

COMET - Overview of the COMET methodology

<!-- -->

Table of contents

1 Model architecture.....	2
1.1 Recursive system architectures.....	3
1.2 PIMs vs PSMs.....	6
2 Development activities.....	6
2.1 Models and process activities.....	8
2.2 Summary of the process.....	8
2.3 Work contexts.....	10
3 Development phases.....	10
3.1 Inception phase.....	11
3.2 Elaboration phase.....	11
3.3 Construction phase.....	12
3.4 Transition phase.....	13

1. Model architecture

The component centre development process is model driven. There are four models involved: *The Business Model, the Requirements Model, the Architecture Model and the Platform Specific Model*. These models contain work products that provide the viewpoint specifications for the component-based system or the individual component that is being developed. The models with their work products are the outcome of the system development activities.

Figure 1 shows the model architecture and depicts the relations between the model world and the real world (where the COMBINE reference architecture represents the developed system).

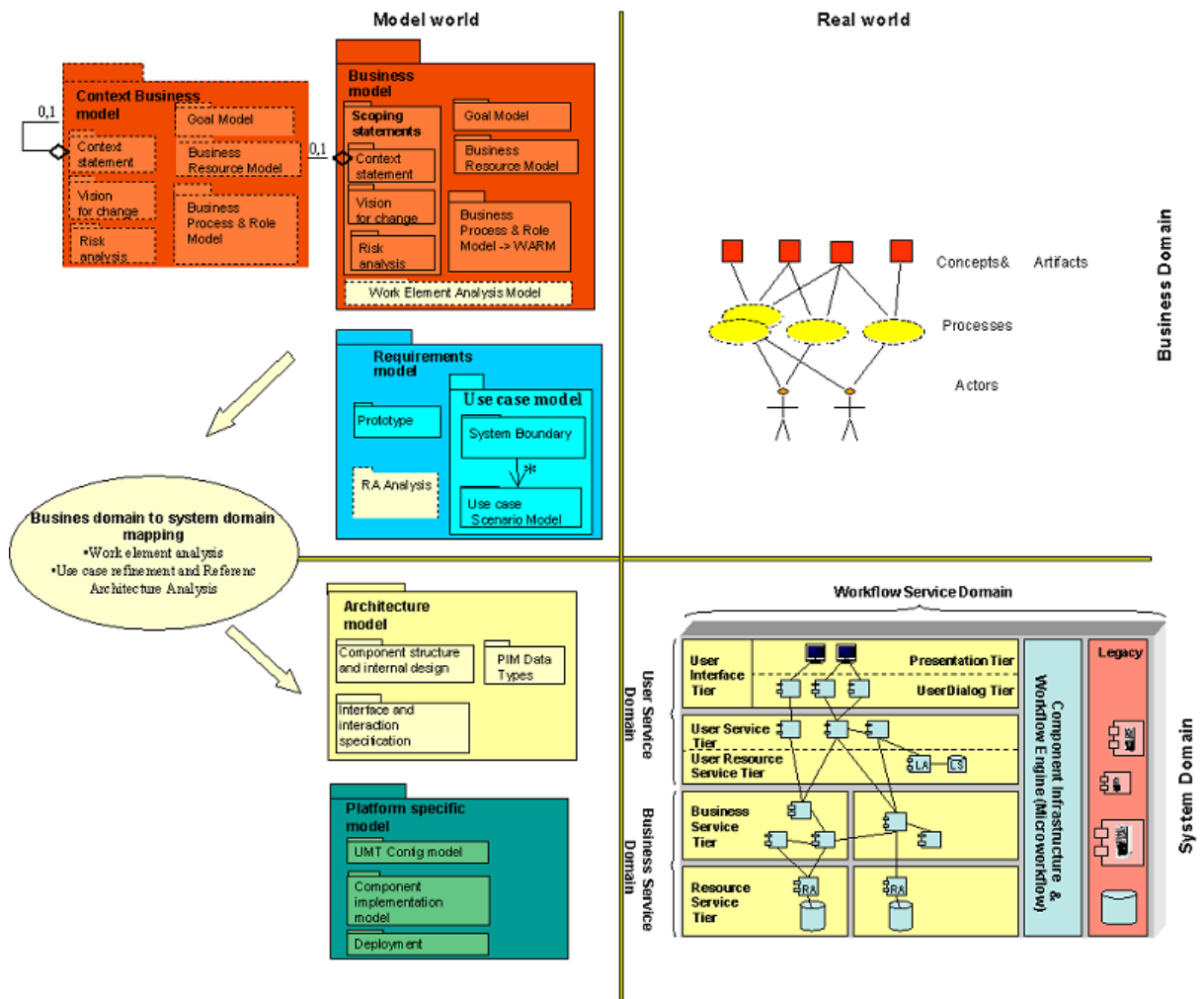


Figure 1: Model Architecture

1.1. Recursive system architectures

The target of consideration during modelling is often referred to as “system”. “System” is used to denote entities on many levels, from virtual enterprises where clusters of businesses compose the “system” down to software objects where collections of data types and operations on these constitute the “system”. These system levels are related. A virtual enterprise consists of interacting businesses and a business consists of interacting actors and technical systems. A software system, one kind of technical system, consists of software components, a software component consists of interacting software objects, and, finally, a software object consists of data types and operations on these. The levels mentioned here are well-established levels, but these are not the only levels that exist. In fact, one can have an arbitrary number of levels by using recursion. For instance, a business may consist of businesses or a technical system may consist of technical systems. Figure 2 shows the common system levels. The curled arrows pointing back to the same level illustrate the recursion.

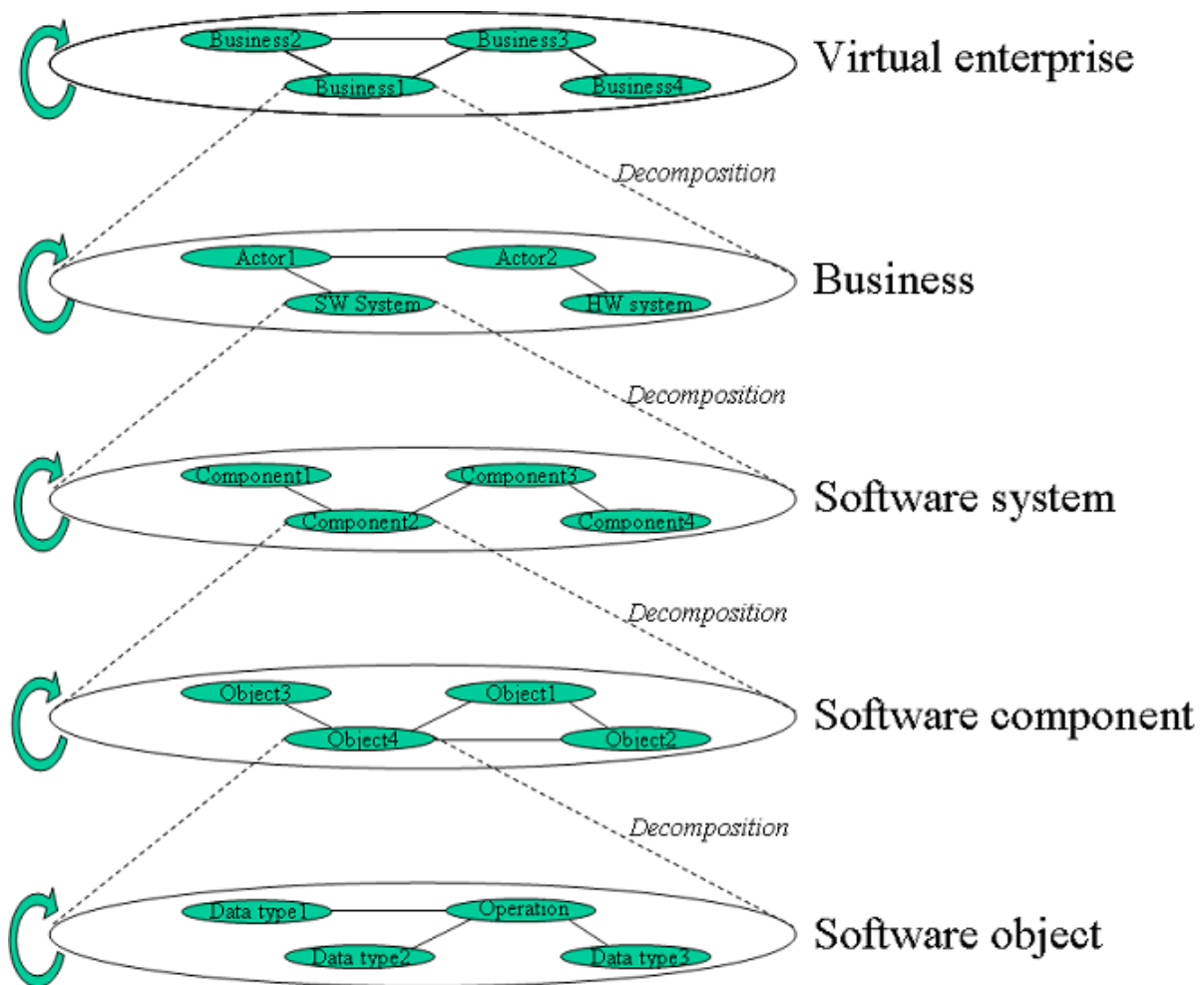


Figure 2: System Levels

For software development, two very important levels are the business level and the software system level. The business level contains human actors and technical systems, some of which are software systems. Models on the business level involving a software system depict the intended use of this software system by the actors. In other words, the collection of business models that involve a software system gives the functional requirements for that system. Models on the business level are therefore of paramount importance in software development to ensure that business needs are catered for by the software systems. However, if these models are not related to the software system level they lose much of their value. The software systems in the business models should be decomposed into a configuration of interacting software components so that the software system requirements are met by this configuration. The properties of the software system as specified at the business level (i.e., the requirements) should be at least the same as the properties of the chosen configuration of software components on the software system level. By “at least the same as” we mean that any property specified at the business level must also be a property of the configuration of software components. This configuration may however have additional properties that are not specified at the business level. In other words, a software system may do more than specified in its requirements.

Using the system hierarchy model presented above, we assign the software system to be developed as the target system. The context system for the software system is the business, which should be described in the business model and addressed in the requirements models. The software system itself is represented as an application component. The building blocks of the application component are software components. The software components themselves can be further decomposed recursively in terms of smaller-grain components. Figure 3 shows a typical application structure consisting of tool, business service and resource service components. The tool can be further decomposed into user interface and user service components.

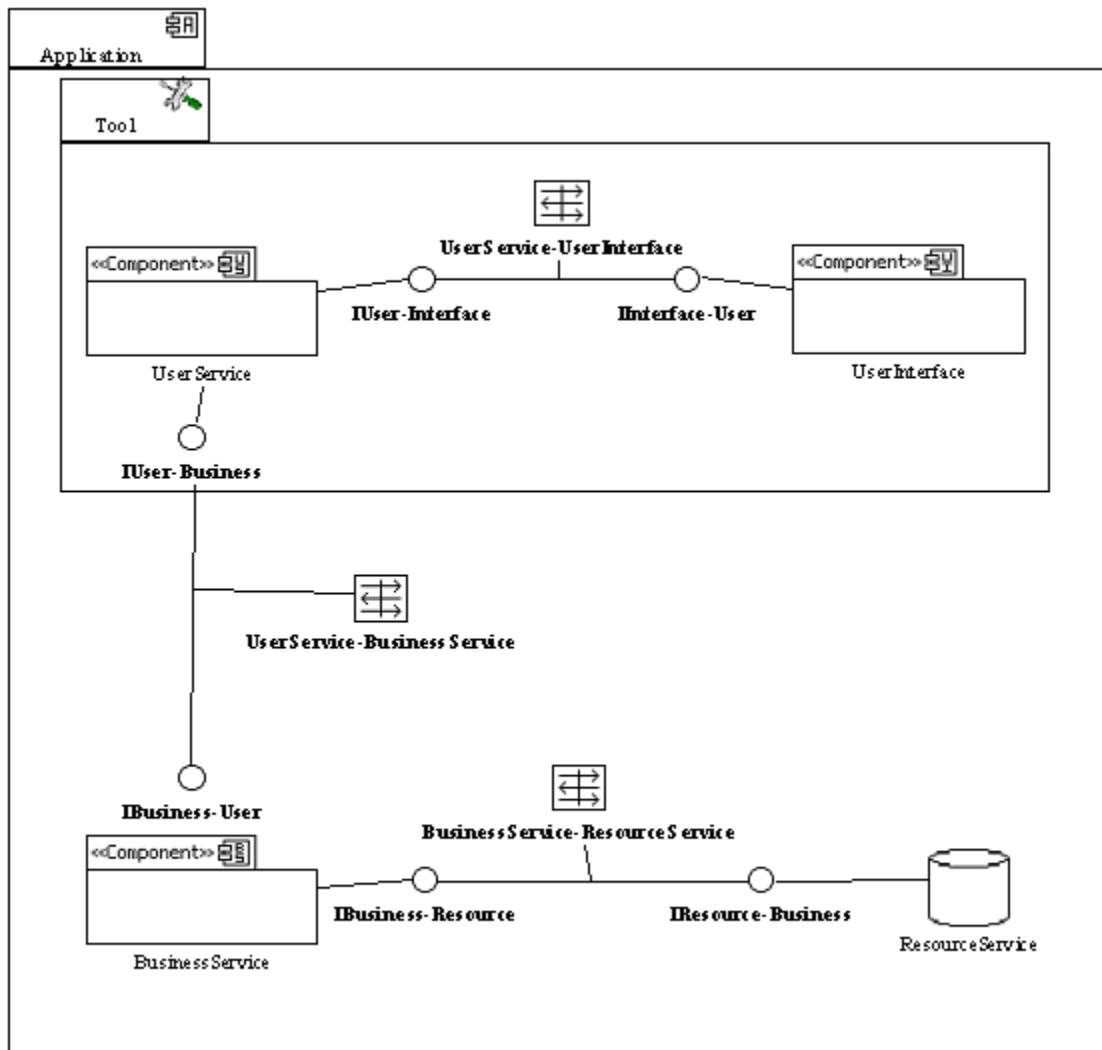


Figure 3: Application structure

In developing new applications one should ideally be able to use or reuse existing software components. The recursive structure allows reuse at various levels of refinement. In order to be able to compose systems from components, one must have a clear description of the collaboration between the components available for reuse.

With relationship to the similar recursive nature of the Kobra methodology, our experiences has shown that it can be useful to specialise the use of notations and techniques for different system levels.

The current focus of the COMET methodology is from the Business system defining the context for a software system, and for the first level of components for that system. For the

Business system level we have found it useful for business people to have a business process and activity focus rather than a component collaboration focus, and for the system specification we have found it useful to utilize the principles of use case modelling.

1.2. PIMs vs PSMs

A platform-independent model (PIM) is expressed in terms of business, requirements and architecture models. COMET defines two types of platform-independent models:

- A specificationally complete PIM defines a complete model of the system specification – the external architectural structure and behaviour – of a component system in terms of a business model, a requirements model and an architecture model as defined by COMET.
- A computationally complete PIM which adds to a specificationally complete PIM a definition of the system realization – the internal design structure and behaviour – of a component system in terms of a design model. The design model is expressed using an action semantics language.

The term PIM in COMET is typically used to mean a specificationally complete model. Currently, COMET does not advocate a design model expressed in a programming language independent action semantics language. Instead, the algorithmic logic of a component is implemented in a given (set of) programming language(s) as part of the platform-specific model.

2. Development activities

An overview of the development process for a Product is shown in Figure 4. This diagram shows the activities performed, and the models produced. This approach is very similar to the Unified Software Development Process.

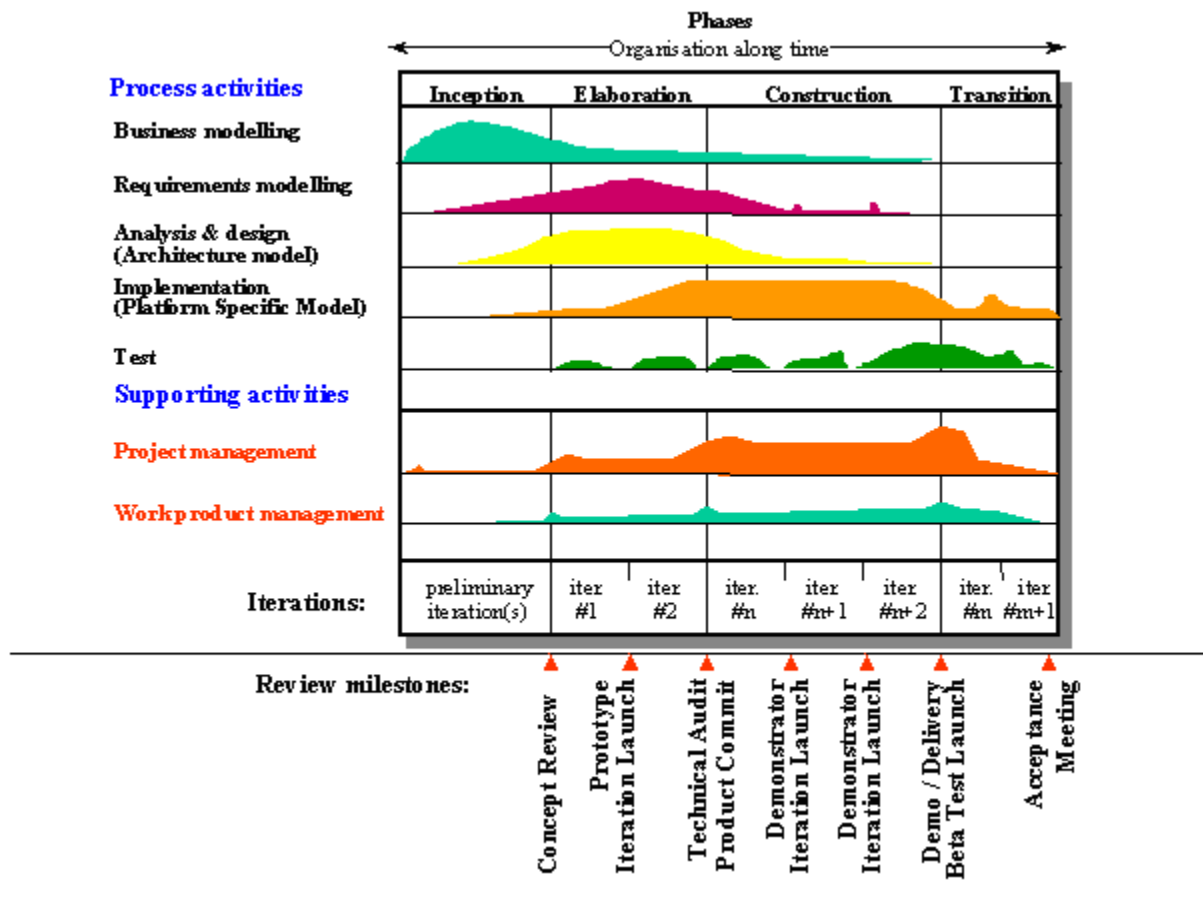


Figure 4: The activity view of development process model

There are four phases in the development process of any Product of the Component Centre: the Inception Phase, the Elaboration Phase, the Construction Phase and the Transition Phase. The completion of a phase means that the Product under development has reached a certain degree of completeness and thus represents a major milestone of the project. The milestones are shown at the bottom of Figure 4.

There are two kinds of activities in the Development Process: process activities and supporting activities. Process activities are activities that are directly related to the system development tasks. They are requirements capture, analysis & design, implementation and test. Supporting activities are activities that support the process activities to ensure that they are carried out effectively and efficiently. They are project management and work product management

The completion of each phase represents a major milestone of the software development process.

The relative importance of the process activities varies during the life cycle of a development

project. In early phases, business modelling, requirements specification and architecture level design tend to dominate, while in later phases the majority of the work is on component design, implementation and testing. This is illustrated in Figure 4, where the curves indicate the effort on each activity as a function of time.

Note that the figure shows an example. Exactly how these curves will look depends on the type of project. In more explorative projects for instance, where the requirements and architecture is difficult to stabilise early, significant effort on requirements analysis and architectural design may persist all the way to the end of the project. In more straightforward projects, on the other hand, these activities will be more or less completed after the first two phases.

2.1. Models and process activities

The models described in Section 1.1 define the work products of the Development Process activities identified in Figure 4.

The Development Process approach is to be understood as a pattern, in the sense that it defines a set of guidelines with ample room for variation, rather than a fixed recipe that can be followed slavishly. This is necessary since each company and each project is unique and must be able to tailor the model to its specific needs.

In principle, all the models can be considered during every phase and iteration, with the exception of the Transition Phase, where the Business Model and the Requirements Model should be stable: if new requirements are identified at this stage of the development process, for example because of changes in the business, it is recommended that these are elaborated in another project. Nevertheless, all the work products are not necessarily considered during each phase. For instance should the overall architecture design become stable during the Elaboration phase there would be minor changes on the this model during the construction phase. Remember that each iteration is an increment of the final product under development.

2.2. Summary of the process

The Component Centre Development Process involves building a set of models and their associated work products, following the iterative and incremental process paradigm. Except from some work products in the business model and the other requirement work product, the work products are UML-based, and each work product is presented with one or more UML diagrams. Figure 5 depicts this and shows the Development Process in a nutshell.

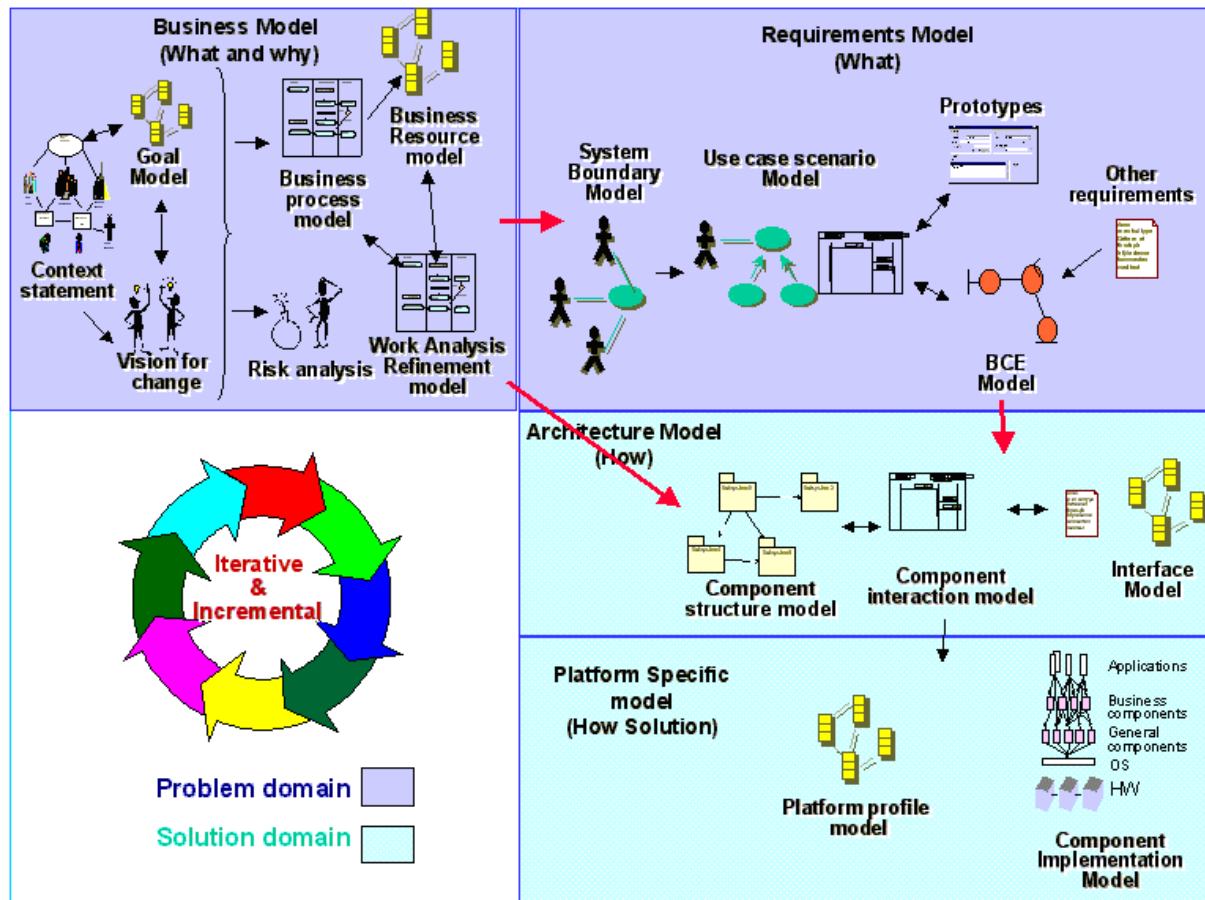


Figure 5: Component centre methodology in a nutshell

The figure shows all Component Centre development process work products. The icons indicate the associated UML diagram(s) or nature of deliverable for each work product and the arrows show the most common path through the set of work products within an iteration.

Starting from the upper left we have the Business model which includes the context statement, vision for change, and risk analysis as well as more formal models of the goals, processes & roles and business resources. The icons indicate use of UML class diagrams for modelling goals and business resources, and UML activity graphs and collaborations for modelling business processes and roles. The Work Analysis Refinement model (WARM) is a refinement of the business process & roles model to identify the required behaviour of the Product under development.

In the Requirements model, the use case diagrams are associated with both the system boundary model and the use case scenario model. UML sequence diagrams or activity graphs are used for detailing the use case scenarios. Prototypes might be developed for various reasons, such as HCI mock-ups, as a vehicle to get fruitful interaction with end users, or for testing the architecture etc. The Other Requirements work product is typically

expressed in text. The BCE model is modelled using UML class diagram with the boundary, control and entity stereotypes.

In the Architecture model, the component structure model uses the package concept of UML and UML class diagram. The component interaction model defines the component interaction using UML sequence diagram and/or UML collaboration diagram. The interface model specifies the interface signatures, pre and post conditions for the operations and the abstract information model represented by the interface. These are modelled using UML class diagram, as well as using prose text and/or OCL.

The platform specific model is typically expressed using UML class diagram and programming languages as well as documentation in various forms (user manual, reference guide, comments within the code etc).

2.3. Work contexts

Four different work contexts are defined to aid a system developer using the COMBINE Component Centre approach.

1. The Business Modelling work context (BusinessModule) defines tool support for developing business models according to the concepts defined in the business metamodel.
2. The Requirements Modelling work context (RequirementsModule) defines tool support for developing requirements models according to the concepts defined in the requirements metamodel.
3. The Architecture Modelling work context (ArchitectureModule) defines tool support for developing architecture models according to the concepts defined in the base component metamodel and the architecture metamodel.
4. Finally, for each specific platform there should be defined a Platform-specific design modelling work context (PSMDesignModule) for implementing the platform-specific model. As much technology specific infrastructure as possible should be automatically generated and hidden from the design developer.

In order to develop a complete PIM, it is essential that the UML modelling tool supports the three first work contexts. The fourth work context should ideally be provided as part of an integrated development environment (IDE) supporting both UML modelling and programming language implementation, but can also provided as a third-party tool.

In COMET, the platform-specific design modelling work context is supported by code generation (in UMT) and IDE tools integrated with the UML modelling tool.

3. Development phases

The COMET methodology addresses how to develop a new Application Component. This might be when you:

- want to automate tasks and processes that until now have been executed manually

- want to perform tasks and processes in a new way, e.g., based on a Business Process Reengineering activity

The suggested procedure for developing a new Application Component concerning the four phases is described below.

3.1. Inception phase

1. Interview users and key stakeholders, study any competing solutions that might exist.
2. Unless one already exists, develop an as is Use Case Model based on information gathered from step 1 (one iteration, focus on System Boundary Model part).
3. Derive the context and identify the use cases and actors using the use case technique. Make a first version of the to be System Boundary Model:
 - Context statement, (derive what is necessary. An informal rich picture or a full Context Business Model)
 - Identify actors
 - Identify use cases and associate extra requirements
4. Make a first draft of the:
 - Vision for change
 - Risk analysis (concentrate on identifying and analyze the business risks at this stage)
 - Goal model, (link to top level process(es))
5. Identify the main business resources (Business Resource Model)
6. Start deriving the business processes (Business Process Model). This step and the one above are conducted together:
 - Identify and study the main business processes
 - Identify which activities are performed entirely by a human, by a human supported by a tool, or entirely and automatically by the Product being developed (as part of the system). This is known as Work Analysis Refinement modelling (WARM)
7. Make a prototype if necessary to:
 - Sell the idea
 - Communicate with stakeholders (e.g. discuss requirements)
 - Reduce business risks
8. Align the Business Model and the System Boundary Model
9. Derive Concept Deliverable including the project proposal
10. Iteration review

Iterate the steps in this phase until the business risks are under control (normally one or two iterations) - or Stop if there is a decision from a review not to proceed.

3.2. Elaboration phase

1. Refine the Vision for change, Context statement and Risk analysis (focus on technical risks),

2. Refine and detail the use case model using the use case template
3. Update and refine the Business Process Model and Business Resource Model if needed:
 - Derive and update the relationships between the Use Case model and the Business Model (goals, activities and resources)
4. Start on identifying subsystems and components either by:
 - Organize the use cases into subsystems (application, tool and business service components (typically in that order) applying the reference architecture analysis:
 - Or identify the tool and user tier components using the use case model, and the other COMBINE components applying the work element analysis
5. Derive the overall static architecture/component structure (Component Structure and Internal Design) based on previous step and extra requirements (quality of service requirements). Extra requirements are picked from the Use Case Model (use case template)
 - Additional strategies for subsystem design might also be used.
6. Start on deriving the interfaces of the components and the component collaboration (Interface and Interaction Specification) either by:
 - Specify the interface contract (operations, protocol and semantics) as well as the abstract information model (entities) visible through the interface by analyzing the use-case scenario descriptions. The entities might also be identified from the Business Resource Model.
 - Specify the Interface contract and the entities using the work element analysis.
7. Make prototype(s) (might evolve to be the first increment of the system) to:
 - Test the architecture
 - Reduce technical risks
 - Communicate with stakeholders (e.g. discuss requirements)
8. Consolidate the results from the steps of the phase to prepare the Elaboration deliverable including the Product Development Plan defining the increments for the Construction Phase (risk driven - use case or a couple of use cases per increment).
9. Iteration review

Iterate the steps in this phase until the technical risks are under control (normally two or three iterations) - or Stop if there is a decision from a review not to proceed.

3.3. Construction phase

1. Review the Construction Authorisation (for the first increment) or the Product Evaluation (for the subsequent increments).
 - Requirements management (store, prioritize and maintain)
 - Model updates (Business Model and Requirements Model)
 - Product development plan updates (risk driven increment plan, use case or a couple of use cases per increment)
2. Evolve the Interface and Interaction Specification for the current increment

3. Design the internals of the components. Use BCE or other design/architectural patterns.
4. Revise the Component Structure if necessary
5. Implement increment:
 - UMT configuration
 - Code generation
 - Implementation
 - Plan and test deployment
6. Integrate and test Product increment.
7. Product evaluation (Increment review)

Iterate the steps in this phase until the requirements are met. Deliver the planned increments - or Stop if there is a decision from a review not to proceed. Iterations are determined by the increments defined by the Product Development Plan.

3.4. Transition phase

1. Review Transition Authorization (for first iteration) or Acceptance Evaluation for subsequent iterations and decide Product modifications required.
2. Implement Product modifications
3. Deploy Product at designated site. Integrate and test
4. Iteration review

Iterate the steps in this phase until the system is accepted (deployed and stable).