incident reports and their processing. The server is a cloud hosted machine that stores, retrieves and processes the incident details.

We utilise one of the discovered patterns to architect the above-mentioned scenario. For demonstration purposes, we have selected and presented the scenario that has an associated pattern available. There are various scenarios where a relevant architectural pattern is not available. In such case, the designer/architect must develop the architecture without any patterns. Patterns only provide a packaged and reusable design rationale for architecting the systems. We illustrate the pattern application based on Figure 9.

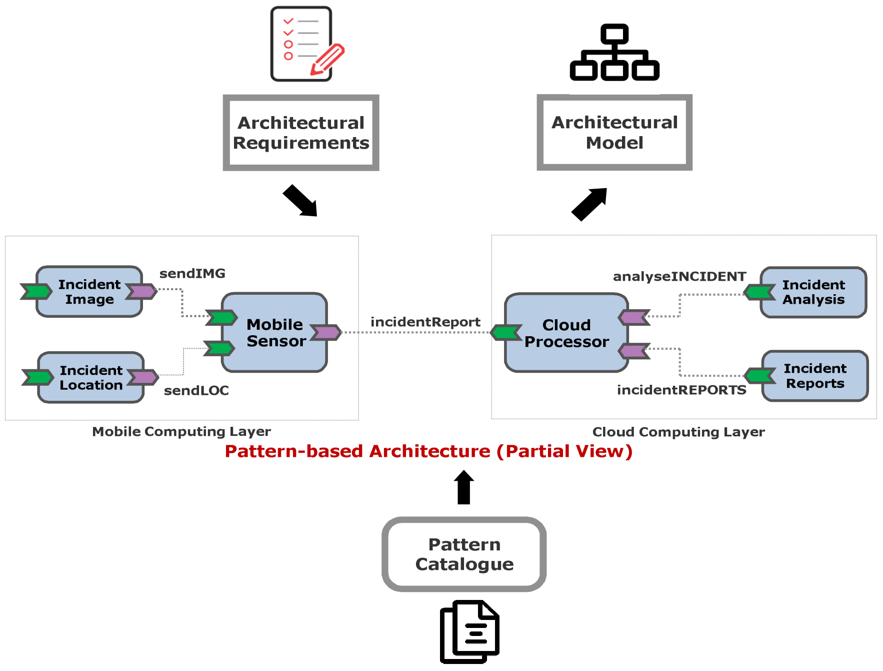


Figure 9. An Overview of the Pattern-based Architecting Process.

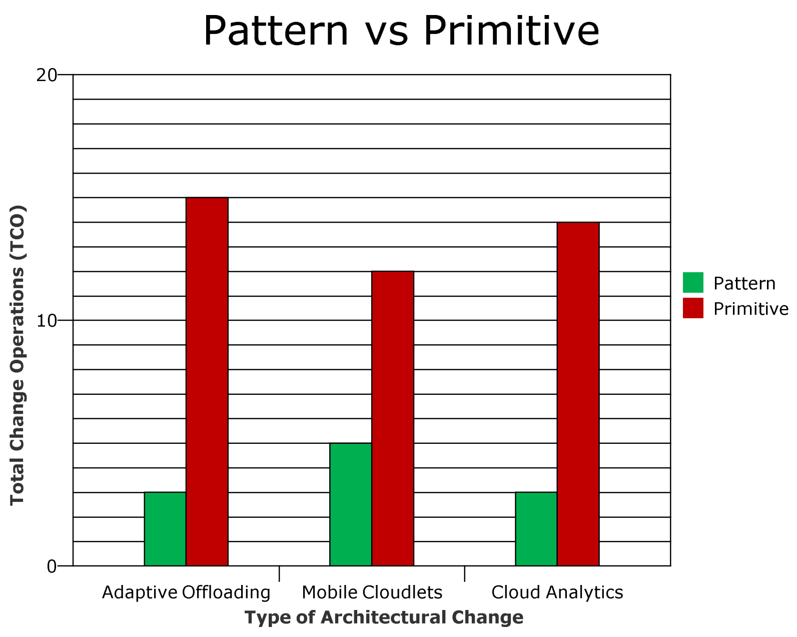
As illustrated in Figure 9, architectural requirements are the foundation for designing the desired architecture. The designer/architect, based on the requirements selects the most appropriate pattern from the catalogue and apply it to achieve reusability in the architectural design process to achieve the architecture [26]. In Figure 9, the *Mobile Sensing and Cloud Analytics* pattern has been applied (cf. Table 4). The pattern enables the integration of the mobile computing layer that senses the incident and send the details to the cloud server. The cloud-based layer manages the processing and analytics of the incident.

6.2 Evaluating the Efficiency of Pattern-based Architecting

After discussing the pattern-based architectural design (cf. Figure 9), we also present the evaluation for the efficiency of pattern-based architecting. In comparison to patterns, primitives are regarded as individual operations that add or remove architectural elements. For example, to accommodate a new component named *IncidentSensor* in the architecture a primitive operation named add a component is executed. A comparison between the pattern and primitive based operations for architecting is illustrated in Figure 10. To evaluate pattern based architectural changes, we compare patterns with primitives and patterns with following parameters. These parameters are adopted from [28] and used for evaluating pattern efficiency.

Total Change Operations To quantify the required efforts for adding, removing or modifying the architectural elements, we count the number of change operators required for implementing a change and call this Total Change Operations (TCO) as illustrated in Figure 10. In [28], TCO is regarded as the total number of architecture change operations required add, remove or modify the architectural elements.

Ratio of Change Operationalization (Primitive vs Pattern) represents the ratio of change operators from pattern to primitive changes expressed as: $1 - (N_{TCO}/E_{TCO})$. N_{TCO} denotes the number of change operations required by the patterns (N), whereas E_{TCO} denotes the number of change operations required by the primitive (E). The ratio is calculated as the sum of the total operations as: 1 – (11/41), i.e.; 74% approx.



	Adaptive Offloading	Mobile Cloudlets	Cloud Analytics
Pattern	3	5	3
Primitive	15	12	14

Figure 10. Overview of Pattern vs Primitive based Architecting Steps

Based on an overview in Figure 10, we conclude that in comparison to patterns, primitive changes require between 8 and 15 change operations (steps) to add/remove/modify an architectural element. Also, patterns when compared to the ad-hoc steps of primitives are regarded as process-based steps for architecting. The evaluation results based on evaluating three patterns suggests that: Pattern-based changes can reuse up to 74% of change operations compared to primitive changes to increase reusability of changes in the architecture-centric software evolution. However, pattern-based change does not support a fine-granular change representation.

6.3 Threats to the Validity of Research

We have presented the proposed solution with preliminary application and evaluation of the patterns. The discovered patterns are less in number, however; we have proposed pattern discovery as an incremental process. This means as the new sources of patterns become available, new patterns can be discovered and maintained in the catalogue. In this section, we highlight some threats to the validity of the research. These threats, if not tackled as ongoing or future research can limit applicability of the results.

(1) Threat I - Applicability of Patterns - We have presented a limited demonstration of the patterns. There is a need for more practical cases and different systems to realise the usefulness of the patterns along with their limitations and potential improvements. Ideally, we aim to analyse the increased reusability and decreased efforts with pattern-based architecting. Above all, the feedback of the pattern users (i.e., designers and architects) is needed to assess the usability of the patterns.

(2) Threat II - Continuous Discovery of Patterns – It is well established that patterns as generic, reusable best practices cannot be invented, therefore, patterns must be empirically discovered. One of the primary threats to the validity of the research is the limited number of patterns that have been discovered. We have outlined a process that uses empirical approaches to discover new patterns by mining new datasets. In comparison to the traditional software systems, mobile cloud systems are new and therefore need innovative patterns to-support their reusability.

7. CONCLUSIONS

Mobile cloud computing represents state-of-the-art mobile computing technology to support context-aware, portable and resource sufficient systems. The mobile cloud systems exploit mobility along with dynamically available software as services to achieve *context-awareness*, *elasticity* and *scalability* etc. that can be best sup-ported by applying reusable practices and solutions. We proposed a 3-step; pattern-driven architecting process that exploits empirically discovered patterns to guide architectural design for MCC systems. A collection of patterns enhances reusability by abstracting (design primitives). Pattern discovery is a continuous process and provides a systemic approach to investigate emerging design problems and their recurring solutions.

We have highlighted some validity threats that include an objective assessment of the applicability, reusability and efficiency of the patterns based on more complex case studies. In addition, to support the vision of pattern-based architecting for MCC, new patterns must be continuously discovered and also validated by the pattern users/designers.

REFERENCES

[1] Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. Wireless Communications and Mobile Computing, 13(18), 1587-1611, Wiley.

[2] M. Satyanarayanan (2011). Mobile Computing: the Next Decade. ACM SIGMOBILE Mobile Computing and Communications Review. vol 15, no 2, pp: 2-10, 2011, ACM.

[3] Baccarelli, Enzo, Nicola Cordeschi, Alessandro Mei, Massimo Panella, Mohammad Shojafar, and Julinda Stefa. "Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study." IEEE Network 30, no. 2 (2016): 54-61.

[4] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee et al. "A View of Cloud Computing." Communications of the ACM 53, no. 4 (2010): 50-58, ACM.

[5] Fernando, N., Loke, S.W. and Rahayu, W., 2013. Mobile Cloud Computing: A Survey. Future Generation Computer Systems, 29(1), pp.84-106.

[6] A. Ahmad, A. Altamimi, A. Alreshidi.: Towards Establishing a Catalogue of Patterns for Architecting Mobile Cloud Software. In 9th International Conference on Software Engineering and Application (SEAS), 2017.

[7] S. Simanta, K. Ha, G. Lewis, E. Morris, and M. Satyanarayanan. A Reference Architecture for Mobile Code Offload in Hostile Environments. In Fourth International Conference on Mobile Computing, Applications and Services (MobiCASE), pp: 274–293, 2013.

[8] G. Lewis, S. Simanta, M. Novakouski, G. Cahill, J. Boleng, E. J. Morris, J. Root. Architecture Patterns for Mobile Systems in Resource-Constrained Environments. In Military Communications Conference (MILCOM), pp: 680–685, 2013.

[9] F. Buschmann, K. Henney, D. C. Schmidt. Pattern Oriented Software Architecture vol 5: On Patterns and Pattern Languages. Wiley and Sons, ISBN-13: 978-0471486480, 2007.

[10] C. Pahl, S. Giesecke, W. Hasselbring. Ontology-based Modelling of Architectural Styles. In Information and Software Technology. vol. 51, no. 12, pp: 1739–1749, 2009.

[11] N. B. Harrison, P. Avgeriou, U. Zdun. Using Patterns to Capture Architectural Decisions. In IEEE Software, vol. 24, no. 4, pp: 38-45, 2007.

[12] IBM Developers Work. Mobile Cloud Computing Devices, Trends, Issues, and the Enabling Technologies. [Online:] http://www.ibm.com/developerworks/cloud/library/cl-mobilecloudcomputing/ , Accessed 10/18/2017.

[13] E. Nakagawa, A. Oliveira, M. Becker. Reference Architecture and Product Line Architecture: A Subtle but Critical Difference. In 5th European Conference on Software Architecture, 2011.

[14] N. Medvidovic and R. N. Taylor. A Classification and Comparison Framework for Software Architecture Description Languages. In IEEE Transactions on Software Engineering, vol 26, no. 1, pp: 70-93, 2000.

[15] B. Wilder. Cloud Architecture Patterns. O'Reilly Media, Inc, ISBN 10: 1-4493-1977-7, 2012.

[16] C. Fehling, F. Leymann, R. Retter, D. Schumm, W. Schupeck. An Architectural Pattern Language of Cloud-based Applications. In Proceedings of the 18th Conference on Pattern Languages of Programs (PLoP), 2011.

[17] Cloud Computing Design Patterns: [online:] accessed on January 5, 2016. http://www.cloudpatterns.org/

[18] Cloud Design Patterns. [online:] accessed on January 6, 2016. http://en.clouddesignpattern.org/index.php/Main_Page

[19] J. Roth. Patterns of Mobile Interaction. In Personal and Ubiquitous Computing, vol 6, no 4, pp: 282 - 289, 2002.

[20] A. Ahmad, A., P. Jamshidi, C. Pahl. Graph-based Pattern Identification from Architecture Change Logs. In 10th Workshop on Systems/Software Arhitecture, pp. 200-213. Springer. 2012.

[21] G. Lewis, M. Novakouski, and E. Snchez. A Reference Architecture for Group-Context-Aware Mobile Applications. In Mobile Computing, Applications, and Services, volume 110 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp: 44–63. Springer, 2013.

[22] J. Kim. Architectural Patterns for Service-based Mobile Applications. In IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp: 1-4, 2010.

[23] Architectural Patterns, Reference Models, and Reference Architectures, [online:] accessed on January 6, 2016. http://www.ece.ubc.ca/~matei/EECE417/BASS/ch02lev1sec3.html.

[24] K. Z. Chen. Integration of Design Method Software for Concurrent Engineering using Axiomatic Design. In International Journal of Manufacturing Technology Management, vol 9, no 4, pp. 242–252, 1998.

[25] A. Ahmad, A. B. Tamimi, N. Saeed, M. Hamayun, M. Fraz. Research Protocol of Software Architecture for Mobile Cloud Systems: A Mapping Study. Technical Report, College of Computer Science and Engineering, University of Ha'il, 2017.

[26] A. Ahmad, P. Jamshidi, C. Pahl. Classification and Comparison of Architecture Evolution Reuse Knowledge – A Systematic Review. In Journal of Software Evolution and Process, vol 26, no 7, pp: 654-691, 2014.m.org/10.1145/161468.16147.

[27] A. Ahmad, M. A. Babar. A Framework for Architecture-driven Migration of Legacy Systems to Cloud-enabled Software. In Proceedings of the WICSA/ECSA 2014 Companion Volume, ACM, 2014.

[28] F. Khaliq, A. Ahmad, O. Maqbool, P. Jamshidi, C. Pahl. Exploiting Patterns and Tool Support for Reusable and Automated Change Support for Software Architectures. In International Journal of Software Engineering. Vol 9, no 1. Pp: 35 – 58, 2016.