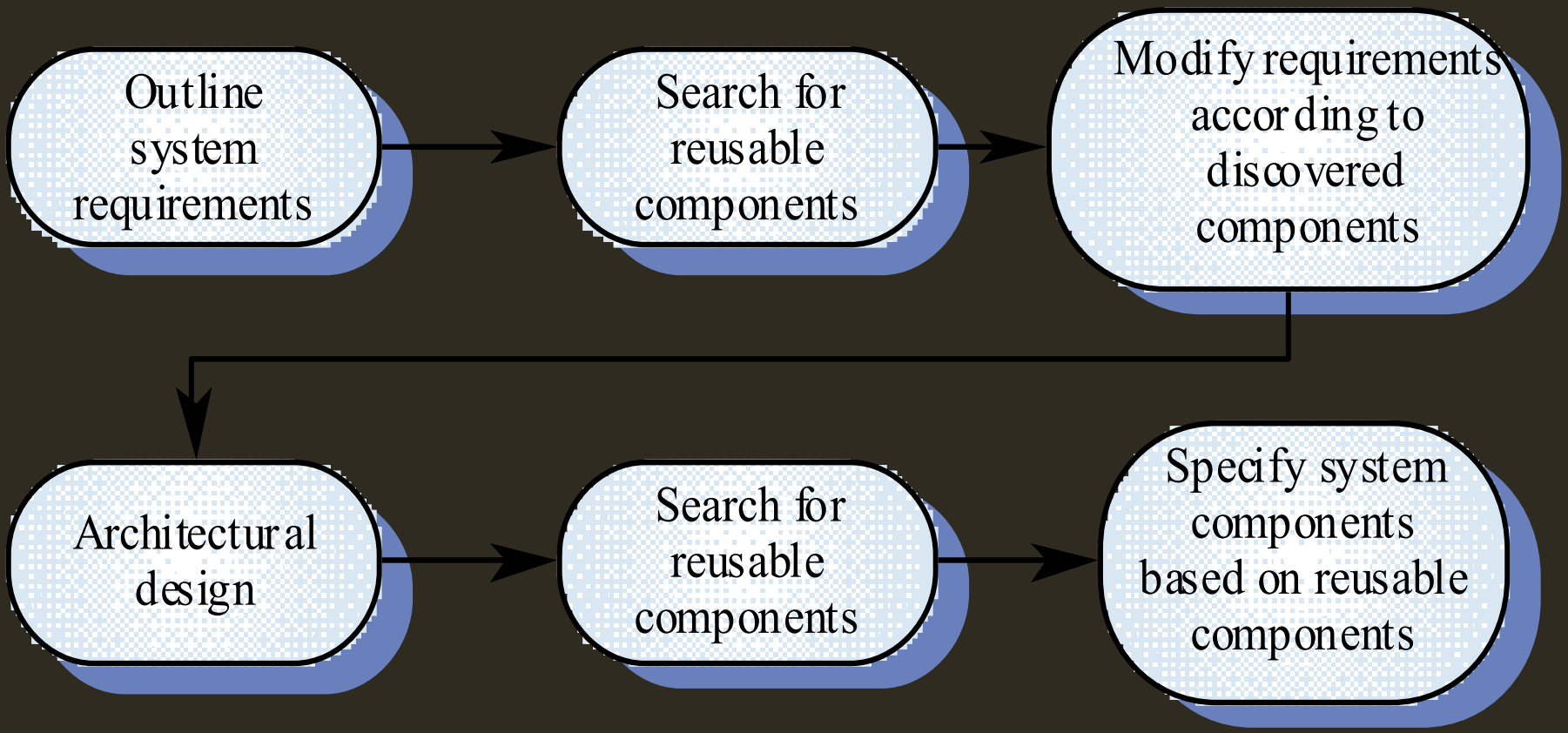
COMPONENT BASED SOFTWARE ENGINEERING

- Component Based Software Engineering(CBSE) is a process that emphasis the design and construction of computer based system using reusable software “components”.

- It emerged from the failure of object-oriented development to support effective reuse. Single object classes are too detailed and specific.

- CBSE embodies the “buy , don’t built” philosophy.

DEVELOPMENT REUSE AS A GOAL



CBSE V/S TRADITIONAL SE

- CBSE views the system as a set of off-the-shelf components integrated within appropriate architecture. Whereas SE seeks to create a system from scratch(building something without tools).
- CBSE does not have any standard development models like UML for SE.
- SE can fulfil the requirements more easily whereas CBSE's fulfilment of the requirements is based on available components.

WHY CBSE?

- CBSE increases quality, especially evolvability and maintainability.
- CBSE increases productivity.
- CBSE shortens development time.
- CBSE is easy to assemble and less costly to build the system constructed from discrete parts.

COMPONENT BASED SYSTEM

The process begins when a software team establish requirements for a system to be built using conventional requirements elicitation techniques.

An architectural design is established , but rather than moving immediately into more detailed tasks , the team examines requirements to determine what subset is directly amenable to ***composition*** , rather than construction.

- For those requirements that can be addressed with available components the following activities take place:

1. component qualification
2. component adaptation
3. component composition
4. component update

- Detailed design activities commence for remainder of the system

- Are Commercial Off-The-Shelf (COTS) components available to implement the requirements ?
- Are internally developed reusable components available to implement the requirements ?
- Are the interfaces for available components compatible within the architecture of the system to be built ?

COTS systems are usually complete applications library the off an applications programming interface (API)

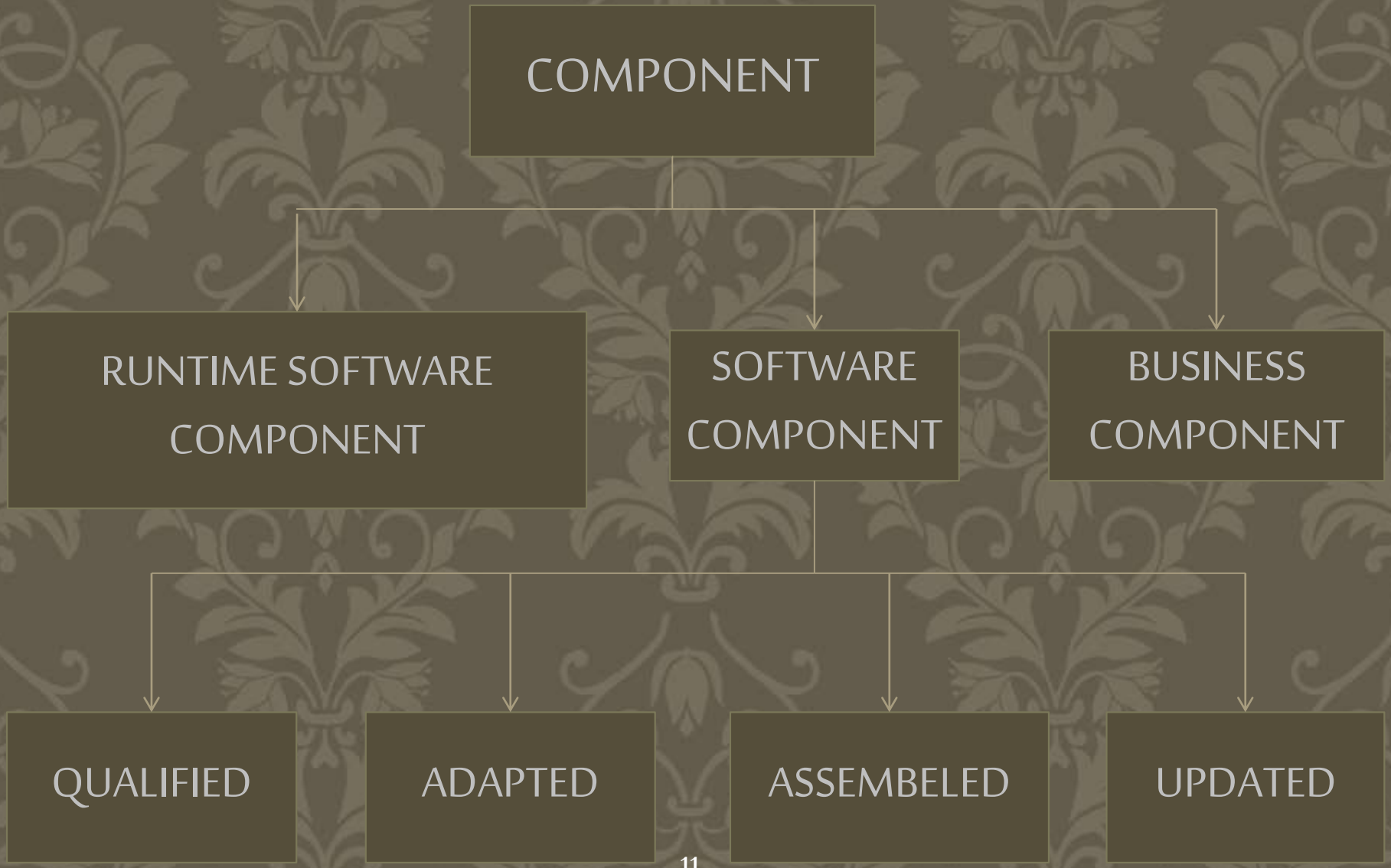
Building large systems by integrating COTS components is a viable development strategy for some types of systems (e.g. E-commerce or video games)

WHAT IS COMPONENT ?

- A nontrivial , nearly independent , and replicable part of a system that fulfils a clear function in the context of a well-defined architecture.
- The component is an independent, executable entity. It can be made up of one or more executable objects.
- It does not have to be compiled before it is used with other components.
- Components do not define types.

COMPONENT CHARACTERISTICS

1. Standardised
2. Independent
3. Compassable
4. Deployable
5. Documented


COMPONENT
RUNTIME SOFTWARE COMPONENT
SOFTWARE COMPONENT
BUSINESS COMPONENT
QUALIFIED
ADAPTED
ASSEMBELED
UPDATED

THE CBSE PROCESS

- The CBSE process is characterized in a manner that not only identifies candidate components but also qualifies each component's interface , adapts components to remove architectural mismatches , assembles components into selected architectural style , and updates components as requirements for the system change.

- The process model for CBSE emphasises parallel tracks in which domain engineering occurs concurrently with component based development.

DOMAIN ENGINEERING

- The intent of domain engineering is to identify , construct , catalog and disseminate a set of software components that have applicability to existing and future software in a particular application domain.
- It provides the library of reusable components that are required for CBSE.
- It includes 3 major activities:

1. Analysis
2. Construction
3. Dissemination

- Domain analysis

1. define application domain to be investigated
2. categorize items extracted from domain
3. collect representative applications from the domain
4. analyze each application from sample
5. develop an analysis model for objects

- Domain analysis is applicable for any software engineering paradigm.
- Software architecture development

STRUCTURAL MODELLING & STRUCTURE POINTS

- Structural modelling is a approach that works under the assumption that every application domain has repeating patterns that have reuse potential.

- Structure point is a distinct construct within a structural model.

CHARACTERISTICS OF STRUCTURE POINT:

- Abstractions with limited number of instances within an application and should recurs in applications in the domain
- Rules governing the use of a structure point should be easily understood and structure point interface should be simple
- Structure points should implement information hiding by isolating all complexity contained within the structure point itself

COMPONENT-BASED DEVELOPMENT (CBD)

- CBD is a CBSE activity that occurs in parallel with domain engineering.
- Architectural style is defined that is appropriate for analysis model created for application to be built.
- Once architecture has been established , it must be populated by components that (1) are available from reuse libraries and/or (2) are engineered to meet custom needs.
- The resultant components are then “composed” (integrated) into the architecture template and tested.

- The existence of reusable components does not guarantee that these components can be integrated easily or effectively into the architecture chosen for a new application. It is for this reason that a sequence of CBD activities is applied when a component is proposed for use.

- CBD consists of following 3 parts:

1. COMPONENT QUALIFICATION
2. COMPONENT ADAPTION
3. COMPONENT COMPOSITION

COMPONENT QUALIFICATION

- It ensures that a candidate component will perform the function required , will properly “fit” into the architectural style specified for system , and will exhibit quality characteristics that are required for application.
- Interface does not provide all the information required to determine if a proposed component can be reused effectively in new application.
- Many factors considered during qualification are: application programming interface (API) , runtime requirements , network protocols , OS interface , exception handling etc.
- Each of these factors is relatively easy to access when reusable components that have been developed in-house are proposed.

COMPONENT ADAPTION

- Even after a component has been qualified for use within an application architecture , conflicts may occur in one or more of the areas. To avoid these conflicts an adaptation technique called component wrapping is often used.
- White-box Wrapping: Integration conflicts removed by making code-level modifications to the code
- Grey-box Wrapping: Used when component library provides a component extension language or API that allows conflicts to be removed or masked
- Black-box Wrapping: Requires the introduction of pre- and post-processing at the component interface to remove or mask conflicts

COMPONENT COMPOSITION

- The component composition task assembles qualified , adapted and engineered components to populate the architecture established for an application. To accomplish this , an infrastructure must be established to bind the components into an operating system.
- Among many mechanism for creating an effective infrastructure is a set of 4 “architectural ingredients” that should be present to achieve component composition. They are as follows:

1. Data exchange model
2. Automation
3. Structured storage
4. Underlying object model

- Data exchange model: similar to drag and drop type mechanisms should be defined for all reusable components allow human-to-software and component-to-component transfer.

- Automation: tools, macros, scripts should be implemented to facilitate interaction between reusable components.

- Structured storage: heterogeneous data should be organized and contained in a single data structure rather several separate files

- Underlying object model: ensures that components developed in different languages are interoperable across computing platforms

REPRESENTATIVE COMPONENT STANDARDS

- Object Management Group (OMG) CORBA- common object request broker architecture
- Microsoft COM - component object model
- Sun JavaBeans Component System

ADVANTAGES OF CBSE

- Reduces development time
- Increases productivity
- Reduction in cost for development
- Reliability is increased
- Management of complexity
- Flexibility

DISADVANTAGE OF CBSE

- Development of components
- Quality of components is questionable
- Component maintenance costs
- Sensitive to changes