|  |
| --- |
| <Project Name> |

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Revision | Comments | Date |
|  |  |  |  |
|  |  |  |  |

References:

|  |  |  |
| --- | --- | --- |
| Title | Revision | File Name |
|  |  |  |
|  |  |  |

Note: Text under ”‘What is this” and displayed in dark red italics (style=Comment) is included to provide guidance to the author and should be deleted before publishing the document. Blue text is example text that should be modified or removed.

To customize automatic fields (which display a gray background when selected), select File>Properties and replace the Title and Subject fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents

Note: The Software Design Description (SDD) captures the complete software design for the system, or a portion of the system.  Following is a typical SDD outline for a project using only traditional natural-language style requirements.

[1 Introduction 3](#_Toc127064432)

[1.1 Purpose 3](#_Toc127064433)

[1.2 Scope 3](#_Toc127064434)

[1.3 References 3](#_Toc127064435)

[1.4 Definitions and acronyms 3](#_Toc127064436)

[1.4.1 General definitions and acronyms 3](#_Toc127064437)

[1.4.2 Specific definitions and acronyms 3](#_Toc127064438)

[1.5 Background 3](#_Toc127064439)

[2 Decomposition description 3](#_Toc127064440)

[*What is this* 3](#_Toc127064441)

[Scope 3](#_Toc127064442)

[Use 3](#_Toc127064443)

[Representation 3](#_Toc127064444)

[2.1 Top level decomposition 3](#_Toc127064445)

[2.2 Hierarchical decomposition 3](#_Toc127064446)

[2.2.1 Entity1 (should be replaced by an entity name) 3](#_Toc127064447)

[2.2.1.1 Type 3](#_Toc127064448)

[2.2.1.2 Purpose 3](#_Toc127064449)

[2.2.1.3 Function 3](#_Toc127064450)

[2.2.1.4 Subordinates 3](#_Toc127064451)

[2.2.2 Entity2 (should be replaced by an entity name) 3](#_Toc127064452)

[2.2.2.1 Type 3](#_Toc127064453)

[2.2.2.2 Purpose 3](#_Toc127064454)

[2.2.2.3 Function 3](#_Toc127064455)

[3 Dependency description 3](#_Toc127064456)

[*What is this* 3](#_Toc127064457)

[Scope 3](#_Toc127064458)

[Use 3](#_Toc127064459)

[Representation 3](#_Toc127064460)

[3.1.1 Top level dependency description 3](#_Toc127064461)

[3.1.2 Entity1 3](#_Toc127064462)

[3.1.2.1 Dependencies 3](#_Toc127064463)

[3.1.2.2 Resources 3](#_Toc127064464)

[3.1.3 Entity2 3](#_Toc127064465)

[3.1.3.1 Dependencies 3](#_Toc127064466)

[3.1.3.2 Resources 3](#_Toc127064467)

[4 Interface description 3](#_Toc127064468)

[*What is this* 3](#_Toc127064469)

[Scope 3](#_Toc127064470)

[Use 3](#_Toc127064471)

[Representation 3](#_Toc127064472)

[4.1.1 Entity1 3](#_Toc127064473)

[4.1.1.1 Interfaces 3](#_Toc127064474)

[4.1.2 Entity2 3](#_Toc127064475)

[4.1.2.1 Interfaces 3](#_Toc127064476)

[5 Detailed design 3](#_Toc127064477)

[*What is this* 3](#_Toc127064478)

[Scope 3](#_Toc127064479)

[Use 3](#_Toc127064480)

[Representation 3](#_Toc127064481)

[5.1.1 Entity1 3](#_Toc127064482)

[5.1.1.1 Processing 3](#_Toc127064483)

[5.1.1.2 Data 3](#_Toc127064484)

[6 Thread module 3](#_Toc127064485)

[7 Data dictionary 3](#_Toc127064486)

[8 Packaging module 3](#_Toc127064487)

[9 Requirement coverage table 3](#_Toc127064488)

# Introduction

Overview of the development goal

## Purpose

Describe the purpose of the document. Who is the intended audience?

## Scope

Describe what parts of the development are covered by this design and what is not.

## References

List the documents that base this document. Document name, title and possibly comments.

## Definitions and acronyms

### General definitions and acronyms

* Identification - *The name of the entity.* Two entities shall not have the same name. The names for the entities may be selected to characterize their nature. This will simplify referencing and tracking in addition to providing identification.
* Type - *A description of the kind of entity.* The type attribute shall describe the nature of the entity. It may simply name the kind of entity, such as subprogram, module, procedure, process, or data store. Alternatively, design entities may be grouped into major classes to assist in locating an entity dealing with a particular type of information. For a given design description, the chosen entity types shall be applied consistently.
* Purpose - *A description of why the entity exists.* The purpose attribute shall provide the rationale for the creation of the entity. Therefore, it shall designate the specific functional and performance requirements for which this entity was created; see IEEE Std 830-1998. The purpose attribute shall also describe special requirements that must be met by the entity that are not included in the software requirements specification.
* Function - *A statement of what the entity does.* The function attribute shall state the transformation applied by the entity to inputs to produce the desired output. In the case of a data entity, this attribute shall state the type of information stored or transmitted by the entity.
* Subordinates - *The identification of all entities composing this entity.* The subordinates attribute shall identify the *composed of* relationship for an entity. This information is used to trace requirements to design entities and to identify parent/child structural relationships through a software system decomposition.
* Dependencies - *A description of the relationships of this entity with other entities.* The dependencies attribute shall identify the *uses* or *requires the presence of* relationship for an entity. These relationships are often graphically depicted by structure charts, data flow diagrams, and transaction diagrams. This attribute shall describe the nature of each interaction including such characteristics as timing and conditions for interaction. The interactions may involve the initiation, order of execution, data sharing, creation, duplicating, usage, storage, or destruction of entities.
* Interface - *A description of how other entities interact with this entity.* The interface attribute shall describe the *methods* of interaction and the *rules* governing those interactions. The methods of interaction include the mechanisms for invoking or interrupting the entity, for communicating through parameters, common data areas or messages, and for direct access to internal data. The rules governing the interaction include the communications protocol, data format, acceptable values, and the meaning of each value. This attribute shall provide a description of the input ranges, the meaning of inputs and outputs, the type and format of each input or output, and output error codes. For information systems, it should include inputs, screen formats, and a complete description of the interactive language.
* Resources - *A description of the elements used by the entity that are external to the design.* The resources attribute shall identify and describe all of the resources *external* to the design that are needed by this entity to perform its function. The interaction rules and methods for using the resource shall be specified by this attribute. This attribute provides information about items such as physical devices (printers, disc-partitions, memory banks), software services (math libraries, operating system services), and processing resources (CPU cycles, memory allocation, buffers). The resources attribute shall describe usage characteristics such as the process time at which resources are to be acquired and sizing to include quantity, and physical sizes of buffer usage. It should also include the identification of potential race and deadlock conditions as well as resource management facilities.
* Processing - *A description of the rules used by the entity to achieve its function.* The processing attribute shall describe the algorithm used by the entity to perform a specific task and shall include contingencies. This description is a refinement of the function attribute. It is the most detailed level of refinement for this entity. This description should include timing, sequencing of events or processes, prerequisites for process initiation, priority of events, processing level, actual process steps, path conditions, and loop back or loop termination criteria. The handling of contingencies should describe the action to be taken in the case of overflow conditions or in the case of a validation check failure.
* Data - *A description of data elements internal to the entity.* The data attribute shall describe the method of representation, initial values, use, semantics, format, and acceptable values of internal data. The description of data may be in the form of a data dictionary that describes the content, structure, and use of all data elements. Data information shall describe everything pertaining to the use of data or internal data structures by this entity. It shall include data specifications such as formats, number of elements, and initial values. It shall also include the structures to be used for representing data such as file structures, arrays, stacks, queues, and memory partitions. The meaning and use of data elements shall be specified. This description includes such things as static versus dynamic, whether it is to be shared by transactions, used as a control parameter, or used as a value, loop iteration count, pointer, or link field. In addition, data information shall include a description of data validation needed for the process.

### Specific definitions and acronyms

List the definitions and acronyms that are used in the SDD

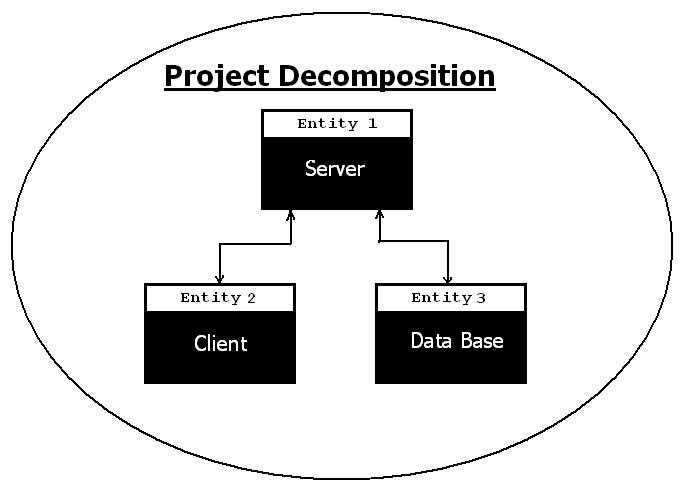
## Background

Specify technical background information that is needed to understand the design

# Decomposition description

Considering the fact that project isn’t very difficult to use and understand, the decomposition will not have alot of top level entities. The system has divided functions of the server, client and Data Base. All three entities are operating actually with information about conveyor working (data).

## Top level decomposition



*Pic 2.1 .Simple decomposition of Conveyor Watching Eye project.*

We can decomposite our project in three main entities :

1. Server (using to manage a data from conveyor, make relations between Client & DB )
2. Client (using to receive data from camera, compute it and send inf to server, etc. )
3. Data Base ( only data store , saving information about conveyors working)

## Hierarchical decomposition

### Entity 1. Server

#### Type

Server is a program wich coordinate all system working, create relations between other parts of system. It represents to be a protected, specialized, optimized program with long-term usage.

#### Purpose

The main purpose of server creating and activity is correct communication between user, client data base. Using server makes us easier to create simple architecture of system, and support it in future. It must be available to install this program on Windows/Linux, and attach data base modules. Command system of server have to be friendly to user.

#### Function

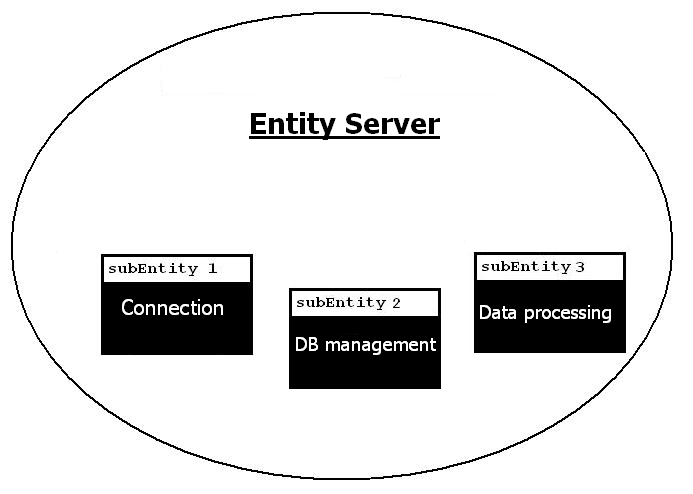
Main function of server is to maintain correct performance of system. Also it includes next functions:

* Processing data
* Receive / Send data
* Authentication
* Create / Delete users
* Grants rights

#### Subordinates

Server entity can be divided itno 3 sub-entities:

* Connection (TCP/IP)
* DB Management
* Data Processing



Pic 2.2. SubEntities of Server.

### Entity 2. Client

#### Type

Client is a program wich has connections with camera(s), or another receiving device, and is using to convert visualization of conveyor to informations about it’s work, with future send to server.

#### Purpose

If server is our system’s heart, then client is it’s eyes and mouth. It receive data from camera detect bread on picture. Main requirement - to have extensive functionality, fast graphics working opportunities, strong correct logic.

#### Function

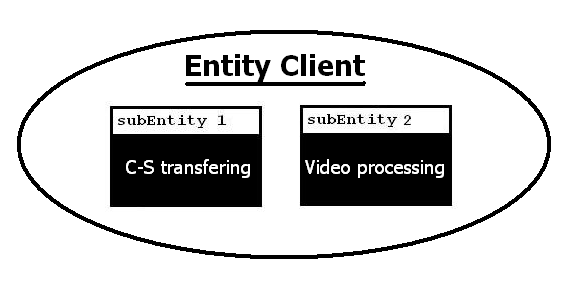
Main functions of client is to operate wih visual data and cooperate with user. Also it has next functions:

* Detecting objects at image
* Sending data to server
* Control conveyor
* Authentication of conveyor
* Display statistic

#### Subordinates

Client entity can be divided to 2 sub entities:

* Client-Server transferring
* Video processing



Pic 2.3. Client subEntities.

### Entity 3. Data Base

#### Type

DataBase is simple collection of data wich consist of information about conveyor working, and let us operate with it, for example see statistic for last weak.

#### Purpose

We use DataBase because operate with huge amount of information. Also we need some systematization of last. DB has advantages in security, performance, systematization, functionality, portability, etc.

#### Function

There are only two functions of DB :

* Save data
* Answer requests from server

#### Subordinates

There are no subordinates.

### 

# Dependency description

## *What is this*

### *Scope*

The dependency description specifies the relationships among entities. It identifies the dependent entities, describes their coupling, and identifies the required resources. This design view defines the strategies for interactions among design entities and provides the information needed to easily perceive how, why, where, and at what level system actions occur. It specifies the type of relationships that exist among the entities such as shared information, prescribed order of execution, or well-defined parameter interfaces. The attribute descriptions for identification, type, purpose, dependencies, and resources should be included in this design view. This attribute information should be provided for all design entities that we listed in the hierarchical decomposition section.

### *Use*

The dependency description provides an overall picture of how the system works in order to assess the impact of requirements and design changes. It can help maintainers to isolate entities causing system failures or resource bottlenecks. It can aid in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. This description can also be used by integration testing to aid in the production of integration test cases.

### *Representation*

Formal specification languages provide for the specification of system functions and data, their interrelationships, the inputs and outputs, and other system aspects in a well-defined language. The relationship among design entities is also represented by data flow diagrams, structure charts, or transaction diagrams.

### Top level dependency description

Use the same UML diagram from the top level decomposition part and add dependency information. Including association, Aggregation, generalization, dependency and realization

### Entity1

For each entity describe the dependencies and resources regarding this entity. Use UML and free text.

#### Dependencies

A description of the relationships of this entity with other entities. The dependencies attribute shall identify the uses or requires the presence of relationship for an entity. These relationships are often graphically depicted by structure charts, data flow diagrams, and transaction diagrams.

This attribute shall describe the nature of each interaction including such characteristics as timing and conditions for interaction. The interactions may involve the initiation, order of execution, data sharing, creation, duplicating, usage, storage, or destruction of entities.

#### Resources

A description of the elements used by the entity that are external to the design. The resources attribute shall identify and describe all of the resources external to the design that are needed by this entity to perform its function. The interaction rules and methods for using the resource shall be specified by this attribute.

This attribute provides information about items such as physical devices (printers, disc-partitions, memory banks), software services (math libraries, operating system services), and processing resources (CPU cycles, memory allocation, buffers).

The resources attribute shall describe usage characteristics such as the process time at which resources are to be acquired and sizing to include quantity, and physical sizes of buffer usage. It should also include the identification of potential race and deadlock conditions as well as resource management facilities.

### Entity2

#### Dependencies

#### Resources

# Interface description

## *What is this*

### *Scope*

The entity interface description provides everything designers, programmers, and testers need to know to correctly use the functions provided by an entity. This description includes the details of external and internal interfaces not provided in the software requirements specification. This design view consists of a set of interface specifications for each entity. The attribute descriptions for identification, function, and interfaces should be included in this design view. This attribute information should be provided for all design entities.

### *Use*

The interface description serves as a binding contract among designers, programmers, customers, and testers. It provides them with an agreement needed before proceeding with the detailed design of entities. In addition, the interface description may be used by technical writers to produce customer documentation or may be used directly by customers. In the latter case, the interface description could result in the production of a human interface view. Designers, programmers, and testers may need to use design entities that they did not develop. These entities may be reused from earlier projects, contracted from an external source, or produced by other developers. The interface description settles the agreement among designers, programmers, and testers about how cooperating entities will interact. Each entity interface description should contain everything another designer or programmer needs to know to develop software that interacts with that entity. A clear description of entity interfaces is essential on a multiperson development for smooth integration and ease of maintenance.

### *Representation*

The interface description should provide the language for communicating with each entity to include screen formats, valid inputs, and resulting outputs. For those entities that are data driven, a data dictionary should be used to describe the data characteristics. Those entities that are highly visible to a user and involve the details of how the customer should perceive the system should include a functional model, scenarios for use, detailed feature sets, and the interaction language.

### Entity1

#### Interfaces

Specify the exported methods and events elaborate on parameters

A description of how other entities interact with this entity. The interface attribute shall describe the methods of interaction and the rules governing those interactions. The methods of interaction include the mechanisms for invoking or interrupting the entity, for communicating through parameters, common data areas or messages, and for direct access to internal data. The rules governing the interaction include the communications protocol, data format, acceptable values, and the meaning of each value.

This attribute shall provide a description of the input ranges, the meaning of inputs and outputs, the type and format of each input or output, and output error codes. For information systems, it should include inputs, screen formats, and a complete description of the interactive language.

### Entity2

#### Interfaces

# Detailed design

## *What is this*

### *Scope*

The detailed design description contains the internal details of each design entity. These details include the attribute descriptions for identification, processing, and data. This attribute information should be provided for all design entities.

### *Use*

This description contains the details needed by programmers prior to implementation. The detailed design description can also be used to aid in producing unit test plans.

### *Representation*

There are many tools used to describe the details of design entities. Program design languages can be used to describe inputs, outputs, local data and the algorithm for an entity. Other common techniques for describing design entity logic include using metacode or structured English, or graphical methods such as UML.

### Entity1

#### Processing

A description of the rules used by the entity to achieve its function. The processing attribute shall describe the algorithm used by the entity to perform a specific task and shall include contingencies. This description is a refinement of the function attributes. It is the most detailed level of refinement for this entity.

This description should include timing, sequencing of events or processes, prerequisites for process initiation, priority of events, processing level, actual process steps, path conditions, and loop back or loop termination criteria. The handling of contingencies should describe the action to be taken in the case of overflow conditions or in the case of a validation check failure.

Use UML state charts and activity Diagrams to describe this

#### Data

A description of data elements internal to the entity. The data attribute shall describe the method of representation, initial values, use, semantics, format, and acceptable values of internal data.

The description of data may be in the form of a data dictionary that describes the content, structure, and use of all data elements. Data information shall describe everything pertaining to the use of data or internal data structures by this entity. It shall include data specifications such as formats, number of elements, and initial values. It shall also include the structures to be used for representing data such as file structures, arrays, stacks, queues, and memory partitions. The meaning and use of data elements shall be specified. This description includes such things as static versus dynamic, whether it is to be shared by transactions, used as a control parameter, or used as a value, loop iteration count, pointer, or link field. In addition, data information shall include a description of data validation needed for the process

# Thread module

Describe the thread module including blocking situations and the relationship between threads

# Data dictionary

This is an optional view for developments that are Data oriented. The Data structure should be organized here.

# Packaging module

Describe the software packages including exes, DLLs, COM / DCOM objects Corba Installs and similar. It is important to specify what components run on which hardware.

# Requirement coverage table

Show that were every requirement is addressed in the design

|  |  |  |
| --- | --- | --- |
|  | SRS Requirement | SDD paragraph |
| 1. |  |  |
| 2. |  |  |
| 3. |  |  |