

# Octopus/UML Sequence Reference

## Version 1.3

Farfarakis Rallis, Ziegler Jürgen  
Nokia Research Center

1.	AIM AND SCOPE .....	1
1.1	Phases and models.....	1
1.2	System Increments and Subsystems .....	2
1.3	System and Subsystem Development Sequence .....	3
2.	DETAILED PHASE DIRECTORY .....	5
2.1	System Requirements Specification .....	5
2.2	System Architecture .....	6
2.3	Subsystem Analysis.....	7
2.4	Subsystem Design .....	8
2.5	Subsystem Implementation .....	9
2.6	Subsystem Testing.....	9
2.7	Increment Integration & Testing .....	9
3.	BIBLIOGRAPHY.....	9

## 1. AIM AND SCOPE

### 1.1 PHASES AND MODELS

The development sequence structures and guides the progress of development work based on the artifacts. At the top level the sequence consists of ordered phases. Each phase has an aim which reflects to a stage within software development, from the initial problem conception to the final concrete software system. In order to achieve this, each phase requests to build at least one model. If within the phase several models are developed, then, each model is assigned a segment of the aim that this phase should accomplish. In subsequent, the models should collectively fulfill the aim of the phase. Each model defines a specific set of artifacts that satisfy the aim in the best possible way.

The development sequence conducts the work to proceed from one phase to the next. A phase defines a work package producing some artifacts that, in an ideal case, should be completed

before work on any of the artifacts of the next phase begins. However in praxis, work on subsequent phases may overlap. Starting the next phase is acceptable only when the artifacts of the earlier phase have been developed to a mature and stable level, thus providing the necessary inputs to the artifacts of the next phase.

In each phase a sequence guides the developer to organise the work of producing the artifacts of the models within the phase. In contrast to the possible serialisation of phases, the models inside a phase must always be developed in parallel. However in some phases, the models may be developed in a sequence, and thus, completion of at least one model will depend on completion of the previous ones. Inside a model, serialisation of artifacts is almost impossible. Completeness of one artifact depends on the completeness of others and vice versa.

**1.2 SYSTEM INCREMENTS AND SUBSYSTEMS**

In Octopus/UML, a system is decomposed into a number of software subsystems. For any domain of the system at least one subsystem is defined. If the domain is large or complex enough, it may be broken into more than one subsystem until each subsystem is a work package of reasonable size (Figure 1-1).

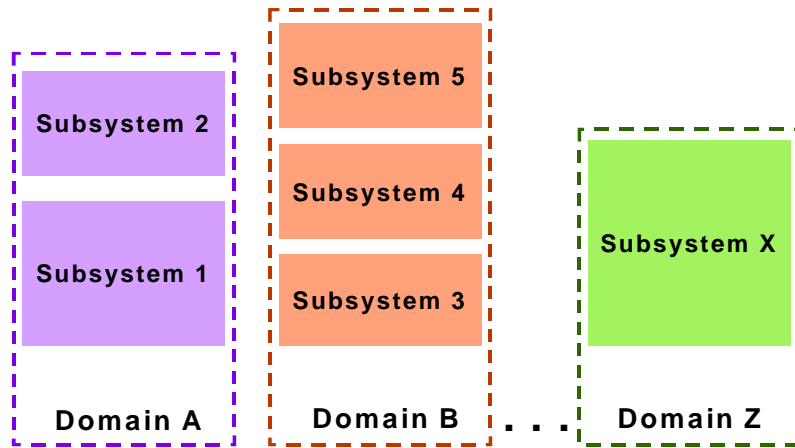


Figure 1-1. Subsystem Decomposition.

A System Increment is the result of developing some of the subsystems simultaneously, and integrating them with each other and with any former increments. Each System Increment is operational, but is incomplete, except for the final increment which will represent the full target system (Figure 1-2). The System Development Pattern defines the generic structure of the development of any such System Increment.

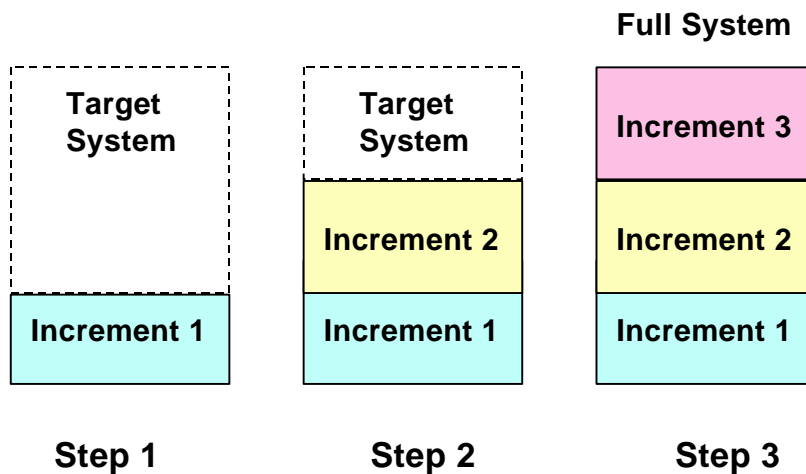


Figure 1-2. Increment Arrangement.

### 1.3 SYSTEM AND SUBSYSTEM DEVELOPMENT SEQUENCE

The development sequence covering the development of an entire system is called the system development sequence. Each project is invited to configure its own system development sequence depending on its individual needs or preferences. However, a standard System Development Pattern can act as a guide when devising those sequences (see Figure 1-3). This pattern provides a standard structure, but it also allows flexibility where adaptation to a project's needs is advantageous.

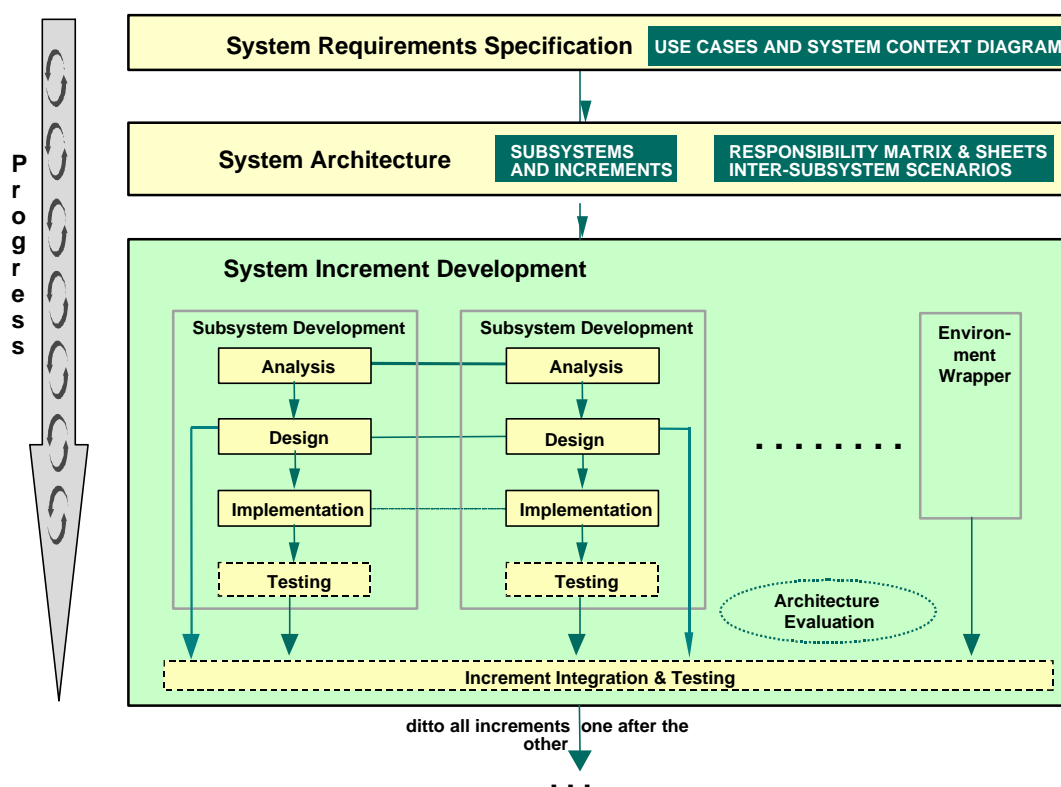


Figure 1-3. System Development Pattern.

According to the System Development Pattern, the system development sequence always starts with two system wide phases, the System Requirements Specification and the System Architecture. After the System Architecture, a series of system increments follows. Each system increment covers a number of subsystems as specified by the System Architecture. For each subsystem the same Subsystem Development Sequence is applied. See Figure 1-4.

The System Development Pattern allows the arrangement of a system development sequence where subsystems are potentially developed in parallel inside a system increment, and/or in series from one increment to the next. The parallel arrangement enables the effective use of more resources than one subsystem requires and it will speed up the project in calendar time. The serial arrangement can be used to achieve incremental and/or evolutionary development. If the subsequent system increment covers subsystems not touched previously, the system is developed by adding more and more subsystems until it is complete, namely the incremental approach. If the same subsystem is covered more than once in a system increment, this subsystem is developed in cycles until it is complete, namely the evolutionary approach.

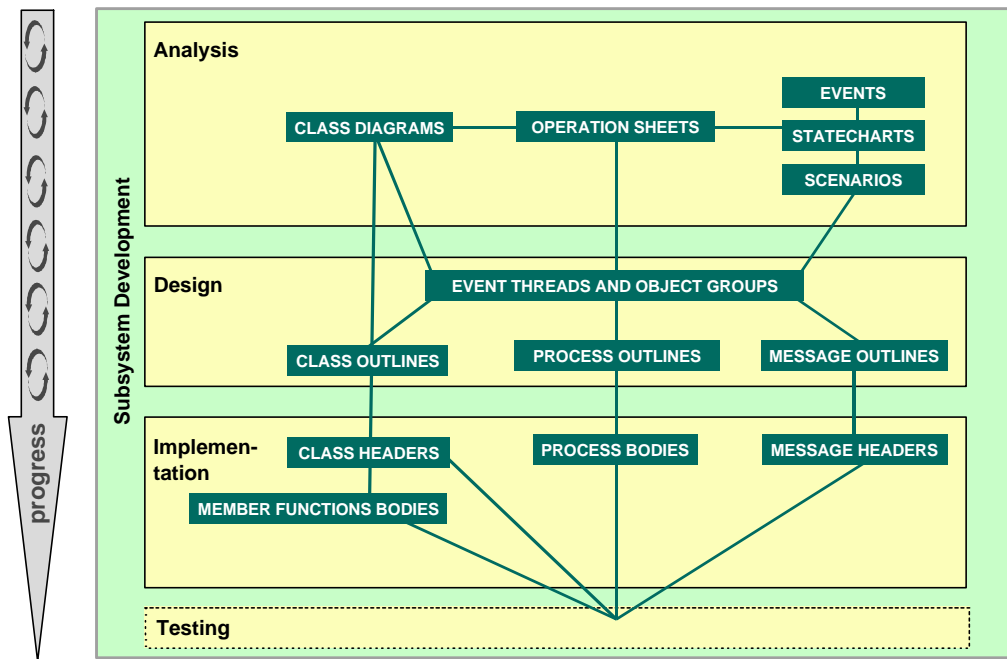


Figure 1-4. Subsystem Development Sequence.

Please note that in the evolutionary approach the project repeats the Subsystem Development Sequence of the same subsystem. Thus, the project may reconsider all decisions made earlier. In practice, each project should find its own optimal combination of all the alternatives, since they all have pros and cons.

Figure 1-5 shows an example of such a practical combination. Only subsystem 1 is developed in the evolutionary way. Its first cycle is done in increment 1 together with subsystem 3. When executing development of increment 2, subsystem 1 is recycled and subsystem 4 and 5 are developed following the incremental way. Finally, subsystem 2 is added to complete the system.

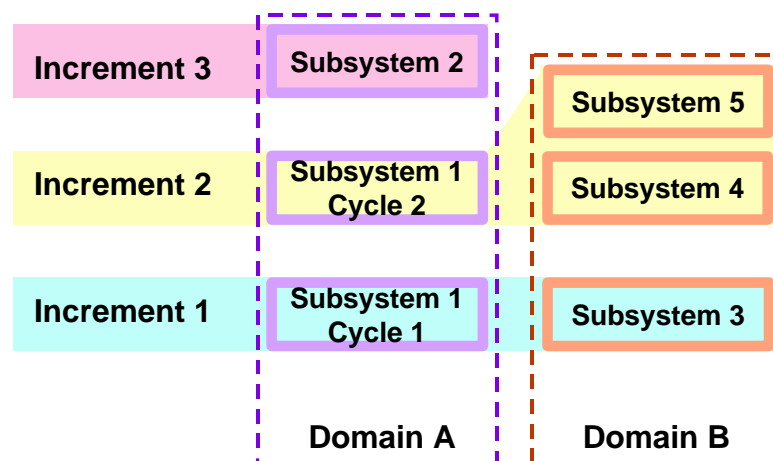
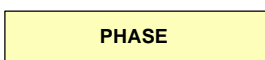

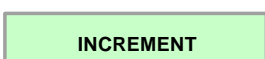
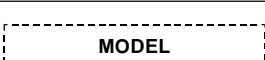


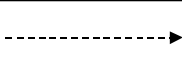
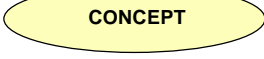


Figure 1-5. Arrangement of the development work.

## 2. DETAILED PHASE DIRECTORY

The detailed phase directory lists and explains each phase in chronological order, displaying a detailed sequence for each phase. The notation that is used to describe the sequence of each phase is presented in Table 2-1.

Table 2-1. Phase Sequence Notation

	<b>Phase: Yellow box with name inside</b>
	<b>Artifact: Dark green box with name inside</b>
	<b>System Increment: Light green box with name inside</b>
	<b>Model: White dotted box with name inside</b>
	<b>Facility: Medium green with name inside</b>
	<b>Solid Arrow: Recommended flow</b>
	<b>Dotted Arrow: Optional flow</b>
	<b>Ellipse: Concept not related to the ones</b>

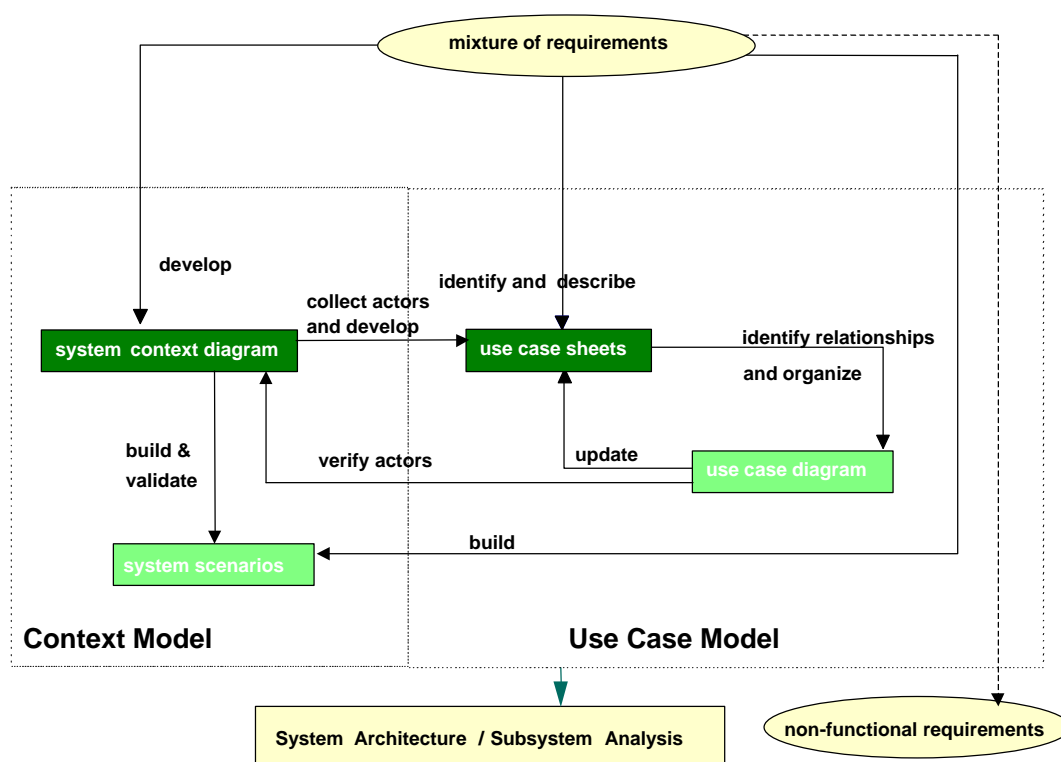
The following sections represent the workflow of the system requirements specification, the system architecture and the system increment development phases.

### 2.1 SYSTEM REQUIREMENTS SPECIFICATION

The aim of the system requirements specification is to restructure the requirements of the target software system, so that their representation supports the development of software. The requirements developed at the product level do not focus on software development, making them unsuitable for an effective start.

However, it is important to notice that system requirements specification, as defined here, is not intended to make the developer fully confident about all the requirements. This phase will uncover the target software system with two limited goals in mind. Firstly, product-related persons except the software developers, should be able to check if the software development is on the right track. Secondly, enough information should be gathered, so that the subsequent system architecture is able to make a sensible decomposition of the system into software subsystems. Elaboration of the requirements, from this phase till the subsystem analysis phase, must be achieved.

## Sequence:



## 2.2 SYSTEM ARCHITECTURE

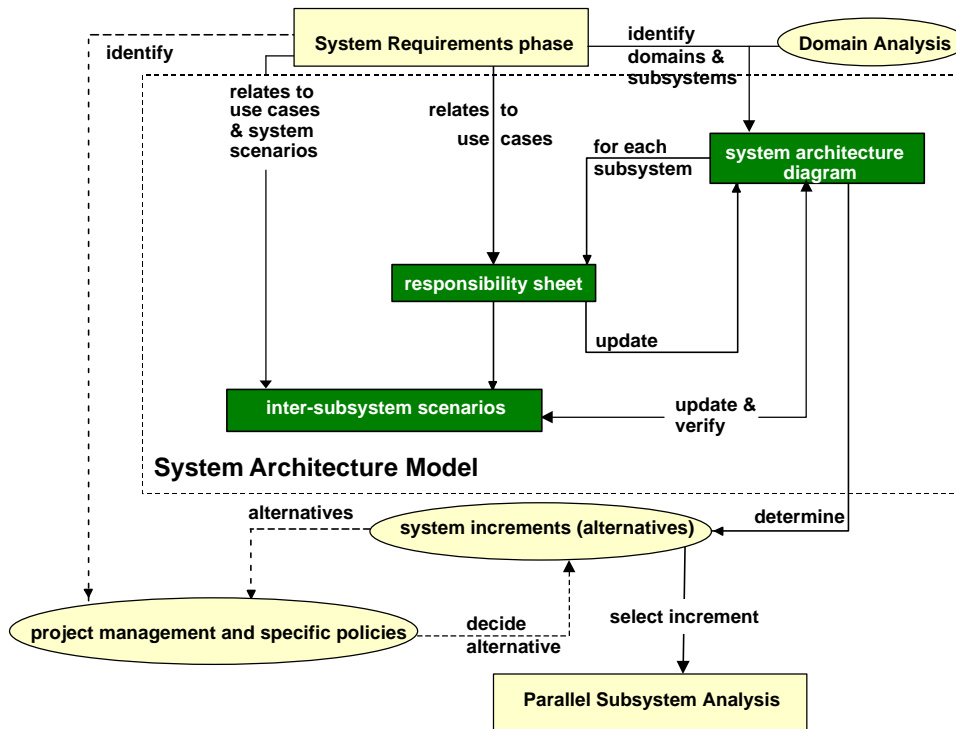
The aims of system architecture are to decompose the system into software subsystems, to describe each subsystem as a black box and to identify the client-server relationships between these subsystems. This work is mainly based on the knowledge gained in system requirements specification. Application domains are the major decomposition criteria. A domain represents “*a separate real, hypothetical or abstract world inhabited by a distinct set of objects that behave according to the rules and policies characteristic of the domain*” [1]. Since domains are independent, system decomposition based on domains results in software subsystems that are loosely coupled and potentially distributed. Each subsystem also initiates its own work package representing the work breakdown of the system and, therefore, it can be used for a decomposition according to size, availability of resources and schedule. The decomposition of the system based on domains, promotes the concept of layered architecture.

During the system architecture phase, further elaboration of the system requirements takes place. The completion of defining the requirements is concentrated in the analysis phase of each subsystem, where development is achieved in smaller packages at an early stage, following a parallel development track and using the advantages, throughout each increment, of the incremental and evolutionary models. This combination of techniques, outweighs the risk of making a non-optimal initial decomposition.

It should be noted that each system increment contains an informal system architecture, which whenever it seems appropriate, can take corrective actions on the architecture of the remaining subsystems.

System requirements specification describes the system in terms of use cases while system architecture describes the system in terms of subsystems. In order to ensure consistency and quality, system architecture demands a mapping of the use cases into subsystems and vice versa. If this mapping is hard to find, or there is a one to one mapping almost everywhere, the development work done is questionable.

**Sequence:**



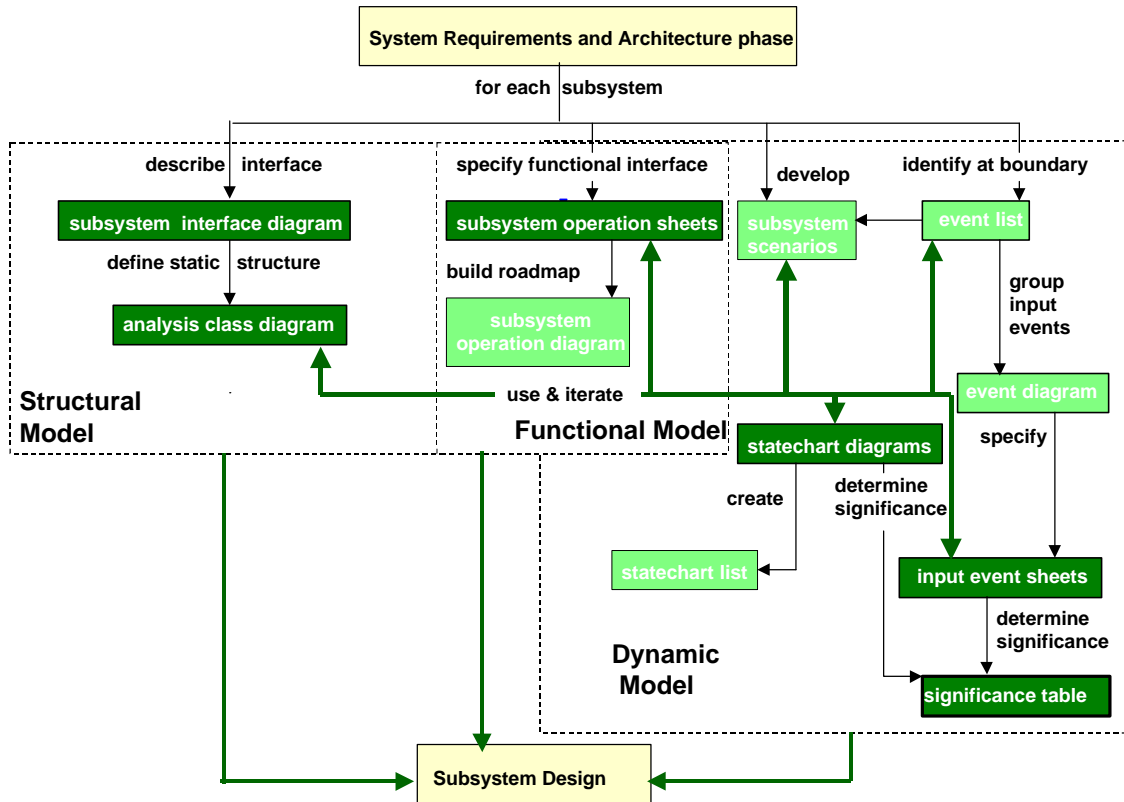
### 2.3 SUBSYSTEM ANALYSIS

The purpose of the subsystem analysis is to clarify and to document what the subsystem is supposed to do. The artifacts to be produced, achieve that purpose and, thus, serve as a basis for a contract between the work the subsystem development team is responsible for, and the interfaces between this subsystem and the other subsystems. Approving these artifacts means that this contract has been established. Therefore, all other subsystem development teams must carefully check these artifacts and vice versa.

Based on the knowledge gathered in the preceding system architecture, subsystem analysis demands to build the structural, functional and dynamic model. These new models are more detailed, technical and structured than the given inputs. However, they should focus on what the subsystem is supposed to do, and not how it should do it. The internal function and structure of the subsystem is deferred to the design of the subsystem, with the exception of the structural model, which defines the classes (concepts) of the subsystem and their associations.

Each model defines a set of artifacts to be produced. All three models are built in parallel. Nonetheless, the starting point and order of progress should be planned and agreed on. Inside the models some partial order exists.

**Sequence:**



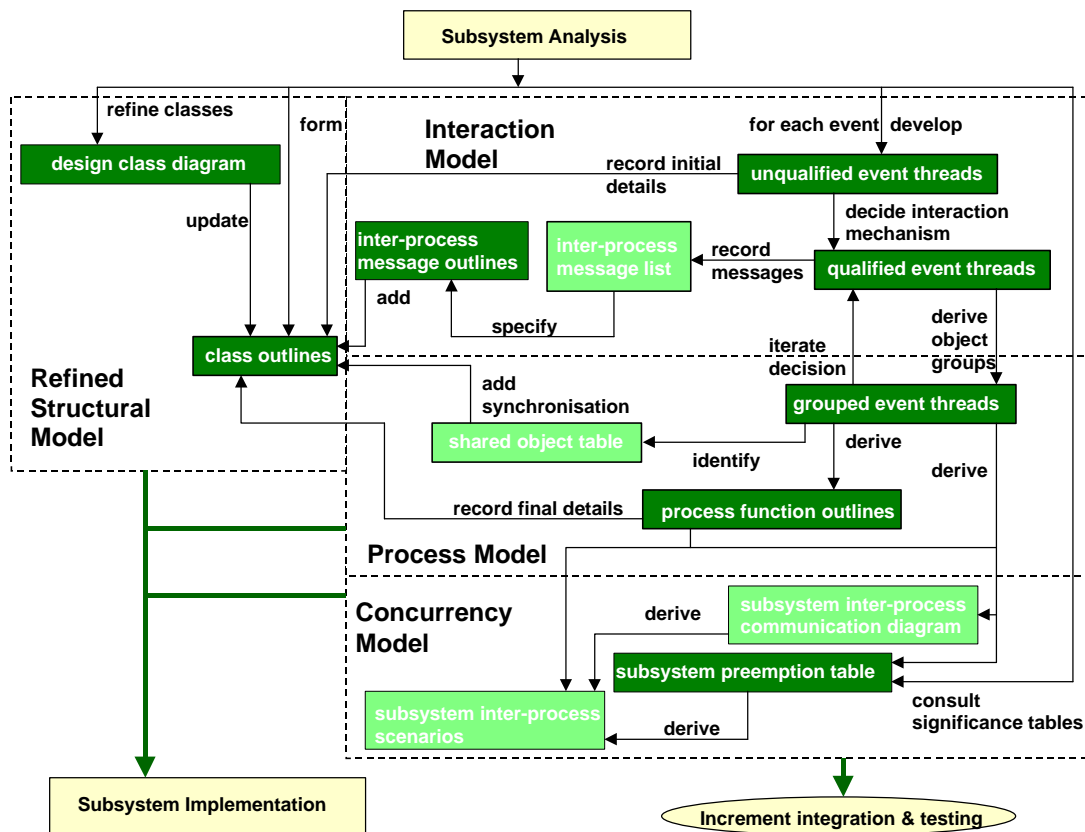
## 2.4 SUBSYSTEM DESIGN

The subsystem design produces a description of how the requirements stated in the former models are achieved. This is the phase where the controlled transition is made from the implicit concurrency model to the explicit concurrency model. Here objects are mapped to processes of a traditional operating system.

The knowledge acquired in the previous development phases is used to construct the four models of the subsystem design phase: the refined structural model, the interaction model, the process model and the concurrency model. All four models capture, through the artifacts, important information regarding messaging techniques (synchronous vs. asynchronous), concurrency, timing constraints, object grouping, transition from objects to processes and performance optimisation.

The aim is to produce a comprehensive software specification one level above implementation. Requirements derived from the chosen operating system and programming language should be considered but they should not drive the subsystem design.

## Sequence:



## 2.5 SUBSYSTEM IMPLEMENTATION

The subsystem implementation translates artifacts, such as the design class diagram, the class and process function outlines produced in the subsystem design phase, into code of the selected programming language and operating system. Since this translation is straightforward no artifacts are required.

## 2.6 SUBSYSTEM TESTING

The subsystem testing verifies the code produced in the subsystem implementation phase. The definition of artifacts for this phase is under development.

## 2.7 INCREMENT INTEGRATION & TESTING

The increment integration & testing merges the results of subsystem design and subsystem implementation, and verifies the increment as a whole. The definition of artifacts for this phase is under development.

## 3. BIBLIOGRAPHY

- [1] M. Awad, J. Kuusela and J. Ziegler, *Object-Oriented Technology for Real-Time Systems*. Prentice Hall, 1996.