**Software Design**

**Description**

**for**

**Producers-Consumers Economic Relationships Model**

**Version 1.0**

**Prepared by Mateusz Golab**

**Cranfield University**

**School of Engineering**

**AMAC**

**1st April 2012**

**Table of Contents**

[1. Introduction 3](#_Toc321888524)

[1.1. Purpose 3](#_Toc321888525)

[1.2. Scope 3](#_Toc321888526)

[1.3. References 3](#_Toc321888527)

[1.4. Glossary 3](#_Toc321888528)

[2. System architecture description 4](#_Toc321888529)

[2.1. Class diagram 4](#_Toc321888530)

[3. Detailed Description of Components 5](#_Toc321888531)

[3.1. Consumer 5](#_Toc321888532)

[3.2. ConsumerState 6](#_Toc321888533)

[3.3. Factory 6](#_Toc321888534)

[3.4. FactoryState 7](#_Toc321888535)

[3.5. Model 8](#_Toc321888536)

[Operations 9](#_Toc321888537)

[3.6. Offer 10](#_Toc321888538)

[3.7. Order 11](#_Toc321888539)

[3.8. Producer 12](#_Toc321888540)

[3.9. Product 13](#_Toc321888541)

[3.10. SimulationManager 14](#_Toc321888542)

[3.11. SimulationPresenter 16](#_Toc321888543)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
| Mateusz Golab | 1/04/2012 | Document created | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

This document describes *Producers – Consumers Relationships Model’s* functionalities. It contains instructions and guides for the developers team. Includes system architecture and components descriptions.

## Scope

The project is desktop application , designed to prepare producers- consumers relationships model and perform simulation.

## References

[1*] Advanced Software Engineering Lecture Notes,* Academic Year 2010/2011 , Dr. Stuart Barnes

[2] *Software Design Specification Example*, B. Hoening, C. Strickland, S. Ayalala, C. Emmons, R.Dhanani, December 2001

[3*] IEEE Standard for Information Technology – Systems Design – Software Design Descriptions*, IEEE Std 1016-2009, IEEE Computer Society

## 1.4. Glossary

Available in *Software Requirements Specification for Producers – Consumers Economic Relationships Model* document *.*

# System architecture description

## Class diagram



# Detailed Description of Components

## Consumer

Type : Class

Created : 03-04-2012 23:54

Last modified : 10-04-2012 19:09

This class represents Consumer which is able to order products from the Producer.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private double cash | Cash of consumer |
| private double salary | Salary of consumer |
| private int id | Consumer’s ID |
| private ConsumerState state | State of the consumer |
| private static int idGenerator | Consumers ID provider |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Consumer | Constructor | double **cash**,  duble **salary** |  |
| public virtual ~Consumer | Destructor |  |  |
| public void makeOrder | Makes order for specified product type | int **productType** |  |
| public void payProducer | Transfers cash to specified Producer when order is placed | double **price** |  |
| public void receiveSalary | Increases Consumer’s cash by salary value |  |  |
| public void setSalary | Sets Consumer’s salary | double **salary** |  |
| public double getCash | Returns Consumer’s cash |  | Consumer’s cash |
| public ConsumerState getState | Returns Consumer’s state |  | Consumer’s state |
| public int getID | Returns Consumer’s ID |  | Consumer’s ID |

## ConsumerState

Type : Enumerator

Created : 06-04-2012 23:33

Last modified : 10-04-2012 19:09

This Enumerator represents Consumer’s possible states

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| PRODUCT\_ORDERED | State of the Consumer when waiting for order |
| NOTHING\_ORDERED | State of the Consumer when idle |

## Factory

Type : Class

Created : 04-04-2012 12:40

Last modified : 10-04-2012 19:13

This class represents Factory which is capable of manufacturing products.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private double efficiency | describes how fast factory can manufacture products |
| private int idleTime | Number of idle cycles of the Factory |
| private FactoryState state | State of the Factory |
| private Product\* currentProduct | Pointer to the currently manufactured Product |
| private static double constructionCost | New Factory construction cost |
| private static double costPerCycle | Cost of maintaining existing Factory for a single cycle |
| private double startUpCost | Cost of changing Factory’s state from IDLE to READY |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Factory | Default constructor |  |  |
| public Factory | Constructor | **double efficiency** |  |
| public virtual ~Factory | Destructor |  |  |
| public int getIdleTime const | Returns number of cycles Factory is in IDLE state |  | Number of IDLE cycles |
| public FactoryState getState const | Returns Factory’s state |  | State of the Factory |
| public double manufacture | Starts manufacturing specified Product | **Product\* product** | Proportion of Product’s completeness |
| public double manufacture | Manufactures current Product |  | Proportion of Product’s completeness |
| public void setState | Sets Factory’s state | **FactoryState state** |  |
| public void incIdleTime | Increases Factory’s idle time |  |  |
| public static void setConstructionCost | Sets Factory’s construction cost | **double cost** |  |
| public static void setRunningCost | Sets Factory’s running cost | **double cost** |  |
| public static void setIdleStartUpCost | Sets Factory’s state changing cost | **double cost** |  |
| public static double getConstructionCost | Gets Factory’s construction cost |  | Factory’s construction cost |
| public static double getRunningCost | Gets Factory’s running cost |  | Factory’s running cost |
| public static double getIdleStartUpCost | Gets Factory’s state changing cost |  | Factory’s state changing cost |

## FactoryState

Type : Enumerator

Created : 05-04-2012 13:33

Last modified : 10-04-2012 19:13

This Enumerator represents Factory’s possible states

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| READY | State of the Factory when ready to manufacture |
| RUNNING | State of the Factory when currently manufacturing |
| IDLE | State without manufacturing and not ready to manufacture |

## Model

Type : Class

Created : 03-04-2012 23:55

Last modified : 10-04-2012 19:44

This class represents Producers-Consumers Economic Relationship Model. Contains all parameters essential to perform simulation.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private double cashPerConsumer | Consumer’s initial cash |
| private double cashPerProducer | Producer’s initial cash |
| private double consumerSalary | Consumer’s salary |
| private double factoryConstructionCost | cost to construct a Factory |
| private double factoryRunningCost | Cost to run factory for one cycle |
| private double idleFactoryStartupCost | Cost of changing Factory’s state from IDLE to READY |
| private int numberOfConsumers | Number of consumers in simulation |
| private int numberOfProducers | Number of producers in simulation |
| private int queueOrdersThreshold | Maximum number of products in Producer’s queue |
| private std::vector<double> productCosts | Costs of particular products |
| private std::vector<double> productLengths | Number of cycles to produce particular products |

## Operations

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Model | Default constructor |  |  |
| public virtual ~Model | Destructor |  |  |
|  |  |  |  |
| public void setConsumerCash | Sets Consumer’s initial cash | **double cash** |  |
| public void setFactoryConstructionCost | Sets Factory’s construction cost | **double cost** |  |
| public void setFactoryRunningCost | Sets Factory’s running cost | **double cost** |  |
| public void setIdleFactoryStartUpCost | Sets Factory’s startUp cost | **double cost** |  |
| public void setNumberOfConsumers | Sets number of consumers | **int numberOfConsumers** |  |
| public void setNumberOfProducers | Sets number of producers | **int numberOfProducers** |  |
| public void setProducerCash | Sets Producer’s cash | **double cash** |  |
| public void setProductManufacturingCost | Sets Particular products manufacturing cost | **std::vector<double> costs** |  |
| public void setProductManufacturingTime | Sets Particular products manufacturing time in cycles | **std::vector<double> cycles** |  |
| public void setQueueOrdersThreshold | Sets threshold for number of products in a queue | **int threshold** |  |
| public void setConsumerSalary | Sets Consumer’s salary | **double salary** |  |
| public int getNumberOfProducers const | Returns Number of producers |  | Number of Producers |
| public int getNumberOfConsumers const | Returns Number of consumers |  | Number of Consumers |
| public double getCashPerProducer const | Returns cash of Producer |  | Cash of Producer |
| public double getCashPerConsumer const | Returns cash of Consumer |  | Cash of Consumer |
| public double getConsumerSalary const | Returns Consumer’s salary |  | Consumer’s salary |
| public double getFactoryRunningCost const | Returns Factory’s running cost |  | Factory’s running cost |
| public double getFactoryConstructionCost const | Returns Factory’s construction cost |  | Factory’s construction cost |
| public double getFactoryStartUpCost const | Returns Factory’s startUp cost |  | Factory’s startUp cost |
| public std::vector<double> getProductCosts | Returns particular products costs |  | Particular products costs |
| public std::vector<double> getProductLengths | Sets Factory’s state |  | Particular products lengths |

## Offer

Type : Class

Created : 07-04-2012 00:10

Last modified : 10-04-2012 19:13

Class representing Offer on Simulation market.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private double price | Price of the offered product |
| private int producerID | Producer’s ID |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Offer | Default constructor |  |  |
| public virtual ~Offer | Destructor |  |  |
| public Offer | Constructor | **int producerID, double price** |  |
| public int getProducerID | Returns Producer’s ID |  | Producer’s ID |
| public double getPrice const | Returns Product’s price |  | Product’s price |
| bool operator() | Operator to compare two Offers | **const Offer & o1, const Offer & o2** | Returns true if o1<o2 |

## Order

Type : Class

Created : 03-04-2012 23:55

Last modified : 10-04-2012 19:13

Represents Order made by Consumer .

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private int id | Order’s id |
| private int consumerID | Consumer’s id |
| private int producerID | Producer’s id |
| private double cost | Order’s total cost |
| private int productType | Product’s type |
| private std::vector<Product> products | Products |
| private static int idGenerator | Order’s ID generator |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Order | Default constructor |  |  |
| public virtual ~Order | Destructor |  |  |
| public Order | Constructor | **int consumerID, int producerID, double cost, int numberOfProducts, int productType** |  |
| public int getNumberOFProducts | Returns Number of Products in the order |  | Number of products |
| public Product & getProduct | Returns specified product | **int i** | Specifie product |
| public int getID | Returns Order’s ID |  | Order’s ID |
| public int getProducerID | Returns Producer’s ID |  | Producer’s ID |
| public int getConsumersID | Returns Consumers ID |  | Consumer’s ID |
| public double getCost | Returns total Order’s cost |  | Total Order’s cost |
| public int getProductType | Returns Product’s type |  | Product’s type |
| public bool isCompleted | Returns true if completed |  | If completed |

## Producer

Type : Class

Created : 03-04-2012 23:54

Last modified : 10-04-2012 19:38

This class represents Producer receiving and realizing Orders from Consumers.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private int id | Producer’s id |
| private double productPrices[5] | Prices of the products |
| private double productCosts[5] | Manufacture costs of the products |
| private double productLengths[5] | Manufacture cycles to produce of the products |
| private std::vector<Factory> factories | Producer’s factories |
| private std::vector<Order> orders | Producer’s orders |
| private int numberOfProduts | Number of products in all orders |
| private int numberOfAllCompletedOrders | Number of all completed orders |
| private int ordersIterator | Iterator of orders |
| private int productIterator | Iterator of products |
| private static int idGenerator | Producer’s idgenerator |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Producer | Constructor | **double cash, int threshold, std:: vector<double> costs, std::vector<double> lengths** |  |
| public virtual ~Producer | Destructor |  |  |
| public bool buildFactory | Creates new factory |  | True if new factory was built |
| public void demolishUnusedFactories | Destroys unused factories |  |  |
| public void realizeOrders | Realizes orders |  |  |
| public void receiveCash | Receives cash | **double cash** |  |
| public double getCash | Returns cash |  | Producer’s cash |
| public double getProductPrice | Returns price of specified product | **int productType** | Specified product’s type price |
| public int getID | Returns ID |  | Producer’s ID |
| public bool acceptOrder | Accepts or not specified order | **Order & order** | If accepted |
| public int getNumberOfOrders | Returns the number of orders |  | Number of orders |
| public int getNumberOfCompletedOrders | Returns number of completed orders |  | Number of completed orders |
| public int getNumberOfFactories | Returns number of factories |  | Number of factories |
| public void payForFactories | Pays for all Producer’s factories |  |  |
| public bool payForProduct | Producer pays for specified product | **int productType** | True if product was bought |
| public bool payForStartup | Producer pays for factory startUp |  | True if factory was started up |
| public void finalizeOrders | Producer finalizes orders |  |  |
| public void increasePrices | Producer increases products prices | **double percentage** |  |
| public Product & getProduct | Returns specified product |  | Specified product |
| public Factory & getFactory | Returns specified Factory |  | Specified Factory |

## Product

Type : Class

Created : 03-04-2012 23:55

Last modified : 10-04-2012 19:13

Represents Product on the Simulation market.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private int id | Product’s id |
| private int cyclesToProduce | Number of cycles to produce |
| private double price | Price of the Product |
| private double manufactureCost | Cost of producing this Product |
| private double completenessFactor | Proportion of completed Product |
| private int typ | Product’s type |
| private static int idGenerator | Product’s ID generator |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public Product | Constructor | **int type, double price** |  |
| public virtual ~Product | Destructor |  |  |
| public void increasePrice | Price increase | **double percentage** |  |
| public double produce | Production process | **double process** | Proportion of completeness |
| public int getProductType | Returns Product’s type |  | Product’s type |
| public double getPrice const | Return’s price |  | Product’s price |

## SimulationManager

Type : Class

Created : 03-04-2012 23:54

Last modified : 10-04-2012 21:21

Represents main controller of the Simulation. Manages almost all operations taking part in Simulation process.

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private Model model | Simulation model |
| private int currentProductType | productType chosen for current cycle |
| private static int cycle | Current Simulation’s cycle |
| private static std::vector<Offer> offers | All offers on the market |
| private static std::vector<Producer> producers | All producers on the market |
| private static std::vector<Consumer> consumers | All consumers on the market |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public SimulationManager | Constructor | **Model & model** |  |
| public virtual ~SimulationManager | Destructor |  |  |
| public SimulationManager | Default constructor |  |  |
| public void demolishUnusedFactories | Destroys unused factories of producers |  |  |
| public void increasePrices | Increases prices |  |  |
| public int getBankruptProducer | Returns bankrupt producer |  | Bankrupt producer |
| public void initializeModel | Initializes specified Simulation model | **Model & model** |  |
| public void initializeModel | Initializes Simulation model |  |  |
| public void transferSalaries | Transfer salaries to consumers |  |  |
| public int choose productType | Randomly chooses productType |  | Randomly chosen productType |
| public void informConsumers | Informs consumers about new productType | **int productType** |  |
| public void findOffers | Finds and sorts all offers | **int productType** |  |
| public void nextCycle | Performs next Simulation’s cycle |  |  |
| public double getConsumersAverageCash | Returns calculated Consumers average cash |  | Average consumers cash |
| public int getNumberOfConsumer | Returns number of consumers |  | Number of consumers |
| public int getNumberOfProducers | Returns Number of producers |  | Number of producers |
| public void realizeOrders | Relizes orders of producers |  |  |
| public void producersPayments | Producers pay for factories maintenance |  |  |
| public Model & getModel | Returns simulation model |  | Simulation model |
| public static Producer & getProducer | Returns specified producer | **int id** | Specified producer |
| public static Consumer& getConsumer | Returns specified Consumer | **int id** | Specified Consumer |
| public static std::vector<Offer> gtOffers | Returns all offers on market |  | All market offers |
| public static double randomNumberGenerator | Generates random numbers | **double low, double high** | Random number |
| Public static bool isProducerInterested | Checks if producer interested in order | **Order order** | True if interested |
| public static void transaction | Performs market transaction | **int producerID, int consumerID, double cash** |  |
| Public static int getCycleNumber | Returns current Simulation’s cycle number |  | Current simulation’scycle number |

## SimulationPresenter

Type : Class

Created : 06-04-2012 16:09

Last modified : 10-04-2012 20:32

This class is responsible for presenting Simulation’s results to the User

**Attributes**

|  |  |
| --- | --- |
| Attribute | Notes |
| private SimulationManager manager | SimulationManager |
| private std::string filename | Name of the file with results |

**Operations**

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Notes | Parameters | Returns |
| public SimulationPresenter | Constructor | **SimulationManager & manager, std::string fileName** |  |
| public virtual ~SimulationManager | Destructor |  |  |
| public void showSimulationState | Shows simulation state after cycle | **std::ostream & os** |  |
| public void showBankruptcy | showsBankruptcy details | **std::ostream & os, int producerID** |  |
| public void saveResults | Saves Simulation results to the file |  |  |
| public bool setParameters | Allows User setting parameters | **std::ostream & os** | True if simulation can be started, false if exit program |
| public void showWelcome | Shows program title | **std::ostream & os** |  |
| public Model & getModel | Returns Simulation’smodel |  | Simulation’smodel |