

COMET - Service architecture metamodel

<!-- -->

Table of contents

1 The COMBINE 4+2 Tier Reference Architecture.....	2
1.1 Component Types.....	3
1.2 Type Libraries.....	6

1. The COMBINE 4+2 Tier Reference Architecture

The basis of the Combine component architecture is a generic system architecture, or reference architecture. The reference architecture defines a set of logical tiers, each of which consists of a set of components (Figure 1).

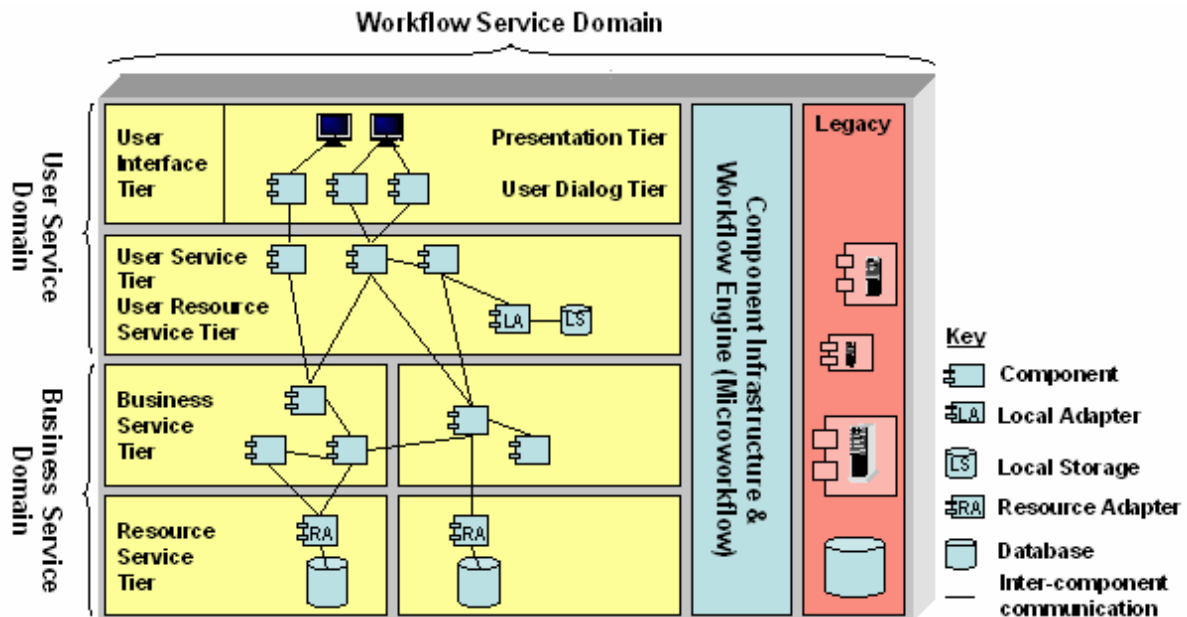


Figure 1: The 4+2 tier architecture

The 4+2 tier reference architecture is separated into a local single user-space, called the user service domain, and a shared transactional business-space called the business service domain. As Figure 1 indicates, the user service domain consists potentially of four local tiers and the business service domain consists of two shared tiers. The 4+2 tiers are as follows:

- **The user interface tier** provides presentation and user dialog logic. Sometimes, it is useful to make the presentation and user dialog separation explicitly, in particular to support reuse of user dialog on multiple platforms with different graphical capabilities, e.g. Web, PDA and Mobile phones.
- **The user service tier** provides the user's model, which may include user session logic and user-side representations of processes and information. It is an abstraction for a set of business services, making the business service provision (and the communication mechanisms) transparent to the user interface tier.
- **The user resource service tier** provides local persistence services. It is only present if persistence capabilities are required by components in the user service tier, e.g. to support disconnected or off-line operations, or to provide smart caching mechanisms. Access to the local storage is provided by local adapters, which typically are APIs to file storage or light-weight single-user databases.

- **The business service tier** provides components that represent business functionality and pervasive functionality (vertical vs. horizontal services). This tier provides enterprise-level services, and is responsible for protecting the integrity of enterprise resources at the business logic level. Components in this tier can be process-oriented, entity-oriented or workflow-oriented. For performance reasons, entity-oriented components are typically not exposed outside of this tier.
- **The resource service tier** provides global persistence services, typically in the form of databases. Resource adapters (e.g. JDBC or ODBC drivers) provide access, search and update services to databases and its data stored in a database management system (DBMS) like Oracle or Sybase.

1.1. Component Types

The figure below shows the component types of the architecture.

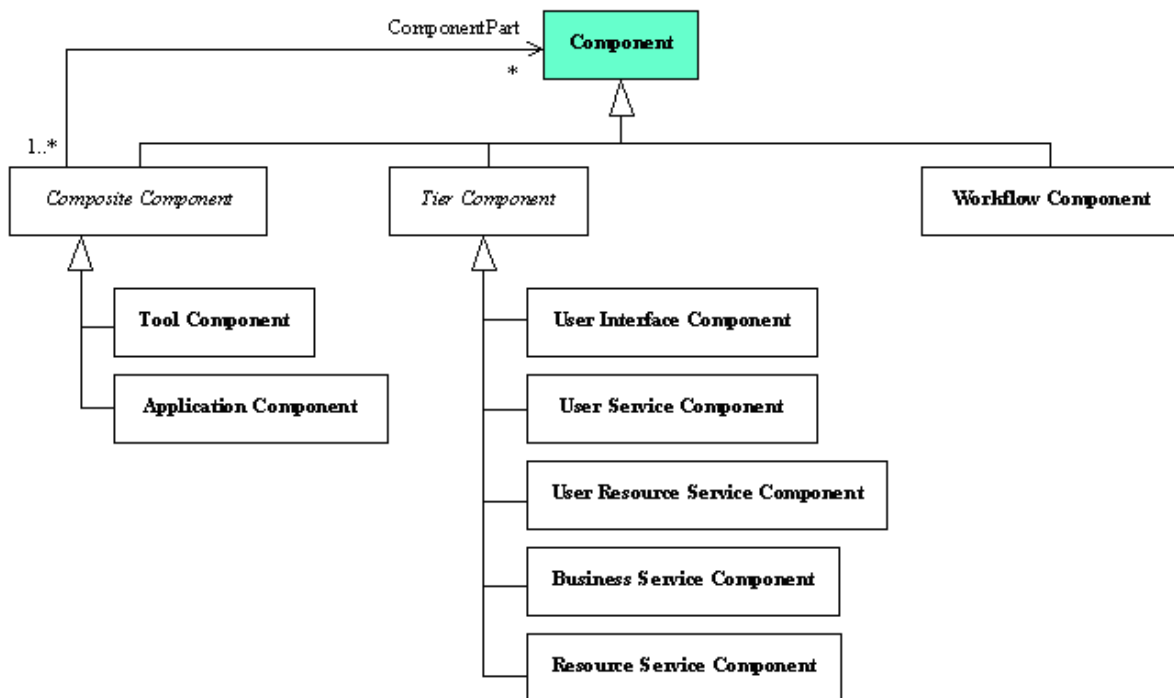


Figure 2: Component types

We have two abstract component types, tier components and composite components.

1.1.1. Tier Components

The tier components are the smallest building blocks components that reside in their corresponding distribution tier.

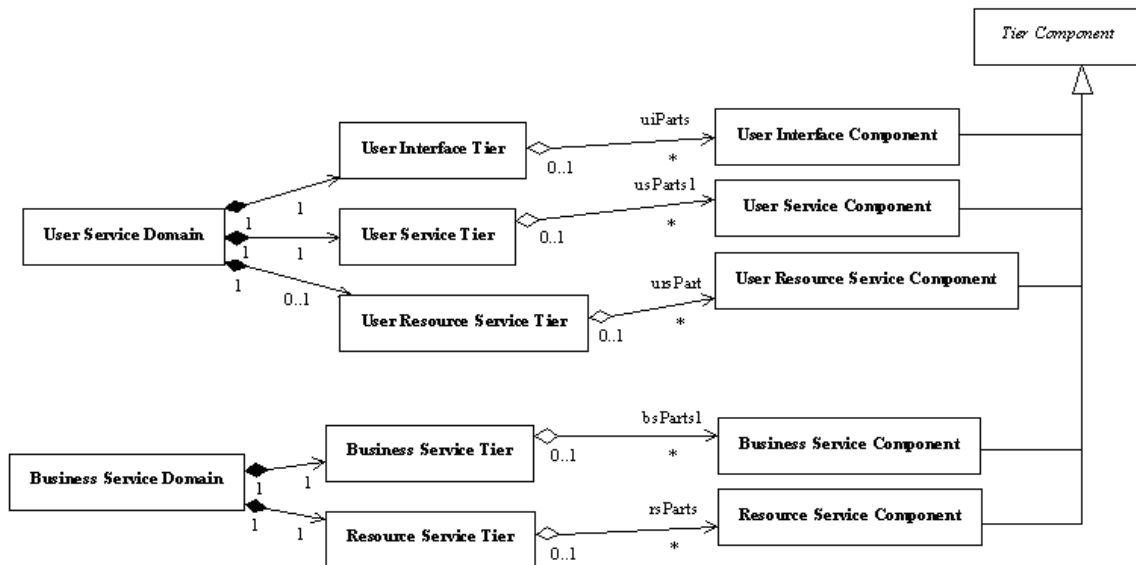


Figure 3: Component distribution

Tier components interact with each other according to the figure below.

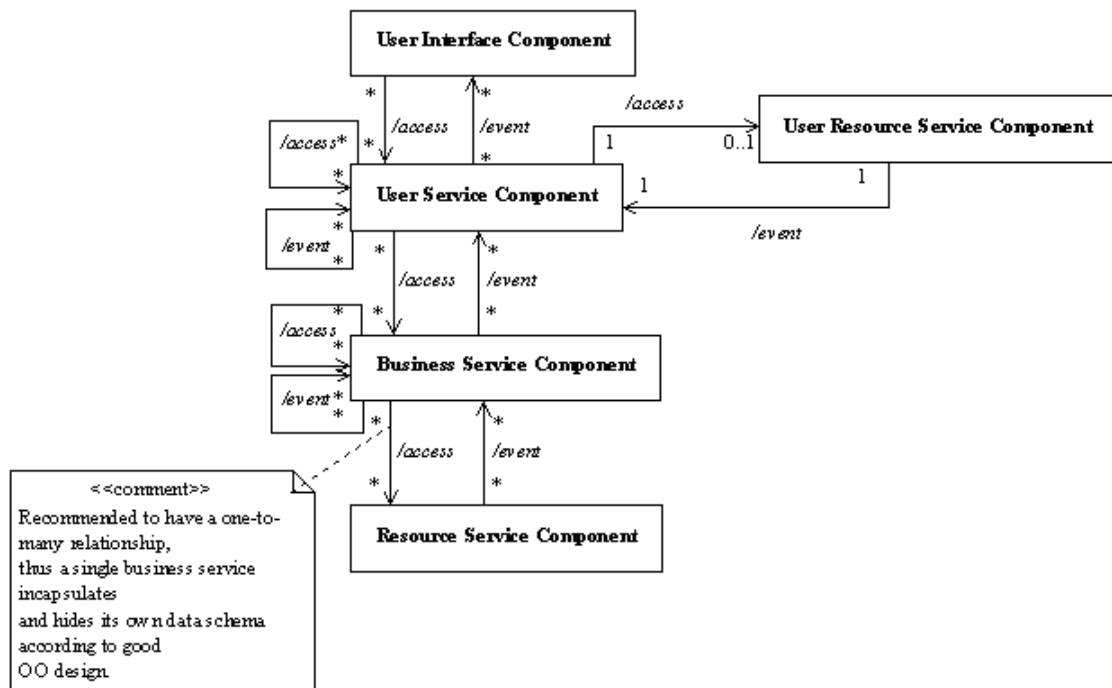


Figure 4: Component Interaction

1.1.2. Composite Components

The composite components are made up of a set of tier components.

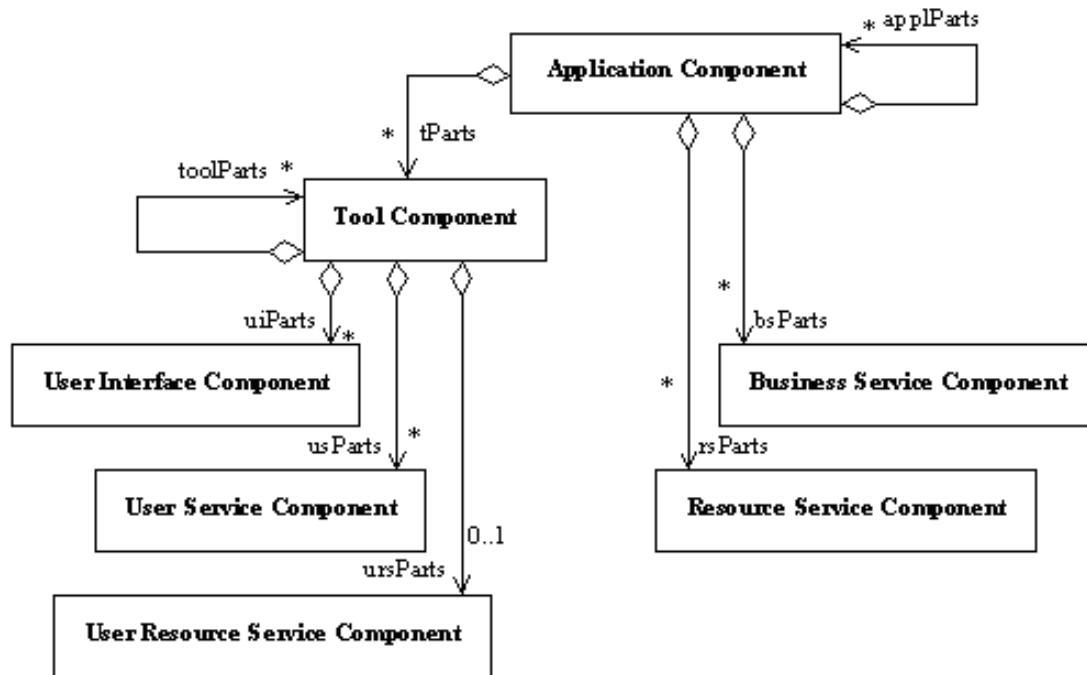


Figure 5: Component compositions

1.1.3. Workflow Components

Workflow components (as used here) are application-specific (contains business workflow definition) and resides in the in the workflow service domain. The workflow service domain is orthogonal to the user service and business service domains. Figure 6 shows how workflow components relate to architectural concepts.

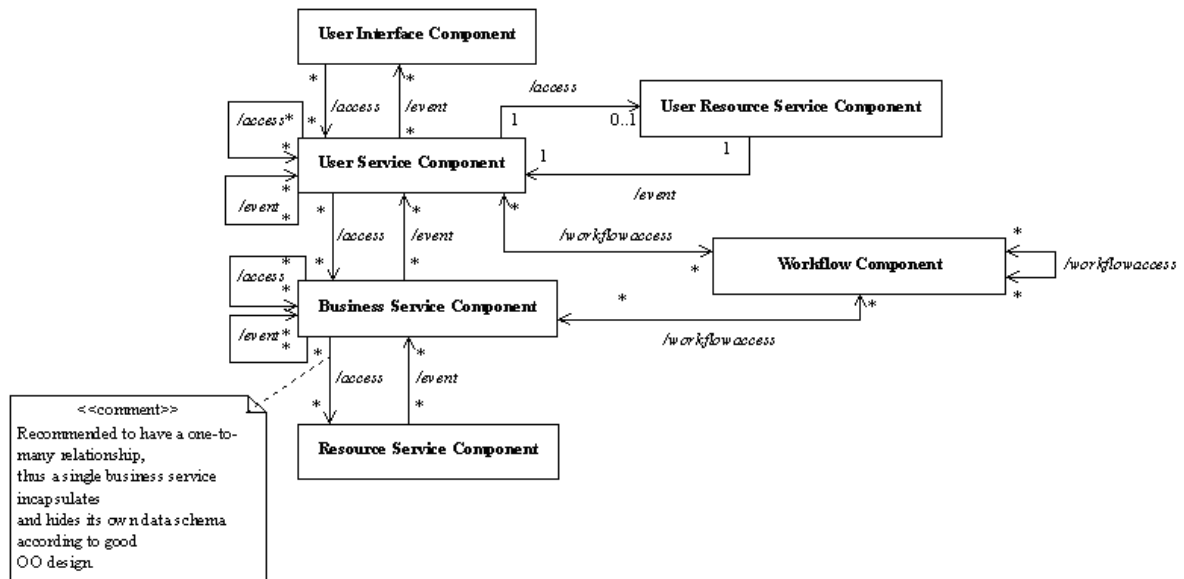


Figure 6: Workflow interactions

1.2. Type Libraries

Type libraries define various data types and primitive classes that are used in the architecture extending the simple, primitive types supported by UML such as string, float, integer, etc. By modelling type libraries, we enforce a single source platform-independent type system that can be mapped to various specific technology or programming language libraries.

1.2.1. Examples

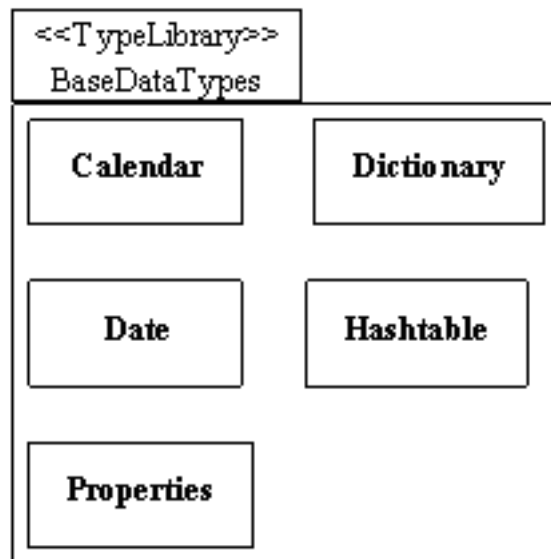


Figure 7: Type Library Example

Figure 7 shows an example of a simple Type Library that defines primitive base data types.