4. Methodology:

The highest-level of data flow diagram is known as the context diagram. A context diagram is a data flow diagram of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system. The context diagram shows the overall business process as just one process and shows the data flows to and from external entities. The context diagram is decomposed into the lower-level diagram which is level 0 data flow diagram. In fact, each process on the level 0 data flow diagram can be decomposed into more explicit data flow diagram, called level 1 diagram and can be further decomposed into next lower-level diagram when it is needed. In general, there are two fundamentally different types of problems that can occur in data flow diagrams which are syntax errors and semantics errors. The syntax of the data flow diagram is how components are interconnected through data flows and what components constitute the subsystem being modeled. The semantics of the data flow diagram, on the other hand, is how data flows are interrelated in terms of data transformations. Syntax errors are easier to find and fix than are semantics errors because there are clear rules that can be used to identify them. There is a set of rules that must be followed by analysts when drawing data flow diagrams in order to evaluate data flow diagrams for correctness;

Rules of data flow diagrams:

1. At least one input or output data flow for external entity
2. At least one input data flow and/or at least one output data flow for a process
3. Output data flows usually have different names than input data flows for a process
4. Data flows only in one direction
5. Every data flow connects to at least one process
6. A unique name (verb phase), a number and a description for a process
7. A unique name that is a noun and a description for a data flow
8. A unique name that is a noun and a description for data store
9. A unique name that is a noun and a description for external entity
10. Every set of data flow diagrams must have one context diagram
11. There is a consistency viewpoint for the entire set of DFDs.
12. Every process is wholly and completely described by the processes on its children DFDs.
13. Every data flow, data store and external entity on a higher level DFD is shown on the lower-level DFD that decomposes it.
14. For every data store, data cannot move directly from one data store to another data store. Data must be moved by a process.

These are the fundamental rules of data flow diagrams. The consistency between context diagram and data flow diagram is very important and the rules for these consistency is captured in rule 10 and 11. Following on the consistency issue, rule 13 addresses aspect on decomposition of the processes to its lower level of DFD and rule 14 addresses aspect of balancing of DFD elements to its lower level of DFD. Syntax rules are used to verify syntax errors within the DFD. The syntax rules are defined as:

Syntax rules of data flow diagram:

I. At least one input data flow for a process
II. At least one output data flow for a process
III. Process from external entity cannot move directly to another external entity
IV. At least one input data flow for a data store
V. At least one output data flow for a data store
VI. Data from one data store cannot move directly to another data store

Among these seven rules six syntax rules are used in order to verify the correctness of the context diagram and level 0 data flow diagram. However, the syntax rules of data store only applied in level 0 data flow diagram. Semantics rules are used to verify semantics errors from context diagram to level 0 data flow diagram. The semantics rules are defined as:

Semantic rules of data flow diagram:

a. The total number and name of external entities in context diagram are the same as in level 0 DFD
b. The total number and name of data flows between process and external entity in context diagram are same as level 0 DFD. The total number and name of external entities in level 0 DFD are same as context diagram. The total number and name of data flows between process and external entity in level 0.
c. DFD are the same as in context diagram

A lecturer can send his or her academic information to the system and can get a list of academicians from the system. DFD is widely used during analysis phase to capture the requirements of any system. The rules for DFD are simple and not complicated. However, the rules are in plain English without formalism. Furthermore, no formal language has been used for semantic specification of data flow diagram; data flow diagram consistency is the extent to which information contained on one level of a set of nested data flow diagram is also included on other levels. The Level 1 DFD presents us with an overview of the system. The following steps explain how this is done.
Step 1. Make each process box the system boundary. All data flows to or from that process are now flows across the lower-level system boundary.

Step 2. Draw, outside the new boundary, the sources and recipients of these flows, as shown on the higher level DFD, (these can be external entities, data stores or other processes. Ensure the labelling is consistent with the higher level.)

Step 3. Identify and draw the processes at the lower levels that act on these data flows. Number the sub-processes with a decimal extension of the higher level number. i.e. Level 1, Process 3 will break down to processes 3.1, 3.2, 3.3 etc. Those processes that cannot be decomposed further, mark with an asterisk in the bottom right-hand corner.

Step 4. Carry out consistency checks using rule 10 and 11.

Step 5. Make sure that all lower level DFDs map onto the Level 1 diagram, by checking data flows.

Step 6. Review the lower levels with the User to be sure that every activity performed by the system is depicted.

When the DFD has progressed as far as it is possible to go, the details must be recorded on an Elementary Process Description (EPD) using a concise and precise narrative. If more than four or five sentences are required, perhaps the process has still to be broken down to another level. Physical DFDs are a means to an end, not an end in them. They are drawn to describe an implementation of the existing system for two reasons:

- To ensure a correct understanding of the current implementation (users are generally better able to discuss the physical system as they know it through people, workstations and days of the week.)
- The implementation itself may be a problem or limiting factor; changing the implementation, rather than the system concept may provide the desired results.

A logical view steps back from the actual implementation and provides a basis for examining the combination of processes, data flows, data stores, inputs and outputs without concern for the physical devices, people or control issues that characterise the implementation. A logical data flow diagram is derived from the physical version by doing the following:

- Show actual data needed in a process, not the documents that contain them.
- Remove routing information; that is, show the flow between procedures, not between people, offices or locations.
- Remove references to physical devices.
- Remove references to control information
- Consolidate redundant data stores.
- Remove unnecessary processes, such as those that do not change the data or data flows. Context level diagrams show all external entities. They do not show any data stores. The context diagram always has only one process labeled 0. It includes all entities and data stores that are directly connected by data flow to the one process you are breaking down. It shows all other data stores that are shared by the processes in this breakdown (these data stores are “internal” to this diagram and will not appear in higher level diagrams, but will appear in lower level diagrams). That last statement is often confusing. Here is another explanation using the hierarchy on the next page. If a data store is used only by processes 3.2.1 and 3.2.3, then it will appear only in the level 2 diagram that includes processes 3.2.1, 3.2.2 and 3.2.3. It will not appear in the diagram that shows processes 3.1 and 3.2 because it is internal to process 3.2.

Primitive DFD: These DFD’s identify who is involved (entities), what the processes are and what data is needed. They do not indicate what technology will be used, nor do they even indicate which processes or data stores will be automated. For you interest, physical DFD’s resemble logical at first glance. But they do have differences such as:

1. Data stores become files but may be changed in order to show distributed processing, etc.
2. There is additional notation to show volumes, locations etc.

Logical DFD’s such as the ones we are drawing in analysis do not include the following:

- audit trails
- backup and restore
- security
- temporary data stores needed only during one business transaction (e.g. holding the details of an online order just in case the customer does actually buy instead of just leaving the site part way through)

Physical Data Flow Diagrams: An implementation-dependent view of the current system, showing what tasks are carried out and how they are performed. Physical characteristics can include. For e.g. Names of people

Logical Data Flow Diagrams: An implementation-independent view of the system, focusing on the flow of data between processes without regard for the specific devices, storage locations or people in the system. Maintain Consistency Between Processes: When developing DFDs in more detail it is important to maintain consistency between levels. No new inputs or outputs to a process should be introduced at lower levels that were not identified at a higher level. However, within a process, new data flows and data stores may be identified. Follow Meaningful Levelling Conventions: Levelling
refers to the handling of local files (those that are used within a process). The details that pertain only to a single process on a particular level should be held within the process. *Data stores and data flows that are relevant only to the inside of a process are concealed until that process is exploded into greater detail.* Add Control on Lower-Level Diagrams Only: The logical data flow diagrams developed to this point do not include control information. No mention has been made of how to handle errors or exceptions. Although this information is necessary in the final analysis, it should not be a concern while identifying the overall data flow. The secondary diagrams (below second or third level) show error and exception handling in the process being exploded.

Some physical control information is unnecessary in logical DFDs. Copy numbers or annotations for documents (e.g. Copy 1, Copy 2, Shipping copy or Accounting copy), procedural orders (e.g. Find the record; Review the record; Annotate the record), or day-of-the-week triggers (e.g. Do on Monday; Do the last day of the month) do not belong with the logical and data aspects of requirements determination. The important elements for understanding a process during logical data flow analysis are not document copy numbers but descriptions of the data needed to perform the process. Assign Meaningful Labels: The descriptions assigned to data flows and processes should tell the reader what is going on. All data flows should be named to reflect their content accurately.

Data Flow Naming: The names assigned to data flows should reflect the data of interest to the analysts, not the documents on which they reside. For example, an invoice contains many different elements of information. Analysts are concerned with the contents of the invoice that are important for a particular process. It may be the invoice number and date of issue or the signature or authorisation associated with the invoice. It is not the sheet of paper itself. Data flowing into a process undergo change. Therefore, the outbound data flow is named differently from the inbound one. Process Naming and Numbering: All processes should be assigned names that tell the reader something specific about the nature of the activities in the process. The names INVENTORY CONTROL, PURCHASING and SALE are too general to be useful in a logical DFD. It is much better to use ADJUST QUANTITY ON HAND, PREPARE PURCHASE ORDER or EDIT SALES ORDER to describe the process. Syntax and structural patterns of the elements of the description logic are exploiting the expressivity of the relevant languages.

The problems described from three perspectives: existing relevant work, formalization and implementation. First, it is shown that existing research results in the identified fields do not provide
enough evidence that semantic specification, interoperability and formal specification of systems (especially in collaborative environments, such as supply chain) can be achieved. While most of the work is focused to achieve these concepts of systems (actually, in most cases, certain levels of systems’ interoperability), the semantic interoperability must be considered as a new, under-developed topic. The attempt to formalize the notion of semantic specification and interoperability is made. This attempt clearly distinct between the notions of semantic specification and “traditional” interoperability, by taking into account the unified view to the interoperability of systems and available formalisms for conceptualization of the systems’ semantics, Formal definition of systems provide the basis for this approach and has significant influence on the choices made in the process of development of methodology for this work.

In order to implement and evaluate semantic specification, interoperability and formal specification enterprises’ realities have to be represented by relevant formal models. In the work on developing different formalisms for enterprise modeling (enterprise architectures, frameworks and database schemas) is presented. Realization of the interoperability value proposition has great impact to the development of new forms of the enterprise collaboration. These forms and associated notions are defined in the context of the issues related to the conventional enterprises’ networking, namely Supply Chain Management. There are already some existing formal models of the collaborative enterprises’ environment. However, these models are not considered as candidate for formal framework for semantic interoperability in supply chain networks, mostly because of lack of relevance. Thus, still there is a need for expressive, explicit, neutral and relevant formal model which will enable the partnering enterprises to exchange the information and services in the supply chain.

The approach is based on the premises that: 1) expressivity of one model can be achieved by selecting adopted industrial reference model for a semantic analysis; 2) explicitness of one model can be achieved by mapping induced enterprise concepts to the formally defined concepts of domain; 3) neutrality can be achieved by semantic enrichment, namely, synthesis of the recognized concepts; and 4) relevance can be achieved by maintaining the mappings between formal definitions of the enterprise concepts and implicit notions of the reference models, used or exploited by the relevant..

The programming interface for the research work is SAP, HTML, XML and or any other design tool to support to the research work.