ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com
Journal DOI: 10.21474/IJAR01

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

MODELING DATA-DRIVEN SYSTEMS: THE I4 REQUIREMENTS ENGINEERING METHODOLOGY (INCEPTION, INTERACTION, ISOLATION, AND INTEGRATION)

Khalil Abuosba.
Ted Rogers School of Information Technology Management, Ryerson University, Toronto, Canada.

Manuscript Info

Manuscript History:

Received: 18 March 2016
Final Accepted: 29 April 2016
Published Online: May 2016

Key words:
system, design, data flow, model, methodology, requirements engineering.

Corresponding Author

Shivangi Mathur.

Abstract

Identifying scopes and requirements is a very challenging process that face systems analysts and architects. Engineering information systems is a very complex process that requires a clear understanding of the scopes and requirements of the system. Distributed information systems are of high complexity, Successful systems design call for design modularity. A top-down approach is one of the successful approaches that can be utilized in decomposing a problem statement. In this research we introduce a design methodology that utilizes existing software engineering tools and methods to achieve goals of design required within distributed systems solutions. We address the challenges by decomposing design problems using a four phase approach: Requirements Engineering, Data Flow Abstraction, Module Boundary Definition, and System Lifecycle Design. The methodology, implicitly, promotes and achieves reusability, interoperability, security, and transparency. Facilitation and formalization of the process of abstracting systems design modules is achieved by abstracting requirements using software engineering tools and specifications. Phases of design are utilizing specific tools that facilitates the goals of design.

Introduction:-

A System may be defined as a collection of components – machine, software and human – which co-operate in an organised way to achieve some desired result – the requirements [1]. Systems may be abstracted as a set of entities that interact using a set of processes. Entities must co-operate, interfaces between components are a vital consideration in system and requirements engineering. Entities interact with other entities using processes. Requirements are statements that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines) [2]. Designing distributed systems is a very complex process and defining requirements accurately is achievable by adopting requirements engineering methods. Requirements engineering may be defined as the subset of systems engineering concerned with discovering, developing, tracing, analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction [1]. In the software engineering industry, the V-Model is considered as one of the most popular requirements engineering methodology. The purpose of V-model is to improve the efficiency and effectiveness of software development activities and reflect the relationship between development activities and test activities [3]. However, the methodology lacks support of properties propagation and properties enforcement. Most of the requirements engineering methodologies separates the phases of design based on the goals that each phase attempt to achieve. For example security is not embedded and the methodologies do not promote implicit security design. Goals of design should be achieved by explicitly specifying them as well as implicitly achieve them by possibly utilizing constraint management utilities.

As an example we demonstrate a problem and we apply a different approach of requirements engineering that implicitly would achieve goals of design including reusability, interoperability, security, and transparency. Big data is an application of Business Intelligence (BI). BI is a term that incorporates a range of analytical and decision support applications in business including data mining, decision support systems, knowledge management systems, and online analytical processing [4]. Big data can be described as large collections of data that can be structured or unstructured and grow so large and quickly that it is difficult to manage with regular database or statistics tools [5]. Big data is defined as large pools of data that can be captured, communicated, aggregated, stored, and analyzed, it is now part of every sector and function of the global economy [6] argues that big data is data which "exceed(s) the capacity or capability of current or conventional methods and systems." NIST defines Big Data as a paradigm where the data volume, acquisition velocity, or data representation limits the ability to perform effective analysis using traditional relational approaches or requires the use of significant horizontal scaling (more nodes) for efficient processing. The problem statement that we are attempting to solve is: How can we abstract a Big Data Processing System that solves the problems data that are archived, at rest, and in-motion using requirements engineering methodologies.

Scope and reqiements:-

We need to understand the system scope at first and then using brainstorming techniques and requirements analysis method identify the goals of design. A good method here is to identify the main deliverables of the systems. Another approach is to look at the systems as a main process with a set of inputs and a set of outputs. The inputs here are data of three types: structured (relational data model), semi-structured data (data model) or unstructured data (no data model). The output at this stage is processed data.

Big data are generated by executing business and personal data transformation processes; these data may be classified as transaction data, web text resources, log data (aka machine data), events. e-mail, social media, sensors, external feeds and live streams, RFID scans, Form Text, explicit geographic positioning information known as geospatial data, audio, still images, videos. Further brainstorming classifies big data as three types of data: archived data, repository data, and flowing data (fig. 1). Flowing data (aka in motion) are dependent five factors: velocity (rate of data flow), variability (change of rate of data flow, structure, and refresh rate), accessibility, transport format, and transport protocols.

Entities and processes:-

We identify the main building block of big data lifecycle based on the functional requirements of the system: Data Acquisition and Collection, Data Serialization, Data Aggregation, Data Mining, Information Dissemination. Fig.2 illustrates a context-level diagram that represents data flows between the main entities of the system: Data Serializer, Data Collector, Data Aggregator, Data Warehouse System, Service Requester, and Analytics are identified.

Based on brainstorming and requirement analysis methods a solution for Big Data Processing system would identify the main building blocks of the system: the Data Acquisition Process, the Data Serialization Process, the Data Aggregation Process, the Data Analysis Process, the Data Mining Process, the Knowledge Representation Process, and the Information Dissemination Process.

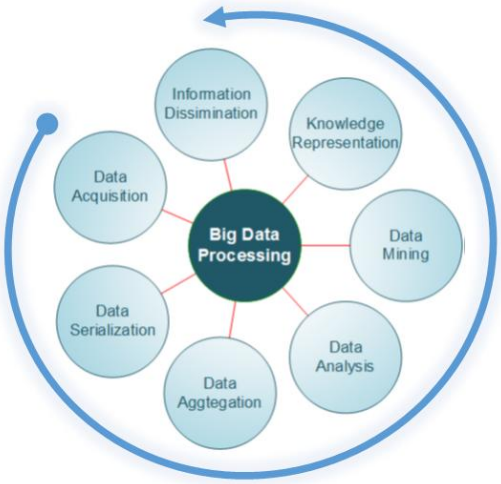

Fig. 1. Early stages of requirement engineering of Big Data Processing problem.

Four-phase decomposition:-

The methodology identifies four phases of system design: the Inception Phase, the Interaction Phase, the Isolation Phase, and the Integration Phase. Inception is achieved using Requirements Engineering, Interaction is achieved using Data Flow Abstraction, Isolation is achieved using Module Boundary Definition, and Integration is achieved using Lifecycle Design.

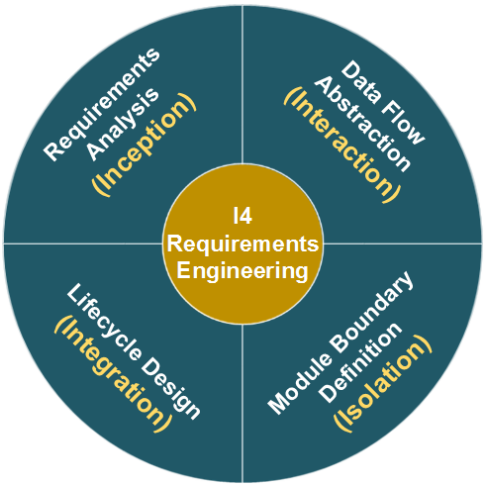

Fig. 2. The Building Blocks of The I4 Requirements Engineering Metodology

The Inception Phase (Process Definition): in this phase, requirements engineering methoods and tools are utilized. The minimum deliverables of this phase is to identify the main entities of the system. System capabilities are abstracted as a set of processes.

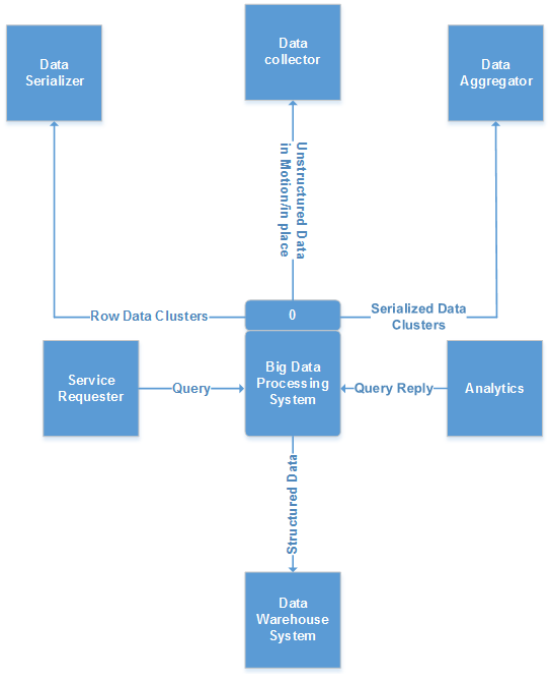

Fig. 3. A Context-level Data Flow Diagram (DFD) for the Big Data Processing System.

Software engineering tools including UML, SysML, and Z-notation would several of the tools that ought to be used. In the context of system design, we find data flow drafting as a major tool that ought to be utilized in the process of abstracting entities and processes. DFDs have advanagous properties over other software engineering tools. At early stages of systems design projects, DFDs promotes implictly modularity, isilation, and security.

In figure 3, the main entities of the systems are identified by drafting a context level data flow diagram; at this stage, we utilize context level diagram to abstract how and what type of data the system would be interacting with.It is noted here that a context-level diagram is a rquirement for the inception phase.

The Intarction Phase (Security Enforecement): in this phase, interaction between entities of the system is specified. Exploding logical level-0 Data Flow Diagram shall achive module boundary defintion. By design, DFD limits entities interactions with the system through processes. No entity has access to data store directly; all entities must interact with data stores through a process as shown in fig. 4.Based on theses processes, modular design may be specified.

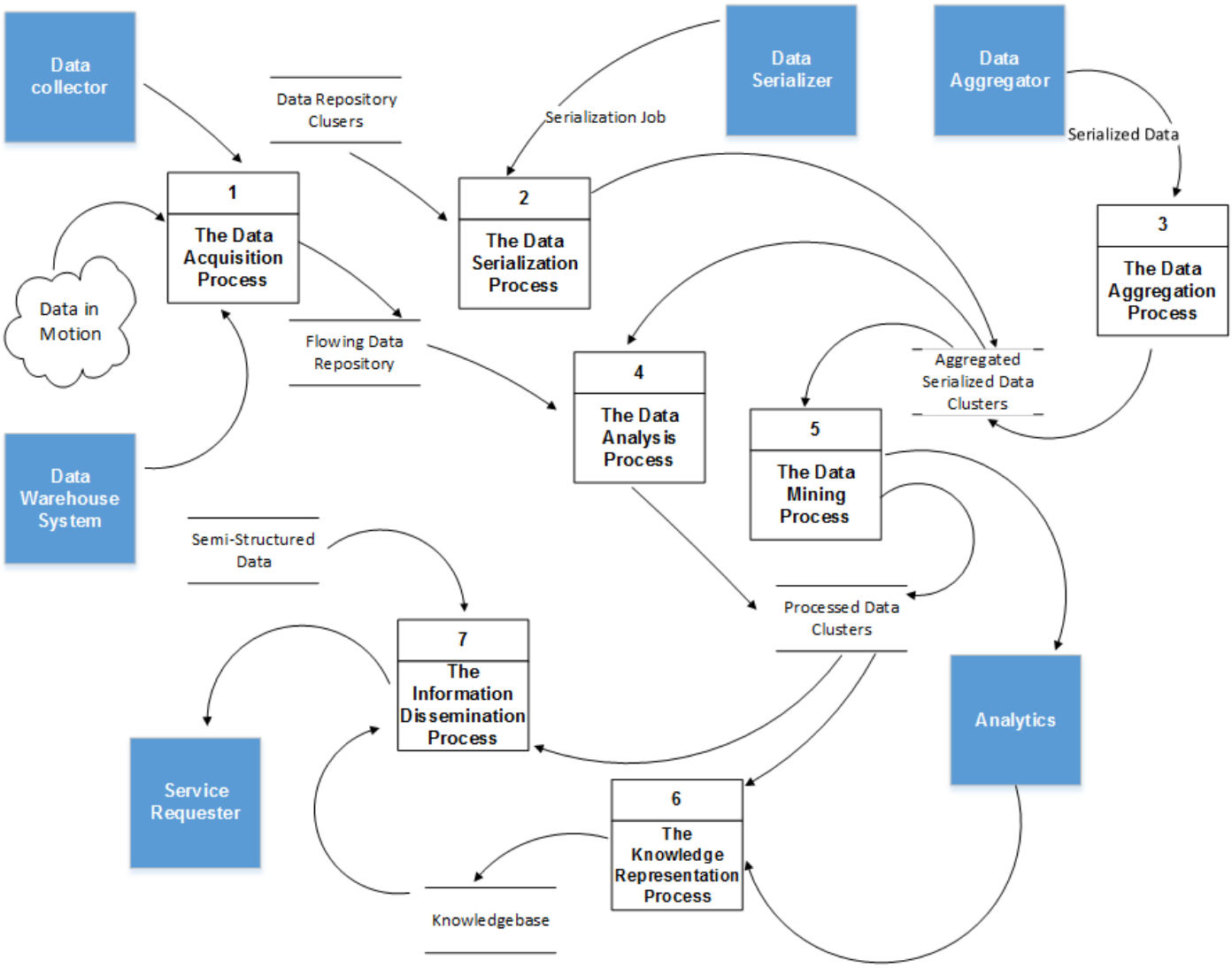

Fig. 4. An example of the Inegration Phase (Lifecycle Abstraction and Design). A Level-0 Data Flow Diagram (DFD) for a Big Data Processing System.

The Isolation Phase (parallism facilitation): in this phase modules boundaries is specified. Modules boundaries may be abstracted and isolated based on four elements (Entity, Process, data store, and Interface). Based on the processs of exploiding the centext-level DFD diagram to a level-0 DFD diagram, modules boundaries can be specified. Interactions between processes would be implemented through specific interfaces.

The Integration Phase (facilitates transpaerncy): The transparency property may be achived by drafting System Life Cycle that integrated all modules into one system. This model can be built using the Lifecycle Modeling Language.

Conclusion and future work:-

Information systems exists in isolation or in distributed environment. Distributed information systems are of high complexity, Successful systems design call for design modularity. In this research we have introduced a methodology that facilitate and formalizes the process of abstracting systems design modules. The methodology, implicitly, achieves four properties: reusability, interoperability, security and scalability. Future work shall be designing a Big Data Processing System Lifecycle using the Lifecycle Modeling Language that conforms to the I4 Requirements Engineering Methodology. Future work will be decomposing each phase into a set of activities that

will be utilizing semi-formal tools including UML, SysML, and lifecycle modeling language. More detailed logical and physical DFDs will be generated. Big data processing lifecycle will be used to demonstrate the process of abstraction. Physical DFD will address specific technologies and different types of data stores.

References:-

1. Hull, Elizabeth; Jackson, Ken; Dick, Jeremy, Requirements Engineering, Springer, 2004.
2. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf
3. S. Mathur, and S. Malik, "Advancements in the VModel. International". Journal of computer applications, 1(12), 2010.
4. Marakas O'Brien, Introduction to Information Systems, 6th Edition, McGraw-Hill, 2013.
5. Raul F. Chong, Clara Liu, DB2 Essentials - Understanding DB2 in a Big Data World, IBM Press, 2014.
6. The National Institute of Standards and Technology's Joint Cloud and Big Data Workshop, http://www.nist.gov/itl/cloud/cloudbdworkshop.cfm
7. Lifecycle Modeling Language Specification, http://www.lifecyclemodeling.org/specification/.
8. K. Abuosba, "Formalizing Big Data Processing Lifecycles: Acquisition, Serialization, Aggregation, Analysis, Mining, Knowledge Representation, and Information Dissemination", IEEE, 2015.