

software models (software design) and related artifacts (Guenec et al., 2000; Liu et al., 2004). UML is used for specifying, visualizing, constructing and documenting all aspects of a software system (Fowler and Scott, 1997; France et al., 2004).

Developing a software application for a biotechnological system is a complex project (Sevella and Bertalan, 2000). The successful implementation of a biotechnological system requires a detailed software design and modeling methodology. In this paper, we present the UML based use case and conceptual class diagrams to implement the simulation environment. Simulation environment will be a software application which will provide a playground for researchers to simulate bioreactor experiments and predict the products of the experiment. Simulation environment is required so that various kind of mathematical models that are developed are integrated or available in one software application for use to researchers. The different mathematical models can be chosen based on the objective of the bioreactor experiment, and the data available to validate the predicted results (Dunn et al., 2003; Rani and Rao, 1999; Schugerl and Bellgardt, 2000). The simulation environment will be a computational tool for researchers to judge the potential of bioreactor experiment and make experimental design based on it. It can also act as a tool for education and teaching purposes.

2. Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems (Fowler and Scott, 1997; Warmer and Kleppe, 1999). It has proven successful in the modeling of large and complex software systems (Liu et al., 2004). The UML is a very important part of the software development process. The primary goals in the design of the UML were:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
5. Integrate best practices.

The UML uses mostly graphical notations to express the design of software projects. The various design diagrams available in UML design are: activity diagrams, use case diagrams, sequence diagrams, collaboration diagrams, class diagrams, state transition diagrams, component diagrams and deployment diagrams (Ambler, 2003; Fowler and Scott, 1997; Warmer and Kleppe, 1999). In this paper, we present use case and class diagrams for the simulation environment to be implemented.

Use Case Diagrams

Use case diagrams are used in almost every software project. It is a great diagram that shows a good overview of the system. It shows the functionality that the system must provide. Use case diagrams are created to visualize the relationships between actors and use cases. They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues few more use cases might become visible and should be included in use case diagrams. The two main components of a use case diagram are actors and use cases, see Figure 1, (Ambler, 2003).



Figure 1. UML components of a Use case diagram: Actor and Use case.

An actor is someone or something that is external to the system, but interacts with the system. It represents a user or another system that must interact with the system under development. Actors are represented as stick figures.

A use case is a sequence of transactions performed by an actor in the system. It is an external view of the system that represents some action the actor might perform in order to complete a task. Use cases are shown as ovals.

Class Diagrams

Class diagrams are one of the most difficult UML diagrams to draw (Fowler and Scott, 1997). For better understanding of the class diagrams, more detailed document describing UML and object oriented principles is required. The scope of this paper doesn't permits for such a detailed description. This section gives an overview of a class diagram.

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. The purpose of class diagram is to show the static structure of the system being modeled. They show the existence of classes and their relationships in the logical view of a system. Class diagrams describe three different perspectives when designing a system: conceptual, specification, and implementation (Ambler, 2003; Fowler and Scott, 1997; Warmer and Kleppe, 1999).

A class is a collection of objects with common structure, common behavior, common relationships and common semantics. In UML, class is represented as a rectangle with three compartments: class name, structure (attributes) and behavior (operations), see Figure 2. A typical class diagram will have many classes. A class is connected to some other class in a class diagram. This connection is called a relationship.

The UML modeling elements found in a class diagrams include (Ambler, 2003):

- a) Classes and their structure and behavior
- b) Association, aggregation, dependency and inheritance relationships
- c) Multiplicity and navigation indicators
- d) Role names

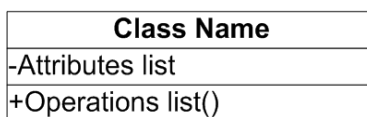


Figure 2. UML representation of a class.

3. Software Design for simulation environment

In various biotechnological systems, a bioprocess occurs in a bioreactor and involves the microbial species (Rani and Rao, 1999). The feed for the microbial species in a bioreactor comes in the form of substrate and nutrients. There are various types of bioreactor used (e.g. batch, fed-batch, continuous, continuous with recycle etc). Each bioreactor type has characteristics specific to it (Berber, 1996; Cinar et al., 2003; Dunn et al., 2003; Schugerl and Bellgardt, 2000). Depending on the substrate type, microbial species involved and bioreactor operational conditions; different kinds of products are produced in the bioprocess (Schugerl and Bellgardt, 2000). There are many kinds of mathematical models available which are used to simulate and predict the product spectrum (Dunn et al., 2003; Schugerl and Bellgardt, 2000). The idea of implementing a simulation environment is to create a software application where user can choose and specify the feed, bioreactor type, operational conditions, microbial species and model types. The simulation environment then predicts the product spectrum over a period of bioreactor operation. Here we present the conceptual, use case and class diagram (conceptual) to implement such kind of system. The implementing platform and language are left open, and can be chosen from C++, JAVA, MATLAB (Simulink) and other object oriented programming languages and platforms.

Conceptual diagram

The conceptual diagram of the simulation environment gives an overview of the various systems involved in the microbial bioprocess occurring in the bioreactor. It describes interactions of the systems involved. For the implementation of simulation environment, four sub-systems are identified, as shown in Figure 3: a) Feed system, b) Bioreactor system, c) Microbial system and d) Product system. The fifth system (Model system) is not shown in the diagram to reduce complexity of the conceptual diagram. The model system is the system through which a model is selected and specified by a user. It is through these models that subsystems communicate with each other.

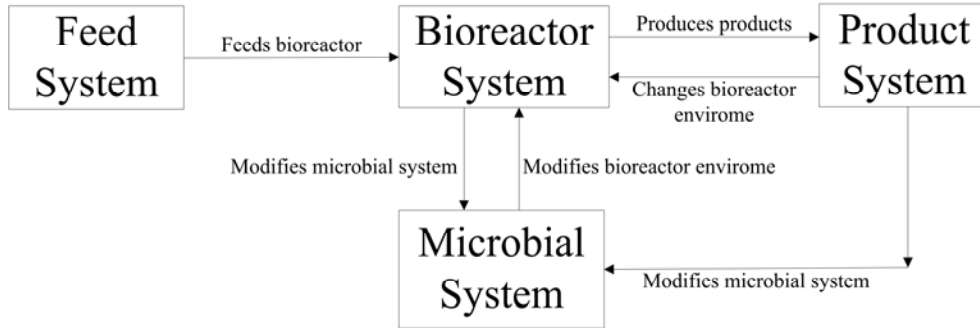


Figure 3. Conceptual diagram of a simulation environment.

Feed system stores the information related to substrate and nutrients being fed to the bioreactor. It feeds the bioreactor, and is the simplest relationship in the simulation environment.

Bioreactor system stores the information about the physical characteristics of the bioreactor and the initial operational conditions set by the user for bioreactor operation. Bioreactor system interacts with the product system by producing the products and hence updating the products concentrations in the product system. Bioreactor system also interacts with the microbial system and modifies the involved microbial species concentration and composition.

Microbial system stores the information about the microbial species involved in the bioprocess. It interacts with the bioreactor system by updating the biomass concentration in the bioreactor.

Product system stores the information regarding the products of the bioprocess. It interacts with the bioreactor and microbial system. Product system modifies the bioreactor system by updating the environmental conditions of the bioreactor. Product system also modifies the microbial system by product inhibition and hence updating the microbial species concentration and composition.

Use case diagrams

Use case diagram 1 (Figure 4) illustrates the interactions of a user with feed system. User sets the substrate type, substrate concentration, substrate feed rate, nutrient composition type, nutrient composition concentration and nutrient composition feeding rate.

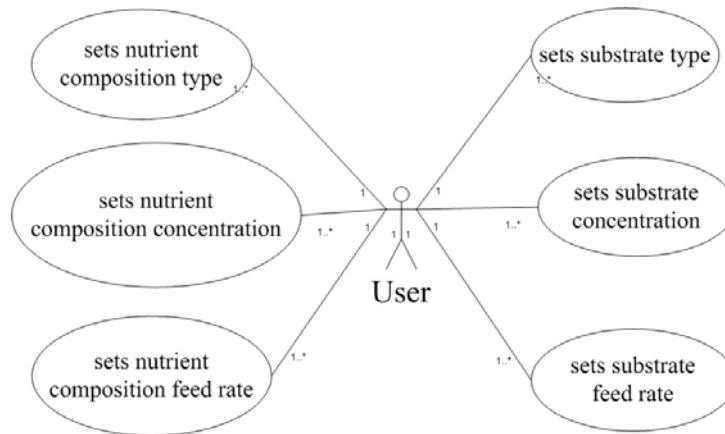


Figure 4. Use case diagram 1: User interactions with feed system.

Use case diagram 2 (Figure 5) illustrates the interactions of a user with the bioreactor system. User sets the bioreactor operation period, bioreactor type, bioreactor volume, bioreactor diameter, bioreactor height, initial oxidation reduction potential (ORP), initial temperature, initial pH and alkalinity.

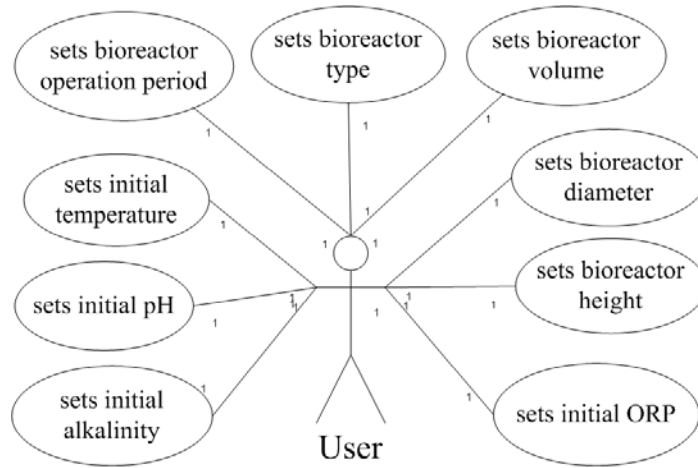


Figure 5. Use case diagram 2: User interactions with bioreactor system.

Use case diagram 3 (Figure 6) illustrates the interactions of a user with the microbial system. User sets the number of species involved in the bioprocess, initial quantity of each species, species specific growth and death rate.

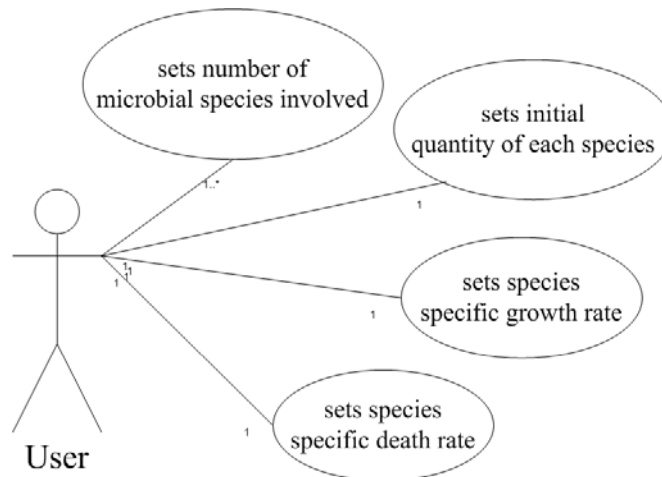


Figure 6. Use case diagram 3: User interactions with microbial system.

Use case diagram 4 (Figure 7) illustrates the interaction of the microbial system with the bioreactor system. Microbial system sets the biomass concentration in the bioreactor.

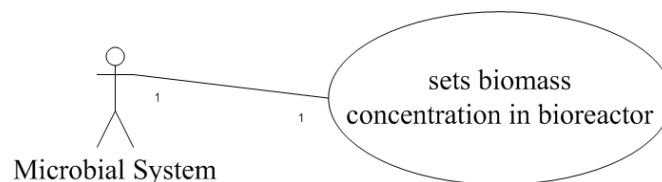


Figure 7. Use case diagram 4: Microbial system interaction with bioreactor system.

Use case diagram 5 (Figure 8) illustrates the interactions of the bioreactor system with the microbial and the product system. Bioreactor system modifies the microbial system by changing the microbial species concentration

and composition. Bioreactor system modifies the product system by producing specific gases, acids and alcohols.

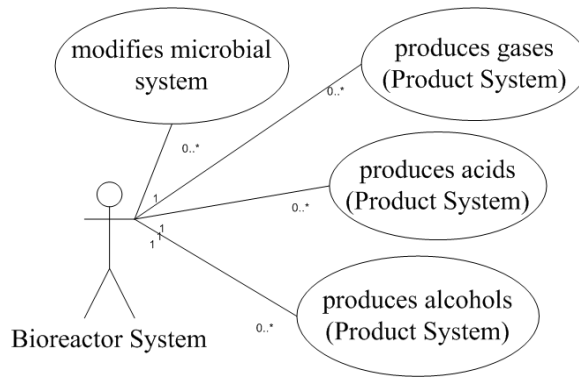


Figure 8. Use case diagram 5: Bioreactor system interactions with microbial and product system.

Use case diagram 6 (Figure 9) illustrates the interactions of the product system with the microbial and bioreactor system. Product system modifies the microbial system through product inhibition. The concentration of the products modifies the growth and death rate of the microbial species, and in turn modifies the microbial species concentration and composition. Product system modifies the bioreactor system by updating the temperature, pH, alkalinity and ORP.

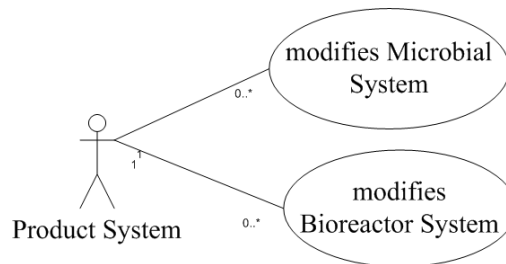


Figure 9. Use case diagram 6: Product system interactions with microbial and bioreactor system.

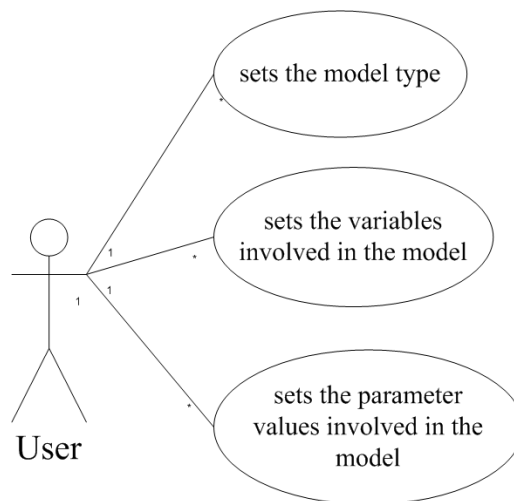


Figure 10. Use case diagram 7: User interactions with model system.

Use case diagram 7 (Figure 10) illustrates the functionality of the simulation environment, where user sets the

model type, variables and parameters involved in the model. It is these models via which the bioreactor, microbial and product system communicate and update each other.

Class diagram: Conceptual

Figure 11 presents the conceptual class diagram of the simulation environment to be implemented. The 14 classes identified for the simulation environment are: feed system, nutrient, substrate, bioreactor system, bioreactor, microbial system, microbial species, product system, gas, acid, alcohol, model class I, model class II and model class III.

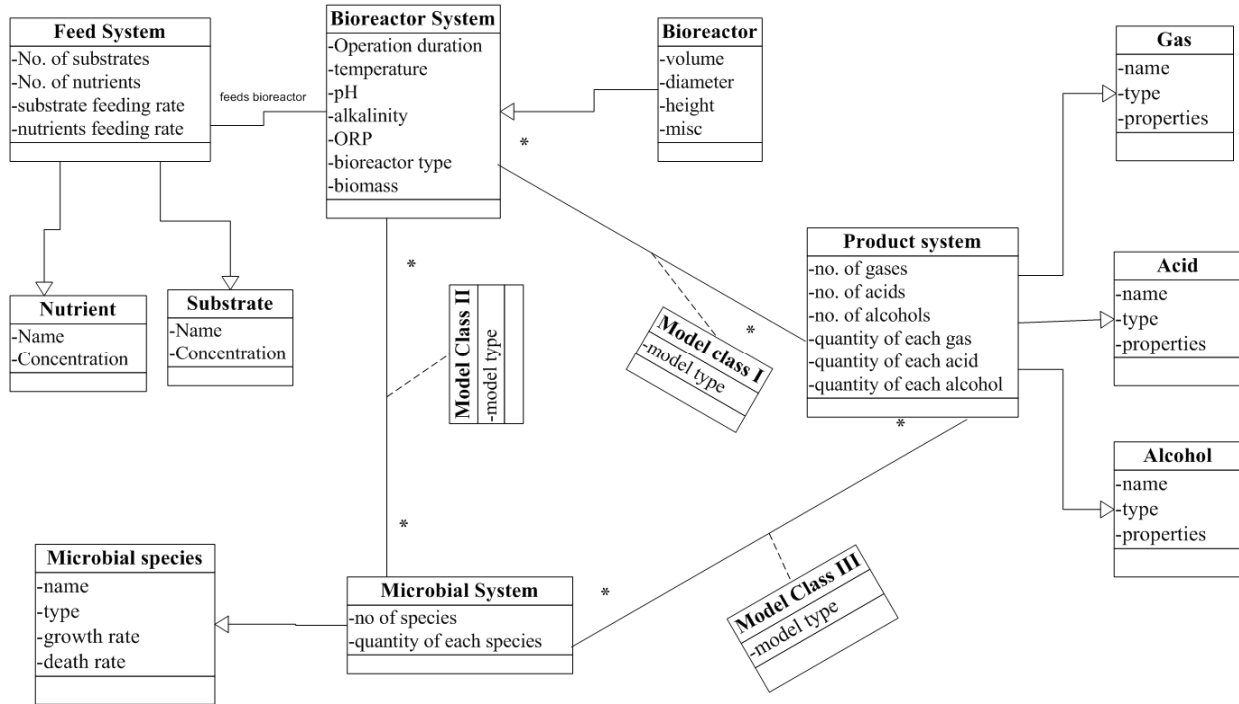


Figure 11. Conceptual Class diagram of a simulation environment.

In the class diagram, model classes are introduced via which bioreactor system, microbial system and product system classes interact with each other. Model classes store various kinds of mathematical models (e.g. differential equations based model, regression based models, neural networks based models and others). Each model class has different models based on the functionality and the interacting classes. The user chooses the model type, variables involved and parameters for each model via which the updating of simulation environment occurs.

The feed system class inherits the features from nutrient and substrate classes. It interacts with bioreactor system by feeding it. The bioreactor system class inherits the features from bioreactor class. Microbial system class inherits the features from microbial species class. Bioreactor system and microbial system classes interact with each other via model class II. Product system class inherits the features from gas, acid and alcohol classes. Product system class interacts with the bioreactor system class via model class I. Product system class interacts with the microbial system class via model class III.

4. Conclusion

In this paper, we have presented the UML based software model (design) to implement the simulation environment. The idea of simulation environment presented is to develop a software application where user can specify the feed type, bioreactor type, microbial species involved and the mathematical model types to predict the product spectrum over a period of bioreactor operation. The use case and conceptual class diagrams are presented for the simulation environment. The implementation of the software application is left open at this stage.

Acknowledgment. The authors acknowledge the support by HydrogenE (research project (2005 - 2008) under Academy of Finland). This work was also supported by the Academy of Finland, (application number 213462, Finnish Programme for Centres of Excellence in Research 2006-2011).

References

- Ambler, S.W. (2003). *The Elements of UML Style*. Cambridge University Press.
- Berber, R. (1996). Control of batch reactors: A review. *Transactions of the Institution of Chemical Engineers*. 74, 3-20.
- Cinar, A., Parulekar, S. J., and Undey, C. (2003). *Batch Fermentation: Modeling, Monitoring, and Control*. Marcel Dekker, Inc.
- Dunn, I.J., Heinzle, E., Ingham, J., and Prenosil, J.E. (2003). *Biological Reaction Engineering: Dynamic Modeling Fundamentals with Simulation Examples*. Weinheim: WILEY-VCH Verlag GmbH & Co. KGaA.
- Fowler, M., and Scott, K. (1997). *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley.
- France, R.B., Kim, D-K., Ghosh, S., and Song, E. (2004). A UML-based pattern specification technique. *IEEE Transactions on Software Engineering*. 30(3), 193-206.
- Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Guennech, A.L., Sunye, G., and Jezequel, J. (2000). Precise modeling of design patterns. *Proc. of the Third International Conference on the Unified Modeling Language*, York, UK, 482-496.
- Liu, J., Liu, Z., He, J., and Li, X. (2004). Linking UML models of design and requirement. *Proc. of the Software Engineering Conference Australian*, 329-338.
- Pressman, R.S. (1992). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc.
- Rani, K.Y., and Rao, V.S.R. (1999). Control of fermenters – a review. *Bioprocess Engineering*. 21, 77-78.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991). *Object-Oriented Modeling and Design*. Prentice Hall.
- Schugerl, K., and Bellgardt, K.H. (2000). *Bioreaction Engineering: Modeling and Control*. Springer-Verlag.
- Sevella, B., and Bertalan, G. (2000). Development of a MATLAB based bioprocess simulation tool. *Bioprocess Engineering*. 23, 621-626.
- Warmer, J., and Kleppe, A. (1999). *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley.