DECISION SUPPORT FOR BIOMASS FEEDSTOCK PRODUCTION ENABLED BY CONCURRENT SCIENCE, ENGINEERING AND TECHNOLOGY (ConSEnT)

BY

YUNG-CHEN LIAO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Agricultural and Biological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Advisers:

Professor Kuan-Chong Ting
Assistant Professor Luis F. Rodríguez

Abstract

In the U.S., more than half of all energy comes from fossil fuels (59.99 of 108.41 quintillion joules) and close to 50% of annual fossil fuel (26.55 of 59.99 quintillion joules) is imported (U.S. Department of Energy, 2010). Fossil fuels accounted for almost all carbon dioxide emissions (5814.4 of 5839.3 million metric tons) in 2008 (U.S. Department of Energy, 2009), which caused significant contributions to the accumulation of greenhouse gasses in the atmosphere. There is, thus, an urgent need to develop viable alternative energy options to address these concerns. It is because of this need that alternative energies have drawn such significant attention in the recent years. There are several potential alternative energy resources under consideration, but biomass-based energy is of particular importance given that it is currently available in large quantities. Specifically, lignocellulosic ethanol is considered among one of the leading alternative energy options in terms of sustainability and availability (Zhang, 2008), and it is expected to replace a significant amount of fossil fuels in transportation sector (Perlack et al., 2005).

The supply chain of biomass-based biofuels is generally comprised of three major steps: (1) biomass feedstock production, (2) biofuel production, and (3) biofuel distribution. The goal of the biomass feedstock production system is the preparation of stable, affordable, and continuous biomass dry matter supply to satisfy the targeted displacement of 30% of petroleum used in transportation sector (Perlack et al., 2005). In order to achieve this goal, an array of scientific discoveries and engineering developments must occur. The research described in this thesis seeks to ensure that an efficient pathway is utilized for the research and technology development of biomass

feedstock production systems, such that science and engineering research can occur in concert with one another, supporting progress in both endeavors. This has been termed a Concurrent Science, Engineering, and Technology approach. A web-based environment for decision support has been designed and implemented for the planning, design, and management of biomass feedstock production. Concurrent Science, Engineering, and Technology is suitable for studying the complexity of biomass feedstock production because it integrates a systems model directly with an informatics platform, which assembles the latest information generated by research teams, and a web-based interface for integrating and utilizing these two resources in a real-time fashion.

This web-based decision support system (DSS) has been named BPSys. It is programmed in the Java™ (Oracle Corporation, Redwood Shores, CA) and integrates the functions of various software packages including: Apache HTTP Server (Apache Software Foundation, Forest Hill, MD), Apache Tomcat (Apache Software Foundation, Forest Hill, MD), Drupal (Drupal Association, Kortrijk, Belgium), MySQL (Oracle Corporation, Redwood Shores, CA) and JFreeChart (Object Refinery Limited, Harpenden, United Kingdom). The DSS is divided into in two parts: (1) the graphical user interface (GUI) of BPSys, a Java™ applet, that will run on the user's local machine and enables the user to work with the analytical model; and (2) an array of server side supporting programs, Java™ servlets, that respond to requests from the graphical user interface and deliver information to the users. Through this platform, users can access the system via web browsers and design biomass feedstock production scenarios for analysis, retrieve the latest attributes describing the scenario, modify attributes, execute models, and save results. With the aid of the system, users can leverage the power of the simulation and analysis tools but are not bothered by how to build and run the models. By utilizing this

platform, the latest knowledge and information regarding the biomass feedstock production system can be leveraged more effectively at the system level, allowing for seamless development of new innovative systems.

Acknowledgements

It is impossible to fulfill this research thesis without the supports from many people. Many thanks to my two advisors, Prof. K.C. Ting and Prof. Luis F. Rodríguez, for their guidance throughout my graduate research and patient while I struggled to search possible solutions. I also want to thank my two other committee members, Prof. Alan C. Hansen and Dr. Yogendra Shastri, for their assistances and reviews on my thesis. I would like to thank the other people Dr. Konstantinos Domdouzis for providing his knowledge in using JFreeChart, and Tao Lin for helping me write and check my thesis. Finall, I would like to share this thesis with my family for their supports. Special thanks to my girlfriend, Shih-Fang Chen, for her encouragements and for the past three years.

Table of Contents

Chapter 1. Introduction

1.1 Potential Alternative Energy Systems

Energy consumption within U.S. can be divided into four major sectors: (1) residential, (2) commercial, (3) industrial, and (4) transportation, and more than 50% of the energy utilized for these sectors are from fossil fuels (U.S. Department of Energy, 2010). These statistics show that human activities heavily rely on fossil energy. However, two key factors, a depleted reservoir (Dresselhaus and Thomas, 2001) and a significant contribution to greenhouse gases emissions (Smeets et al., 2009; U.S. Department of Energy, 2009) make fossil fuel unreliable and have triggered a search for alternative energy sources.

Alternative energy has drawn significant attention and interest in the recent years due to three major driving forces: climate change, energy security, and rural economic development (Sagar and Kartha, 2007; Koh and Ghazoul, 2008). During the past few years, it is thought that increasing green house gases (GHG) concentrations, mainly generated from the usage of petroleum-based energy, is a major factor causing and severe climate changes (Smeets et al., 2009). Bioenergy is seen as a cleaner and more sustainable energy source, with the potential to ease the climate change (Sagar and Kartha. 2007). Energy security refers to "the availability of sufficient supplies at affordable prices" (Yergin, 2006) and depleting crude oil reserves, continuously increasing demands, and rising oil prices has urged oil-importing countries to seek energy independence via developing domestic alternative energy sources. The stimulation of rural economic development via the establishment of bioenergy industries is anticipated to be an additional benefit from the implementation of biofuel production. These three

motivating factors attract researchers around the world to seek alternative energy solutions for the replacement of petroleum-based energy.

Different regions, however, require solutions and resources suitable for their regional environment. Solutions should be geographically appropriate and use a combination of renewable energy resources. Among the potential alternative energy resources, biomass resources are widely available from both cultivated and waste materials involved in many processes.

1.2 Biomass Feedstock Production

If biomass-based energy is to replace petroleum-based energy within the coming decade, then the reliable provision of biomass feedstocks must occur (Perlack et al., 2005). Biomass-based biofuels may be one of the only available options for replacing a significant portion of petroleum-based transportation fuels. (Perlack et al., 2005). The production chain of biomass-based biofuel production is comprised of multiple subsystems beginning with producing biomass feedstocks and culminating with distributing biofuel from biorefineries to end users (Figure 1.1) (Giampietro and Ulgiati, 1997). Biomass Feedstock Production is the production of raw material for conversion to liquid fuels and the delivery of those feedstocks for biofuel production. Biofuel Production is the conversion of biomass into liquid fuels. Biofuel Distribution is the delivery of biofuels to retail outlets accessible to the public. This research work mainly focuses on the operations within in the first subsystem, Biomass Feedsotck Production (BFP).

Figure 1.1. Biofuels production chain.

According to Perlack et al (2005), biomass can be acquired via two major resources: forest resources and agricultural resources. There are three usage platforms: (1) electricity, (2) transportation fuels, and (3) biobased products. In particular, lignocellulosic ethanol converted from biomass is considered to be among the most substantial and sustainable alternative energy sources (Zhang, 2008); however, the primary remaining challenge for development of a BFP system is the preparation of stable, affordable, and continuous biomass dry matter supplies to satisfy the expected displacement of 30% of petroleum used in transportation sector. Beside adequate provision of materials, production of biomass must be affordable and sustainable.

1.3 Engineering Solutions for Biomass Feedstock Production

In order to deliver sufficient biomass at competitive prices, the integration of scientific discoveries and engineering designs from various disciplines into the BFP system is needed. Thus, a research scheme for the investigation of biomass feedstock production processes by integrating sciences, engineering, and technologies related to biomass production was proposed and funded within the Energy Biosciences Institute, a research collaboration between the University of Illinois, the University of California at Berkeley, and the Lawrence Berkeley National Laboratory (Ting, 2009). Thus, a research program in biomass feedstock production engineering has emerged including five different tasks: Pre-harvesting Crop Monitoring, Harvesting, Transport, and Storage, and System Informatics and Analysis task (Figure 1.2). The pre-harvest crop monitoring task will develop the technology to observe crop physiological state and field conditions to assist better management during growing energy crops. The harvesting task will develop harvest equipment including cutting and collecting energy crops from agricultural fields. The transport task will develop techniques and a strategy to move harvested biomass

between various locations. The storage task will develop methods for the preservation of the quantity and quality of biomass after harvest. The system informatics and analysis task will develop computer models at the system level for the integration of various BFP components and conduct analysis to assess the system performance.

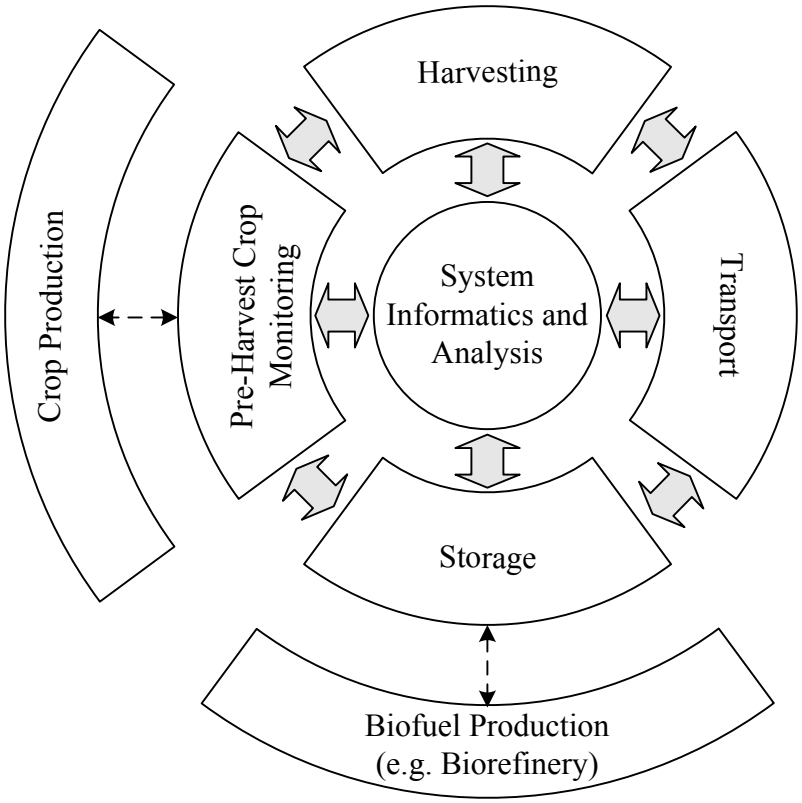


Figure 1.2. The scope of the BFP system (base image from Ting, 2009).

The gap between Crop Production and Biofuel Production is bridged by maintaining contact with investigators in both of these areas. The connection between the defined BFP system and crop production indicates that factors like species of energy crops, farm size, and farm location influence what future biomass feedstock production systems might be comprised of and how they will operate. The same is expected to be true of Biofuel Production including factors such as biorefinery capacity and conversion technology. The double-headed arrows between the five tasks have two meanings. The arrows between the tasks suggest that they are inter-related; and the arrows with the

system informatics and analysis task suggest that the four other tasks are considered concurrently via system-level databases and models.

1.4 Concurrent Science and Engineering

Ting (2002) proposed the concept of Concurrent Science and Engineering (CS&E), a system informatics and analysis platform for integration of information to conduct system-level analysis in "real-time". It has been implemented for system level decision support for phytomation systems and extended to advanced life support systems for long term space exploration (Ting et al., 2003). This thesis has implemented the concept of CS&E for BFP systems and the outcome is a **Con**current **S**cience, **En**gineering, and **T**echnology (ConSEnT)-based computational platform deployed via the Internet. To enable the ConSEnT platform, a data warehouse has been developed to manage data needed for systems analysis; a computer model, capable of integrating each of the subsystems and assess the performance of the whole system for decision support has been built. A user interface capable of using the model and database in a concurrent fashion to deliver decision support information has been implemented. As Ting (2002) suggested, CS&E should provide users with a web-based interface to access analysis tools.

Figure 1.3 shows the connections between components within the ConSEnT platform. It houses a database, a simulation and optimization tool, and the web-based user interface that can be accessed through a web browser. The double-arrowed lines between the user interface and other components represent an interface that can respond to user input. The simulation/optimization model and database are provided members of the research team described in Ting (2009). The overall objective of the work described in this thesis is to implement the user interface where decision support can occur, with

support from the database and simulation and modeling tools.

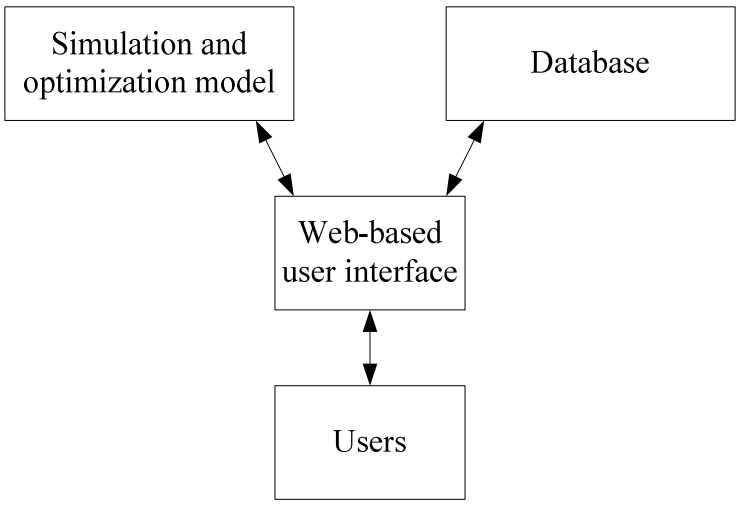


Figure 1.3. Schematic diagram of the ConSEnT platform.

1.5 Objectives

The ultimate goal is to demonstrate the utility of Concurrent Science, Engineering, and Technology (ConSEnT) for biomass feedstock production systems engineering. To achieve this goal, the objectives of this study are:

(1) develop a web-based computational environment for the provision of on-line access to databases and modeling tools,

(2) providing a user-friendly interface for supporting decisions, and

(3) establishing a framework for future expansion into a broader systems informatics and analysis tool in BPSys.

1.6 Layout of Thesis

This thesis is arranged into seven chapters. The operational challenges faced by biomass feedstock production systems are discussed in Section 2.1 of the literature review. Section 2.2 reviews the published research efforts in systems analysis of agricultural production systems. Section 2.3 describes the types of decision support systems and their applications. Section 2.4 shows the general server architecture for

Java-based web applications.

Chapter 3 presents the research methodologies applied in this thesis work. Sections 3.1 – 3.3 define the system functionality, components and architecture of BPSys. Section 3.4 reveals the performance indicators to measure the performance of BPSys. After developing a fundamental design for the system, section 3.5 describes the approach taken in this thesis to design and implement the graphical user interface of BPSys.

Chapter 4 provides the procedures for delivering decision support functionality including screen pictures of the graphical user interface. Chapter 5 describes the Java™ classes implemented for the decision support system, following the information flow. A case study is presented in Chapter 6 and Chapter 7 describes the conclusions and potential future work.

Chapter 2. Literature Review

Design of an optimized BFP system requires systematic research approaches. A decision support platform would facilitate the delivery of the outcome of the systematic research scheme as introduced in Chapter 1. Under the scheme, there are four operation tasks and a system task. The first two sections in this chapter will cover the existing work related to these four operation tasks, as well as the review of systems analysis method and its applications in agricultural production. The third sections will brief the backgrounds of decision support systems. The last section deals with the web technology that enables the development of web-based decision support system.

2.1 Biomass Feedstock Production System

The biomass feedstock production system is comprised of several distinct tasks and each major task may be divided into several sub-tasks. The four major tasks include: pre-harvesting crop monitoring, harvesting, transportation, and storage. The following subsections cover the research these four tasks one by one.

2.1.1 Pre-Harvest Crop Monitoring

Through remote sensing technology, pre-harvesting crop monitoring technologies have gathered crop information by collecting multi- and hyper-spectral imagery and correlating to biophysical and agronomic conditions (Pinter et al., 2003). This facilitates the non-invasive monitoring of crops and provides opportunity for precision crop management. Potential enhancements of management practices include irrigation, nutrient application, and pest control as well as the predictions of crop yields.

When growing crops, remote sensing can also improve an irrigation schedule by assessing crop water stress through thermal infrared (Alderfasi and Nielsen, 2001;

Wanjura and Upchurch, 2002) and is cost effective and affordable (Moran et al., 1994). In addition, remote sensing technology is capable of detecting nutrient distributions throughout the farm, identifying areas where nutrient stress may be likely to be found (Blackmer and Schepers, 1996; Blackmer et al., 1996). Protecting the crop from the invasion of weed competition, pests, or disease is also a potential benefit from remote sensing technology (Hanks and Beck, 1998; Pinter et al., 2003). Overall, pre-harvest crop monitoring is aiming at deriving an optimized crop yield by providing timely site-specific management tools. Aircraft-based imagery could help estimate the crop productivities and yield variations and make better and quicker decisions (Yang et al., 2000).

2.1.2 Harvesting

The harvesting operation is a necessary step by removing standing crops from the field and preparing them to be transported to storage or refinery facilities. This task consists of multiple operations including: cutting, collection, size reduction, packing, and in-field transportation (Sokhansanj et al., 2002; Hess et al., 2007; Domdouzis et al., 2009). Cutting breaks the physical connection of the crop with the ground. Size reduction is the action of chopping cut crops into smaller pieces for more efficient packing. In general, packing formats of biomass comprise bale (round or square), briquette, and pellets from the size-reduced crops (Prochnow et al., 2009). Infield-transportation is the process of moving the biomass from within the field to the roadside, where the trucks upload the packed biomass.

Harvesting schedule is an important consideration in the harvest operations. The harvesting schedule is strongly influenced by the time window when field operations may occur. In some regions, this is limited by the local climate conditions (Hess et al., 2007). The harvesting timing affects the biomass quality, and final energy yield (Lewandowski

and Heinz, 2003). The energy yield within Miscanthus fields harvested between December and March could decrease 14 – 28 % (Lewandowski and Heinz, 2003). In addition, the required storage area and the necessary transportation fleet is influenced by the packing density of the biomass (Rentizelas et al., 2009).

2.1.3 Storage

Harvest of biomass generally occurs within a fixed time during the year; however, provision of biomass materials to conversion facilities must occur continuously throughout the year. Thus, appropriate storage facilities are important to keep sufficient quantities of biomass available. The options for storing biomass include uncovered and covered on-farm storage and centralized storage facilities (Shastri et al., 2009; Sokhansanj et al., 2006; Browne and Hunter, 1998; Huisman et al., 1997). Centralized storage refers to a storage that serves multiple farms and may include a controllable storage environment. Uncovered on-farm storage has the lowest operating cost; however, significant biomass losses may occur particularly given higher moisture content and lead to decreasing energy conversion efficiency (Rentizelas et al., 2009). Decisions regarding storage operations may involve the fraction of biomass stored on fields relative to centralized storage facilities in order to strike a balance between the biomass quality and resource inputs. Proper locations of storage facilities are likely to decrease transportation costs for moving biomass from fields to the storage facilities and delivering from the storage facilities to downstream conversion facilities.

2.1.4 Transport

Many analyses of BFP systems have shown that transport operations are costly within the biomass supply chain. In previous studies on harvesting corn stover (Kumar et al., 2006; Rentizelas et al., 2009), transportation cost could range from 20% to 40% of the

total biomass supply cost. For energy crops, based on the reports by Shastri et al. (2009), transportation cost was accounted for 22 % of total cost of switchgrass production including harvesting, transport, and storage, and 13 % of miscanthus production (Shastri et al., 2010). Biomass transportation involves movement of harvested biomass from the fields to centralized storage, the fields to conversion facilities, and from centralized storage facilities to conversion facilities. The transport task can be separated into three operations: loading, transporting, and unloading (Kumar et al., 2006). The currently available carriers could be barge, rail, and road truck. The cost for each carrier varies with the load and the distance (Hess et al., 2007).

2.2 Systems Analysis

Kropff et al. (2001) pointed out that a factor, such as water content in crops, in an agronomic system is significantly influenced by multiple factors. For the optimization of using resource, multiple objectives of the system might be contradictory. Modeling and analysis tools, such as optimization and simulation, make it possible to consider the inter-relationships within various systems and subsystems simultaneously.

From the aspect of systems concept and analysis (Blanchard and Fabrycky, 1990), systems can be considered as a collection of several interrelated components with varying key parameters. The ability to predict the outputs (system performance) in reality and study the trends of changes in output associated with the varied input can provide essential insight into the system under study. The knowledge derived from such analysis can be very comprehensive and useful when designing a novel system or improving an existing one.

Systems approach has been applied to various agricultural and other systems. Considering biofuel production systems, for example, decreasing particle size has a

positive effect on the conversion rate of the biomass to ethanol (Torget et al., 1988; 1996). However, producing smaller biomass particles requires more energy consumption. It is possible that the energy required to produce the smallest particle sizes is higher than the energy content retained within the produced fuel; thus, such a system would be unsustainable, despite the fact that the conversion process is optimized. Utilizing systems analysis is seen as a promising method for the consideration of problems encountered in bioenergy system.

2.2.1 Systems Analysis Steps

Systems analysis has been applied to a variety of plant production systems, such as controlled environment plant production systems (Ting, 1997a; 1997b) and bioregenerative life support system for long duration space exploration (Rodriguez et al., 2003). The steps for conducting systems analysis consist of eight steps (Ting, 1997a; 1997b) (Figure 2.1).

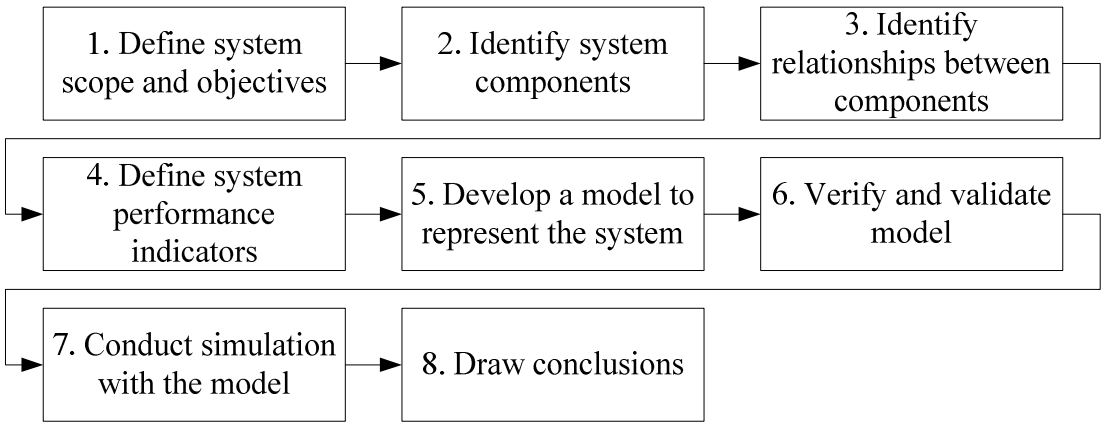


Figure 2.1. The eight steps of systems analysis.

The initial step is to frame the system scope and clarify its objectives. Given a defined system scope, the second step involves the discretization of the system into a representative set of individual components responsible for the functionality of the system. During this step, a necessary and sufficient set of attributes, and constraints for

describing each component should be established. Third, the inter-relationships defining the interactions between system components should be identified. Next, one should select system performance indicators capable of evaluating the workable system designs. There are several to choose from, depending on the perceived constraints and system objectives, such as the system workability, environmental sustainability, or resources allocations and managements. Fifth, a model is developed for analysis. Depending on cost, either a computerized mathematical model or a scaled physical system is chosen. A computerized mathematical model is chosen because it is frequently more cost-effective. In this case, a computer model is implemented as a software program, which depicts the behaviors of system components and their interactions as defined in the third step. It is a powerful tool to answer what-if-type questions. The remaining steps for systems analysis in this section are focused on conducting systems analysis with the computer models. The sixth step is verification and validation of the computer model. These two processes are to examine whether the model performs as expected and the outcomes are correct compared with the real data or the past experience on the similar system. Seventh, the developed model is then utilized to study the system responses under representative situations in an effort to seek advantageous configurations. And finally, the analysts will draw conclusions about the system and disseminate their results. Often dissemination may result in more research, in which case the process may repeat from any of the previous steps, or a new system design may emerge for implementation.

2.2.2 Applications for Agricultural Production

There have been many applications of system analysis for agricultural production systems in terms of economic performance, environmental impact, and routinely operation management. Some important examples are APSIM (McCown et al., 1996;

Keating et al., 2003), I-FARM (van Ouweker et al., 2003), POLYSYS (De La Torre Ugarte and Ray, 2000), DSSAT (Jones et al., 2003), and IBSAL (Kumar et al., 2006; Kumar and Sokhansanj, 2007). The systems studied include the development of models ranging from the consideration of crop growth environments—i.e. the farms—to the whole production system, including harvest, collection, transportation, storage, and crops delivery.

APSIM (McCown et al., 1996; Keating et al., 2003), the Agricultural Production Systems Simulator, is a modularized modeling framework consisting of multiple inter-connected models that predict how the biophysical process within a farm system and crops responds to practical management. The outcomes are represented by not only economic and ecological indicators of the farming system, but also crop productivity. APSIM is facilitated with a simulation engine that coordinates the modules responsible for simulating different components in the farming system. One unique feature in APSIM is that users are allowed to configure the combinations of the modules to model different farming system and it can be not only used to simulate the outcomes of farming system not only in 10-years term but also on a daily basis. .

The purpose of I-Farm (van Ouweker et al., 2003) is to promote the integrated farming system which grows crops and livestock. The calculation power of I-FARM is assembled from the existing models done by different researchers and programmed in multiple languages. The mixed model cluster calculates the system performances of a farming system, which grows crops and raises livestock simultaneously. I-FARM inter-relates ecosystem, economic, and community impacts from farming activities, land use change, and crop rotation at the farm scale.

POLYSYS (De La Torre Ugarte and Ray, 2000), the Policy Analysis System, is a

modeling framework that estimates the impact of the changes to the policy, environment, and economic conditions within the agriculture sector at the national level over long periods of time and records the results for each year. With the comparison of the recursively estimated results for each year with the changed conditions to the baseline scenario, POLYSYS is able to identify where the direct and indirect impacts are from and provide the traceable outputs. With these records of system state and performance along a simulation horizon, POLYSYS is useful to recognize the direct and indirect path in which the differences from the default initial system conditions may lead to final outcomes.

The decision support system for agrotechnology transfer (DSSAT) was developed to model the cropping systems (Jones et al., 2003). It integrates several modularized models, including soils, multiple crops, weather, pests, management practices and resource competition for water and light among soil, plant, and atmosphere. DSSAT provides an interface, which allows users to add a new crop. DSSAT has been applied to study the impacts of agronomic operations (i.e. fertilizer usage, pest control, and irrigation management) on the cropping system, combined with the economic and socioeconomic analysis for dealing with economic and production issues. It has also been used to study the cropping system behaviors under uncertain situation, such as the influence of climate change.

IBSAL (Kumar et al., 2006), the integrated biomass supply analysis and logistics model, is used to simulate the supply network of biomass feedstocks from the fields to a biorefinery. This network consists of four major activities: (1) collection, (2) storage, (3) preparation, and (4) transportation of biomass feedstock and is simulated with the consideration of the work rate of equipments and the capacity of storage facilities. It has been used to simulate the supply chains of switchgrass (Kumar and Sokhansanj, 2007),

assess biomass collection and transportation systems (Kumar et al., 2006), and conduct analyses for providing a corn ethanol plant with corn stover for power and heat production (Sokhansanj et al., 2010).

BioFeed (Shastri et al., 2009; 2010) develops a simulation/optimization modeling framework to minimize the cost for producing biomass over the biomass feedstock production system including growing crops to delivering to a conversion facility. The three performance indicators selected are supply chain cost, greenhouse gas emission and energy consumption. In contrast to the system level results, it also provides elaborate information on the operational level—i.e. the transportation fleet schedule and biomass harvesting schedule—which is provided on a daily basis. BioFeed has been applied to analyze the biomass feedstock production systems of switchgrass (Shastri et al., 2009) and miscanthus (Shastri et al., 2010).

2.3 Decision Support Systems

Instead of replacing decision makers, the objective of DSS is to provide them with valuable information helpful in solving complex problems. The theoretical studies of decision-making process were carried out in Carnegie Institute of Technology (Cyert et al., 1956; 1958; Simon, 1959).The recent applications of decision support systems (DSS) have taken many different forms for diverse purposes.

Presently, DSS is seen as a platform to provide decision makers organized and valuable information to solve complex problems more efficiently and effectively (Power, 2002). With the significant improvement in computing technology, decision support systems are capable of handling a massive amount of information and executing complex computations in a vastly shorter period than ever before. The basic components for building a computerized decision support system encompass three basic elements: user

interface, database, and models (Power, 2002). The user interface allows interaction with the DSS; the database provides the ability to handle massive data transaction including storage and retrieval of the information; and models aim to provide the capacity of processing information according to the user inputs.

2.3.1 Classification

Depending on the purpose and the information technologies utilized within the decision support system, they can be grouped into several types: 1) communication-driven, 2) data and document-driven, 3) knowledge-driven, 4) model-driven, and 5) web-based decision support systems (Nilsson and Ziemke, 2007; Power, 2002). Communication-driven decision support system is a system with interactive software tools to facility communication and consensus between distributed groups when finding solution (Power, 2002). It is also called a group decision support system (GDDS). Data-driven DSS is defined as a system that makes a large database available to users via an interface to retrieve and display data; document-driven DSS is a system which provides user with document retrieval capability to search and retrieve existing documents such as product manuals, government policies, and news (Power, 2002). Knowledge-driven DSS is in particular a system applying data mining technique or artificial intelligent tools to codify rules, relationships, and facts (Power, 2002). The codified knowledge can be stored and retrieved later to assist decision-making. Model-driven DSS provides users with computer models to integrate information and analysis tools to compare modeled results in order to aid better decision-making on complex problem (Power, 2002). Web-based DSS is a DSS that is accessible to users via Internet (Power, 2002). Shim et al. (2002) concluded that the usability of DSS would be dramatically improved accompanying with the advancements in web technology and the

dissemination of Internet connection.

Among these five, the web-based DSS is the most often developed in recent years, as the new opportunities emerge from the recent advances of the Internet (Power, 2002). Through the functionality of the Internet, decision support systems are able to share information with a much larger audience in a real-time fashion, overcoming the limitation of geography. Users can access decision support systems remotely, often through a web browser.

Any kind of decision support system described above can be implemented as a web-based DSS and becomes a very powerful tool for a broader use (Power, 2002; Shim et al., 2002). However, as pointed out by Bhargava et al. (2007), several considerations need to be kept in mind, both research opportunities and potential issues when utilizing web-based DSS. First, few research papers focused on the architecture design of web-based DSS, although there was a great amount of applications in web-based DSS. Second, the stability of the server connection is a significant challenge when building web-based DSS. Since data retrievals from multiple databases and model executions might require longer duration to complete, this problem should be addressed. Third, the technology suggestions made by empirical results for different types of DSS were deficient.

2.3.2 Application in Agricultural Production Systems

Agricultural production systems are very complex and require multidisciplinary expertise for their planning and implementation. There have been various DSSs developed for different purposes and built on different technologies. In the following paragraphs, some examples of DSSs designed for the agricultural production system are introduced.

The Aberdeen University Harvesting Decision Support System (AUHDSS) is a model-driven DSS developed for providing optimum solutions for the management of forest biomass supply chains (Mitchell, 2000). It deals with the operation of managing standing trees as biomass feedstock for the energy conversion, including harvesting of conventional forest, biomass storage, and the refinery (Mitchell, 2000). One key output from AUHDSS is an estimated delivery cost. Mitchell (1995) also developed a suite of DSSs that considered the cost of growing short rotation coppices including willows or poplars.

Some DSSs within agricultural areas are embedded within a geographical information system (GIS) to visualize the present and potential location of resources. Ayoub et al. (2007) developed a web-based DSS integrated with GIS technology, the general Bioenergy Decision System (gBEDS), which optimized forestry residue production for power plants and visualized the power plants on a map with the potential quantities of biomass. Frombo et al. (2009) also established a GIS-based DSS to assist the planning of logistic system of woody biomass for energy production. Instead of using GIS map to visualize geographical results, the GIS map here acts as an interface to sketch the harvested area and point plant location. The system will then calculate the optimized capacity of the plant as a function of energy production techniques—for example, gasification and combustion.

In addition to the DSSs for woody biomass production, DSS4Ag (Hoskinson, 2007) was developed for producing agricultural-derived biomass. The Decision Support System for Agriculture (DSS4Ag) considers an agricultural production system where grains and residuals as a biomass feedstock are produced simultaneously. DDS4Ag maximizes the grower's profits by optimizing fertilizer cost with the consideration of

historical yields, the current fertilizer cost, and the forecast crop price.

While the DSSs described so far focus on providing information for decision making by the calculating capability from computer models, there is a web-based DSS, Farm Decision Outreach Central (farmdoc), focusing on supplying integrated and timely information which covers multiple topics such as the finance, law, marketing, and policy (Irwin et al., 2004) for commercial producers through a web site. Apart from the information provision, a comprehensive suite of decision tools entitled Farm Analysis Solution Tool (FAST) dealing with financial issues is available for download (Irwin et al., 2004).

2.4 Java-based Web Applications

The functionality of a modern web site has changed greatly from the conventional web site hosting and serving static web pages. The original definition of a web site is no longer sufficient such that the term "web application" is created. The web applications refer to a client/server architecture in which web pages are created with dynamic content based on the user's requests from client side (Shklar and Rosen, 2009). During the creation of dynamic web page, the contents are queried from the sources according to the user's request. The dynamic web contents commonly are retrieved from the other sources such as a database. There are many computer languages, such as PHP, Java, and ColdFusion, capable of accessing the sources.

Java™, one of the alternatives, has been a superior language because of its effective integrations with the Internet environment. Java-based Internet protocols and technologies have been published continuously to strengthen the web-enabled abilities for different applications since Java™ was released by Sun Microsystems in 1995 (Kleijnen and Raju, 2003). For web application, a Java-based web technology, servlet, was

officially released in 1998 and maintained regularly. According to the released document (Mordani, 2010), servlet is a special Java™ class which is designed to handle HTTP requests and assemble HTTP responses. A Java servlet is executed and managed by a servlet runner, also known as servlet container, to generate the requested web contents (Shklar and Rosen, 2009; Mordani, 2010). Through servlet container, a Java™ servelt is able to receive HTTP requests from the browser running on the user's computer.

The general architecture of a Java-based web application and procedure of handling an HTTP request are shown in Figure 2.2 (Basham et al., 2004; Kleijnen and Raju. 2003). It contains five components: a web browser, web server, servlet container, and third-party software such as a database application (Basham et al., 2004; Kleijnen and Raju. 2003). Once the web server receives an HTTP request from a browser, it redirects the HTTP request to the servlet container. Upon the arrived of the HTTP request at the servlet container, the container, first, analyzes the request; second, identifies what program should handle the HTTP request; and third, calls the appropriate servlet to handle the request. Steps 4 and 5 are executed to retrieve necessary data from other third-party software packages, e.g. database, to create web contents. When the servlet has completed its run, it transmits an HTTP response, containing the requested items, back to the web browser.

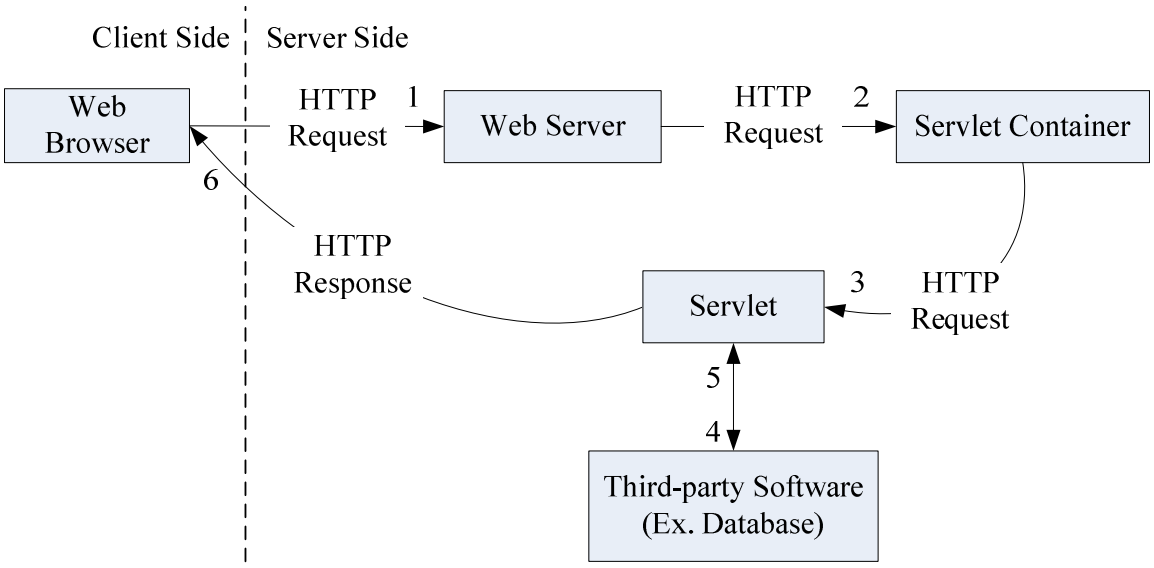


Figure 2.2. Architecture of web application for dynamic content web page.

Chapter 3. Design of Decision Support System

The eight steps of systems analysis are applied here to guide the development of a DSS for biomass feedstock production (see Section 2.3 for a brief introduction). They are discussed over the course of the next five chapters. The following narrative describes how these steps have been implemented. The first four steps of systems analysis are applied to the development of Biomass Production Systems, referred to as BPSys here, the central component within ConSEnT providing decision support functionality. The first step is to define the system objectives and scope and is presented in Section 3.1. In Section 3.2, the components comprising BPSys for achieving system objectives are discussed. The methodology for establishing relationships between system components is established and translated into the system architecture in Section 3.3. The performance indicators used to evaluate BPSys are defined in Section 0. The development of a model for these purposes involves a software engineering design approach for the clarification of the functionalities of graphical user interfaces within BPSys; this is described in Section 3.5. Verification and validation of the model is discussed first in Chapter 4 where the functionality and the usage of the user interface is described with the corresponding screen shots and subsequently in Chapter 5. A case study is presented in Chapter 6 and conclusions are presented in Chapter 7.

3.1 Objectives of BPSys

BPSys is a component of the concurrent science, engineering and technology computational platform, ConSEnT. There are two primary objectives: decision support, and communication. Decision support refers to allowing users to access the resources of BPSys including the provided model, BioFeed (Shastri et al., 2009; 2010), and database

through web browsers. Communication refers to providing the research team with a web site to introduce research work, publish results, and discuss related topics with other researchers. The components which are capable of serving the objectives are introduced in Section 3.2.

3.2 Components of BPSys

This section describes not only the necessary components for achieving the objectives, but also the suitable software packages. Tables 3.1 and 3.2 itemize the major tasks necessary to achieve the objectives of the web site and the DSS respectively. First, the web technology that supports the tasks in Table 3.1 is examined in Section 3.2.1. Then,the components for the tasks listed in Table 3.2 are discussed in Section 3.2.2.

Table 3.1. The four major tasks for the objective of communication.

Task	Description
1	Provide administrative tools for authorization and management of users
2	Allow users to post new articles
3	Allow users to leave comments
4	Allow users to upload files to be shared with other users

Table 3.2. The eight major tasks for the objective of providing decision support.

Task	Description
1	Allow users to select suitable models from the server
2	Retrieve input data from the database associated with the selected model
3	Allow users to modify the input data of the models
4	Allow users to launch the execution of the models on the remote server
5	Provide users with tools for analyzing and saving generated results
6	Provide users with the graphic tools for displaying information efficiently
7	Provide users with an easy-to-use graphical interface for tasks 1 to 6
8	Provide a client side application executable in a web browser

3.2.1 Communication

The listed tasks in Table 3.1 are commonly seen in current active social networking websites and several well-developed web content management systems

(WCMS) can deliver such functionality, given the support of a web server and a database application. WCMS are software tools for assisting website development and maintenance (Mooney and Baenziger, 2008) and have been used to establish websites for academic organizations and research groups (Mooney and Baenziger, 2008; Cao and Yu, 2010; Coombs, 2009; Das et al., 2009). The key components necessary for constructing a WCMS include a web server and a database. Several WCMS are available on the market; Drupal™ (The Drupal Association, Belgium) has been chosen for the communication platform because of its well-known extensibility and its large amount of available plug-in modules, providing additional functionality. This tool supports the convenience of allowing users to develop their website via a web browser. The web server and database selected to support the WCMS were Apache Http Server (The Apache Software Foundation, Forest Hill, MD) and MySQL database (Oracle Co., Redwood Shores, CA), respectively. These were chosen because they were compatible with Drupal™ and the Linux-based Operating System, Ubuntu 8.04. The following version numbers for the selected software components were Drupal 6.20, Apache HTTP Server 2.2.8_1ubuntu0.19, and MySQL 5.0.51a-3ubuntu5.8. These tools were organized together to establish the decision support system.

3.2.2 Decision Support

When considering the components for decision support, they can be effectively classified into two categories: the client-side components, which run on the user's computer, and the server-side components, which run on the server. The client-side components that provide the web-based user interface of the ConSEnT platform are described in Section 3.3. Via the web-based user interface, the user can access various resources on the server by communicating with server-side components. The environment

where the client-components are executed must be platform-independent and executable in an HTML page. Platform independence is necessary because the client environments can be quite diverse. Java™ has been selected to program the user interface because it possesses superior cross-platform compatibility, acceptable graphical user interface design, and powerful web-enabled applications, such as the Java™ applet which is used to develop the client. A Java™ applet is a small application that can be embedded and distributed directly with an HTML web page. When the web page is opened in a web browser, the Java™ applet will be executed on the client system and is displayed as a part of the web page. Java™ is also a suitable computer language for the server-side programs.

The architecture of a Java-based web application was introduced in Section 2.4 and can be taken advantage of to design the decision support application. The remaining choices are to select the web server, servlet container and the necessary third-party software. Because the web server application, Apache HTTP Server, chosen previously is unable to work directly with Java™ servlets, a servlet container will need to be added and chosen carefully to make sure that the container can be called by Apache Http Server to run a servlet when it is needed by the HTTP request. Apache Tomcat (The Apache Software Foundation, Forest Hill, MD) was selected because it can work with Apache HTTP Server by enabling the connecting module, mod_proxy, which is included with Apache Http Server (Apache Software Foundation, 2010). Within the module, the developer customizes a list specifying precisely what servlets will handle which HTTP requests. When this module is enabled, mod_proxy will be called to redirect the HTTP request to the proper application.

In order to build a web application for decision support purposes, a database for

data storage and software execution model is necessary. MySQL is again selected for data storage. The software package called by a Java™ servlet to execute models is highly dependent on what software package the model is programmed by. In this case, GAMS, the General Algebraic Modeling System (GAMS Development Corporation, Washington, DC), is used to run the model as BioFeed (Shastri et al., 2009; 2010) is coded in this language and the model will be utilized to provide decision support. A location is necessary to host the model, keep user records, and store results for BPSys. In the next section, the architecture of BPSys and the collaboration between the selected software packages are introduced.

3.3 Relationships between Components of BPSys

After defining the system components, the next step of systems analysis is to establish the relationships between system components. The relationships between the components are translated into a layered system architecture for BPSys and a data flow between these components under different situations. The first subsection will discuss the layer structure, the components within each layer, and general procedure for handling user requests. The following subsections will display the data flow between the system components under different use cases.

3.3.1 Layered Architecture of BPSys

The layered architecture of BPSys is displayed in Figure 3.1. The components in Figure 3.1 are classified into the same layer because of their functionalities. From bottom to top of the presented architecture, Layer 1, containing DSS Website (A) and the BPSys Client (B) is users interact with the system and where all requests are initiated; Layer 2, containing Apache HTTP Server (C), mod_proxy Module (D), Apache Tomcat (E), and Servlets (F), is where of the requests are received and processed; and Layer 3, containing

Drupal (G), MySQL (H), BPSys Folder (I), and GAMS (J), is where data are sourced for users requests. DSS Website refers to the exhibit of research context and related discussions. Generally, each request is processed through a pathway moving from Layer 1 to 2, 2 to 3, 3 to 2, and 2 to 1, although this may vary depending on the specific scenario. These relationships will be elaborated in the next four subsections 3.3.2 – 3.3.5 with a description of information flow between these components. The primary use cases that the system has been designed for include: web content management, data management, file input and output, and model execution. The first use case involves the DSS Website, while the other three are for the BPSys client.

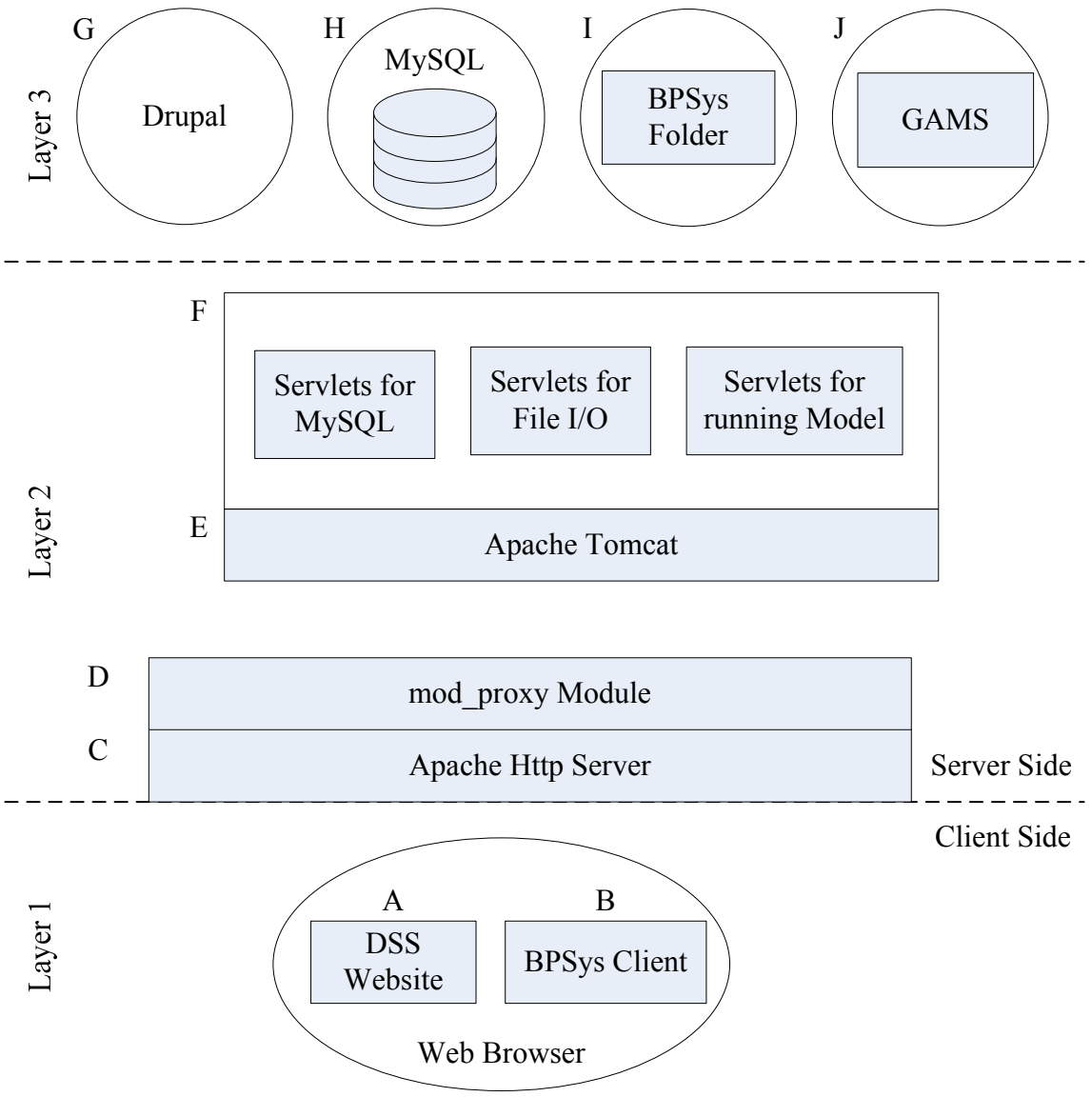


Figure 3.1. Layered system architecture of BPSys.

3.3.2 Web Content Management

Figure 3.2 shows the data flow related to browsing the website and posting new web content. The numbers beside the arrows represent the order of the data flow. When the Apache HTTP Server receives an HTTP Request regarding the website, the request is transferred over to Drupal. Drupal will gather the data from the MySQL database needed for the requested web page and create the web page in steps 2 – 4. After the creation of the web page, it is returned through the Apache HTTP Server.

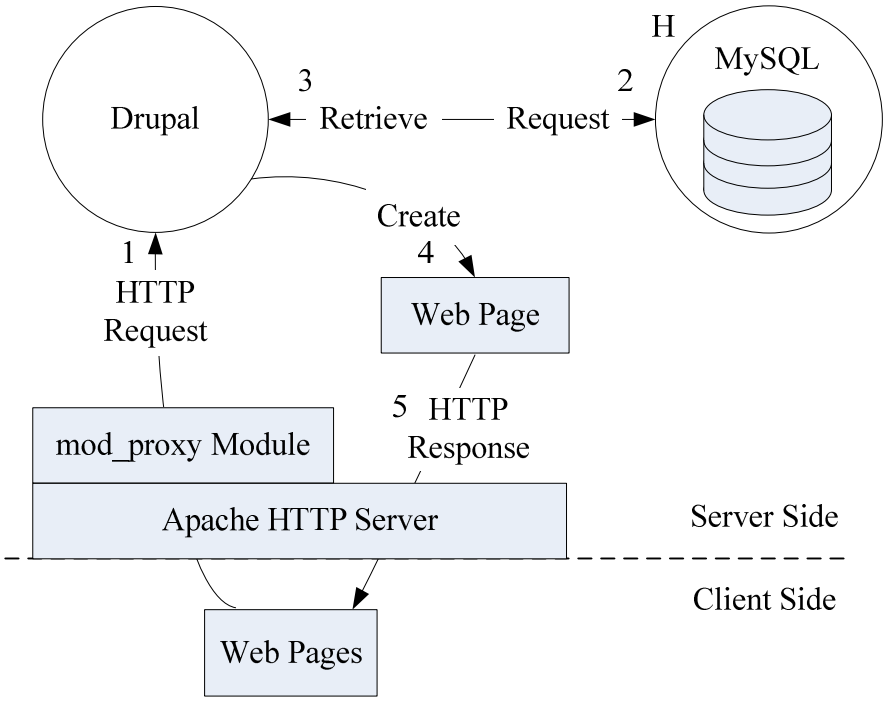


Figure 3.2. The information flow of web content management.

3.3.3 Database Management

The user will interact with the BPSysClient to browse and edit data stored within the database. The BPSysClient requests the data from the MySQL database for these actions. The mod_proxy module redirects the HTTP Request to Apache Tomcat, which calls servlets prepared for MySQL for data management (Figure 3.3). In general, there are two types of data management: data retrieval and data storage. The response from the MySQL database also has two parts: the retrieved data, and the message of whether the data are stored successfully. The response is put within an HTTP response, which is returned to the client side.

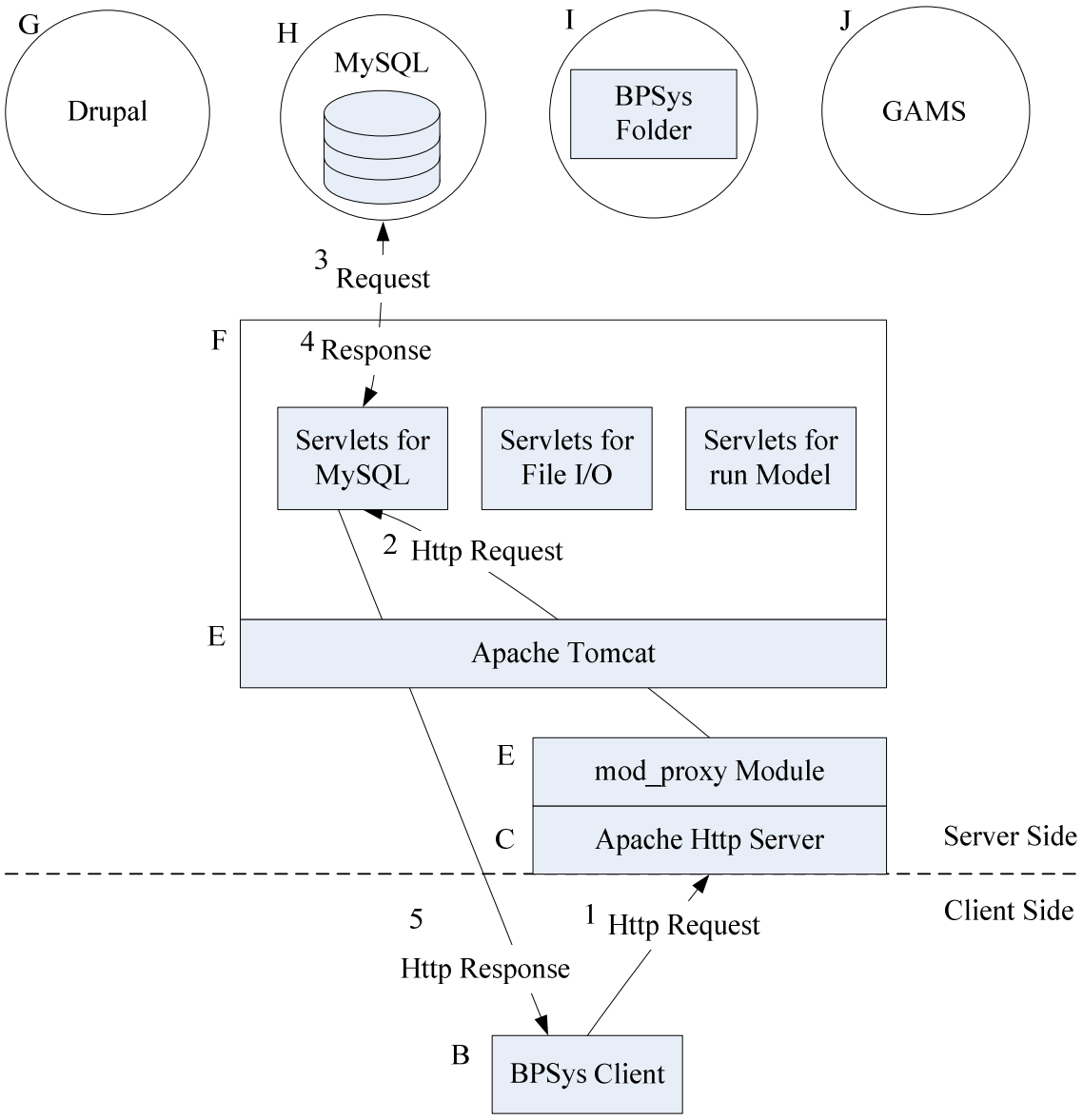


Figure 3.3. The data flow of MySQL access.

3.3.4 File Input and Output

Servlets for file input and output have been designed to deal with the input and output of the files. In this case, servlets designed for file input and output are called from Apache Tomcat and directly interact with the server side filesystem.

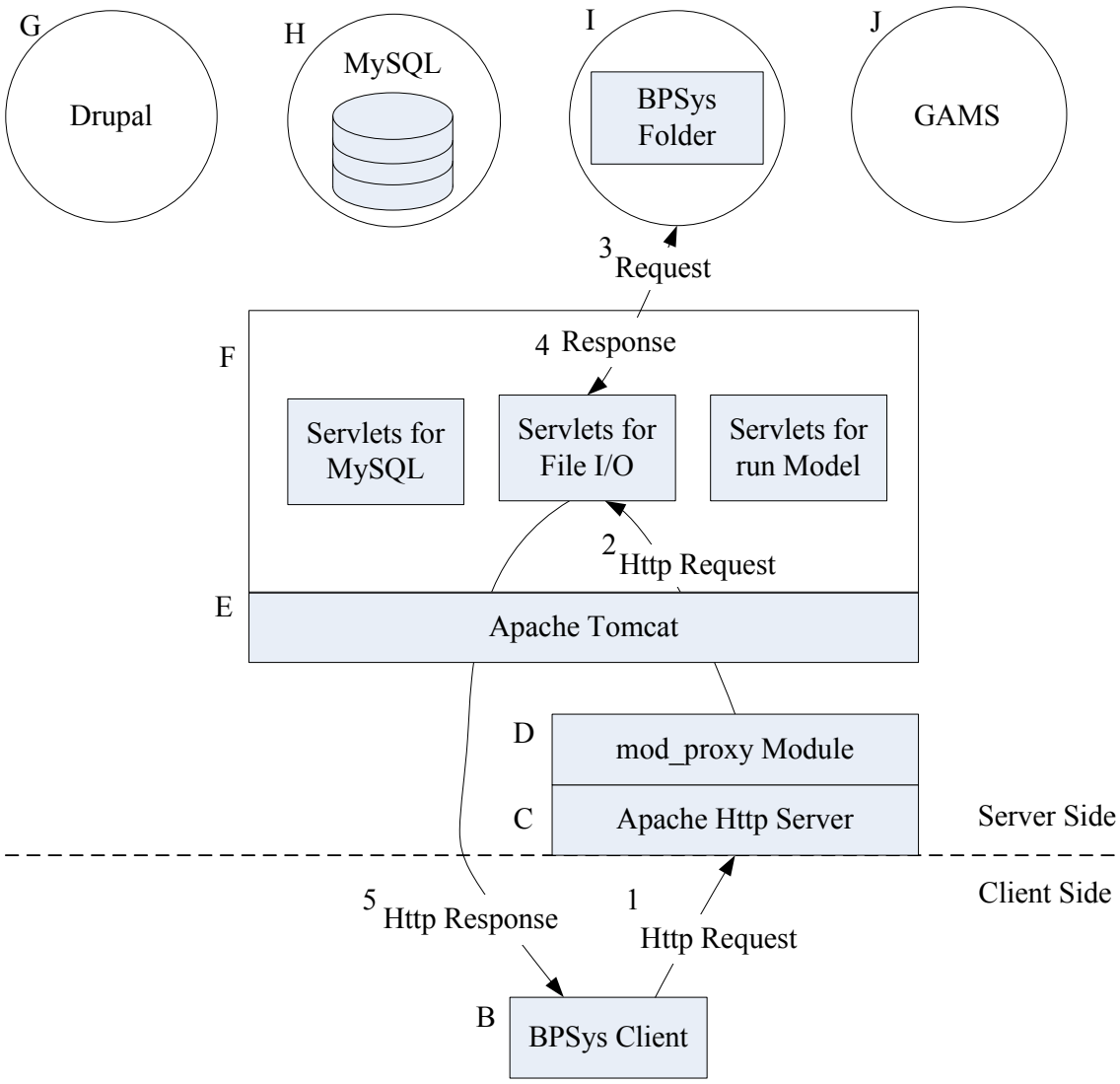


Figure 3.4. The data flow of file input and output.

3.3.5 Model Execution

Data flow for execution of models is depicted in Figure 3.5. Servlets for running the model have been designed for supporting this task. These servlets execute the model back end in response to calls from Apache Tomcat. The servlet waits for the response from the models and responds when the problem is solved. Once the model has completed successfully, the servlet returns the response to the BPSys client.

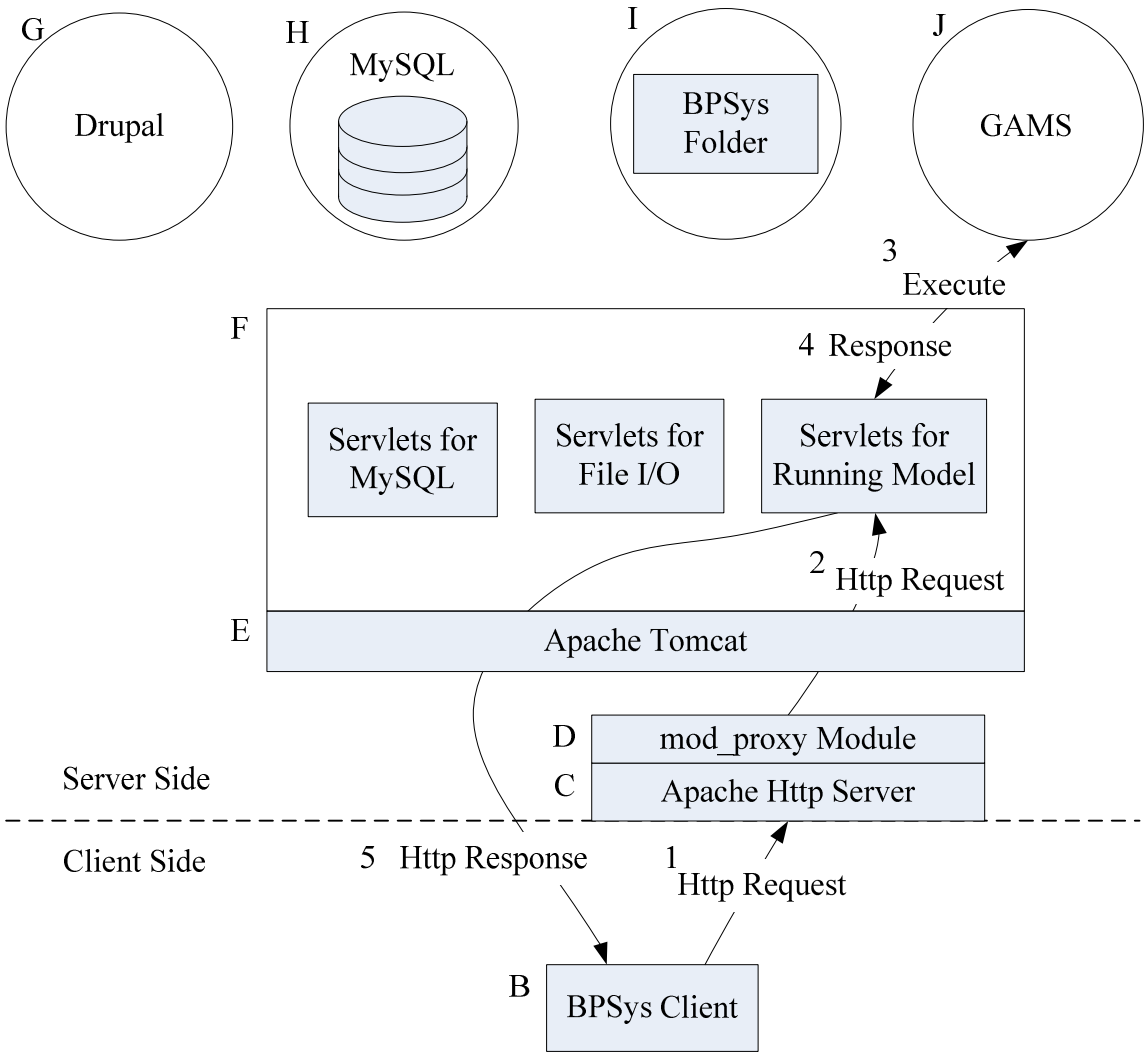


Figure 3.5. The data flow of model execution.

3.4 Performance Indicators of BPSys

The previous three sections have applied the first three steps of systems analysis to describe the system objectives, system components, and relationships. The next step of system analysis is defining the system performance indicators before the implementation. The purpose of this research work is to aid decision support on equipment selections and technology development via a graphical user interface. Therefore, the performance indicators should measure the effectiveness and usefulness of the information provided

for decision-making, the effectiveness of the presentation of information, and the friendliness of graphical user interface. A user experience evaluation of BPSys should be conducted to cover the three measurements because it is very likely that the effectiveness of the total experience will be affected negatively by a cumbersome interface design and/or inappropriate presentation of information. An evaluation of the interface and the user experience has been designed, and will be implemented in the future, but is not presented as part of this thesis.

3.5 The User Story Method

Given that user friendliness is a key performance indicator, the design of the user interface becomes crucial for a DSS. To address this challenge, a standardized approach for interface design has been implemented here. This section describes one popular and practical programming method, the user story, as it is applied to guide the design and implementation of computer programs (Steinberg and Palmer, 2004). The user story method is applied to develop the BPSys user interface and its functionalities. The intent of the user story method is to encourage the software developer to program an application from a user's perspective, not the programmer's, so that the product actually meets customer's demands and is useful (Cohn, 2004).

The method starts from a description, which might be a sentence or a short paragraph, to express how the desired task should be accomplished in the user's language before the programmer starts to program. The written description is called a user story. At the beginning, the first version of the description is often too general and vague to implement for the software programmer. The role of the software developer is to help the user clarify their demands and break down their first draft into multiple descriptions of more basic tasks. This requires an iterative process to make the description classified,

clarified, and fundamental enough for the programmer to implement. The programmer begins to develop the application after a detailed user story appears and makes sure that every user story is met in the application delivered to the user later.

In this thesis, the users' opinions to guide the user interface design are from Prof. K.C. Ting, Prof. Luis Rodriguez, Prof. Alan Hansen, Prof. Yogendra Shastri and other team members and the user stories were defined within the team. The sponsor of this research program, the members of the Energy Biosciences Institute, is the envisioned user. The requirements in Table 3.2 are used as the draft user stories, placed into two categories and extended into a detailed description. These requirements are divided into two main categories, in which one is about system analysis and the other is about database management. Section 3.5.1 and 3.5.2 describe the user stories for these two purposes. There are ten main stories for the system analysis from the user's angle. Among the defined ten stories, five (3, 4, 5, 7, and 8) are divided into multiple subtasks. Generally, these user stories are listed in order of complexity of implementation and in some cases they leverage functionality from one another. For example, before selecting the model (the story 1), the story 2 cannot be completed because the default input data are unknown. Also, the user is not able to modify input data (the story 4) without the display of input data. In the next chapter, the demonstration of how the graphical user interface supports these stories will be introduced.

3.5.1 User Stories for System Analysis

1. Select models

   The user can select the model from the list showing the available models on the server.

2. Retrieve the default input data from the server

The application retrieves the default input data from the server after the user selects the model.

3. Visualize scenario

The user can see the flow chart of the modeled system and scenario and modify the input data via the visualization.

3.1 Flow chart of the modeled system

The flow chart is composed of box shaped icons and lines representing the tasks and the sequences of the modeled system.

3.2 Display the input data

The application displays the default input data retrieved from Task 2 in assorted tables organized by the task in the modeled system.

4. Modify input data of the models

The user will have two options for modifying input data: modifying the value of an attribute describing pieces of equipment included in the analysis or the modifying the combination of the equipment.

4.1 Modify the value

The user can directly click on the cell with a table that needs to be modified and type in the desired value with the keyboard.

4.2 Modify the combination

The user can directly click a check box to select and deselect the equipment used in the model.

5.Execute model

The user can execute the model on the server from their browser. The model can be executed once or in batches.

5.1 One time execution

The user can execute the model based on the set of input data once.

5.2 Batch execution

The user can execute the model multiple times while varying input data to observe the impact on the results.

6. Retrieve the result

The result is sent back from the server to the client automatically after the completion of the execution.

7. Review the result

The user can observe the results in the useful pre-defined formats.

7.1 Display the result in tables

The user can view the result via multiple tables to displaying the values of various performance indicators.

7.2 Visualize the result

The application can create figures to illustrate the content visually.

8. Result management

The user can save a copy of the result on his or her computer and open the saved files on BPSys.

8.1 Save result

The files are saved to the user's local file system.

8.2 Read result

The user can read the result via a text file editor or spreadsheet editor and run further analyses.

9. Parametric analysis

The user can select independent variables to vary and the corresponding dependent variables and create output chart to view the relationships graphically.

10. Auto save

BPSys will automatically keep track on the user's activity and save the every input information and result.

3.5.2 User Stories for Database Management

BPSys will require information provided by research collaborators to enable ConSEnT, thus a MySQL database was included for this purpose. It is expected that BPSys will provide the user with a graphical interface to work with MySQL. The user stories for database management include five simple functions: view database, upload data, download data, delete data, and define variables. The descriptions are as follows.

1.View database

1.1 View the databases existing in the MySQL server

The user can see the database list existing in the MySQL database by clicking on the server icon.

1.2 View the tables existing in a database

The user can see the table list in the database by clicking on the database icon.

1.3 View the contents of a table

The user can view the content stored in the table displayed in a table by clicking on the database icon.

2. Upload data

2.1 Create a new table in a database

The user can create a new table in the selected database in MySQL

database.

2.2 Add column(s) into an existing table

The user can add a new column in the selected table. The user can edit the column by the field name and the variable format.

2.3 Add record(s) into an existing table

The user can add a new record in the selected table. The user can type in the data or use the common copy and paste commands to speed the process.

2.4 Update the value of a cell

The user can modify an existing record in the selected table. The user first selects the cell by directly clicking on the cell and then typing the updated value. If the user is certain about this change, the user can click on the “Submit” button to commit.

2.5 Import data from a comma-separated variable (CSV) file

The user can import a great amount of records into a database table by reading a CSV file.

3. Save data

The user can save the whole table onto his or her computer in CSV format.

4. Delete data

4.1 Delete database

The user can delete an existing database in the MySQL database.

4.2 Delete table

The user can delete a table existing in the database.

4.3 Delete column(s)

The user can delete columns, which are redundant.

4.4 Delete records(s)

The user can delete records from an existing table.

5. Define Variables

The user can define the primitive variable formats for each column in the table.

The variable formats should include text, float, date, and integer.

Chapter 4. Software Implementation

This chapter will introduce how to use the user interface. Section 4.1 – 4.3 covers the basic operations of the initialization, authentication, and graphical components. The remainder of the chapter provides instruction regarding how to use the implementation of the user stories. All of the previously defined user stories, except for the user story 10, Auto save, have been implemented.

4.1 Run BPSys

BPSys is programmed in Java™ as an applet. Being an applet, it can be executed within a web page by adding an applet tag with the appropriate link in the HTML source code of the web page (Zakhour et al., 2006). When a user accesses the web page tagged with BPSys in step 1 the web browser tries to run BPSys. Figure 4.1 outlines the process of running BPSys. First, before BPSys executes, the system checks whether the appropriate Java™ Runtime Environment (JRE) libraries needed to run the applet are available. If the current JRE on the client is up-to-date, the applet is downloaded and BPSys is initialized. Once BPSys is executed, a window asking for the user to authorize the execution of an application downloaded from the Internet is displayed (Figure 4.2). BPSys is executed immediately after user approval. If the current JRE is not up to date or it is not installed on the client side, the user is redirected to the Java™ JRE library download page. Once the latest JRE is installed, the user is redirected to try again. It is required to acquire user authorization because BPSys uses several file I/O classes which save and open files on the user's local machine and such functionality is prohibited in all default applets to prevent any intentional damage to the client machine. The only way to solve this problem is to request the permission of the user, making the user aware of this

potential security risk.

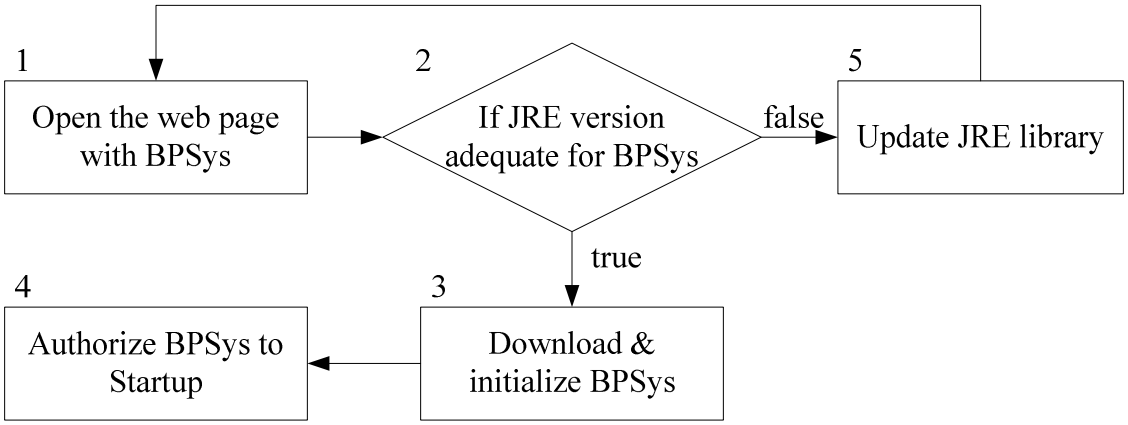


Figure 4.1. The steps of executing BPSys through a web browser.

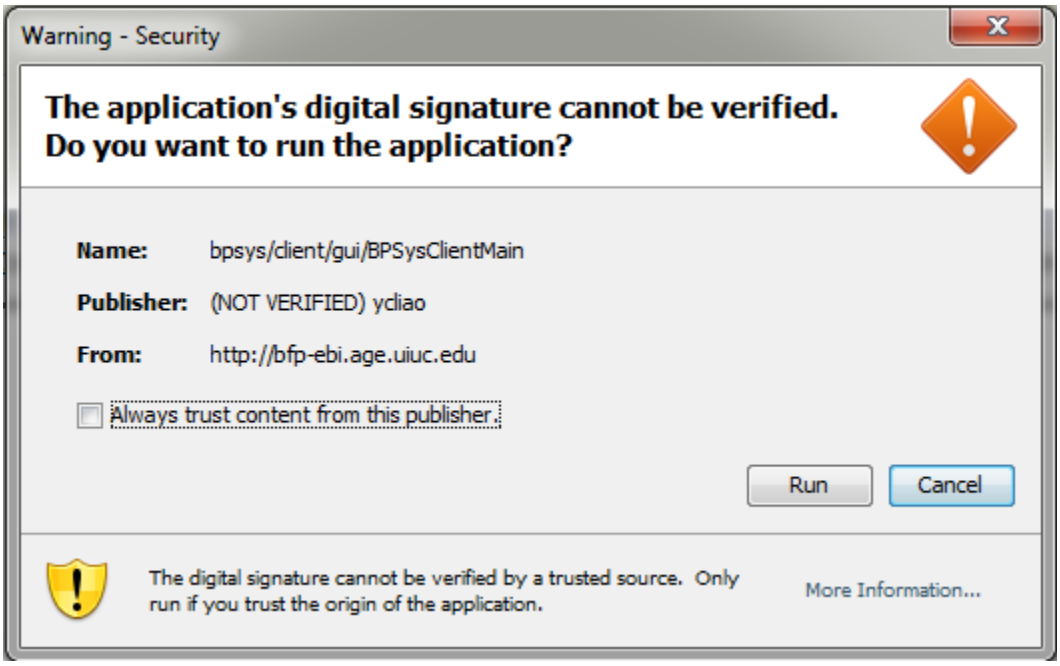


Figure 4.2. The authorization window to run BPSys.

4.2 Login to BPSys

Figure 4.3 displays a screen shot of BPSys after successful initialization. The first step after initializing BPSys is to login. Users have to authenticate by entering their user name and password in a window which entitled “Enter User Name”, highlighted by the red rectangle in Figure 4.3. The username is used to access the data in MySQL server and create a root folder for the user, which will be used for running model and storing any

generated data. Once the user enters the correct username and password, the window "Enter User Name" disappears and BPSys is ready to be used.

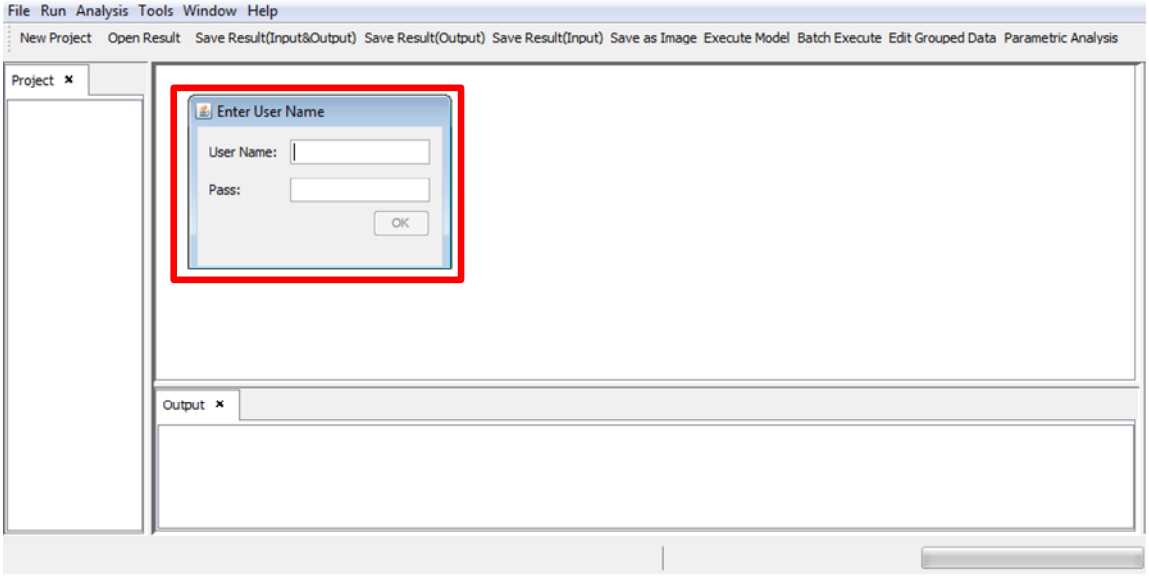


Figure 4.3. Login screen shot of BPSys.

4.3 Components of the BPSys GUI

Figure 4.4 is a screen shot of the BPSys after a successful login. Before the introduction to the functionality and purpose for each component of the graphical user interface (GUI), it might be helpful to introduce the components of the GUI and their purpose. The components are categorized into two types: functional components and informational components. Functional components are programmed to have responses when they detect mouse clicks or keystrokes, while informational components display information or reflect system status, such as a progress bar. The following sections will describe the use of these two types of components in BPSys in turn.

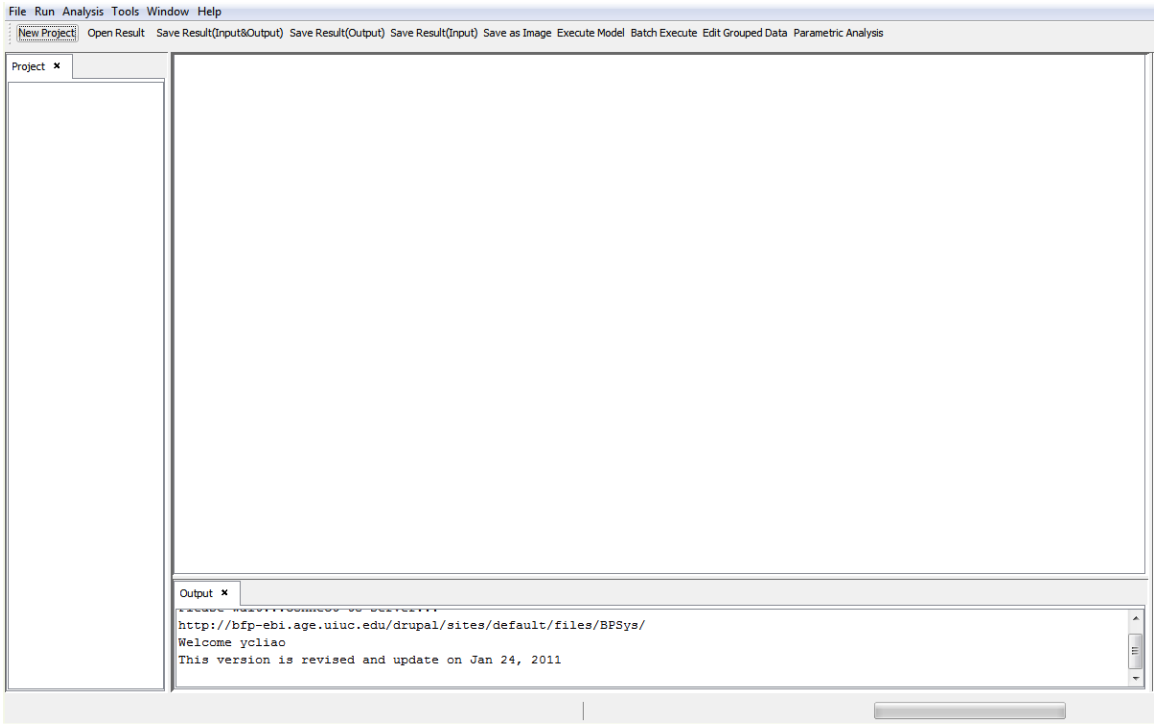


Figure 4.4. The initialized screen shot of BPSys after successful login.

4.3.1 Functional GUI Components

The functional components are underlined in red and blue and labeled "1" and "2" in Figure 4.5. They provide key interfaces for users to initiate tasks on BPSys. In component 1, underlined by the red line, there is a menu bar where the main functions in BPSys can be found and are gathered into key categories; in component 2, underlined by the blue line, these is a tool bar which contains frequently used buttons of key functions.

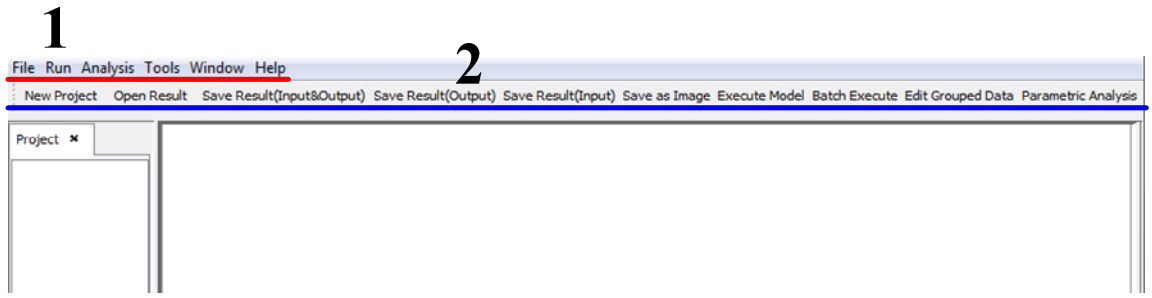


Figure 4.5. The functional components of BPsys.

4.3.2 Informational GUI Components

The informational components are emphasized in Figure 4.6 by differently

colored rectangles, labeled “1”, “2”, “3”, and “4”. In component 1, enclosed by the a red rectangle, is a tab pane component where all standard tree view components for viewing the file system are displayed; component 2, enclosed by the blue rectangle, is a tab pane component where all the principle graphical displays of data are displayed; component 3, surrounded by a light brown rectangle, is a tab pane component where the system output and other textual meta-data are displayed; and component 4, within a green rectangle, is a progress bar, which displays an indicator, regarding whether the system is processing or idling. The next subsection will explain each of these components in detail.

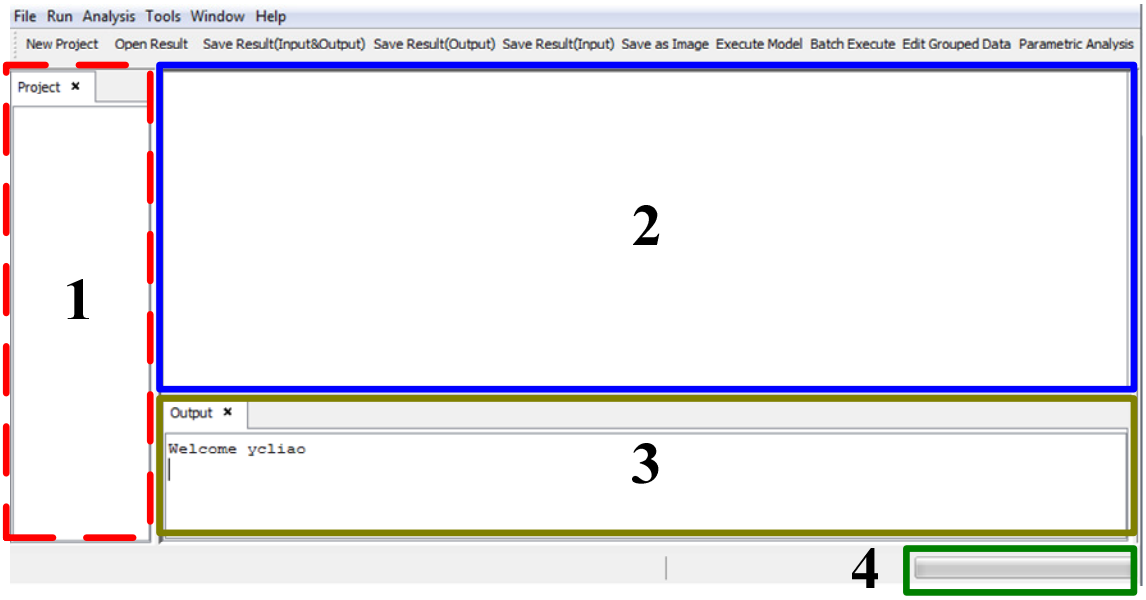


Figure 4.6. The display components of BPsys.

4.4 User Interface for Systems Analysis

Section 4.4 provides an overview of the functions within BPSys, which implement the user stories of systems analysis defined in Section 3.5.1. In total, there are ten user stories for systems analysis that have been identified; however, the tenth story has not yet been implemented since it was decided to be beyond the scope of this thesis. Until now screen shots have been displayed as they appear when BPSys first initializes in order to focus on

the relative locations of the components; some are adjusted to a suitable size to suit the illustrations presented here. The screen shots in the following subsections are captured from a computer operated under the Windows 7 Professional (64-bit) operating system. The installed JRE library version is 1.6.22. The browser used to run BPSys is Firefox version 3.6.12.

4.4.1 Select Model and Retrieve Data

The procedures for selecting a model and retrieving data are illustrated in Figure 4.7. The location of the menu item where a new project is created is shown in Figure 4.8. Once the menu item is clicked, a window listing the available modeled scenario appears—the "Create Project" window (Figure 4.9). Steps 2 to 4 are occur within the "Create Project" window. Clicking "OK" causes the system to gather the necessary information in step 5. During step 5, BPSys downloads the necessary data from the server. The retrieved data are used to visualize the modeled scenario and the default input data, as described in Section 4.4.2. After the data are downloaded, BPSys will complete the creation of the new project by displaying it as a node and expanding it to a tree structure in the "Project" window (Figure 4.10).

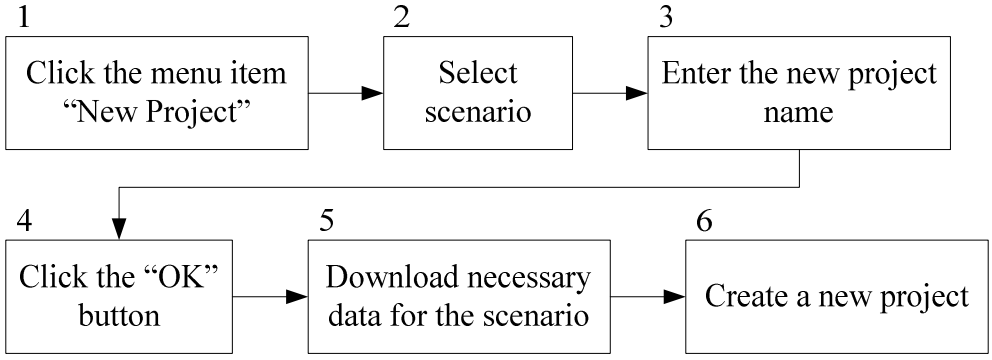


Figure 4.7. The procedure of selecting model and download default data.

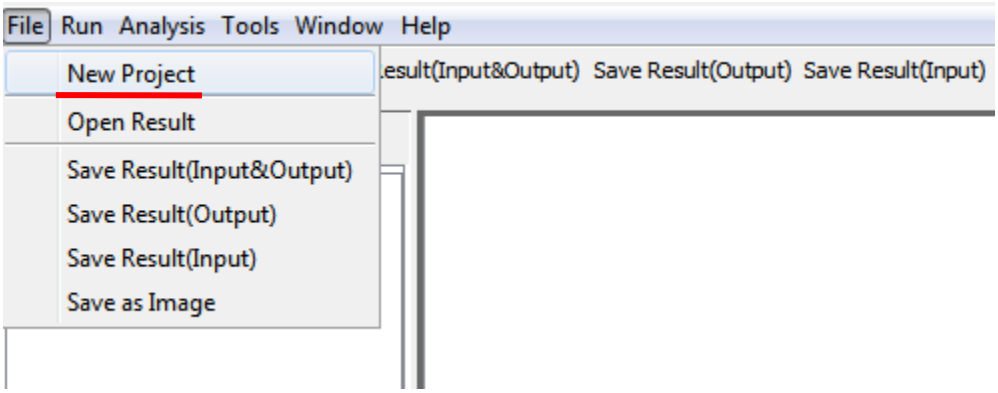


Figure 4.8. The “New Project” menu item.

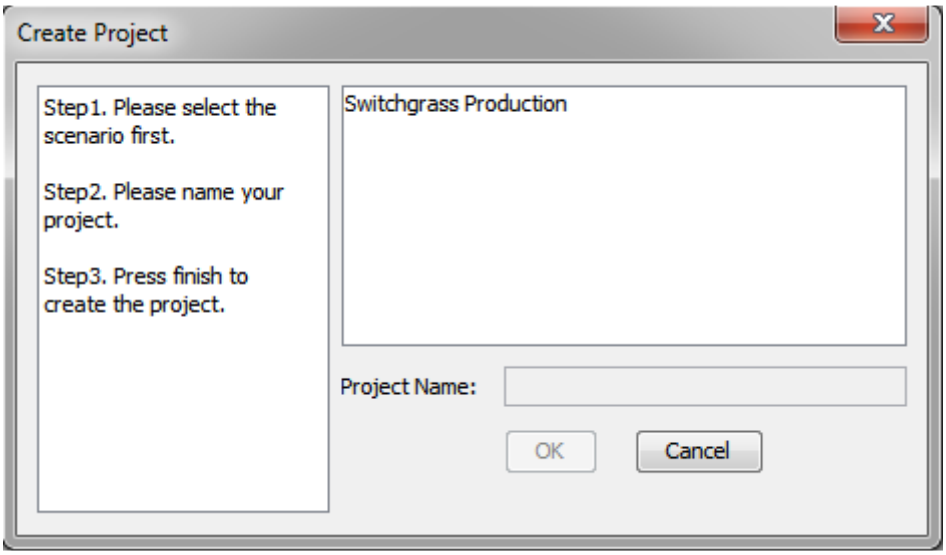


Figure 4.9. The “Create Project” window.

Figure 4.10. The screen shot of the new project in the “Project” window.

4.4.2 Visualize Scenario

The created project will be displayed in the “Project” window as a tree node and has four default child nodes (Figure 4.11). The “Switchgrass Production” node, underlined in Figure 4.11, represents the modeled scenario and the other three nodes, the

"Results" node, the "DataGroups" node, and the "Parametric Analysis" node, are folders for information produced via BPSys depending on the selections of the user. To visualize the scenario in the project, the user will double click the "Switchgrass Production" node creating two tabs, one in the upper tabbed pane (component 1 in Figure 4.12) and one in the lower tabbed pane (component 2 in Figure 4.12). Component 1 depicts a materials flow diagram of the biomass feedstock production system under study. Component 2 is a display of the related input data of the scenario. To view the content in each tab the user needs to click the desired locations within each tab. Each of the ten buttons in the flowchart stands for a step in the biomass feedstock production supply chain, and the buttons connected with red lines represent that they are sequential in nature. If a red line has more than one branch, there exist multiple possible pathways for materials to flow through the system. Some of the buttons are surrounded with a green rectangle and some are not; highlighted buttons have a corresponding attributes table in the "InputData" tab pane, where the user may select equipment for analysis and manipulate attribute values. Figure 4.13 (a) shows the flowchart and Figure 4.13 (b) shows the contents in the "InputData" tab pane where eight tab panes contain corresponding attribute tables.

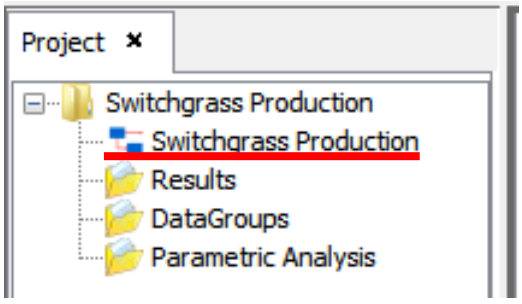


Figure 4.11. The scenario node.

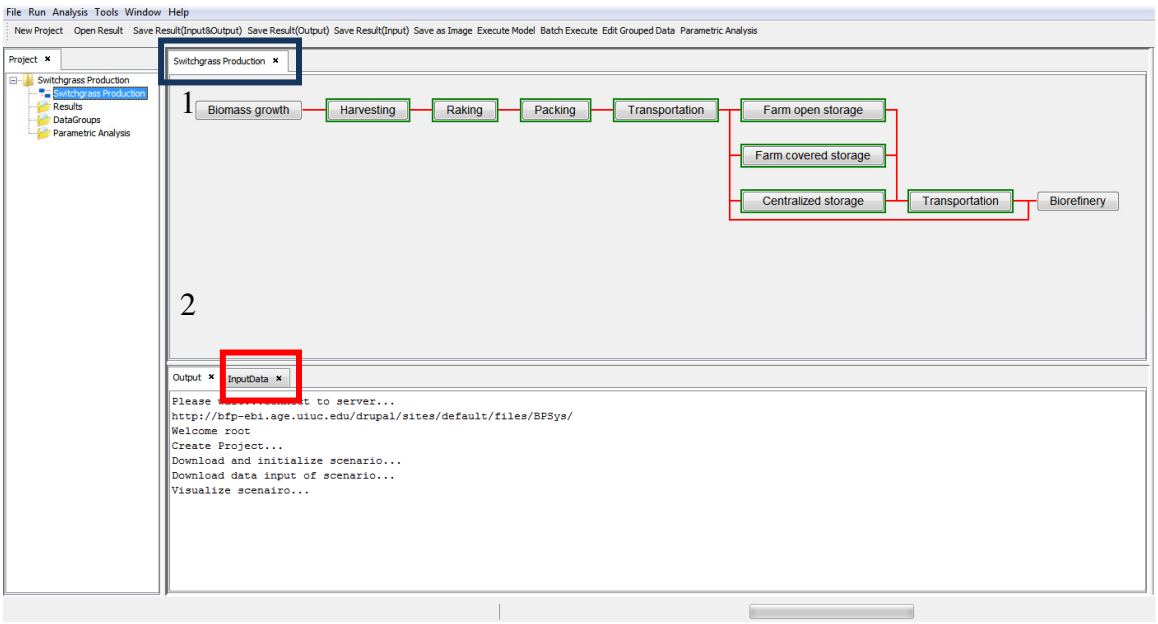
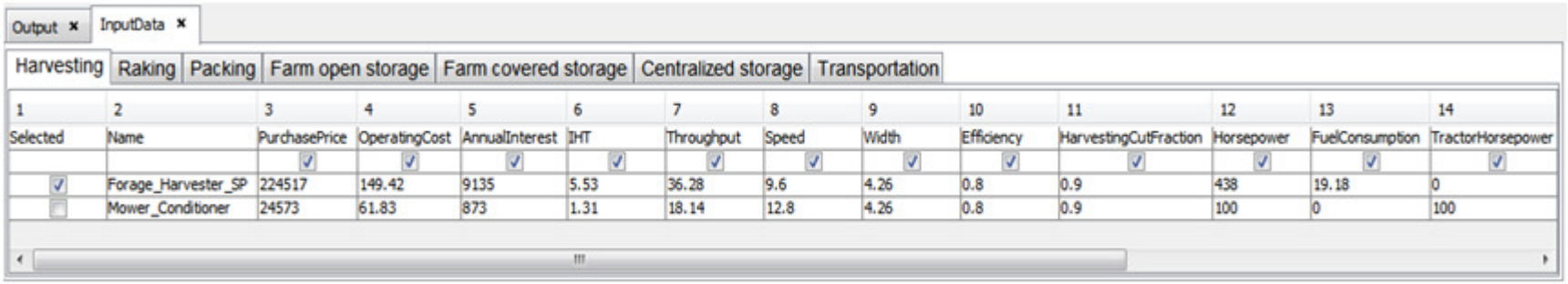
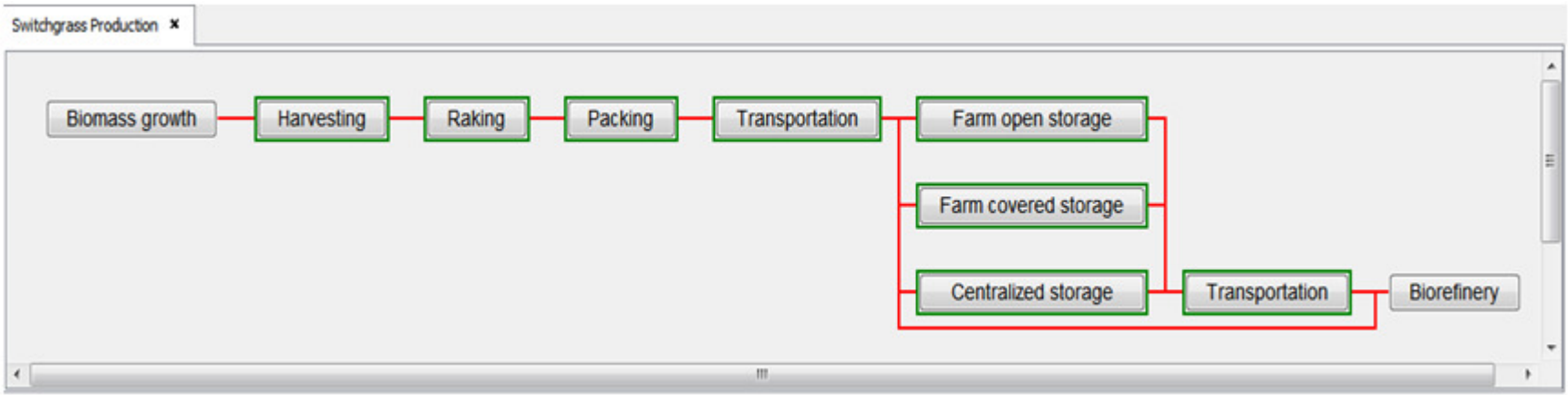


Figure 4.12. The screen shot of visualized scenario.



1	2	3	4	5	6	7	8	9	10	11	12	13	14
Selected	Name	PurchasePrice	OperatingCost	AnnualInterest	IHT	Throughput	Speed	Width	Efficiency	HarvestingCutFraction	Horsepower	FuelConsumption	TractorHorsepower
☑	Forage_Harvester_SP	224517	149.42	9135	5.53	36.28	9.6	4.26	0.8	0.9	438	19.18	0
☐	Mower_Conditioner	24573	61.83	873	1.31	18.14	12.8	4.26	0.8	0.9	100	0	100

(a)



(b)

Figure 4.13. The visualization components of the modeled scenario: (a) The tab pane of the scenario's flowchart (b) The tab pane of the attributes tables.

4.4.3 Modify Input Data of the Models

Input data can be modified via two approaches. Equipment can be selected and unselected by checking and unchecking the checkboxes by a red rectangle in Figure 4.14. Alternatively, specific values may be modified by performing three steps: double clicking the cell, making the background of the cell become white (Figure 4.15. The screen shot of editing a cell in an attributes table; and pressing “Enter” or any of the arrow keys on the keyboard.

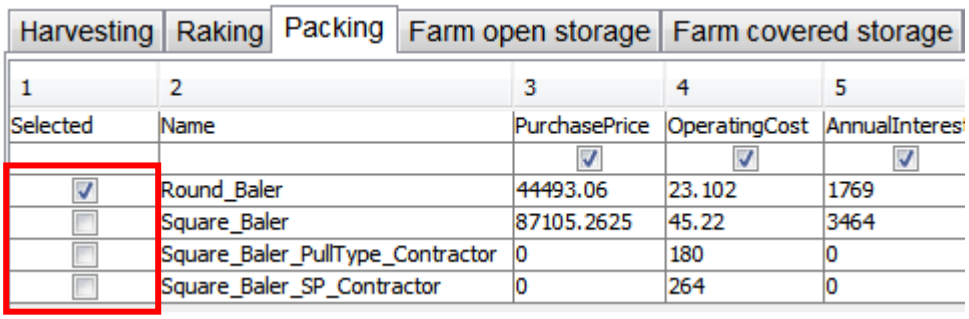


Figure 4.14. The checkbox column of selecting equipment.

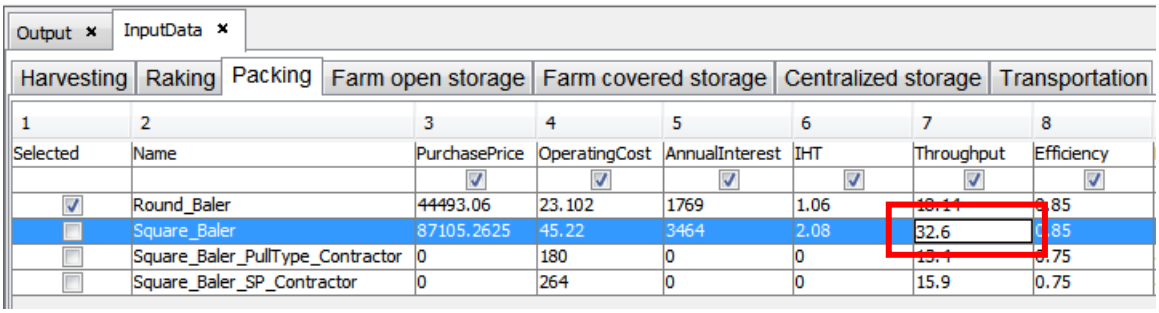


Figure 4.15. The screen shot of editing a cell in an attributes table.

4.4.4 One Time Execution

There are two approaches to executing the models once the scenarios have been constructed: one time execution and batch execution. This section discusses one time execution. The menu item for one time execution of the model once is underlined in Figure 4.16, on the run menu, in the menu bar. When the menu item is clicked, BPSys starts to prepare the input data and then runs the model. This includes the collection of the

selected equipment data, as specified by the user. Then, these data are uploaded to the server and utilized for model execution. While the model is running, the progress bar is filling and the "Output" window" (component 3 in Figure 4.6) is printing out the messages received from the modeling software regarding the current progress. Once the analysis is complete, a message window will appear to notify the user (Figure 4.17). The results are then automatically retrieved from the server and put under the "Results" node in the "Project" window (Figure 4.18).

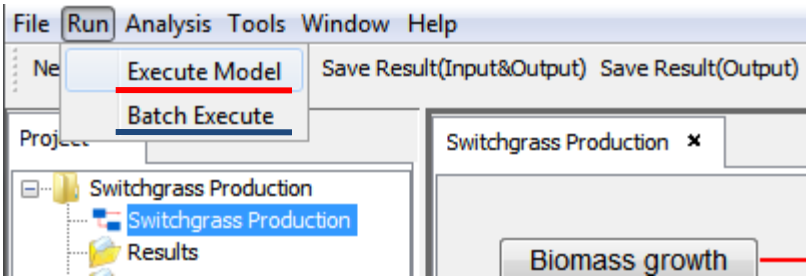


Figure 4.16. The "Execute Model" menu item.

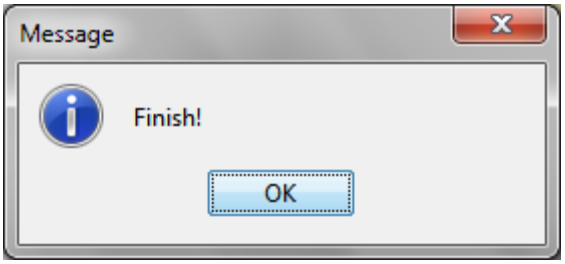


Figure 4.17. The window to signal the completion of the model.

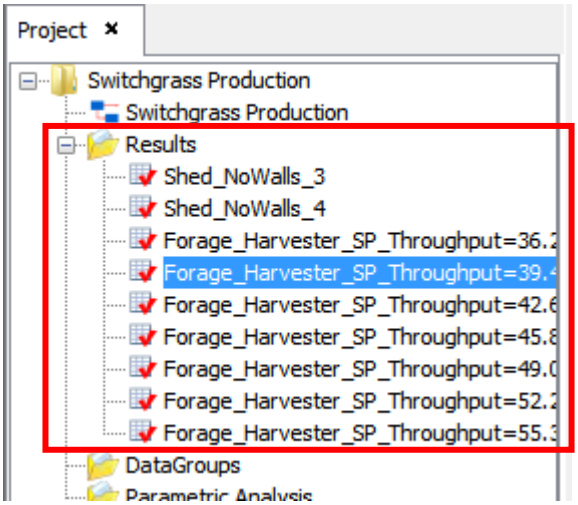


Figure 4.18. The generated result under the "Results" node in the "Project" window.

4.4.5 Batch Execution

The menu item for selecting batch execution is highlighted by the blue line shown in Figure 4.16. Clicking on the "Batch Execute" menu item causes the appearance of the "Batch Execute Configuration Dialog" window (Figure 4.19). The window contains two major parts: the configuration toolkit, on the left, and the list of batch executions to be run, on the right. Batch execution is similar to the basic execution of the model, requiring two additional inputs from the user: equipment selection and attribute value modification.

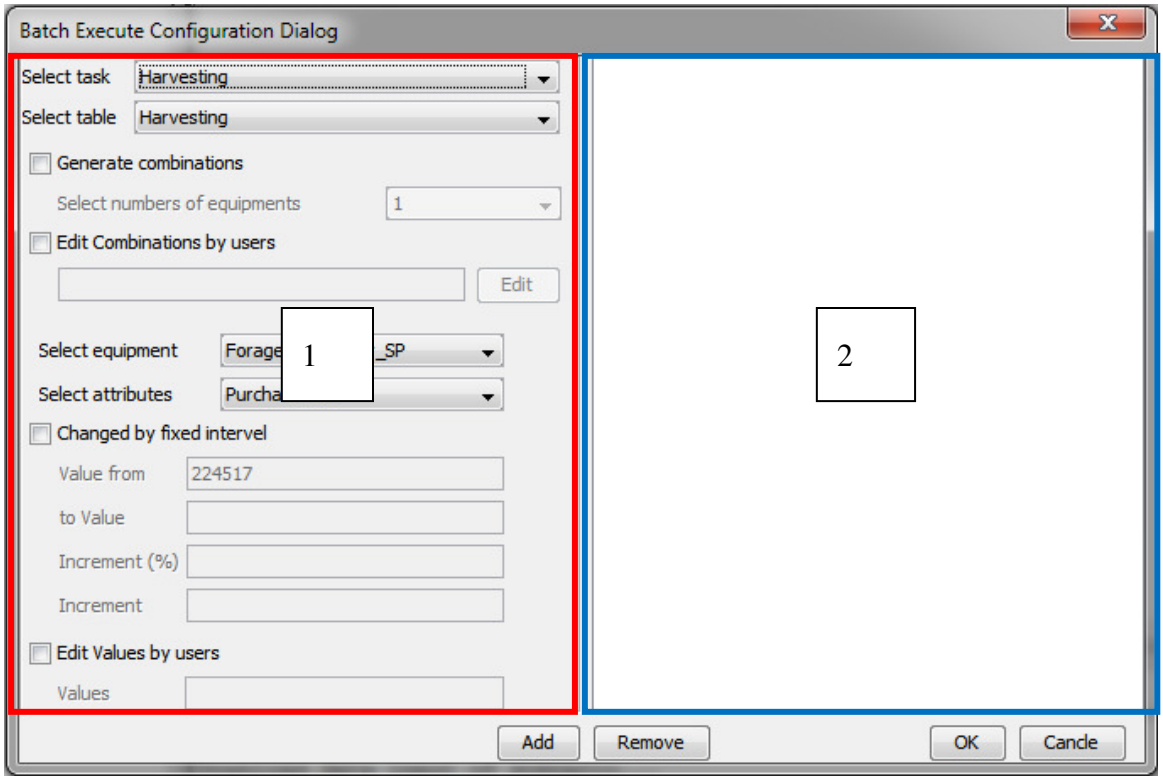


Figure 4.19. The configuration dialog for batch execution.

The steps for configuration of a batch execution are depicted in Figure 4.20. The first step is to select the biomass feedstock production task to be studied, and the second step is to select the equipment table within the task of interest. These are selected by choosing items from the lists underlined in red (Figure 4.21). The items available under "Select task" list are displayed in Figure 4.22, which reflects the editable tasks outlined in

green in Figure 4.13 (a). Each of these tasks may be described by several database tables (Figure 4.23). Next the user would select how the input data are varied by selecting one of the four checkboxes underlined in blue in Figure 4.21. The first two checkboxes control equipment selection; the last two control attribute modification.

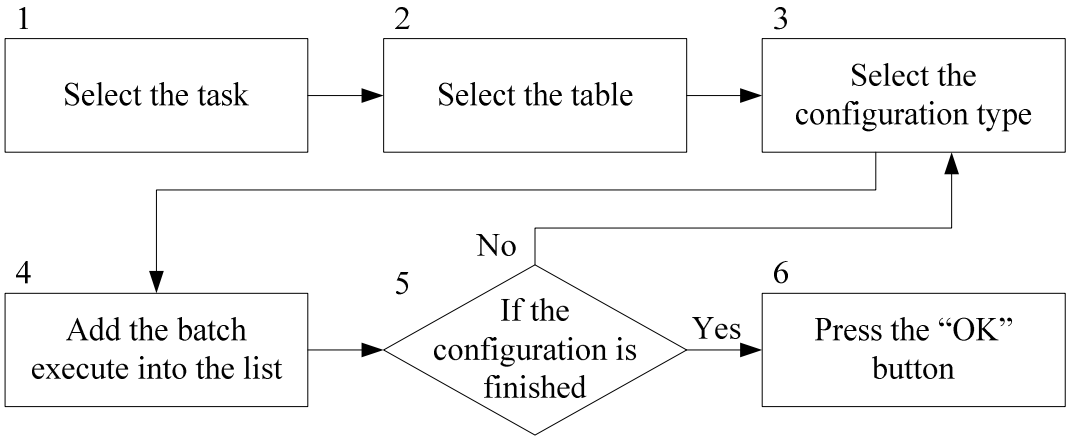


Figure 4.20. The flowchart of configuring the batch executions.

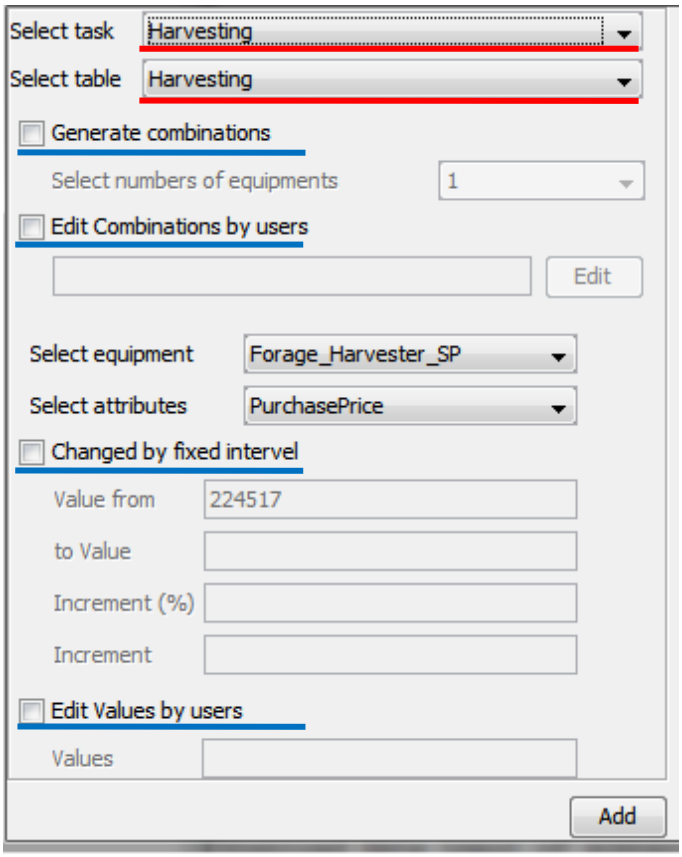


Figure 4.21. The task list and the table list to locate the attributes table.

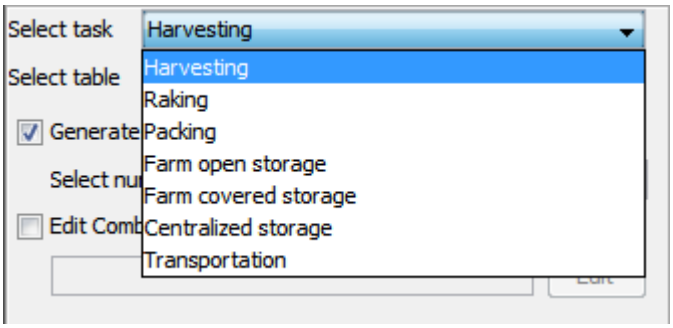


Figure 4.22. The selectable tasks in the task list.

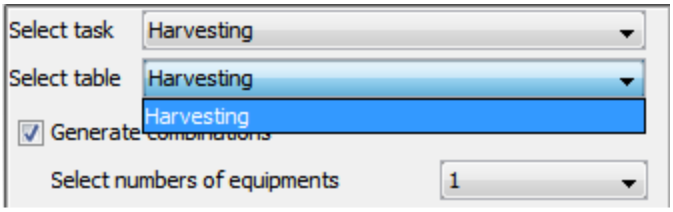


Figure 4.23. The selectable attributes table in the table list.

Combinations of equipment are based on the quantity of equipment available for consideration in the database. For example, the "Transportation" task has seven types of trucks available for analysis and the user can select any number of these for batch execution. Thus, if "Generate combinations" is selected, all possible combinations among the available equipments are executed. For example, if the user chooses "1" from the drop down list, there are seven combinations which use a single machine. If "2" is selected, all the combinations of any two trucks will be calculated and used to run the model.

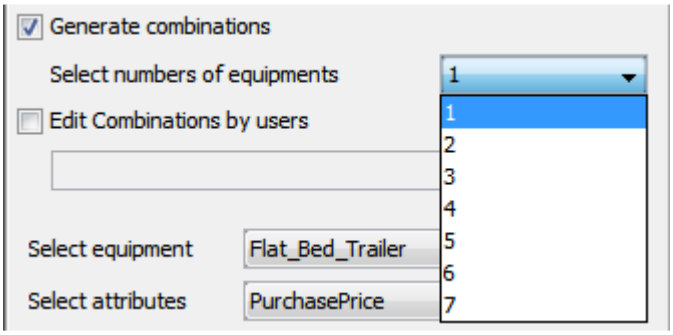


Figure 4.24. The numbers of selecting equipment.

By checking the checkbox "Edit Combinations by users" in Figure 4.24,

equipment combination can be arbitrarily edited by the user Figure 4.25 to initiate the configuration window (Figure 4.26). The user can select the desired equipment from the list on the left-side, as in Figure 4.26, and click the “Add” button. This adds the combination to the list on the right-side, as in Figure 4.27. Items may be similarly removed by selecting and clicking “delete” on the right. Once the user is satisfied with the list of equipment and clicks “OK”, the user-defined equipment selections will be displayed in the text field underlined in Figure 4.28.

Figure 4.25. The GUI components for editing equipment combination.

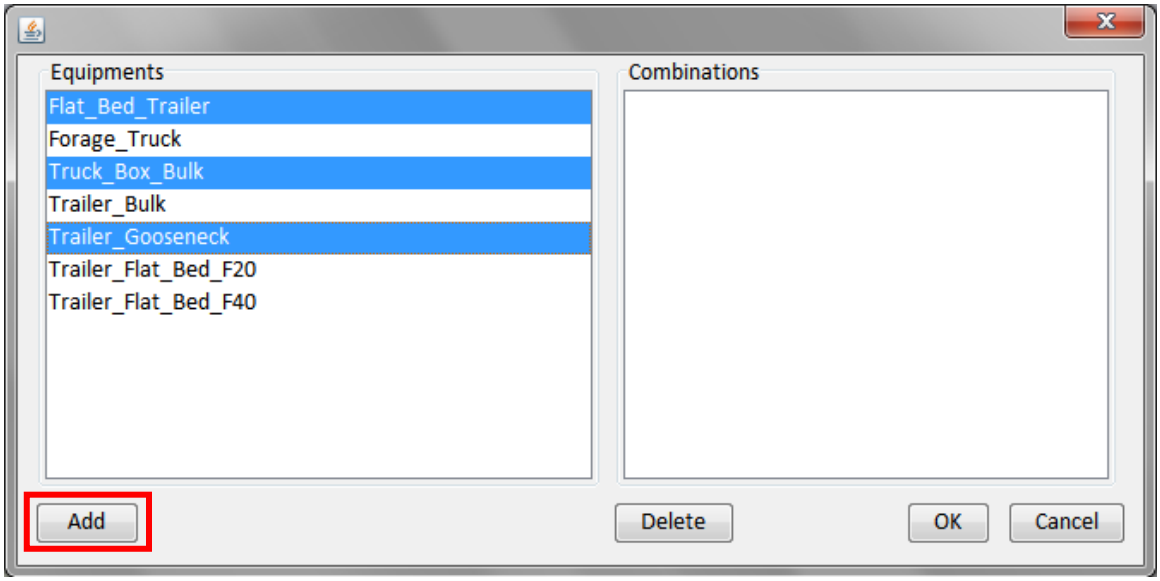


Figure 4.26. Choosing equipment.

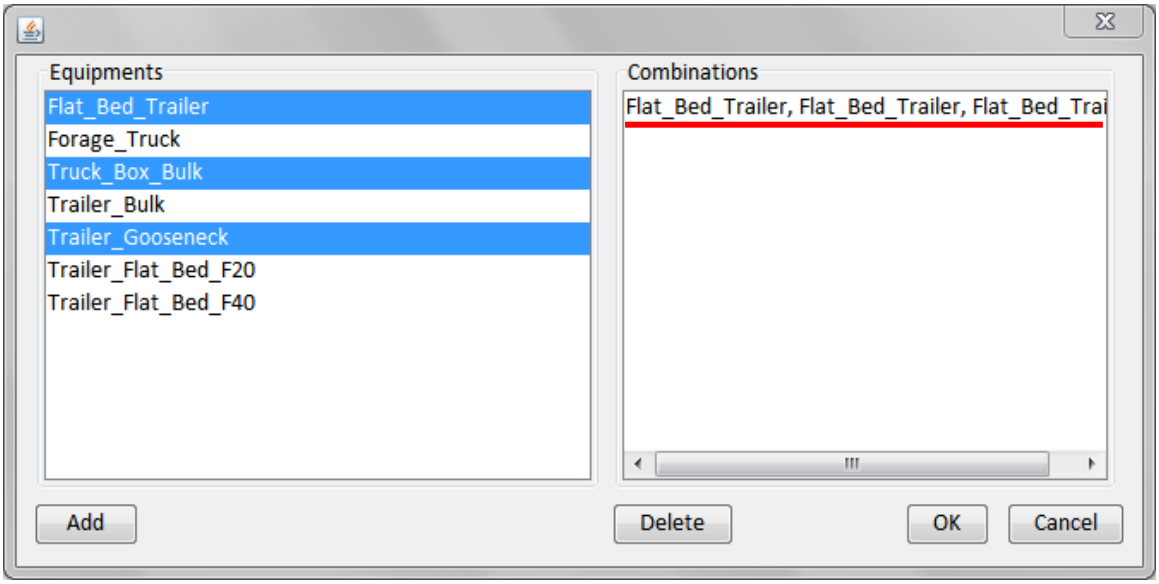


Figure 4.27. Add the equipment combination.

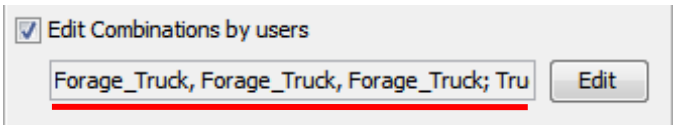
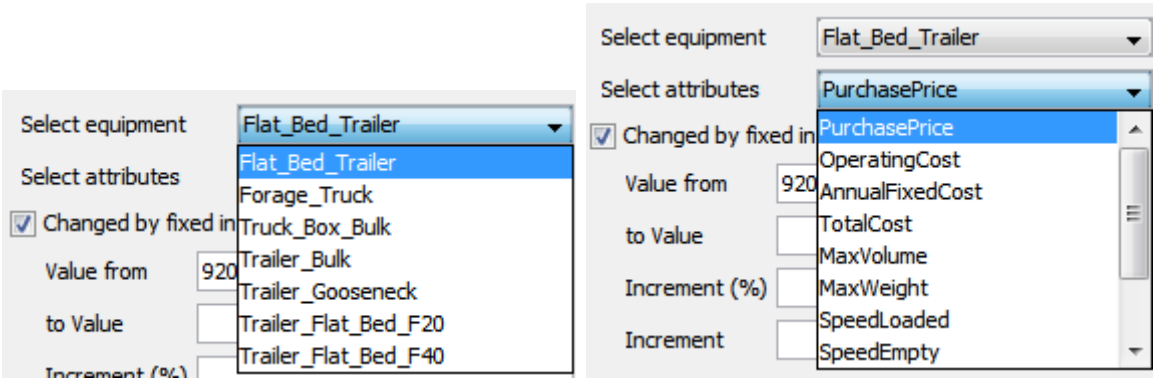


Figure 4.28. The user-defined equipment selections.

A user may also select to run a batch execution while modifying attribute values. The attribute to be changed should be specified by choosing the equipment and the attribute name from the lists as shown in Figure 4.29 (a) and (b), and entering the desired range of values to be considered.

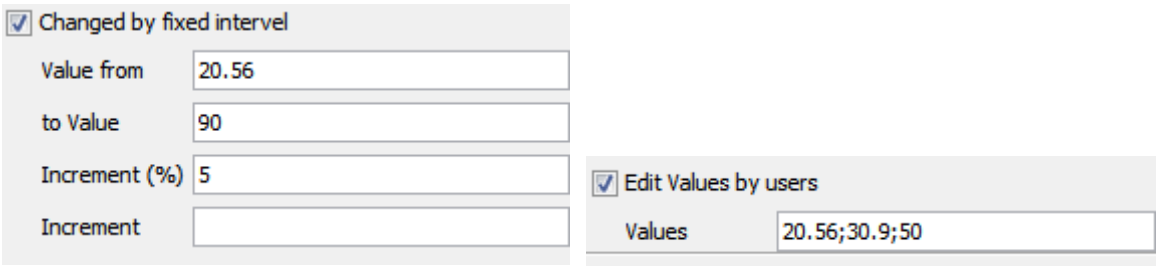


(a) (b)

Figure 4.29. The selection lists to choose the attribute for batch executions (a) Equipment list (b) Attribute list.

Just as modifying equipment combinations, there are two methods for varying the

range of attribute values to be considered. This is illustrated in Figure 4.30. Figure 4.30 (a) demonstrates how to configure a set of values to be varied by a fixed interval, defined by a percentage of the specified range. The user can also specify a set of values directly by entering the values into the text field, separated by semicolons, as in Figure 4.30 (b).



(a) (b)

Figure 4.30. The two approaches to define the values of the selected attribute: (a) Fixed interval attribute (b) User-defined attributes.

After the user finishes configuration of a batch through steps 1 to 3 (Figure 4.20), it can be added into the list by clicking the "Add" button underlined by the blue line in Figure 4.31. For example, Figure 4.31 shows a batch execution that increases the attribute "MaxVolume" of the Forage_Truck, in the Transportation task, from 141.58 to 200 by intervals of 5 percent of this range. This has been added onto the list at right (marked by the red line). After the configuration is finished, the user presses "OK" to send the list to the server for execution.

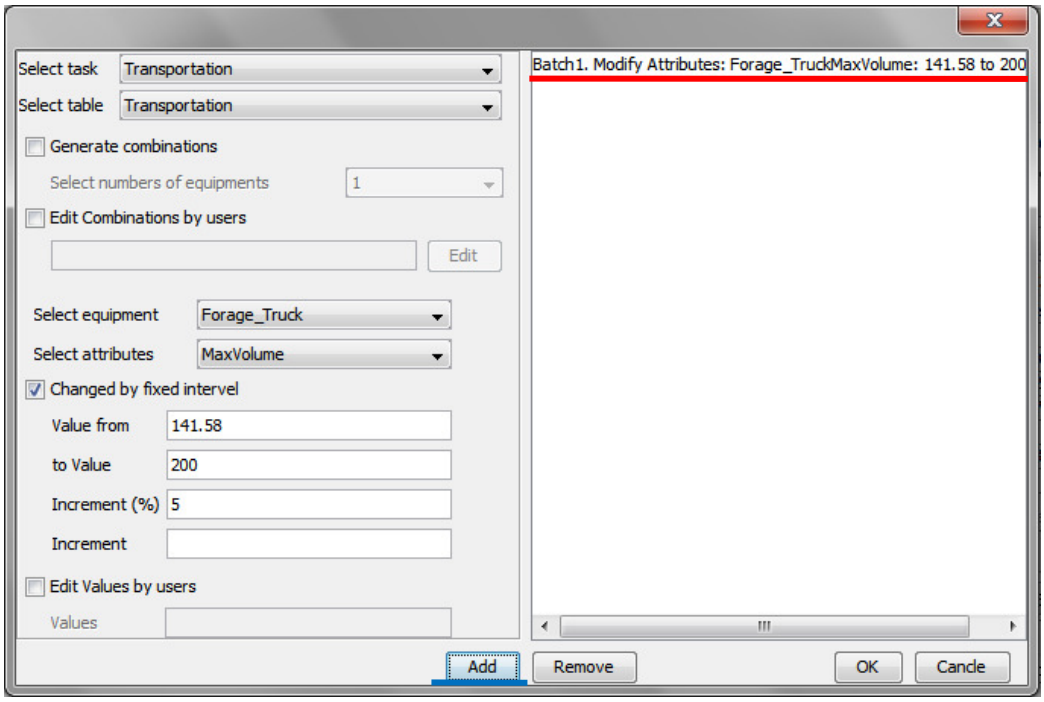


Figure 4.31. The screen shot of adding a batch execute into the execution list.

While the server is processing these requests, the "Execute Progress" window (Figure 4.32) appears, indicating the progress with six components: the fraction of processed batches, highlighted by the red line; the description of the current batch being processed, highlighted by the red rectangle; the progress bar displaying the percentage of completed batch executions, highlighted by the red dashed rectangle; the fraction of processed executions within a batch, highlighted by the blue line; the description of the current iteration, highlighted by the blue rectangle; and the progress bar displaying the progress of current model being executed, highlighted by the blue dashed rectangle. The results generated are put under the "Results" folder to be reviewed later. Before the requested batches are complete, the webpage should not be closed and the Internet connection should be maintained, or the batch will fail to complete.

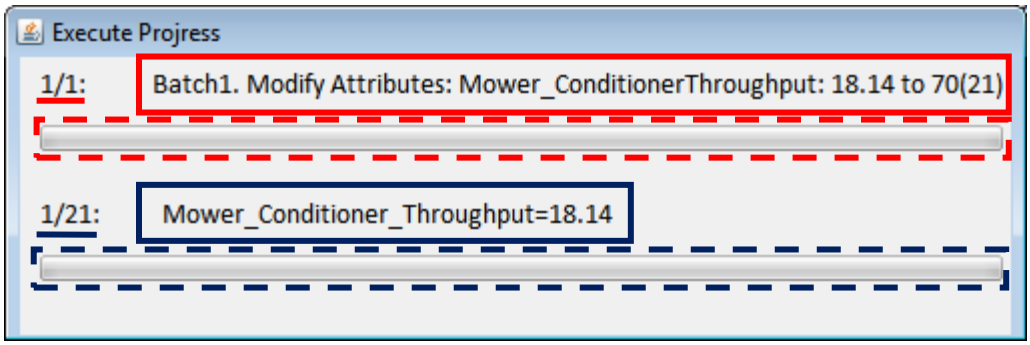
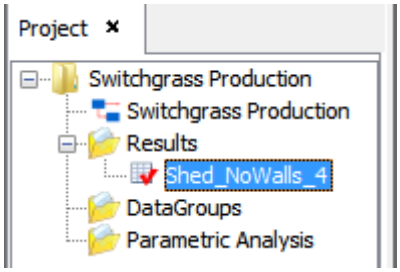


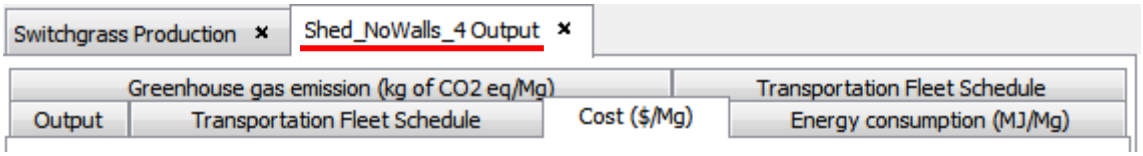
Figure 4.32. The progress window of the batch execute.

4.4.6 Retrieve and Review the Results

To display the output of the result, the user clicks on the result node (blue background in Figure 4.33 (a)) under the "Results" folder. This creates a tab pane labeled with the name of the results file (underlined by the red line in Figure 4.33 (b)). Multiple tab panes are created to display various types of output generated. This function is the manifestation of user story 7.



(a)

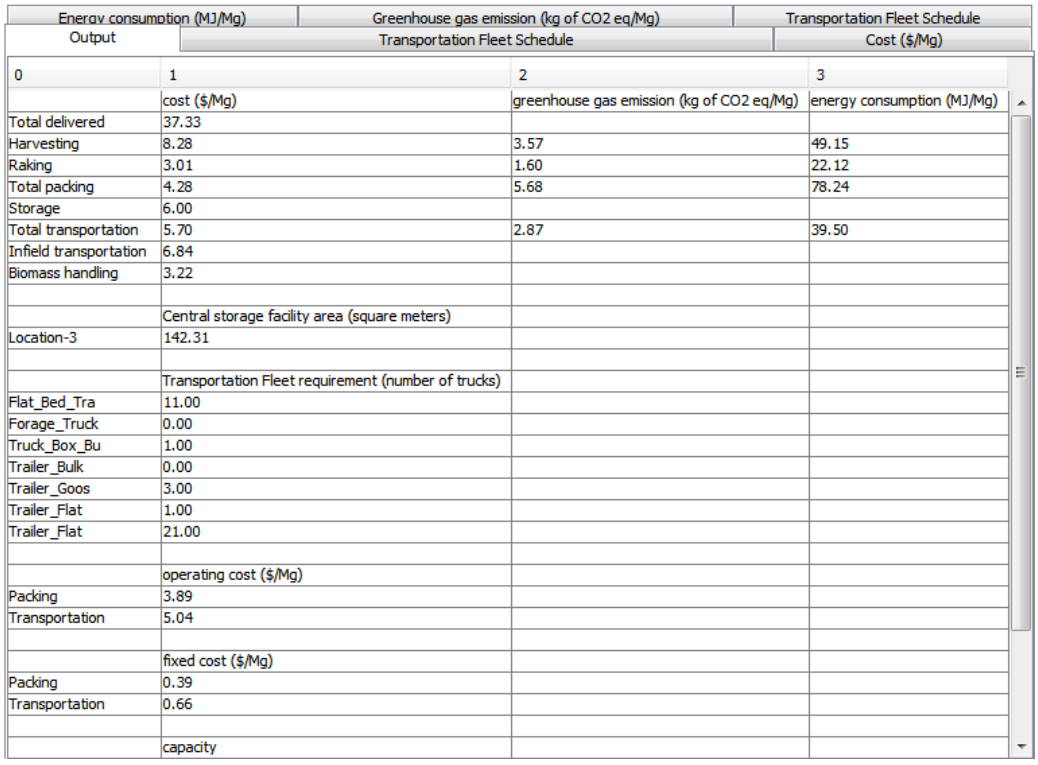


(b)

Figure 4.33. The visual components of a result: (a) The result node "Shed_NoWalls_4" (b) The result tab "Shed_NoWalls4 Output" to display the output of the result "Shed_NoWaslls_4.

The tab panes included in Figure 4.33 (b) are the "Output", the "Transportation Fleet Schedule", the "Cost ($/Mg)", the "Energy consumption (MJ/Mg), the "Greenhouse

gas emission (kg of CO_2 eq/Mg)", and the "Transportation Fleet Schedule Figure". When each of these tabs is clicked, the content within the tab pane is displayed (Figure 4.34 - 41). The "Output" tab (Figure 4.34) displays the system level performance indicators. The "Transportation Fleet Schedule" tab (Figure 4.35) shows the trucking schedule the entire year. The "Cost ($/Mg)" tab pane (Figure 4.36) is a pie chart for the distribution of costs within each task. The "Energy consumption (MJ/Mg)" tab pane (Figure 4.37) is the pie chart for the distribution of energy usage. The "Greenhouse gas emission (kg of CO2 eq/Mg)" tab pane (Figure 4.38) has the pie chart of the distribution CO2 emission of the system. The "Transportation Fleet Schedule" tab pane (Figure 4.39) is a line chart for visualization of the fleet schedule.


Energy consumption (MJ/Mg) | Greenhouse gas emission (kg of CO2 eq/Mg) | Transportation Fleet Schedule
Output | Transportation Fleet Schedule | Cost ($/Mg)

0	1	2	3
	cost ($/Mg)	greenhouse gas emission (kg of CO2 eq/Mg)	energy consumption (MJ/Mg)
Total delivered	37.33		
Harvesting	8.28	3.57	49.15
Raking	3.01	1.60	22.12
Total packing	4.28	5.68	78.24
Storage	6.00		
Total transportation	5.70	2.87	39.50
Infield transportation	6.84		
Biomass handling	3.22		
	Central storage facility area (square meters)		
Location-3	142.31		
	Transportation Fleet requirement (number of trucks)		
Flat_Bed_Tra	11.00		
Forage_Truck	0.00		
Truck_Box_Bu	1.00		
Trailer_Bulk	0.00		
Trailer_Goos	3.00		
Trailer_Flat	1.00		
Trailer_Flat	21.00		
	operating cost ($/Mg)		
Packing	3.89		
Transportation	5.04		
	fixed cost ($/Mg)		
Packing	0.39		
Transportation	0.66		
	capacity		

Figure 4.34. The "Output" tab pane.

Energy consumption (MJ/Mg) | Greenhouse gas emission (kg of CO2 eq/Mg) | Transportation Fleet Schedule

Output | Transportation Fleet Schedule | Cost ($/Mg)

0	1	2	3	4	5	6	7
Transportation fleet schedule							
Day	Flat_Bed_Trailer	Forage_Truck	Truck_Box_Bulk	Trailer_Bulk	Trailer_Gooseneck	Trailer_Flat_Bed_F20	Trailer_Flat_E
1	11.00	0.00	1.00	0.00	3.00	1.00	21.00
2	11.00	0.00	1.00	0.00	3.00	1.00	21.00
3	11.00	0.00	1.00	0.00	3.00	1.00	21.00
4	11.00	0.00	1.00	0.00	3.00	1.00	21.00
5	11.00	0.00	1.00	0.00	3.00	1.00	21.00
6	11.00	0.00	1.00	0.00	3.00	1.00	21.00
7	11.00	0.00	1.00	0.00	3.00	1.00	21.00
8	11.00	0.00	1.00	0.00	3.00	1.00	21.00
9	11.00	0.00	1.00	0.00	3.00	1.00	21.00
10	11.00	0.00	1.00	0.00	3.00	1.00	21.00
11	11.00	0.00	1.00	0.00	3.00	1.00	21.00
12	11.00	0.00	1.00	0.00	3.00	1.00	21.00
13	11.00	0.00	1.00	0.00	3.00	1.00	21.00
14	11.00	0.00	1.00	0.00	3.00	1.00	21.00
15	11.00	0.00	1.00	0.00	3.00	1.00	21.00
16	11.00	0.00	1.00	0.00	3.00	1.00	21.00
17	11.00	0.00	1.00	0.00	3.00	1.00	21.00
18	11.00	0.00	1.00	0.00	3.00	1.00	21.00
19	11.00	0.00	1.00	0.00	3.00	1.00	21.00
20	11.00	0.00	1.00	0.00	3.00	1.00	21.00
21	11.00	0.00	1.00	0.00	3.00	1.00	21.00
22	11.00	0.00	1.00	0.00	3.00	1.00	21.00
23	11.00	0.00	1.00	0.00	3.00	1.00	21.00
24	11.00	0.00	1.00	0.00	3.00	1.00	21.00
25	11.00	0.00	1.00	0.00	3.00	1.00	21.00
26	11.00	0.00	1.00	0.00	3.00	1.00	21.00
27	11.00	0.00	1.00	0.00	3.00	1.00	21.00
28	11.00	0.00	1.00	0.00	3.00	1.00	21.00

Figure 4.35. The table of the transportation fleet schedule.

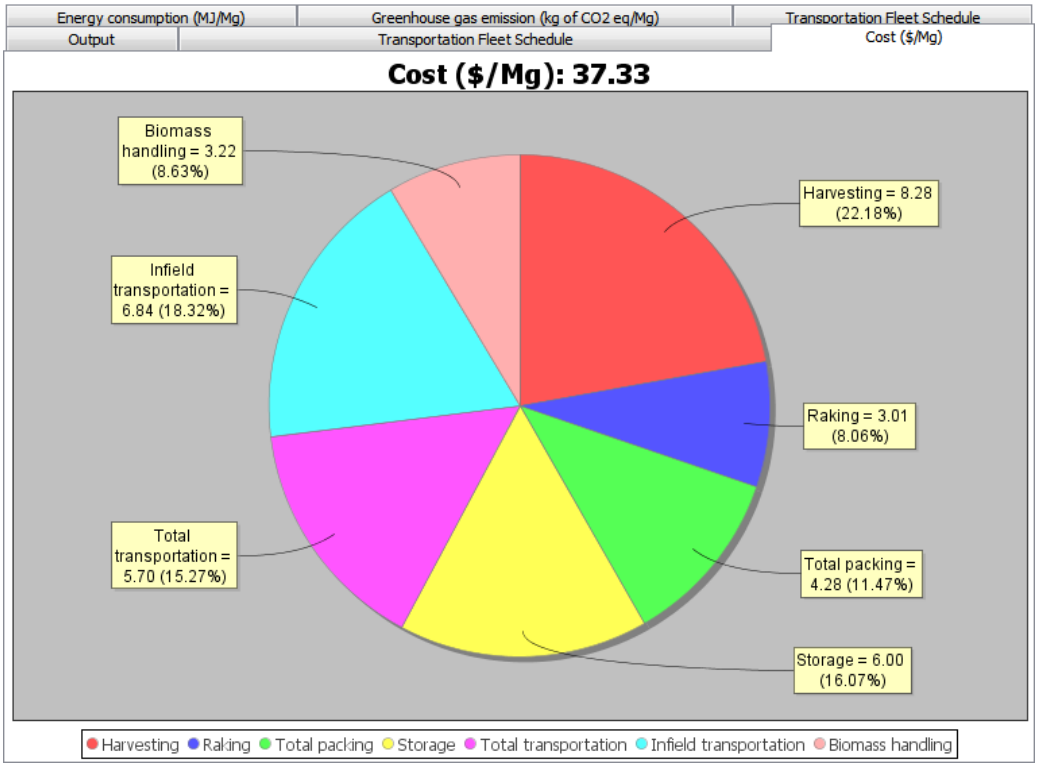


Figure 4.36. The “Cost ($/Mg)” tab pane.

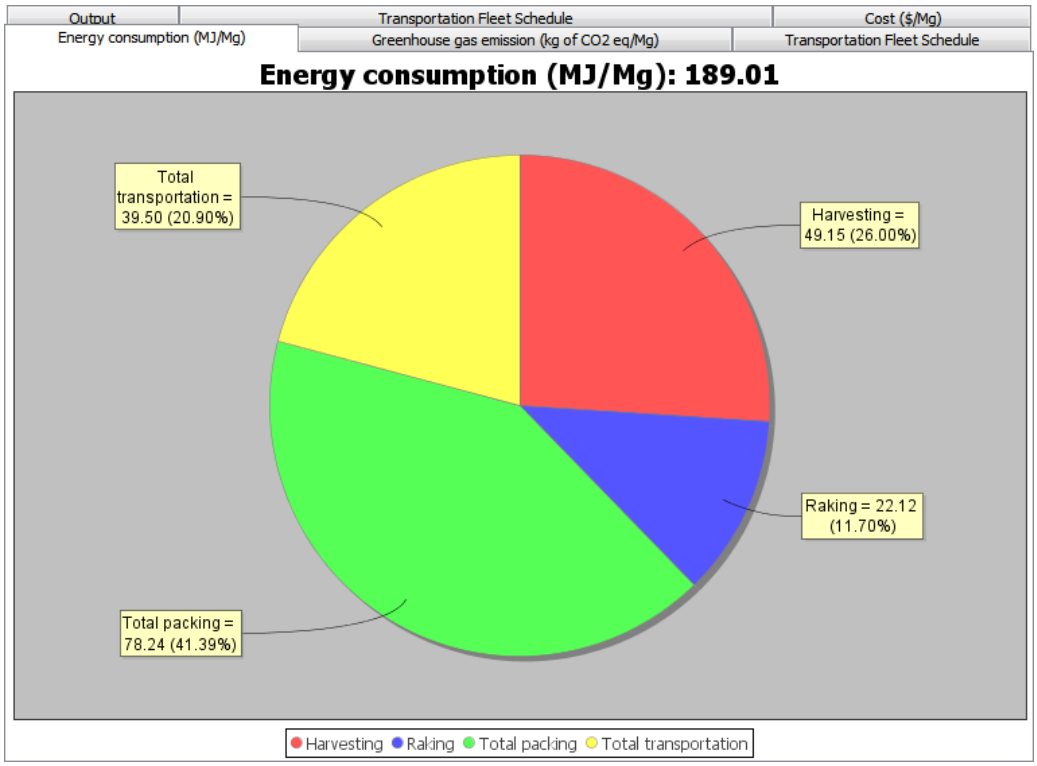


Figure 4.37. The "Energy consumption (MJ/Mg)" tab pane.

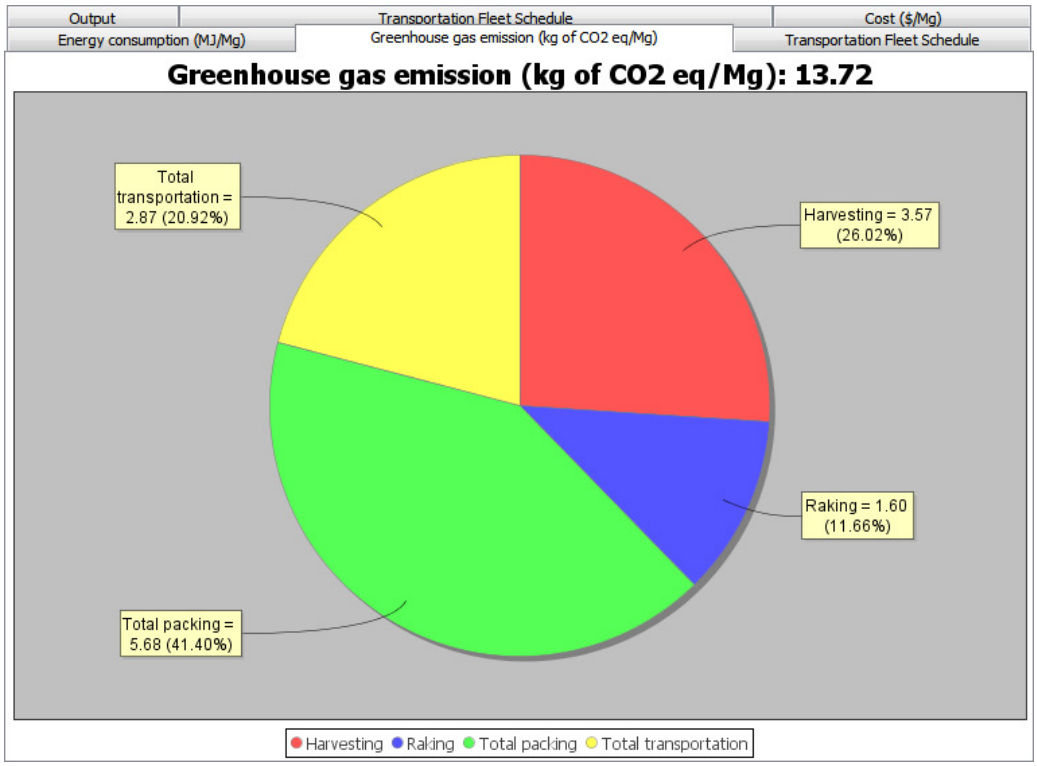


Figure 4.38. The "Greenhouse gas emission (kg of CO2 eq/Mg)" tab pane.

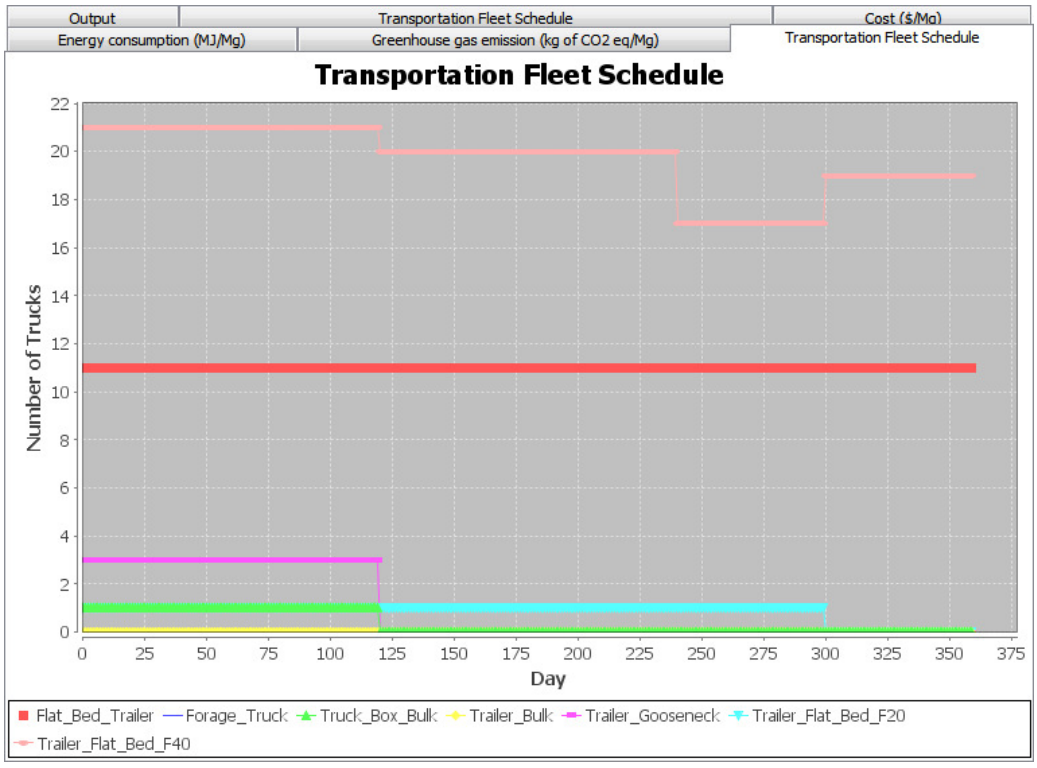


Figure 4.39. The "Transportation Fleet Schedule" tab pane.

4.4.7 Save Result

The results can be saved in CSV file for analysis outside of BPSys (Figure 4.40). First the user should select the result(s) to be saved from the Project window by clicking on the node to make its background become blue (Figure 4.41 (a)). Then the user should click the "Save Result(Input&Output)" menu item (Figure 4.41 (b)). This makes the "Save" dialog box appear automatically (Figure 4.42).

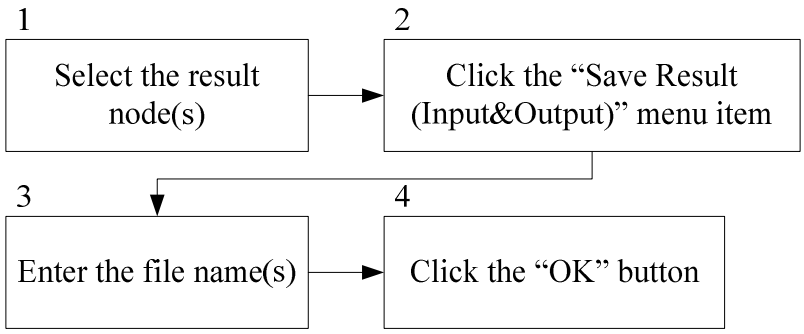
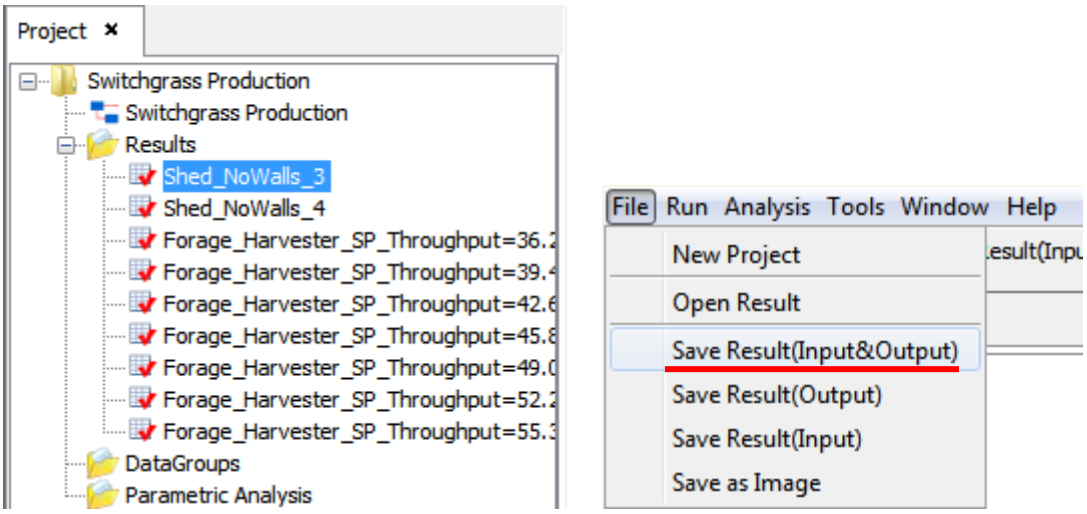


Figure 4.40. The flowchart to save result(s).



(a) (b)

Figure 4.41. The adjustment of the dimensional size for saving a image: (a) The mouse cursor on the edge between the “Project” window and the main tab pane (b) The mouse cursor on the edge between the “Object” window and the main tab.

In the dialog, the default file name for the input file and output file of the selected result are displayed in the “File name” text field underlined by the red line (Figure 4.42). The user can modify the file names, as desired, and press “Save”. The file extension, “*.in” stands for the file of input data; the sub file title, ”*.out” stands for the file out output data.

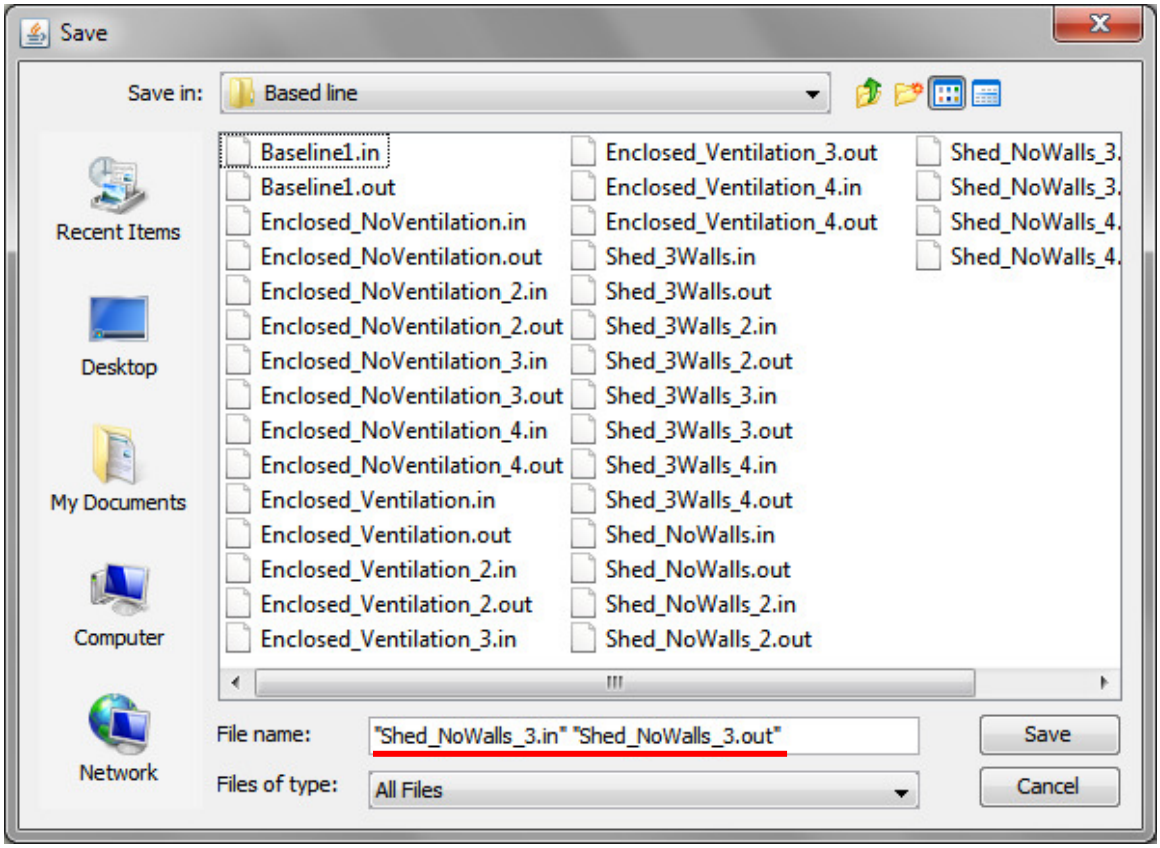


Figure 4.42. The "Save" dialog.

The user can also save the figures in BMP image format. The steps of saving the figure as a BMP image are shown in Figure 4.43. First the user should display the figure shown on the screen by clicking the appropriate tab on the tab pane. Then the user should click "Save as Image" menu item (Figure 4.44). The "Save" dialog (Figure 4.42) will appear automatically.

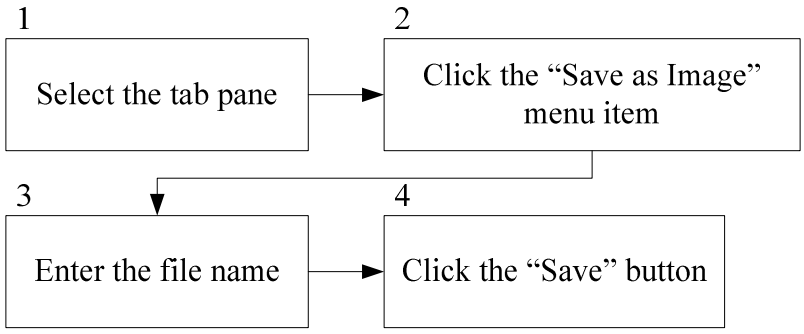


Figure 4.43. The flowchart of saving the figure as an image.

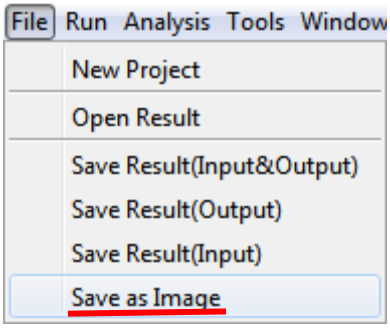
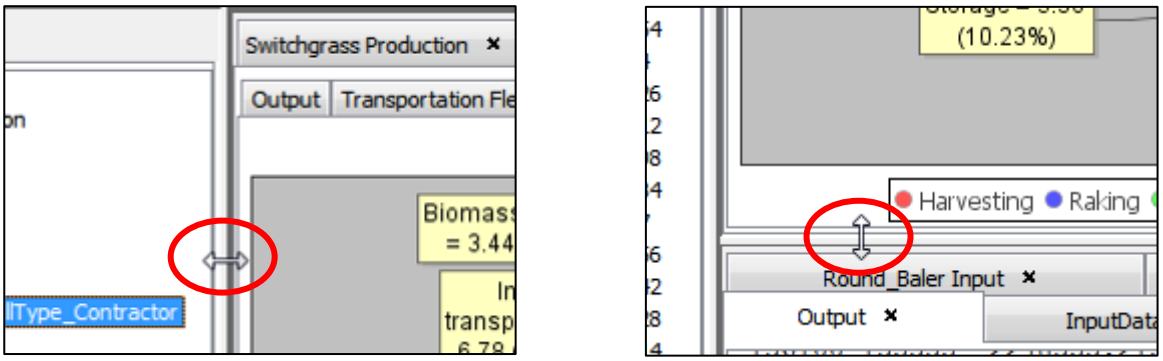


Figure 4.44. The "Save as Image" menu item.

The size of the saved image will depend on the current size of display of figure. The width and height of the figure can be adjusted by moving the mouse cursor to the edges of the Project window (Figure 4.45 (a)) and the Output window (Figure 4.45 (b)), until the double arrow appears, and dragging the mouse cursor. The functions to save text and figure outputs are the manifestation of user story 8.1 "Save result".



(a) (b)

Figure 4.45. (a) The mouse cursor on the edge between the "Project" window and the main tab pane (b) The mouse cursor on the edge between the "Object" window and the main tab.

4.4.8 Read Result

To open a saved result, the user should click "Open Result" on the file menu (Figure 4.46). The "Open" dialog (Figure 4.47) will appear. The user selects the output files of interest from the dialog and clicks the "Open" button. A node under the "Results" node will be created and the user can click as described in Section 4.4.6.

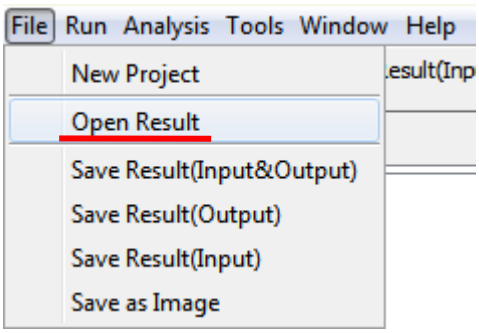


Figure 4.46. The "Open Result" menu item.

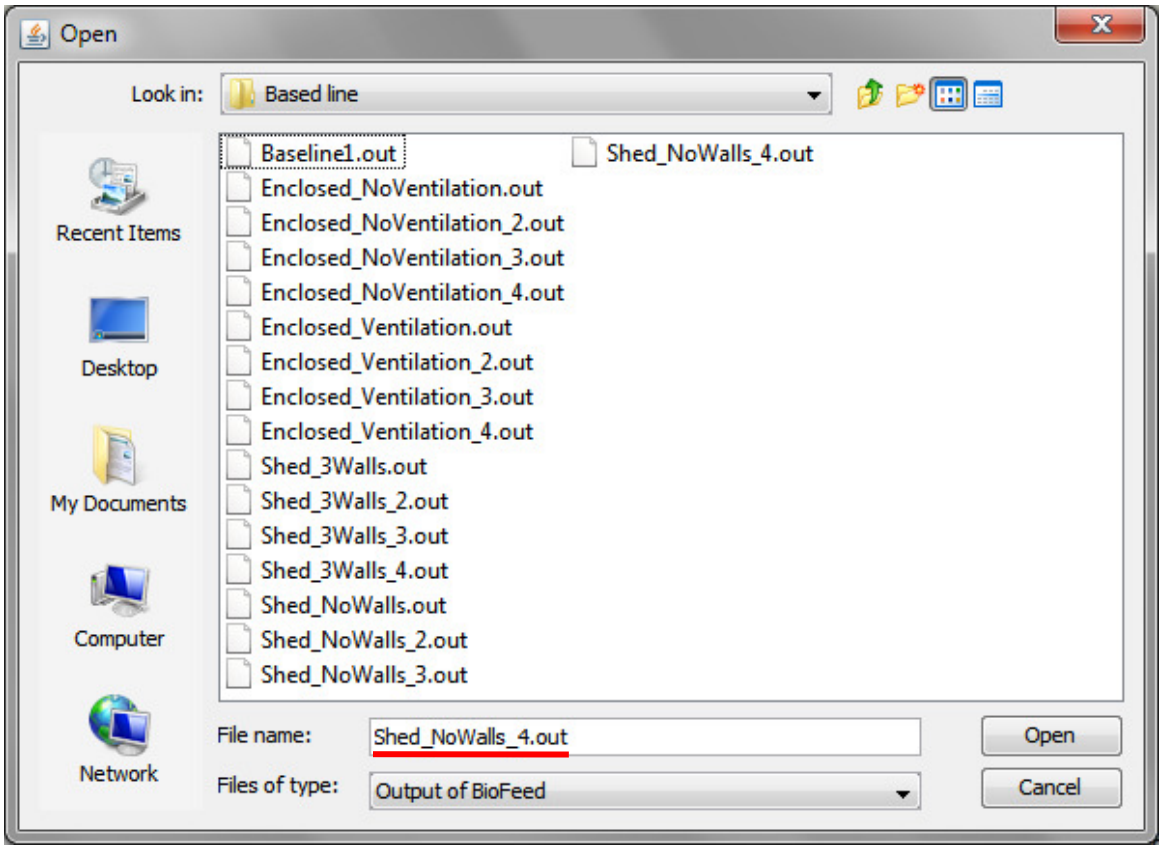


Figure 4.47. The "Open" dialog.

4.4.9 Parametric analysis

This function provides a convenient tool to study the impact of specific attributes on the system performance. To do so, the user must select the dependent and independent variables from a batch of results and create plots to present the impacts. The procedure for parametric analysis is shown in Figure 4.48. It consists of nine steps. Steps 1 to 3 create a group of results, called a DataGroup; steps 4 to 9 prepare the data set and draw

the plot.

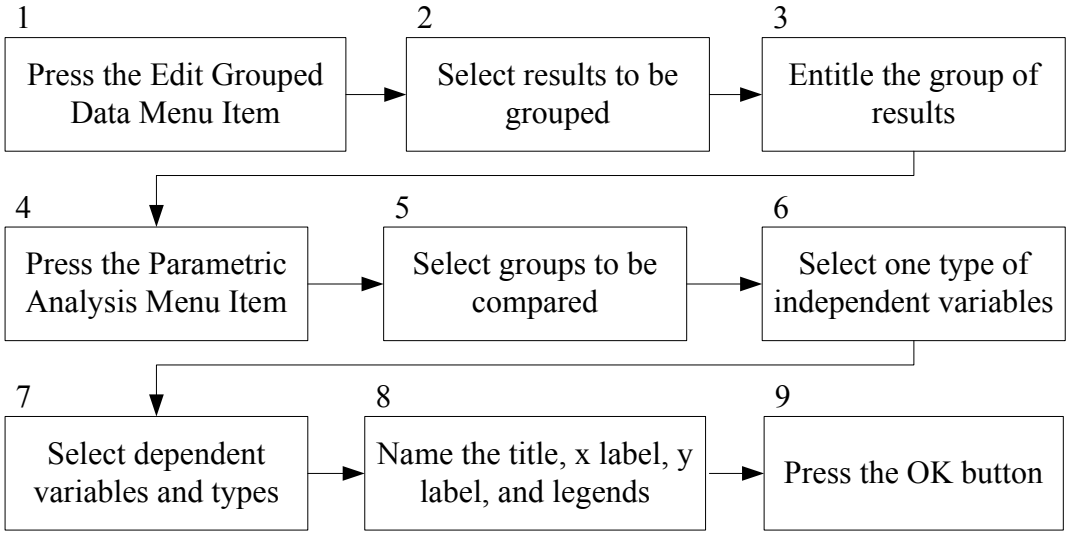


Figure 4.48. The procedures of parametric analysis.

First, the user should create DataGroups by pressing "Edit Grouped Data" (Figure 4.49). A dialog will appear (Figure 4.50). Then the user can select the results from the list on the left side of the dialog (Figure 4.50) and add them to "Selected Results" by clicking the double arrows. After making the selections, the user should add a description to the group and click "Ok". A node describing the group will display in the DataGroups node, in the Project window (Figure 4.51).

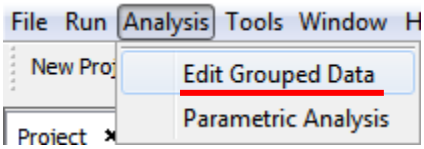


Figure 4.49. The Edit Grouped Data Menu Item.

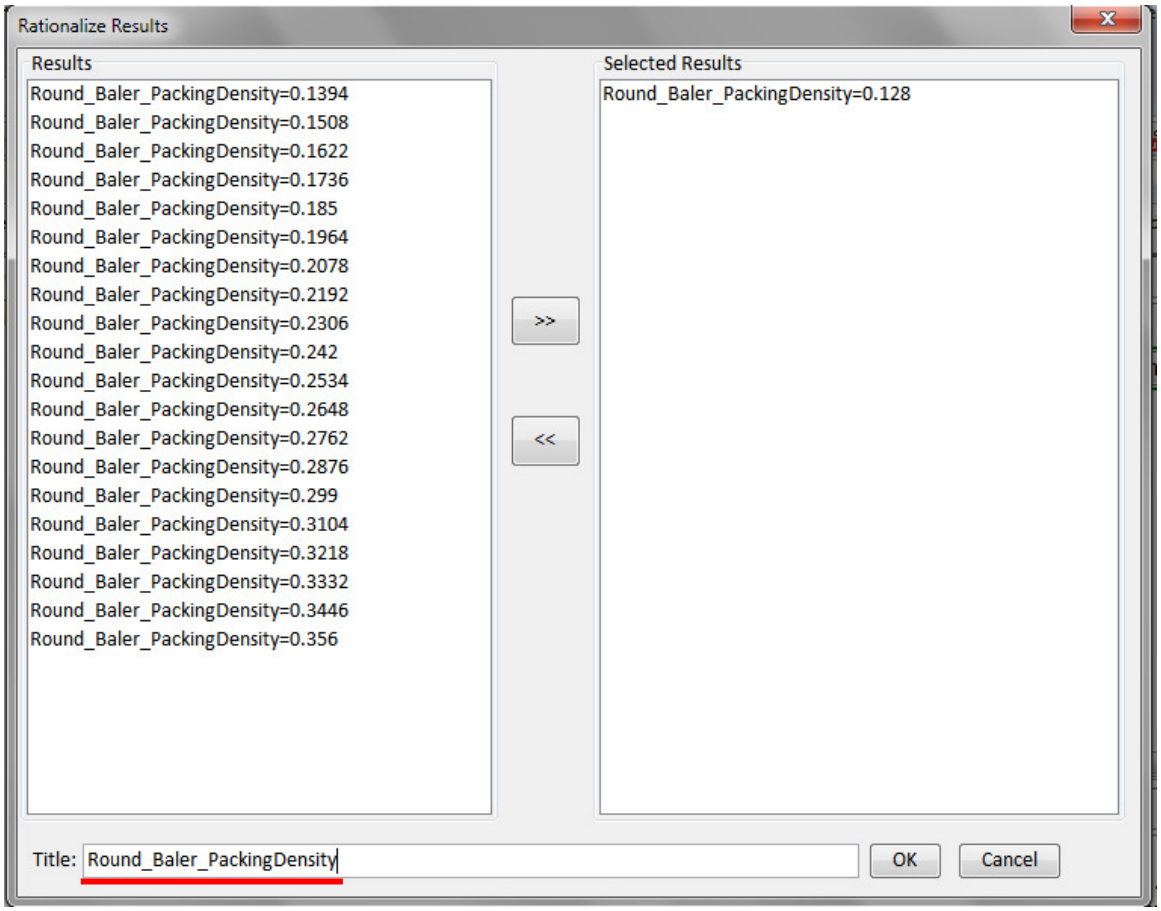


Figure 4.50. The dialog for selecting results.

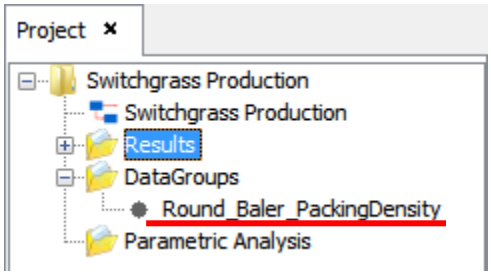


Figure 4.51. The node symbolizing the group of selected results under the folder node DataGroups.

Next, the user should begin creating the chart by clicking “Parametric Analysis” in the Analysis menu (Figure 4.54). The “Parametric Analysis” dialog box will display (Figure 4.53). Steps 5 to 9 will be completed by the user in this window.

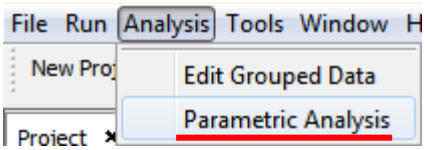


Figure 4.52. The Parametric Analysis Menu Item.

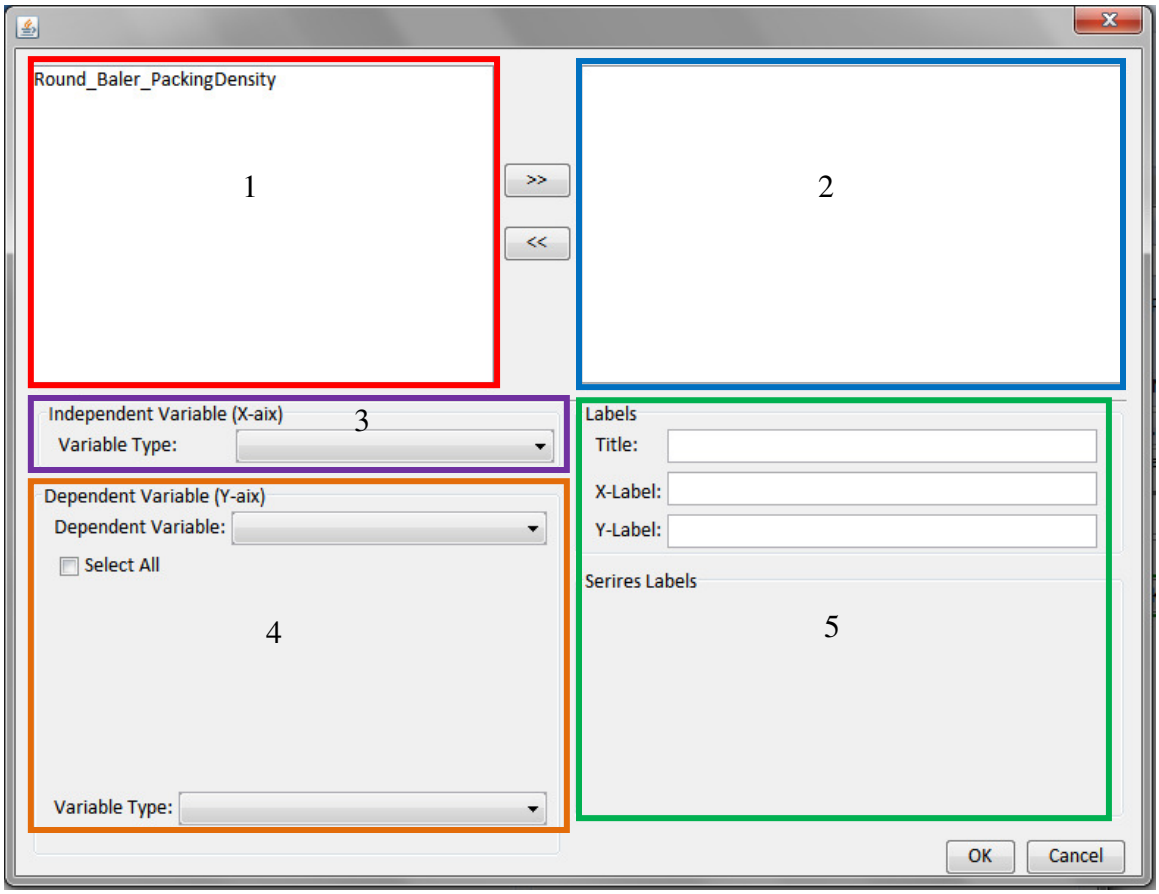


Figure 4.53. The window to configure parametric analysis.

The user can select the groups of results to be compared from the left (part 1 in Figure 4.53) and press the double arrow button to move it to the right (part 2 in Figure 4.53). When an item is moved into the right list, the contents in parts 3 and 4 will be enabled (Figure 4.54).

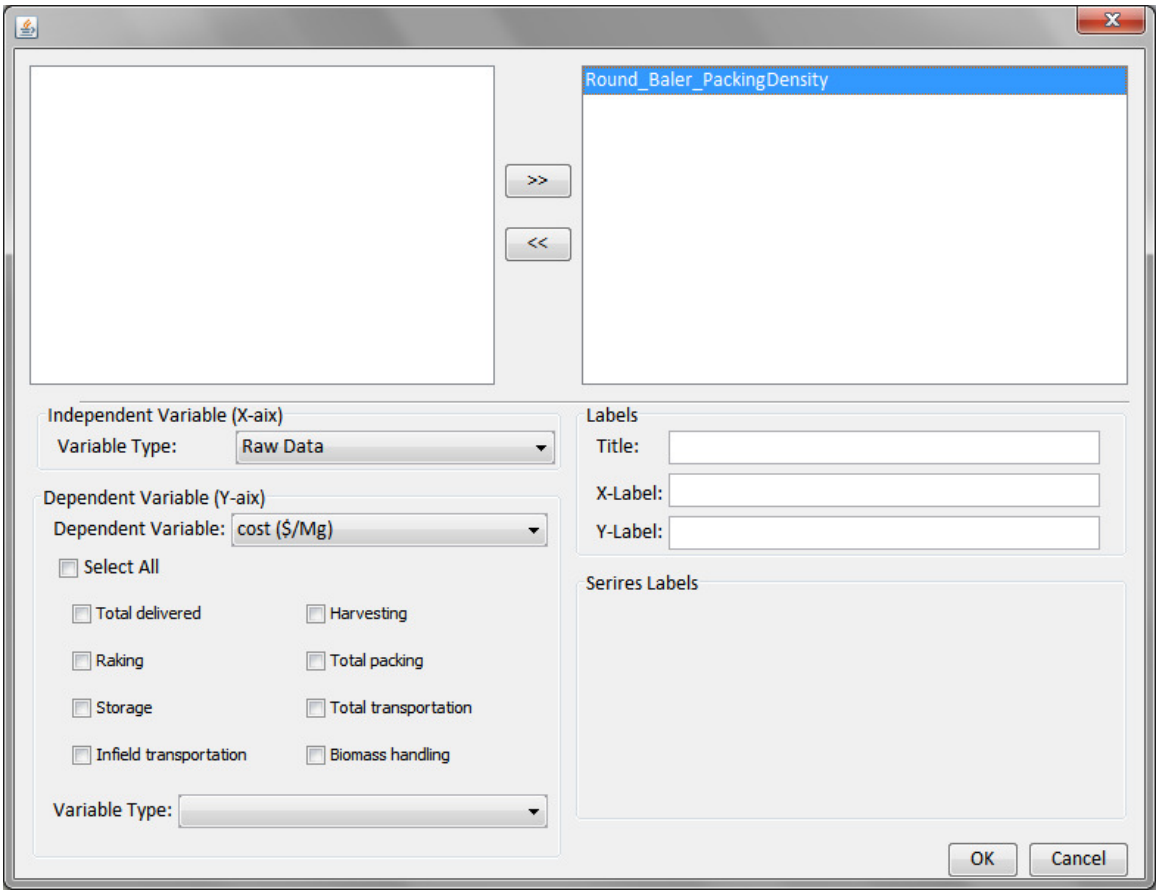


Figure 4.54. The window contents when an item is moved to the right list.

The user should choose how to display the x-axis values by clicking the item list in part 2 (Figure 4.55). “Raw Data” displays x-axis values with no additional processing; “Difference Between Raw Data” displays x-axis values as difference from the first value of the raw data series; and similarly, “Change Ratio of Raw Data” displays x-axis values as ratios from the first value of the original series.

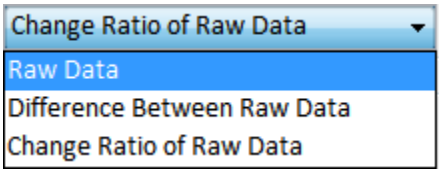


Figure 4.55. Types to prepare x-axis values.

The user will now need to select the system performance indicator they would like to be plotted. Tasks to be plotted will be selected by clicking checkboxes (Figure

4.56). Again the user will decide the format to plot data on the y-axis, under the "Variable Type" list. Axis labels can be edited by the user as well (part 5 in Figure 4.53). The user presses the OK button to finish.

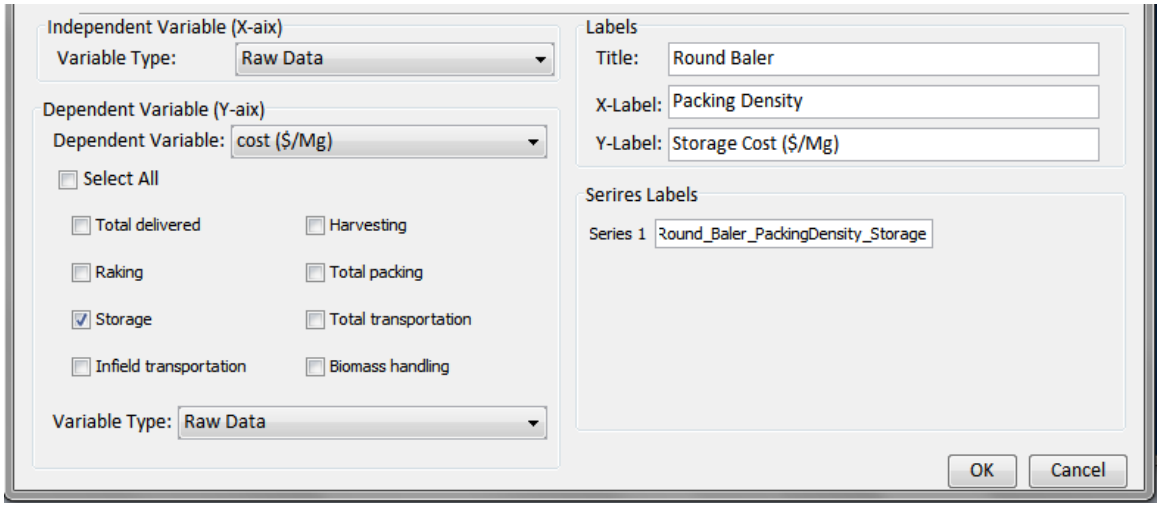


Figure 4.56. The example of the configurations of independent variable, dependent variable and chart labels.

After clicking the OK, a node will be put under the folder node Parametric Analysis to symbolize the comparison (Figure 4.57) and the chart (Figure 4.58) will be displayed with other charts. This function is the manifestation of the user story number nine. The next chapter will discuss more about the Java classes necessary which enable the graphical user interfaces.

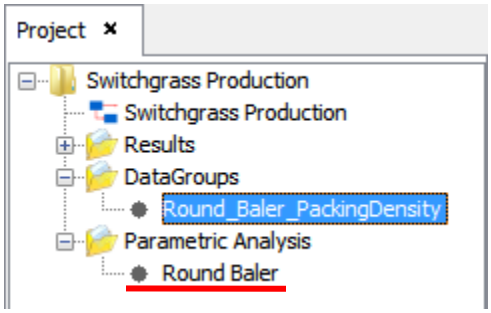


Figure 4.57. The parametric analysis node.

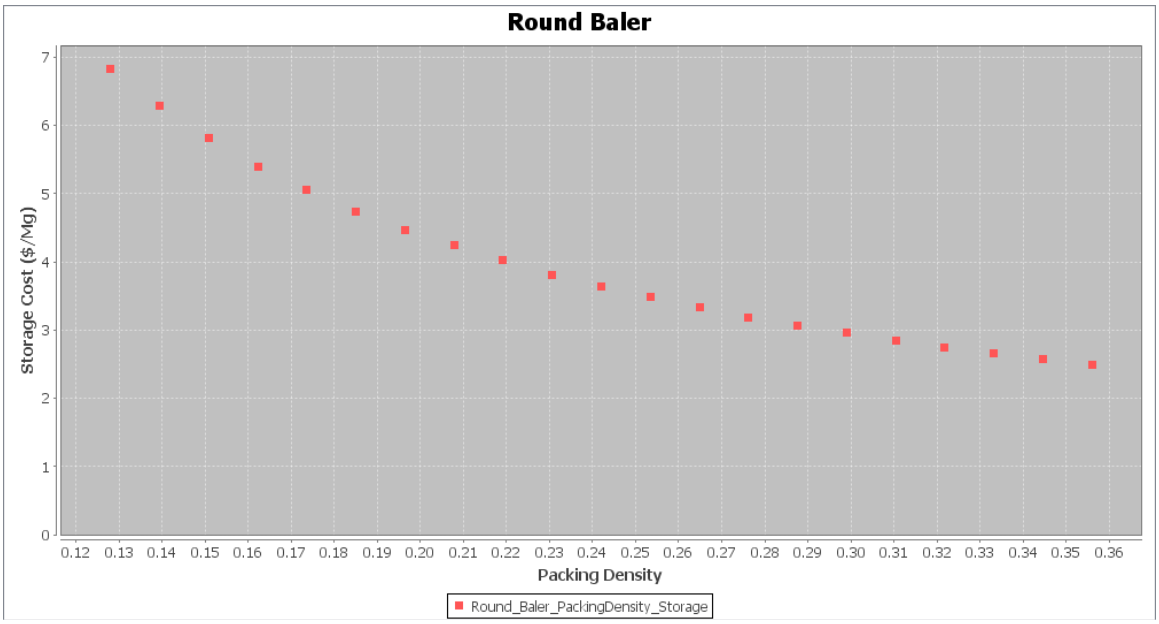


Figure 4.58. The chart for the parametric analysis of a group of results.

Chapter 5. BPSys Procedures

Chapter 5 will discuss the developed Java classes, and the information flow associated with the user stories. The developed classes include two parts: the client side and server side. Table 5.1 lists the client-side and server-side Java classes divided into their specific purposes. These classes are designed to satisfy the defined use cases (Section 3.5). The classes are explained by applying the unified modeling language (UML), including class diagrams, use case diagrams, and sequence diagrams. See Fowler (2003) for an explanation for how to read and interpret UML diagrams.

Table 5.1. List of client-side and server-side packages.

Client-side Package	**Description**
System	Responsible for handling routine operations, e.g. string handling
client.com	Responsible for handling sending requests to the server and retrieving information
client.modeling	Responsible for handling information used for decision support
client.gui	Responsible for providing container components for the user interface
client.gui.basic	Responsible for providing basic user interface components extended from Java Swing components to display
client.gui.database	Responsible for providing components used to build user interface for database
client.gui.project	Responsible for providing components used to build user interface for decision support
Server-side Package	**Description**
bpsys.server.com	Responsible for handling server requests from the client side and returning the requested resources

The use case diagram is charted according to the defined user stories (Section 3.5) and used to describe the interactions between the user and the system via the available

system functions (Figure 5.1). Each oval stands for a use case, named for a different user story. The use cases linked with a solid line are directly called by the user. Dash lines are called by the other use cases. The use cases will be explained in Sections 5.2 with the sequence diagrams.

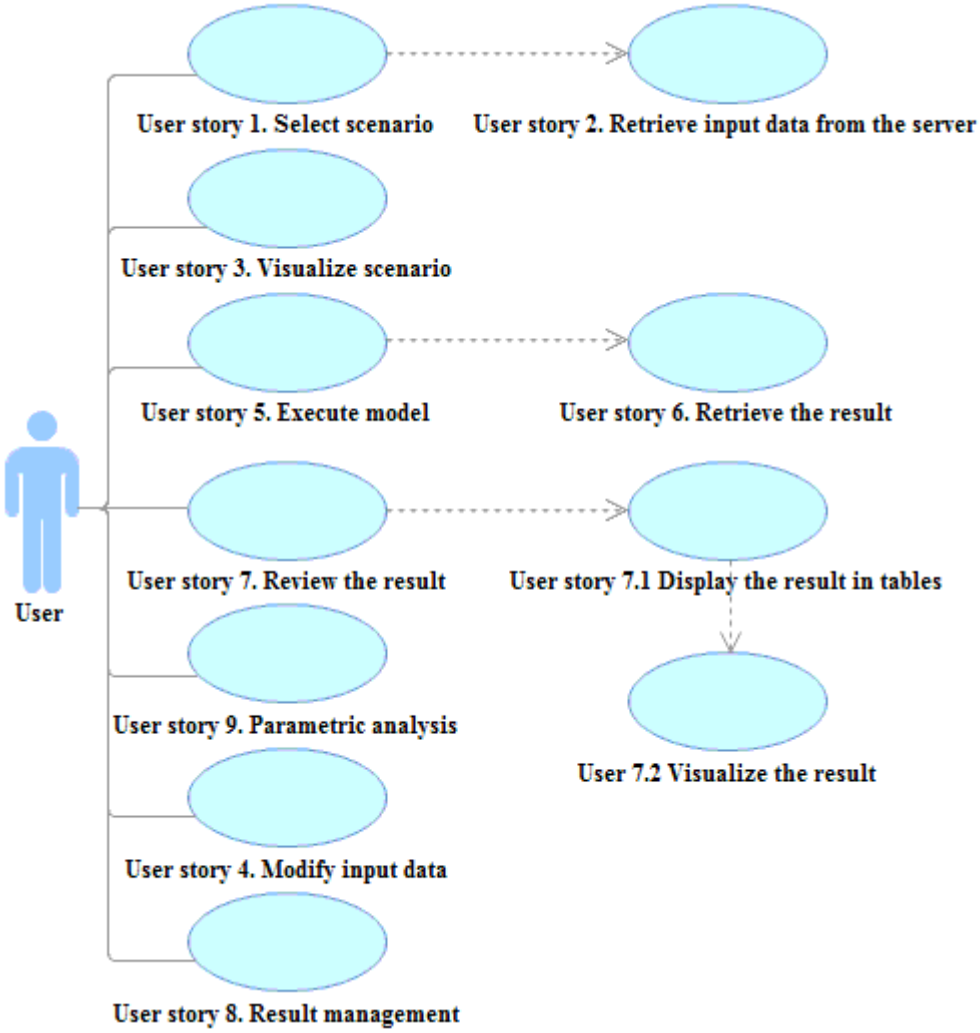


Figure 5.1. BPSysClient use case diagram.

The remaining sections describe how BPSys utilizes the developed Java classes and how BPSys accesses computational resources to provide the user stories. The major Java classes are introduced in the class diagrams in Section 5.1. Section 5.2.1 explains the procedure for creating a project to manage the selected model and results. Section 5.2.2 is about the creation of visualized scenario and attributes tables. The procedure of model

execution is included in Section 5.2.3 and 5.2.4. Section 5.2.5 reveals the procedure of presenting results. Section 5.2.6 discusses the procedures for conducting parametric analysis.

5.1 Java Classes

This section describes the Java classes that make information management and presentation possible with the user interface. The emphasis of this section is on (1) the purposes of the classes, (2) how they are related to the user interface and (3) the relationships between the classes. This is presented via the Unified Modeling Language (UML) (Fowler, 2003). The classes are divided into three groups here, generally according to what they are designed to achieve. Each group of classes is responsible for multiple functions in the user interface. This classification is necessary to follow the logic because the implementation of single function in user interface involves normally more than one class.

The first group of classes introduced in Section 5.1.1 is responsible for the functions of creating a project, a combination of user story 1 and 2. Visualizing the modeled scenario, user story 3, is considered in Section 5.1.2. Modifying input data, and executing the model, user stories 5 and 6, are described in Section 5.1.3. These functions make the users feel comfortable while working with models. The next group of classes introduced in Section 5.1.4 is responsible for handling results, user stories 7 and 8, including presentation, saving, and opening result sets. The Section 5.1.5 describes our approach to parametric analysis for the comparison of results, user story 9. Finally, Section 5.1.6 describes the classes used to communicate between the client and server. The information flows and operations between them are discussed in the following sections Section 5.2 with associated sequence diagrams.

5.1.1 Generating a Project

Ten classes in BPSys are involved in the process of creating a project to retrieve the model results. Project is the core class referring to the other classes directly or indirectly. The UML diagram of the Project class (Figure 5.2) shows that it is supported by five classes: Scenario, Result, DataGroup, ParametricAnalysis, and DependentVariableExtractor. The Projectclass helps the user manage the model, the output of the model, and the meta-information obtained from analysis. These are all symbolized as nodes in the Project window in Section 4.4. The Scenario class processes the information necessary for describing the selected model, such as the order of production chain, and location of the table in the database. The Result class is the container of output from a single execution of the model and related input data. The DataGroup class organizes the grouped Result objects for further processing. The ParametricAnalysis class manages the DataGroup objects for visualization and comparison. The DependentVariableExtractor class is responsible for recognizing and gathering useful information with model output and then creating a Result object.

This design (Figure 5.2) helps the classes specialize themselves and facilitates future development. Each class can be easily replaced or improved in the future. This can also simplify information management because pertinent information is encapsulated.

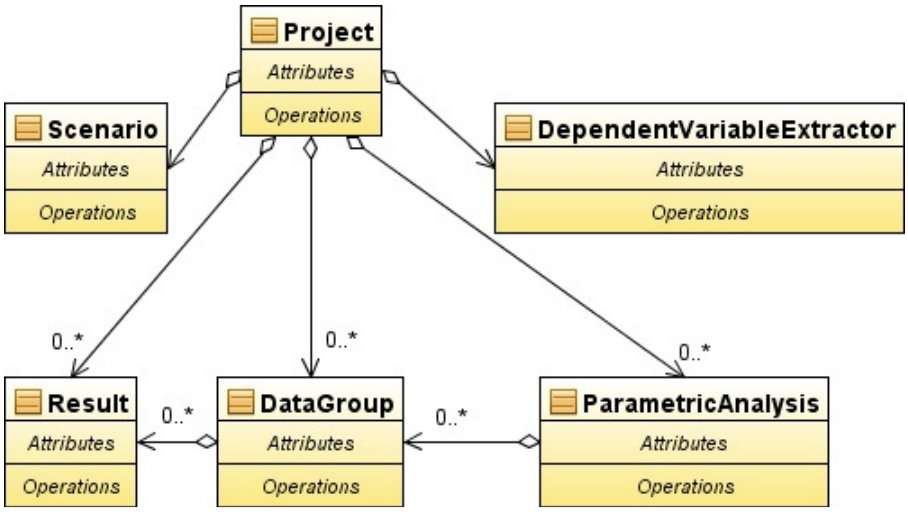


Figure 5.2. A class diagram of the Project class. Boxes represent Java classes developed for the purpose of this research. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

The Scenario class (Figure 5.3) is supported by three classes: ScenarioPanel, ResultOutputReportCreator, and Task. The Task class is the basic element in a modeled scenario; it stores the reference to the previous and the next Task object and connects the object to the database. The ScenarioPanel class creates the object that displays the flowchart of the modeled scenario (Figure 4.13). ResultOutputReportCreator is designed to organize the simulation results into the tables and also create the charts.

The input needed to construct a Scenario object is acquired from a Java-based properties file associated with the selected model from the server (Appendix A). The properties file passes parameters to the BPSys Client along the pathway shown in Figure 3.4 and is designed to be opened readily using native Java commands provided by the Properties Class (java.util package). The contents of the properties file gathers information for multiple purposes, including creation of a project (Section 5.2.1), visualization of the modeled scenario (Section 5.2.2), execution of the model (Section

5.2.3), and presentation of the results (Section 5.2.5).

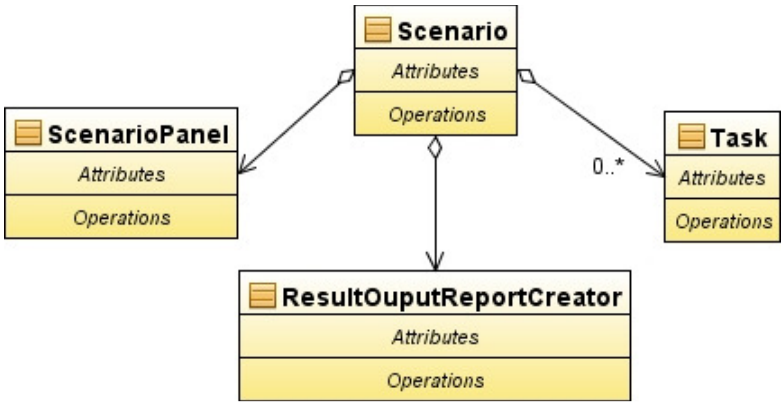


Figure 5.3. Class diagram of Scenario class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

5.1.2 Visualizing a Modeled Scenario and Attribute Table

The ScenarioPanel class (Figure 5.4) is supported by two classes: TaskButton and JPanel. The ScenarioPanel class uses the array of the Task objects in the object of the Scenario class to draw the scenario flowchart. For each Task object, there will be a TaskButton object declared in the ScenarioPanel to represent it. While the ScenarioPanel is being created, the equipment data for each task is downloaded from the database and used to construct a table as in Figure 4.15. The procedure for instantiating a graphical scenario is presented in Section 5.2.2.

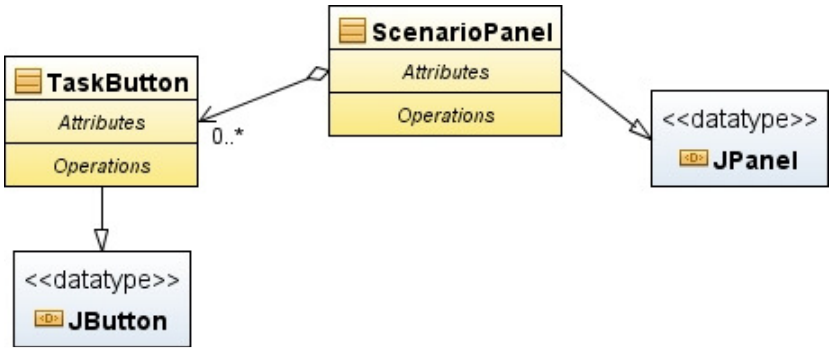


Figure 5.4. Class diagram of ScenarioPanel class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

5.1.3 Executing the Model

The user story Execute Model is among one of the most complicated cases. It includes three sub tasks and majorly relies on the classes Project, Scenario, and Servlet which are used to send request to server (Section 5.1.6). The sequence of function calls for the single execution of the model is introduced in Section 5.2.3. For batch execution see Section 5.2.4.

5.1.4 Presenting Results

When a project receives raw output from the server, it depends on several classes to extract useful information, create tables, and make charts. A class, named Result, is designed for containing the input to and output from the model. Within the Result class, two types of data are handled by two different classes: ResultOutput and ResultInput (Figure 5.5). The Scenario class, described above, has access to the resource properties file, which is needed for creating the ResultOutput object. The Result class is able to create visual components to display input and output data as well. The visual components of output are created via the ResultOutputReportCreator object accessed through the

associated Scenario object.

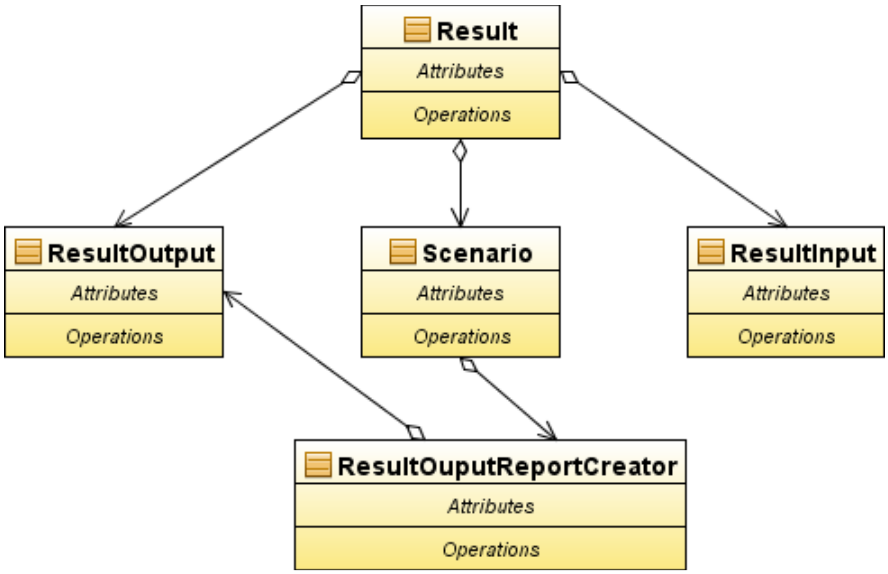


Figure 5.5. Class diagram of Result class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part.

The ResultOutput class depends on the ProfileVariable class and the DependentVariable class (Figure 5.6). These two Java classes handle different kinds of variables in the result set. The ProfileVariable class is for storing a continuous series of data; for example, the number of trucks number required per day throughout a whole year for moving biomass (Figure 4.35). This is as opposed to the DependentVariable class for storing variables which describe discrete values; for example, the cost for each task in the production system is handled by a DependentVariable object. Arrays of ProfileVariable objects and DependentVariable objects are created from the raw output data by the DependentVariableExtractor object in the Project object and they are used to visualize the charts and tables later. The procedures for extracting theses variables are described in Section 5.2.5.1.

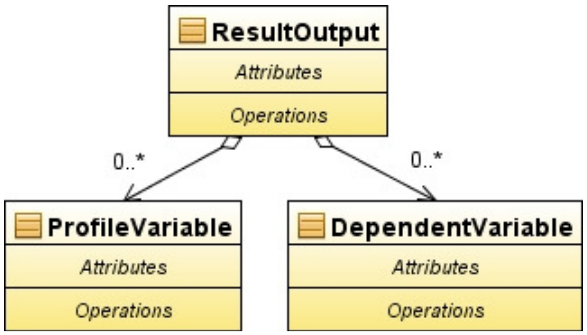


Figure 5.6. Class diagram of ResultOutput class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

The ResultInput class contains the data fed into the model to generate the output. The class diagram (Figure 5.7) shows that the ResultInput class refers to an array of IndependentVariable objects. The IndependentVariable objects hold the data describing each task in the modeled system. The IndependentVariable class has three attributes for describing the input data of a task. The attribute "Name" stores the name of the attributes table (Figure 4.15); the array "fieldname" stores the first row in the table, including the attribute name and the unit, except the first cell "Selected" (Figure 4.15); the array "values" stores the values on the selected equipment.

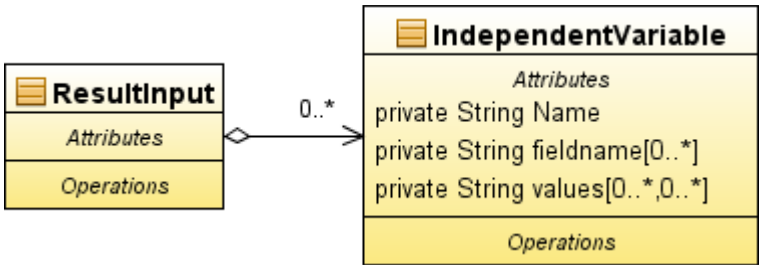


Figure 5.7. Class diagram of ResultInput class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

5.1.5 Comparing Results

The classes in this section implement the parametric analysis. The ParametricAnalysis class (Figure 5.8) is supported by three classes: XYSeriesDataset, DataGroup, and JFreeChartCreator. The DataGroup contains a set of grouped Results objects and make it reusable. The DataGroup object is symbolized as a node under the GroupData node (Figure 4.51) in the Project window (Figure 4.10). The XYSeriesDataset class gathers data from a DataGroup object. It will also output a standardized dataset, which is sent to a JFreeChartCreator object to generate plots (Figure 4.58). The JFreeChartCreator class takes advantage of an open source Java library, JFreeChart (Gilbert, 2009), to create the charts. The purposes and use of the developed classes is explained in Section 5.2.6.

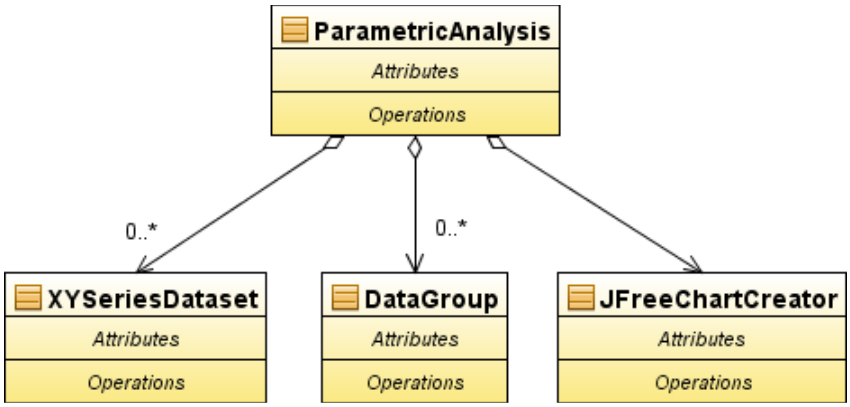


Figure 5.8. Class diagram of ParametricAnalysis class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part. The label "0..*" represents the cardinality of objects in the relationship; in this case, there may be zero to many, represented by the "*", objects that may be part of the whole.

5.1.6 Supporting Infrastructure: Sending Requests

The classes introduced in this section are used to transmit data between the client side and the server side. Figure 5.9 displays the classes used on the client side to send requests to the server for various services including downloading data, executing the

model, and uploading information. The requested services and necessary information are stored in the RequestInfo class in Figure 5.9. The necessary information is dependent on the requested services. For example, it will include the file name if trying to download a file from the server, which is not required for downloading data from the database. The Request class is used to create an HTTP request to transmit a RequestInfo object to a servlet specified by the internet address stored in the attribute "url" (Figure 5.9). Calling the operation "processRequest()" will proceed to execute the request. Once the request is completed successfully, the operation "getReceivedData()" will retrieve the information associated with the requested service.

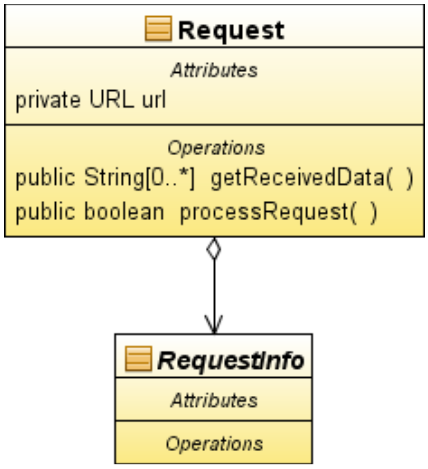


Figure 5.9. Class diagram of Request class. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part.

The server side classes that receive requests are displayed in Figure 5.10. The three servlets referred to in Figure 5.10 are designed for unique purposes. The download servlet handles requests for downloading information stored in either the properties file or the database. The runmodel servlet is responsible for executing the model. The upload servlet uploads the input data and store in on server side files and the database. Each of the three servlets are extended from the Java™ class HttpServlet. When the three servlets

receive a request from the client, the RequestInfo object sending with the request is read and analyzed by the responsible servlet to conduct the service. When the request is completed, the responsible servlet sends the message and the requested information to the client.

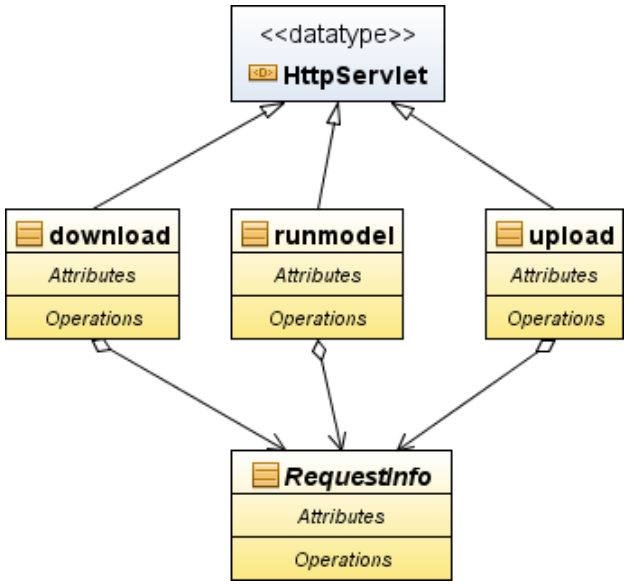


Figure 5.10. Class diagram of servlet classes. Arrows with diamonds at the tail represent whole-part relationships, where the diamond end signifies the whole and the arrow signifies the part.

5.2 Sequence Diagrams & Algorithms

5.2.1 Generating a Project

Figure 5.11 displays the procedure for creating a project with six steps executed after the user selects the model through user interface. Step 1 is to download the descriptions of the model from a properties file on the server. Step 2 begins the construction of a Project object by passing in the downloaded properties of the model and the project name. Step 3, a Scenario object is constructed using the constructor method of the Project class. The properties of the model are passed into the constructor method of the Scenario class. Step 4 is to create the objects of the Task class within the construction method of the Scenario class. After creating the Task objects, the default input data is

downloaded from the server in step 5. Step 6 finishes the construction of the project and displays the project node (Figure 4.11).

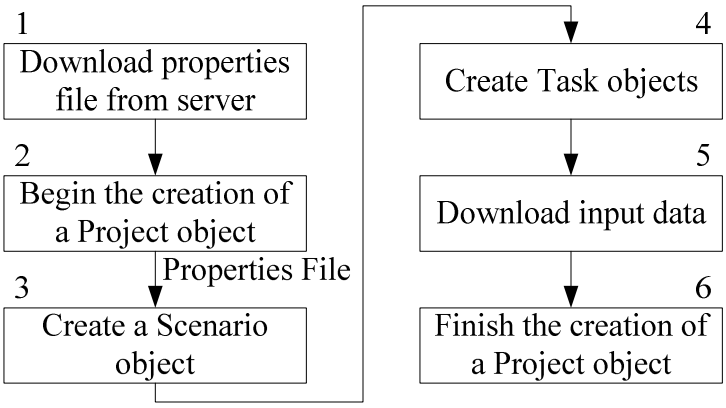


Figure 5.11. Flowchart of creating a project.

Figure 5.12 shows the sequence diagram for creating a new project. In this diagram, the classes involved are shown with the functions called for creating a project. The involved classes are listed on the top row of blue boxes in the diagram. The label of the box contains two pieces of information separated by the colon. The label on the right side represents the name of the class. The label on the left side represents the name taken to create the objects of the class. In Figure 5.12, the classes used to create a project include ProjectManager, Project, and Scenario. The call of a function is initiated from a labeled line with an arrow and the blue bar represents the length of the call. If the called function returns some information, the blue bar is ended with the company of a line with an arrow back to where this function is called. The labels aside or above the lines describe the depth of the call with the numbers and the name of the called functions with the text. The sequence diagrams are presented in this chapter to describe the behaviors of the program that conduct the steps outline in the flowcharts (for example creating a project Figure 5.11).

Step 1 in Figure 5.11 is finished within the segment between the arrows 1.1 and

1.2. Step 2 is represented by the arrow 1.3.1. Step 3 is completed between the arrows 1.3.1.1 and 1.3.1.2. Step 4 is accomplished between the arrows 1.3.1.1.1 and 1.3.1.1.2. Step 5 is finished between the arrows 1.3.3 and 1.3.4. The returned arrow 1.4 represents Step 6. Once the project is created, it will be assigned and displayed in Project window as a root node (Figure 4.10). The two coming subsections describe the detailed procedures of steps 4 and 5 respectively.

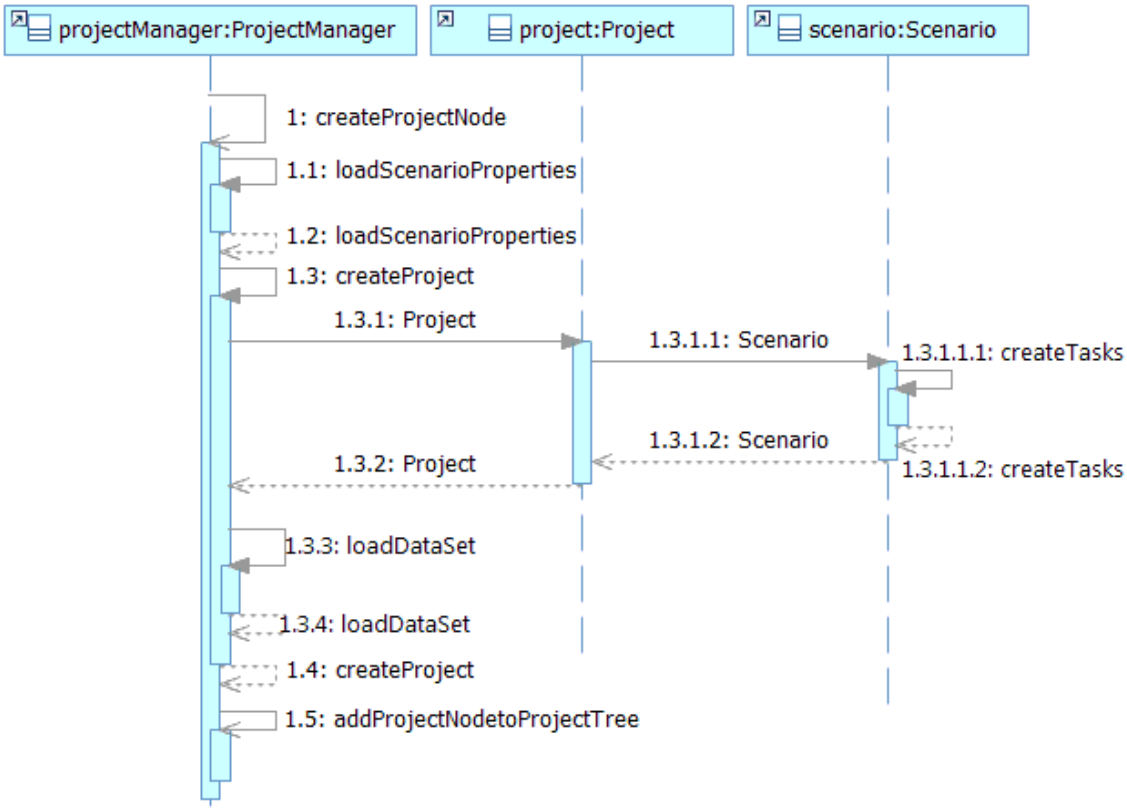


Figure 5.12. Sequence diagram of creating project.

5.2.1.1 Creating Tasks

This section discusses the process for extracting the information from the properties file for the creation of the Task objects and the ScenarioPanel. The contents in the properties file relates to declaring Task objects and extraction of the raw data displayed in Figure 5.13. The first line in Figure 5.13, NumofTask=9, represents how many Task objects the Scenario object consists of; the following lines, from 2 to 17,

provide the attributes for declaring the Task objects. Each declaration of the Task class needs two attributes, the name of the Task and the boolean value describing whether it is editable. When the value of editable property is equal to true, there is a set of input data applied to the Task object and it can be edited to meet user's needs. When the value is equal to false, it represents a set of input data provided by the model for this Task, but modification by the user is not permitted. If the editable property doesn't exist for certain tasks, there is no input data provided in the MySQL database. For instance, there is no editable property for "Task1" in Figure 5.13 and there is currently no table describing biomass growth at this time. The procedure of creating the Task objects is as shown in Figure 5.14. First, the number of Tasks is read in from the properties file. If the number is greater than 0, it enters the creation loop. In the loop, the program reads Task name and the editable property and then creates the Task N. The N refers to the number following the text "Task" in Figure 5.13. Table 5.2 lists all the Task objects created after the loop is completed. There are nine Task objects and seven of them specify input data that could be edited. In Table 5.2, the column "Editable" indicates whether data is made available for editing by the user. If so, the table in the MySQL database is specified in the "Table name in MySQL" column in Table 5.2.

```
1   NumofTask=9
2   Task1.class=BiomassGrowth
3   Task2.class=Harvesting
4   Task2.editable=true
5   Task3.class=Raking
6   Task3.editable=true
7   Task4.class=Packing
8   Task4.editable=true
9   Task5.class=FarmOpenStorage
10  Task5.editable=true
11  Task6.class=FarmCoveredStorage
12  Task6.editable=true
13  Task7.class=CentralizedStorage
14  Task7.editable=true
15  Task8.class=Transportation
16  Task8.editable=true
17  Task9.class=Biorefinery
```

Figure 5.13. The content format in the properties file for creating Task objects.

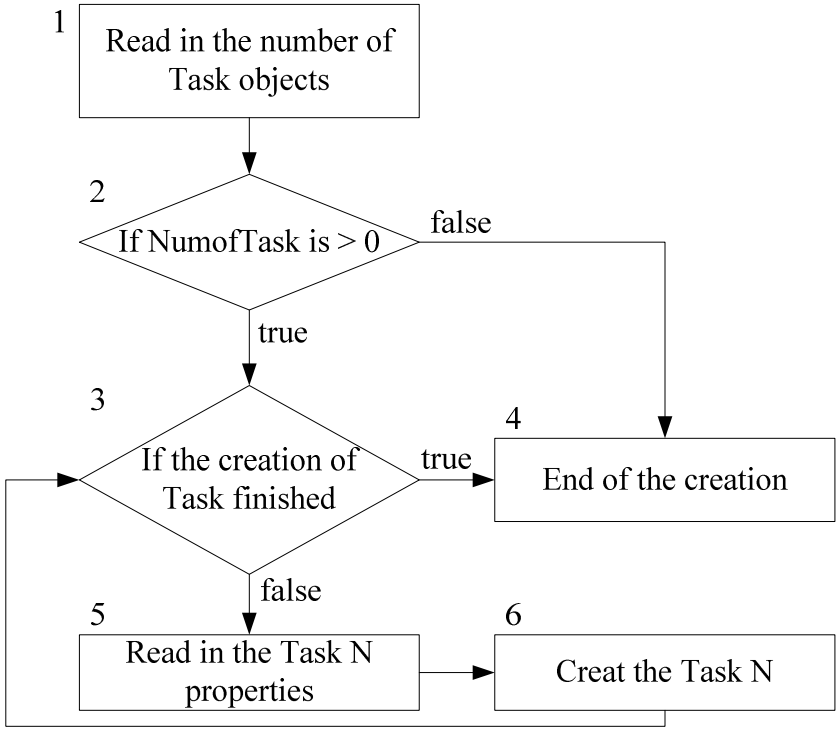


Figure 5.14. The flowchart of creating Task objects.

Table 5.2. The properties of created Tasks.

ID	Task name	Editable	Table name in MySQL
Task1	BiomassGrowth	null	
Task2	Harvesting	true	Harvesting
Task3	Raking	true	Raking
Task4	Packing	true	Packing
Task5	FarmOpenStorage	true	FarmOpenStorage
Task6	FarmCoveredStorage	true	FarmCoveredStorage
Task7	CentralizedStorage	true	CentralizedStorage
Task8	Transportation	true	Transportation
Task9	Biorefinery	null	

Figure 5.15 shows the sequence diagram of creating Task objects. Steps 1 and 2 in Figure 5.14 are completed before entering the loop in Figure 5.16. The arrows inside the loop perform steps 3, 5, and 6 repeatedly.

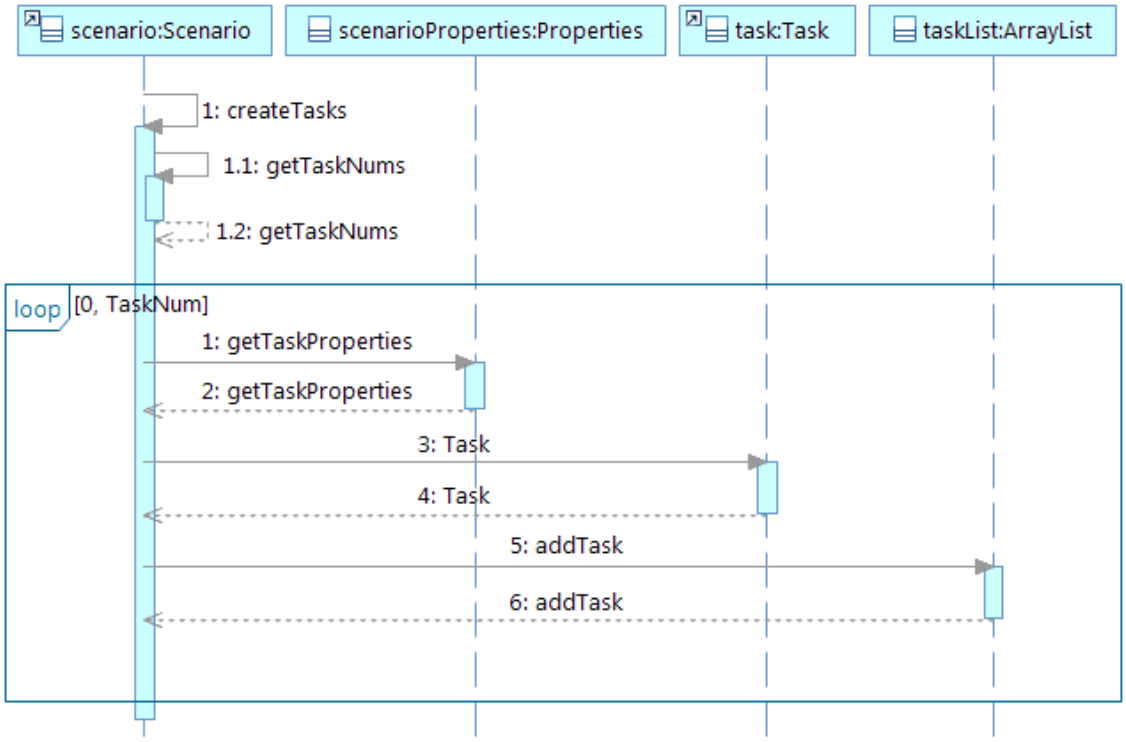


Figure 5.15. Sequence diagram of creating Task objects.

5.2.1.2 Downloading Input Data

This section describes the steps for downloading input data from the MySQL

server used to support the BioFeed model and BPSys (Figure 5.16). The array of Task objects is passed into the method to start the download. If the "editable" property of Task N is equal to true or false, the name of the Task N is then used as the table name to construct a query and download the input data of the task from the table in MySQL server. Although the project is created and the related data is downloaded, the visualization of the modeled scenario and the tables for modifying attributes values are not yet created. The next section 5.2.2 introduces the procedures for creating these visual components.

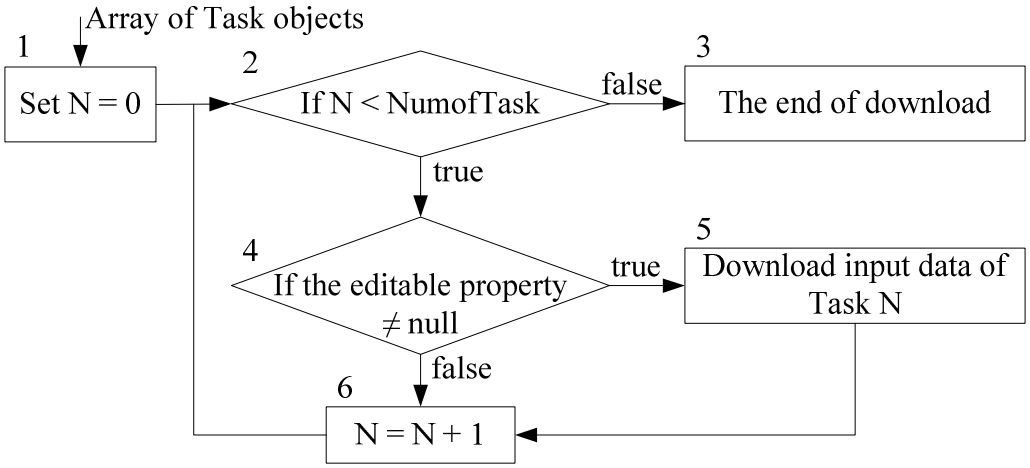


Figure 5.16. The flowchart of downloading input data. NumofTask is the variable retrieved from the content in the properties file shown in Figure 5.13. It is set to inform that the number of Task object should be created for the selected scenario.

Figure 5.17 shows the sequence diagram for downloading input data. The array of Task objects (Figure 5.16) are prepared by the segment between the arrows 1 and 1.2 and passed to the loop shown in Figure 5.17. The loop repeats as it performs steps 2 – 6 in Figure 5.16.

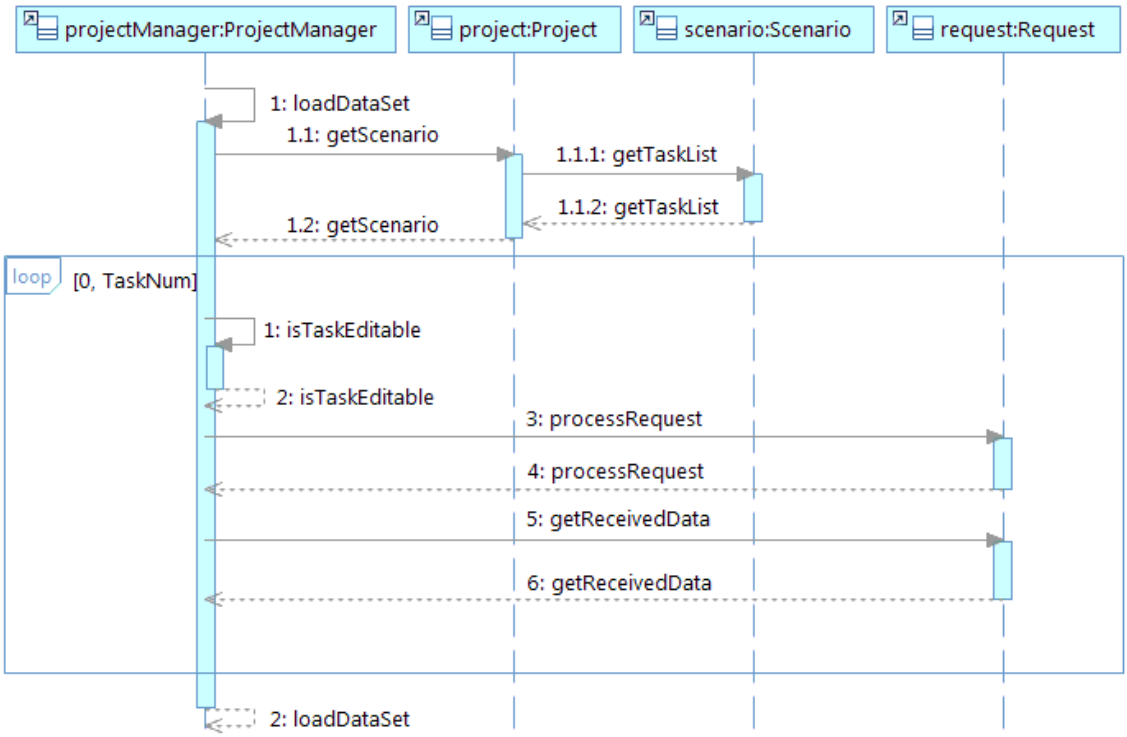


Figure 5.17. Sequence diagram of downloading input data.

5.2.2 Visualizing a Modeled Scenario and Attribute Table

The procedure shown below, in Figure 5.18, will occur if the user double clicks the scenario node (illustrated in Section 4.4.2). When a scenario node is double clicked, the user interface of BPSys will identify which scenario node is double clicked and call the corresponding Scenario object to create the visual components. Steps 2 and 3 are then performed within the Scenario object. When these two steps are finished, the Scenario object returns the visual components to the user interface to be displayed.

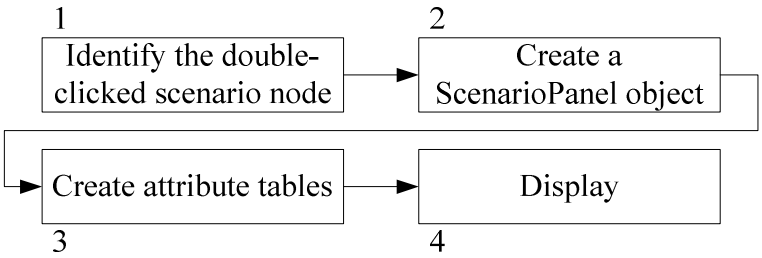


Figure 5.18. Flowchart of visualizing modeled scenario and attribute tables for tasks.

Figure 5.19 shows the sequence diagram of visualizing scenario and tables. Step 1

in Figure 5.18 is represented by the segment between the arrows 1 and 1.1. Steps 2 and 3 are completed by calling functions via the arrow 1.1 and 1.2 separately. The next two subsections will show steps 2 and 3 in more detail.

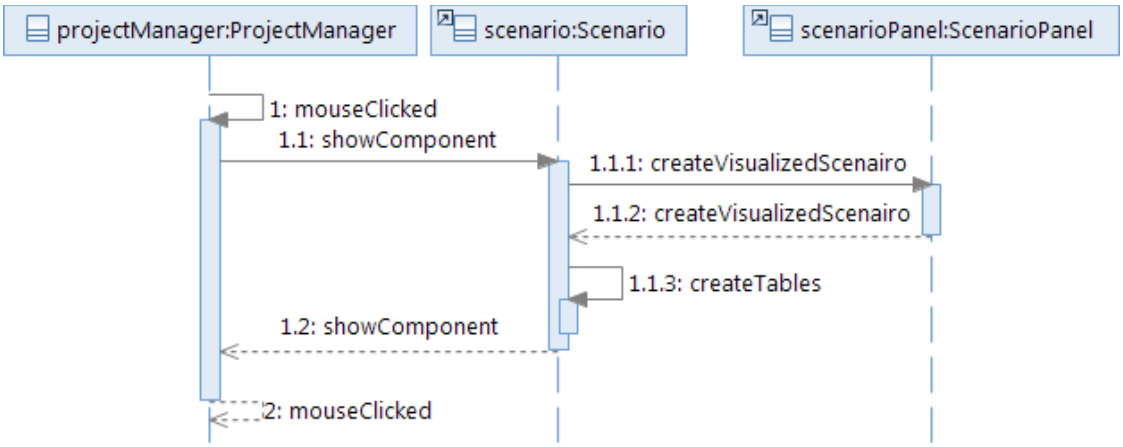


Figure 5.19. Sequence diagram of visualizing scenario and tables.

5.2.2.1 Creating a ScenarioPanel Object

The instructions for creating the ScenarioPanel are acquired from the properties file (Figure 5.20). There are three key pieces of information in the properties file involved in assembling the scenario flowchart. The first the number of the buttons is established from value of NumofBox. The second is the description of the task the button represents, lines 2 through 11. The last is a description of the connections the button should have in the process flow chart. A task in the scenario can be connected to one or many buttons in the scenario flowchart (Figure 4.13). For example, Box5 and Box9, labeled by the text "Transportation" in Figure 4.13, both refer to Task8 (Table 5.3). Thus, buttons can arbitrarily connect to as well as be connected by others if these connections are reasonable in the modeled scenario.

```
1   NumofBox=10
2   Box1.Task=Task1
3   Box2.Task=Task2
4   Box3.Task=Task3
5   Box4.Task=Task4
6   Box5.Task=Task8
7   Box6.Task=Task5
8   Box7.Task=Task6
9   Box8.Task=Task7
10  Box9.Task=Task8
11  Box10.Task=Task9
12  Box1.next=Box2
13  Box2.next=Box3
14  Box3.next=Box4
15  Box4.next=Box5
16  Box5.next=Box6,Box7,Box8,Box10
17  Box6.next=Box9
18  Box7.next=Box9
19  Box8.next=Box9
20  Box9.next=Box10
```

Figure 5.20. The content in the properties file for creating a ScenarioPanel object.

Table 5.3. The referred task and connection(s) for each button.

	Referred task	Input Connections	Output Connections
Box1	BiomassGrowth (Task1)		Box2
Box2	Harvesting (Task2)	Box1	Box3
Box3	Raking (Task3)	Box2	Box4
Box4	Packing (Task4)	Box3	Box5
Box5	Transportation (Task8)	Box4	Box6,Box7,Box8,Box10
Box6	FarmOpenStorage (Task5)	Box5	Box9
Box7	FarmCoveredStorage (Task6)	Box5	Box9
Box8	CentralizedStorage (Task7)	Box5	Box9
Box9	Transportation (Task8)	Box6,Box7,Box8	Box10
Box10	Biorefinery (Task10)	Box5	

Figure 5.21 shows the flowchart for creating the visualized scenario. It starts

with the passing of the properties file into the constructor of ScenarioPanel class. The first block indicates the initialization of Java™ Swing component. The second block retrieves the value of NumofBox in the properties file to set the number of iterations in the loop. In the loop, the properties of the button N are retrieved to declare and establish each button. After the completion of the loop, the buttons are added into the GUI component list of the ScenarioPanel object for displaying them later on the screen. The coordinate of each button must be adjusted because the initial coordinate for any Java™ Swing component is (0, 0). Each button is located based on the order of the connections. The links are drawn as red lines after the coordinates for all buttons are certain.

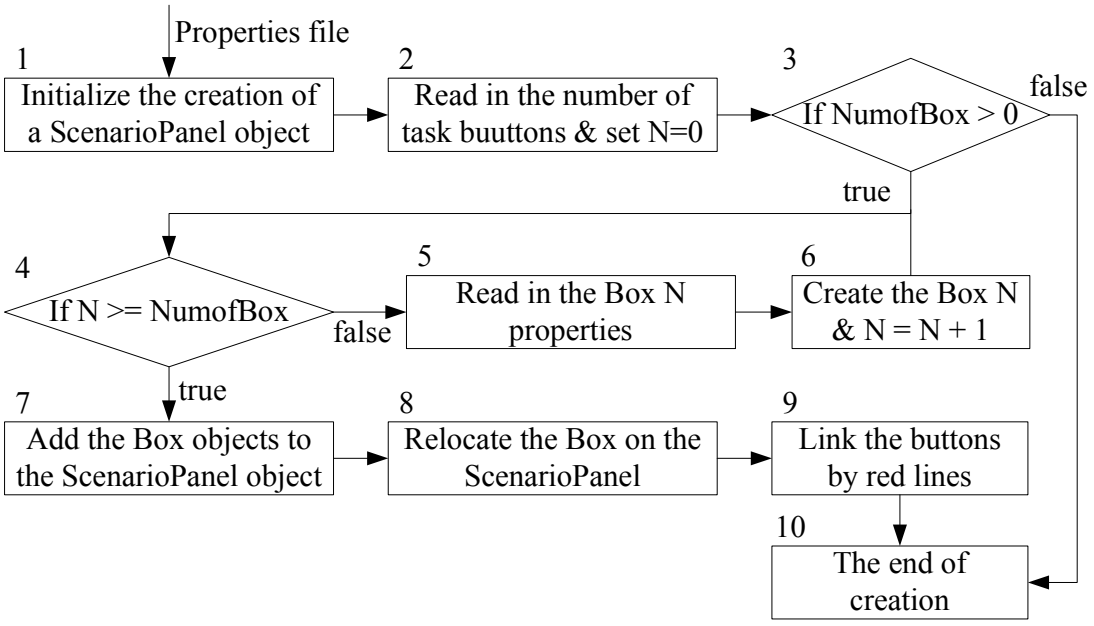


Figure 5.21. The flowchart of creating a ScenarioPanel object.

Figure 5.22 shows the sequence diagram of creating a ScenarioPanel object. The segment between the arrows 1 and 2 is for steps 1 and 2 in Figure 5.21. The buttons shown in Figure 4.13 (a) are created by calling the method, createVisualizedScenario, of ScenarioPanel along the arrow 3. Through the arrow 3.1, steps 3 – 7 are accomplished. Steps 8 and 9 are finished by calling the function, arrangeButton, of ScenarioPanel via

the arrow 3.2.

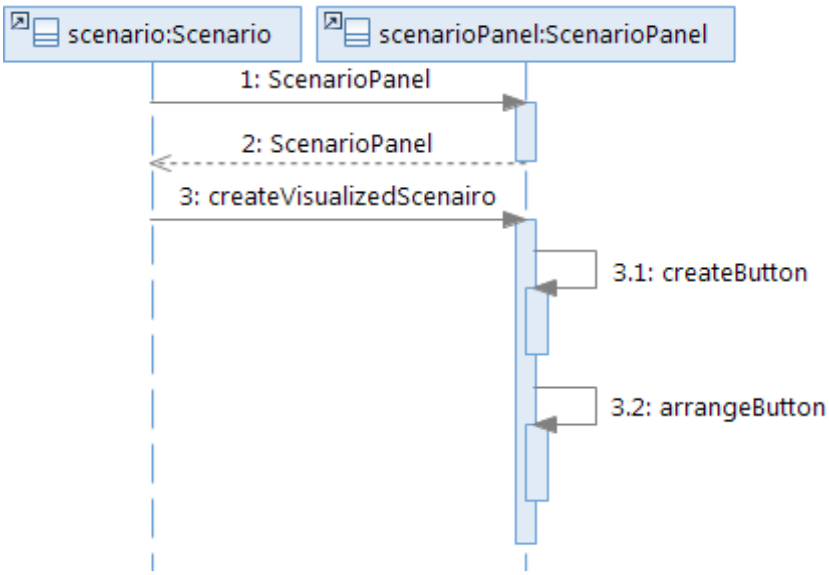


Figure 5.22. Sequence diagram of creating a ScenarioPanel object.

The buttons put on the panel not only play a role for interpretation of the scenario but provide users the ability to interact with the data. Users can view the attributes table of a task via clicking the corresponding button. The following section 5.2.2.2 describes the procedures for creating tables and how to connect the attributes tables with task buttons and allowing the user to browse the tables with a button click.

5.2.2.2 Creating Attribute Tables

The flowchart Figure 5.23 shows the steps necessary for creating tables, displaying them, and creating links with buttons in the ScenarioPanel. The tables are created and put into an array of tables in a loop if the value for the editable property of the task is true (Figure 5.13). After the loop is completed, the next step is to display the attributes tables in the JTabbedPane where every table occupies a tab (Figure 5.24). The label of the tab is the task name. The tables are passed into the corresponding Task objects. Lastly, the connection of the table and the button is via the Task object. If there is an attributes table declared for the Task object, the Task object declares a mouse action listener to monitor

the mouse click on the button of the Task object. If the button is clicked, the tab containing the table is presented.

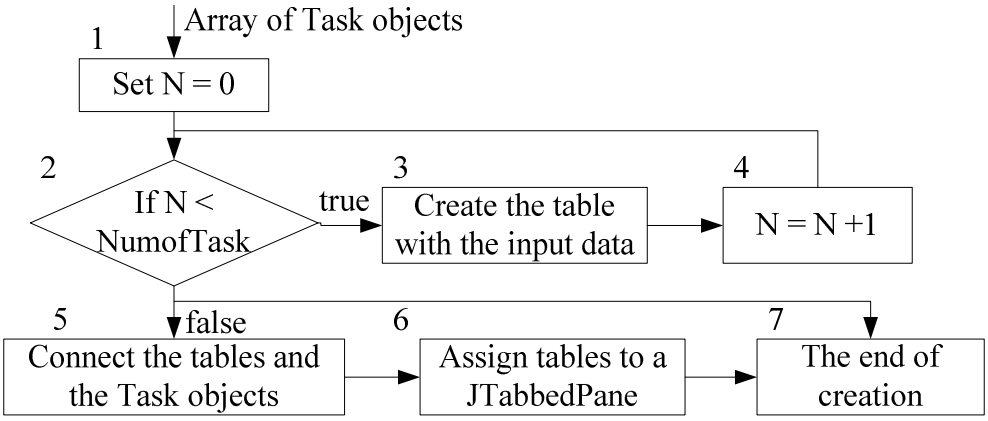


Figure 5.23. The flowchart of creating tables of input data.

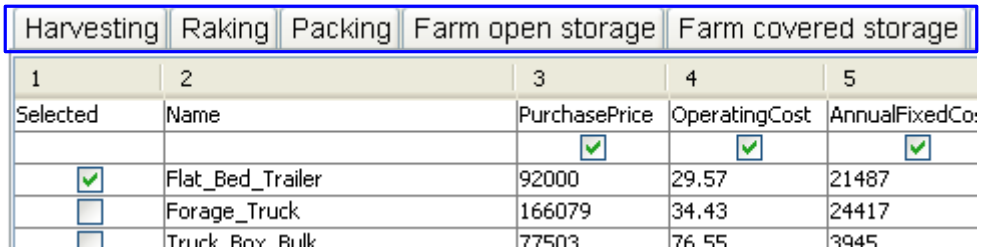


Figure 5.24. The tabs of attributes tables in a JTabbedPane.

Figure 5.25 shows the sequence diagram of creating attribute tables for a selected scenario. Steps 1 – 4 in Figure 5.23 are finished by the part of program between the arrows 1 and 1.1. Step 5 is carried out by the function setTable of Task class. Step 6 is finished between the arrows 1.3 and 1.4.

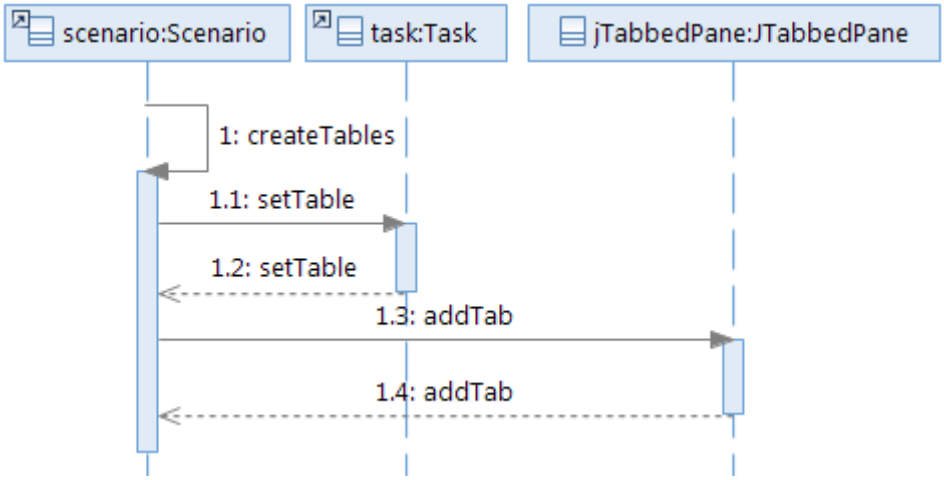


Figure 5.25. Sequence diagram of creating attribute tables.

Through the steps demonstrated in the section, the Task object, the containers of the input data for each task in the scenario, the visual components for the scenario and input data are established. The links between the graphical visualizations are also made. The next section introduces the procedure to run the model remotely on the server.

5.2.3 Conducting Single Model Execution

The flowchart of model execution is illustrated in Figure 5.26 and consists of three major steps. The first one is to collect data describing the selected equipment attributes and organize into a format defined by the model. The second step is to upload the data to the server. The third step involves asking server to run the model and returning the output. The following subsections describe these three steps in detail.

Figure 5.26. The abstract flowchart of model execution.

Figure 5.27 shows the sequence diagram of model execution. The arrow 1 represents the initiation of model execution. The segment between 1.1 and 1.4 involves the collection of input data for the selected Scenario object (step 1 in Figure 5.26). The segment between the arrow 1.5 and 1.6 uploads the data. The segment between the arrow

1.7 and 1.8 handles execution of the model and returning results in Figure 5.26.

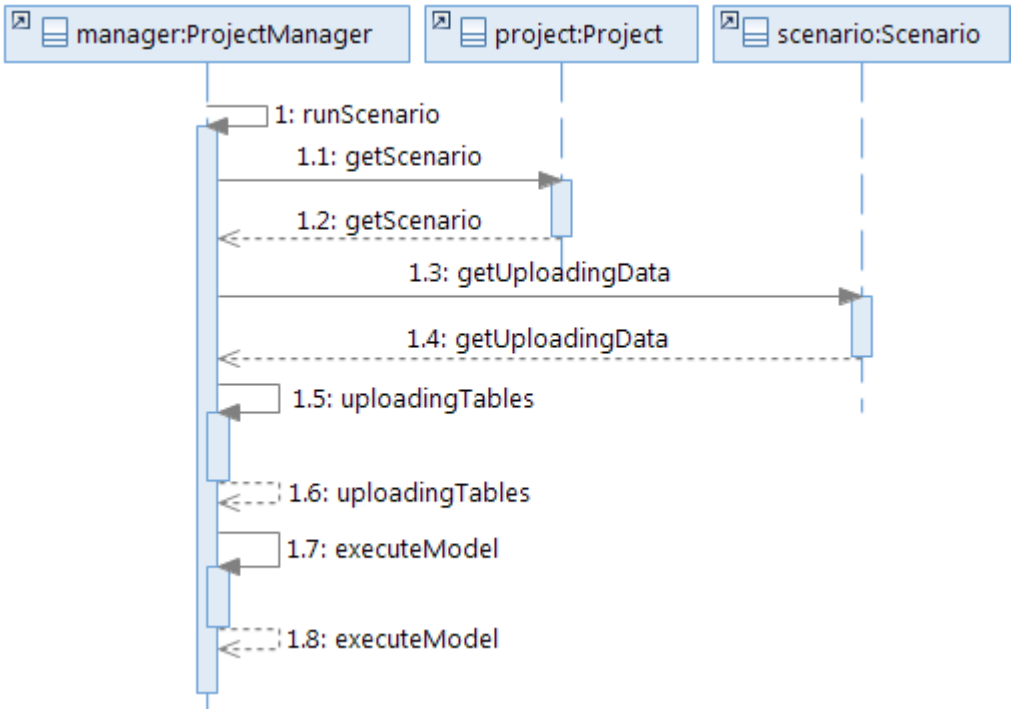
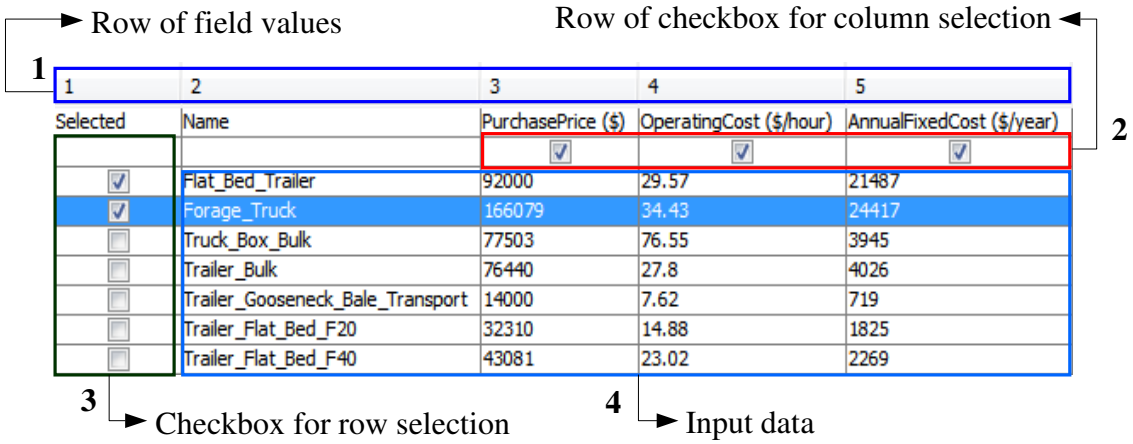


Figure 5.27. Sequence diagram of model execution.

5.2.3.1 Collecting Input Data

This section depicts what data should be extracted and what formatting occurs for the selected model. The model, BioFeed, utilized by BPSys requires three different input files for each task for exectution. They are called ColumnIdentifiers, Types, and Table. Taking the tables in Figure 5.28 as an example, the three files are shown in Figure 5.29, with the first two pieces of equipment selected. The “Types” file (Figure 5.29 (a)) stores the values under the “Name” for the selected rows or the name of the selected equipment. The “ColumnIdentifiers” file (Figure 5.29 (b)) stores only the headings for each column (part 1 in Figure 5.28). The “Table” file (Figure 5.29 (c)) stores the data within each of the selected rows (parts 2 and 4 in Figure 5.28). These three input files will be stored in CSV format with different filenames on the server and read by BioFeed when the model is executed. The actual contents and ways to arrange input data of three input files are

illustrated in Appendix B.



1	2	3	4	5
Selected	Name	PurchasePrice ($)	OperatingCost ($/hour)	AnnualFixedCost ($/year)
☑	Flat_Bed_Trailer	92000	29.57	21487
☑	Forage_Truck	166079	34.43	24417
☐	Truck_Box_Bulk	77503	76.55	3945
☐	Trailer_Bulk	76440	27.8	4026
☐	Trailer_Gooseneck_Bale_Transport	14000	7.62	719
☐	Trailer_Flat_Bed_F20	32310	14.88	1825
☐	Trailer_Flat_Bed_F40	43081	23.02	2269

Figure 5.28. The attribute table for preparing input file.

(a)	(b)	(c)
Flat_Bed_Trailer Forage_Truck	Name PurchasePrice OperatingCost AnnualFixedCost	Name,Flat_Bed_Trailer,Forage_Truck PurchasePrice,92000,166079 OperatingCost,29.57,34.43 AnnualFixedCost,21487,24417

Figure 5.29. The content of input files: (a) Types (b) ColumnIdentifiers (c) Table files.

Figure 5.30 shows the procedure for producing the input data files for each attribute table. The array of attribute tables is passed into the loop. The size of the array is dependent on the selected scenario. For example, the size for the illustrated scenario in Chapter 4 is seven. The types of input files and saved filenames required for each attribute table is identified within the loop consist of steps 2 – 5. Information regarding what necessary types and filenames of input files are created is recorded according to the properties file illustrated in Figure 5.31.

The contents in Figure 5.31 provide the types of input files and the associated filenames for each attributes table. The text before the equality is the combination of the

table name and the input file type, separated by a period. The text after the equality specifies the filename. During step 3, the program will read every line to search the properties file regarding the required input files for a given table. Step 4 then extracts the proper data for each input file from the attribute table. Once all the data has been collected for BioFeed, there are twenty-one files of data. Next these files will be uploaded to the server.

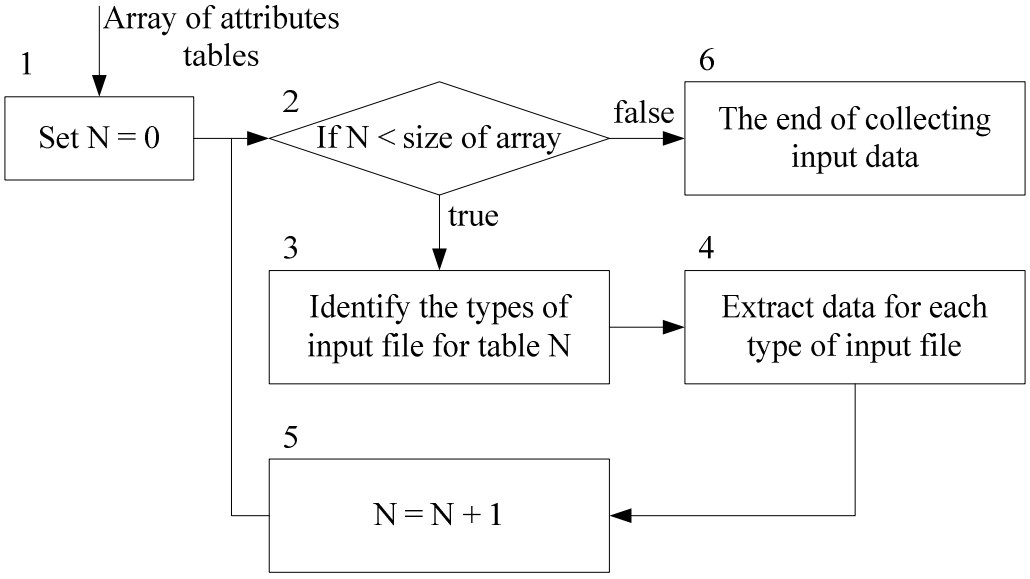


Figure 5.30. The flowchart of collecting input data.

```
1   harvesting.ColummIdentifiers=HarvesterAttributes.csv
2   harvesting.Types=HarvesterTypes.csv
3   harvesting.Table=Harvesting.csv
4   raking.ColummIdentifiers=RakeAttributes.csv
5   raking.Types=RakerTypes.csv
6   raking.Table=Raking.csv
7   transportation.ColummIdentifiers=TransportationAttributes.csv
8   transportation.Types=TransportationTypes.csv
9   transportation.Table=Transportation.csv
10  farmopenstorage.ColummIdentifiers=FarmOpenStorageAttributes.csv
11  farmopenstorage.Types=FarmOpenStorageTypes.csv
12  farmopenstorage.Table=FarmOpenStorage.csv
13  farmcoveredstorage.ColummIdentifiers=FarmCoveredStorageAttributes.csv
14  farmcoveredstorage.Types=FarmCoveredStorageTypes.csv
15  farmcoveredstorage.Table=FarmCoveredStorage.csv
16  centralizedstorage.ColummIdentifiers=CentralizedStorageAttributes.csv
17  centralizedstorage.Types=CentralizedStorageTypes.csv
18  centralizedstorage.Table=CentralizedStorage.csv
19  packing.ColummIdentifiers=PackingAttributes.csv
20  packing.Types=PackingTypes.csv
21  packing.Table=Packing.csv
```

Figure 5.31. Content format of properties file for preparing input files.

5.2.3.2 Uploading Data to Server

The procedure of uploading input data to the server is shown in Figure 5.32. The key steps in the procedure of uploading input data to server are steps 3 and 4 in Figure 5.32. In step 3, the client-side application identifies the correct file name for each set of collected data according to the context in the properties file (Figure 5.31). The text after the equality is the filename where the data should be saved. Step 4 sends an http request with the files to the server. Each request carries one set of collected data to the server at each time. After the server receives the set of data from each request, it will be saved into a CSV file waiting for BioFeed model to read. Given the contents of Figure 5.31, there

are twenty-one HTTP requests created for the upload of files to the server. The model is ready to execute when the all files are uploaded.

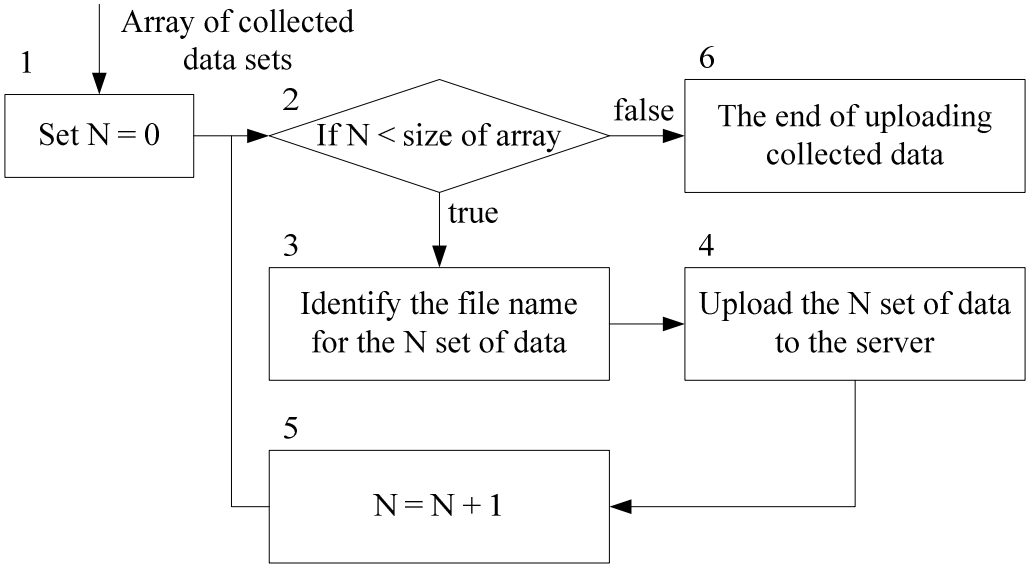


Figure 5.32. The flowchart of collecting input data.

5.2.3.3 Executing Model

When data upload is successful, the client-side application creates another Http request to ask the server-side application to run the model. As steps 2 and 3 shown in Figure 5.33, the client-side application passively waits for the response from server during the execution until the output is ready to be retrieved. When the client receives the output, the client application uses it to create a Result object for further analyses.

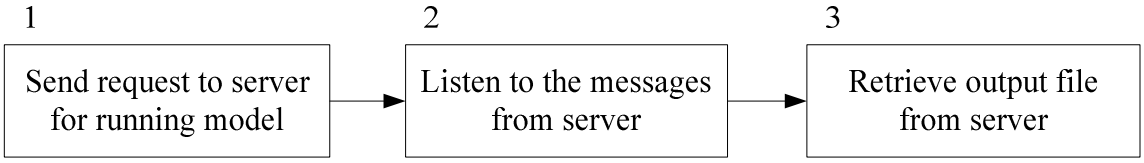


Figure 5.33. The flowchart of execute mode.

Figure 5.34 shows the sequence diagram for executing the model. The segment between the arrows 1 and 2 conducts step 1 in Figure 5.33. The segment between the arrows 3 and 4 is for step 2. While the model is executed on the server, BPSysClient

waits for the message from the server via the calling the method of Request, readObject, along the arrow 3.1. The segment between the arrows 5 and 6 is used to retrieve the output from the server after BPSysClient receives the completion message of the execution. The last segment is to get the data from the Request object and create the Result object.

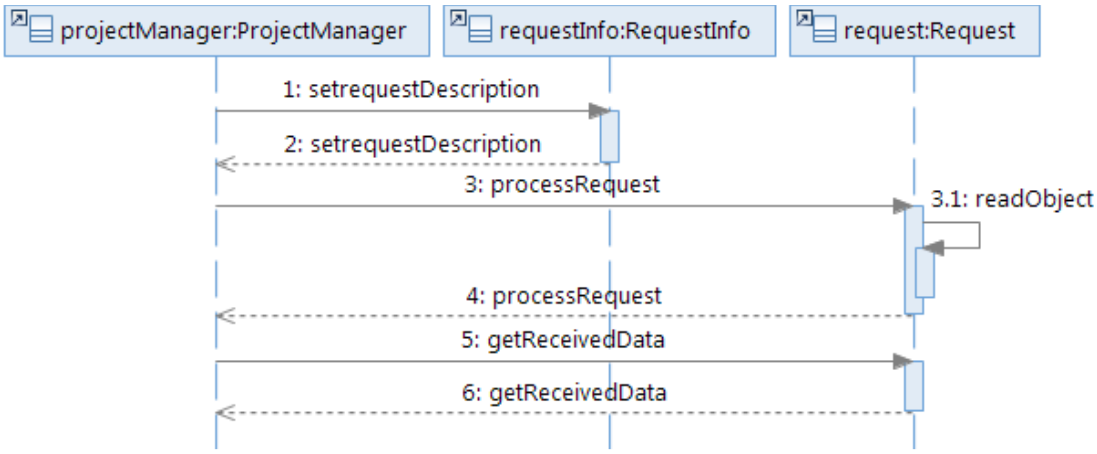


Figure 5.34. Sequence diagram of executing model.

5.2.4 Conducting Batch Execution

While the function of executing model is provided, it is not able to generate enough output for parametric analysis, thus the function of batch execution has been developed. A batch execution requires one more step, to define the parameters of a batch execution. Figure 5.35 shows the flowchart for conducting batch execution of the model. Step 1 defines the batch execution following the steps introduced in Section 5.2.3. Steps 2 – 5 are the loop to execute the model included within the predefined batch execution. Steps 3 – 5 are identical with the steps in Section 5.2.3 for conducting a single model execution. The batch execution is ended when all executions are finished.

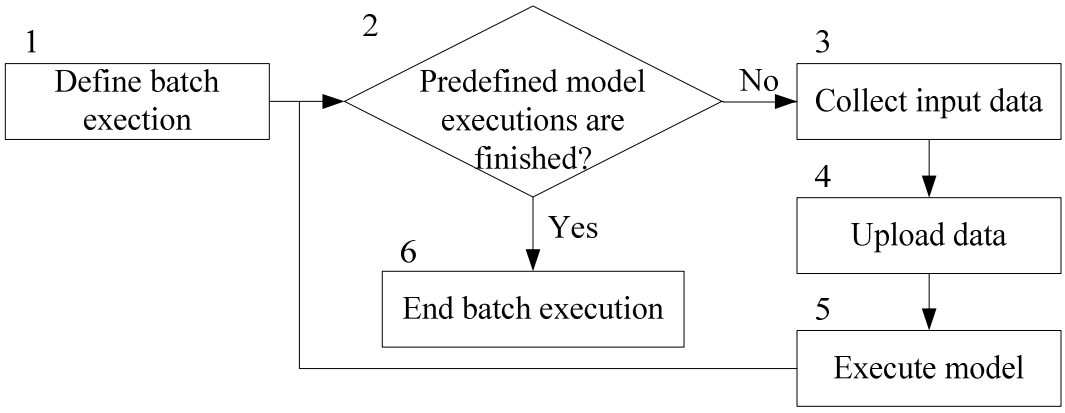


Figure 5.35. The flowchart to conduct batch execution.

5.2.5 Presenting Results

After a raw result is received, BPSys will go through the process shown in Figure 5.36. Steps 1 and 2 create the ResultOutput and ResultInput objects. These two objects are then used to create a Result object in step 3. At this stage, the demonstrated charts and tables in Figure 4.34 – Figure 4.39 are not created. Steps 4 and 5 generate these figures and they are assigned to the Result object. The following two subsections discuss steps 1 – 4 in more detail.

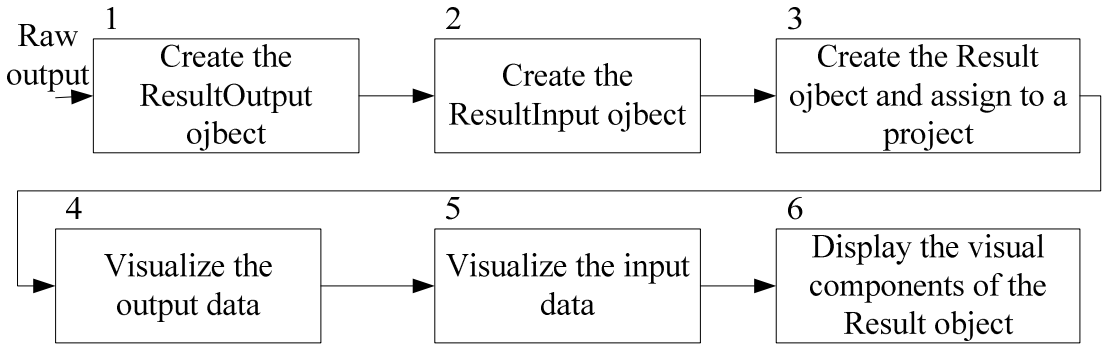


Figure 5.36. The flowchart of visualizing results.

Figure 5.37 shows the sequence diagram of visualizing results. The arrow 1 initiates step 1 in Figure 5.36 and the arrow 2 returns the created ResultOutput object to end step 1. Step 2 is initiated by the arrow 3 and ended by the arrow 4. Once the ResultOutput and ResultInput objects are created, they are used to create a Result object

along the directions of the arrows 5 and 6. Steps 4 – 6 are completed by calling the functions depicted in arrows 7, 7.1, and 7.2.

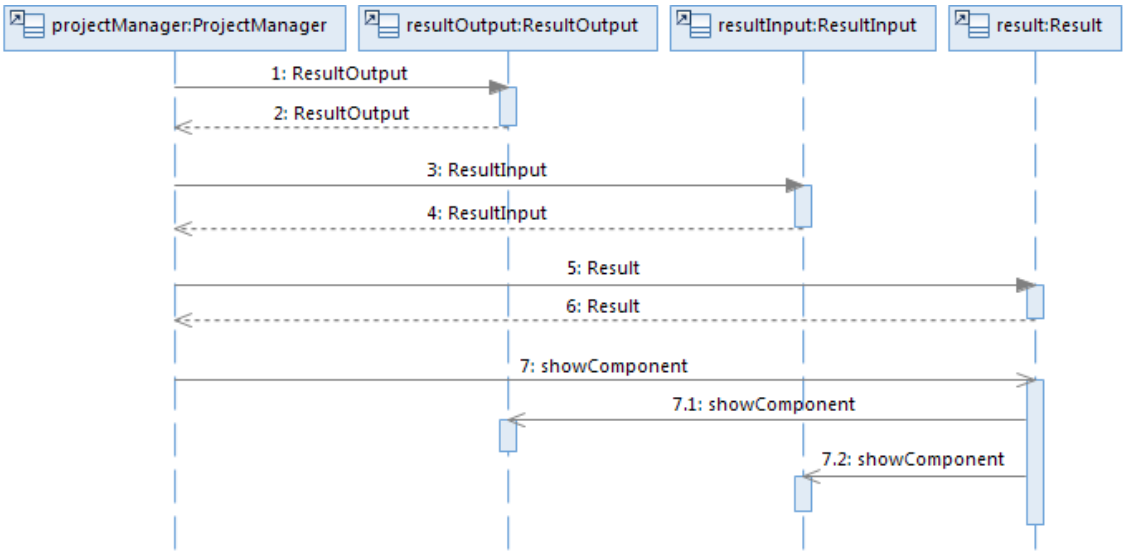


Figure 5.37. Sequence diagram of visualizing results.

The visualization components in a ResultOutput object include tables which display DependentVariable objects, tables which display ProfileVariable objects, pie charts which visualize DependentVariable objects, and the profile charts which visualize ProfileVariable objects.

5.2.5.1 Creating ResultOutput object

This section will show how the raw output of the model is processed in order to create a ResultOutput object. The most important step here is to identify the dependent variables and extract the values. Two types of dependent variable classes designed for the ResultOutput class where introduced in Section 5.1.4: the ProfileVariable and DependentVariable class. The program will identify which class the extracted variable belongs to. The extraction operation follows the steps outlined in Figure 5.38. This is completed by a DependentVariablesExtractor object. First, create the DependentVariableExtracor object based on variable keys obtained from the properties

file displayed in Figure 5.39. The text after the equal sign in the first line in Figure 5.39 lists keys utilized for locating and extracting data necessary for creating DependentVariable objects; similarly the second line lists keys for extracting data necessary for creating ProfileVariable objects. The categories of the DependentVariable are separated by the semicolons. These keys are stored in an array format within the DependentVariableExtractor object. Next, each line in the output data matching these keys is assigned a specific number as an identifier for the type of variable. According to these added identifiers, the variables can be segregated from each other via steps 3 and 4.

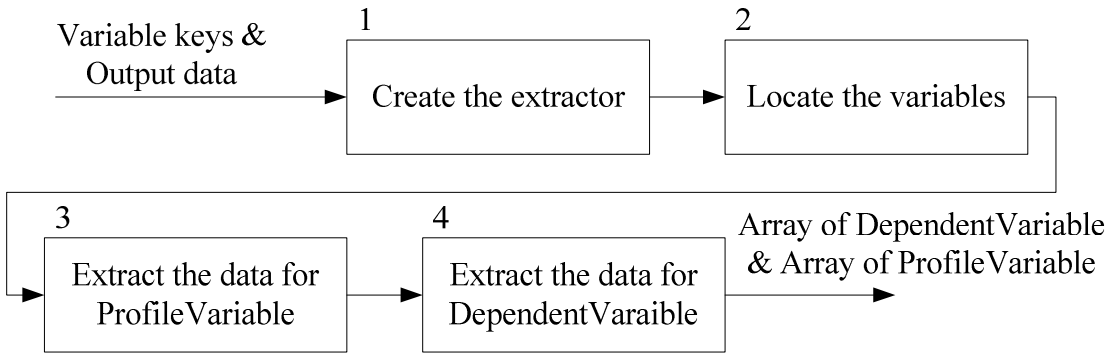


Figure 5.38. The flowchart to extract variables.

```
1    ResultOutput.PieChartVariableKeys=greenhouse gas emission;energy
     consumption;Fleet requirement;operating cost;fixed
     cost;capacity;cost;area
2    ResultOutput.ProfileVariableKeys=schedule
```

Figure 5.39. The content of the properties file for creating the ResultOutput.

Figure 5.40 shows the sequence diagram for creating a ResultOutput object. The arrows 1 and 2 complete step 1 in Figure 5.38. The arrow 3 calls the DependentVariableExtractor function of setResultOutput to set the raw output for locating the variables in step 2 in Figure 5.38. Steps 3 and 4 are accomplished via the arrows of 5-7. The extracted variables are used to create a ResultOutput object.

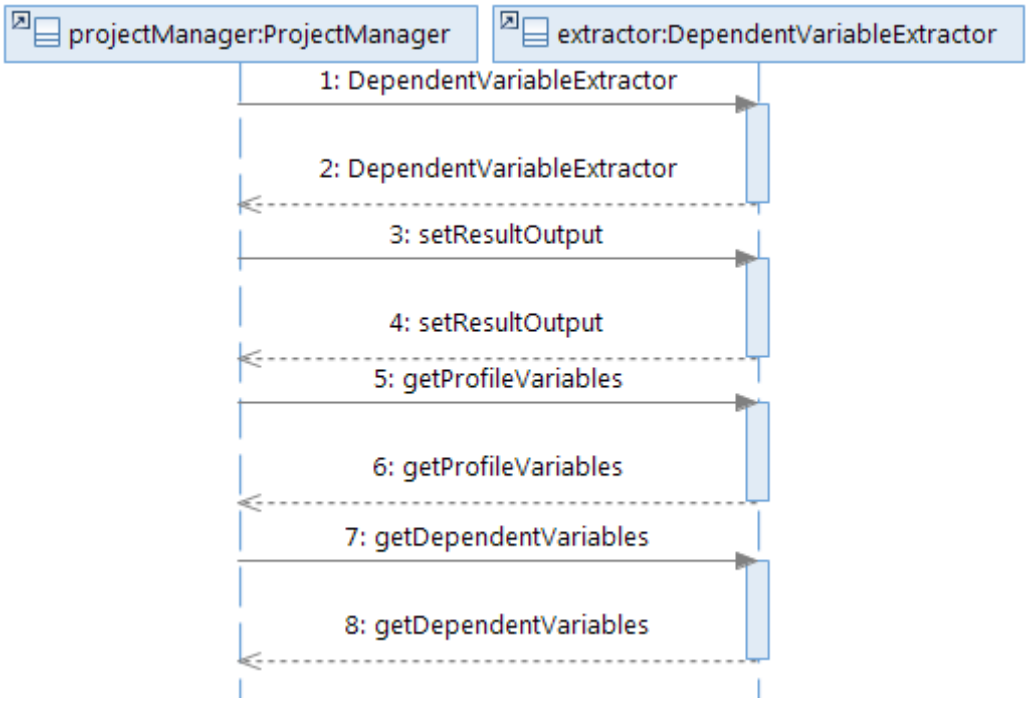


Figure 5.40. Sequence diagram of creating ResultOutput object.

The original contents and formats in the output file from BioFeed are provided in Appendix C. The arrays of the DependentVariable objects for the output in Appendix C are listed in Table 5.4. There are nine sets of dependent variables created from the BioFeed output file. For the output materials generated by different models, the keys for extracting the dependent variables can be customized. There is also a ProfileVariable object for storing the information about daily truck usage throughout the year. The two arrays of DependentVariables and ProfileVariable objects are used to create a ResultOutput object.

Table 5.4. The extracted DependentVariables objects in an output file.

Cost ($/Mg)	
Total delivered	43.25
Harvesting	9.16
Raking	3.05
Total packing	4.65
Storage	8.8
Total transportation	7.57
Infield transportation	6.79
Biomass handling	3.24

(a)

Greenhouse gas emission (kg of CO2 eq/Mg)	
Harvesting	7.96
Raking	1.64
Total packing	7.16
Total transportation	3

(b)

Energy consumption (MJ/Mg)	
Harvesting	109.74
Raking	22.56
Total packing	98.67
Total transportation	41.41

(c)

Capacity (Mg/Day)	
Biorefinery	1249.5

(d)

Central storage facility area (square meters)	
Location-3	0

(e)

Transportation fleet requirement (number of trucks)	
Flat_Bed_Trailer	34

(f)

Operating cost ($/Mg)	
Packing	4.44
Transportation	5.94

(g)

Fixed cost ($/Mg)	
Packing	0.2
Transportation	1.62

(h)

Area (square meters)	
Total on-farm storage	416889.08
Total silage pit	114396.92

(i)

5.2.5.2 Creating ResultInput object

The procedure of creating a ResultInput object is very similar to creating ResultOutput object. The program checks and extracts the selected rows in each attribute table. An IndependentVariable object is created to store the name of attribute table in lower case, the column names of the table, and the values within the selected rows. These three attributes in the IndependentVariable object are for storing the selected input data.

Taking the attribute table for Transportation task displayed in Figure 5.41 as an example, the contents of these three attributes, if the first two pieces of equipment are selected, are displayed in Table 5.5.

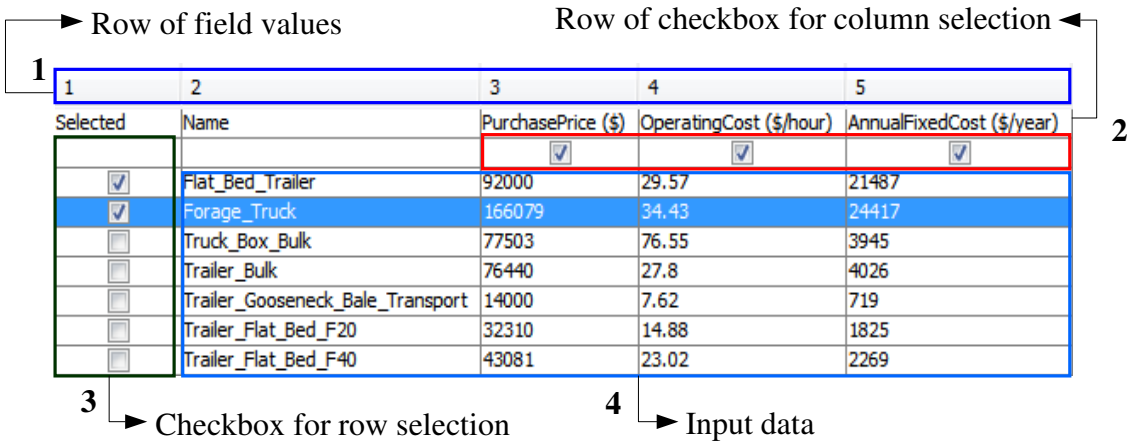


Figure 5.41. The fraction of attribute table for transportation task.

Table 5.5. The attributes values of an IndependentVariable object for transportation task.

Name	transportation
fieldname	Name;PurchasePrice;OperatingCost;AnnualFixedCost
values	[Flat_Bed_Trailer;92000;29.57;21487], [Forage_Truck;166079;34.43;24417]

5.2.5.3 Visualizing ResultOutput

Figure 5.42 shows the process for visualizing the ResultOutput object. The contents of properties file (Figure 5.43) will guide the visualization of the ResultOutput object by managing the flow through steps 1, 3, 5, and 7, which checks what type of visual component and the quantity of each type are to be included in this report. There are four types of visual components currently: Table, ScheduleTable, PieChart, and Profile. The text “ResultOutput.TableNum=1” in Figure 5.43 ensures that there is a table showing the DependentVariable objects to be created; similarly, the text “ResultOutput.ScheduleTableNum=1” ensures that there is a table to display

ProfileVariable objects; the text “ResultOutput.PieChartNum=3” ensures that there are three pie charts to visualize the fractions of a performance indicator described by a DependentVariable object (please refer to Section 5.1.4); and the text “ResultOutput.ProfileNum=1” ensures that a profile chart for visualizing the trend of a ProfileVariable object is included. Steps 1, 3, 5, and 7 are to check the number visualization types and steps 2, 4, 6, and 8 are to create them. At this moment, the quantity of visual components are known but the data sources for creating them have not been identified. To create the table for displaying the dependent variables, the information on the title of the table, and the data to be shown is needed and this is read from the properties file. The following paragraphs describe how the charts and table are created in steps 2, 4, 6, and 8.

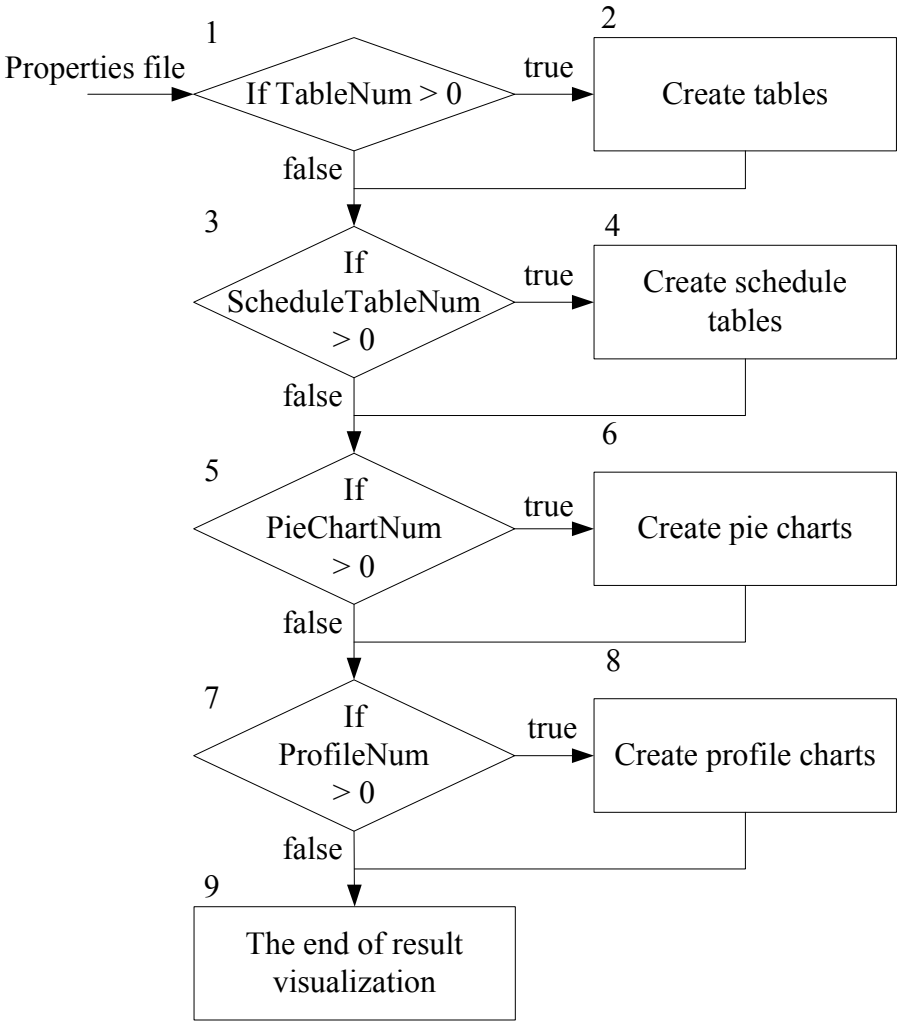


Figure 5.42. The flowchart of visualizing ResultOutput.

```
1    ResultOutput.TableNum=1
2    ResultOutput.ScheduleTableNum=1
3    ResultOutput.PieChartNum=3
4    ResultOutput.ProfileNum=1
```

Figure 5.43. The content of the properties file for visualizing ResultOutput.

Step 2 creates the table for displaying DependentVariable objects (refer to the figure). The content of the properties file required for step 2 is shown in Figure 5.44. The text “Table1” in the contents provides the descriptions used to create the table. If the first line in Figure 5.43 is “ResultOutput.TableNum=2”, there should be another pieces of description mentioning in Figure 5.44 about “Table2”.

The first line in Figure 5.44 represents that the title for the table, “Output” in this case. The text after the equal sign in the second line shows the dependent variables with the unit: cost ($/Mg), greenhouse gas emission (kg of CO_2 eq/Mg), energy consumption (MJ/Mg), central storage facility area (square meters), transportation fleet requirement (number of trucks), operating cost ($/Mg), fixed cost ($/Mg), and area (square meters). They can be separated by comas or semicolons. A semicolon suggests that there is an area in the table to display the independent variables enclosed by semicolons. If there are multiple dependent variables enclosed within a semicolon and separated by comas, they are organized in parallel. Blocks are partitioned by empty lines. The first block is generated to display the dependent variables, cost ($/Mg), greenhouse gas emission (kg of CO2 eq/Mg), and energy consumption (MJ/Mg), is shown in Table 5.6. Table 5.7 shows the remaining parts to display the dependent variables 4 to 9 mentioned above.

```
1  ResultOutput.Table1=Output
2  ResultOutput.Table1.Data=cost ($/Mg),greenhouse gas emission (kg of
   CO2 eq/Mg),energy consumption (MJ/Mg);Central storage facility area
   (square meters);Transportation Fleet requirement (number of
   trucks);operating cost ($/Mg);fixed cost ($/Mg);area (square meters)
```

Figure 5.44. The content of the properties file for displaying DependentVariable objects.

Table 5.6. The part of the "Output" table to display cost, greenhouse gas emission, and energy consumption.

	Cost ($/Mg)	Greenhouse gas emission (kg of CO2 eq/Mg)	Energy consumption (MJ/Mg)
Total delivered	43.25		
Harvesting	9.16	7.96	109.74
Raking	3.05	1.64	22.56
Total packing	4.65	7.16	98.67
Storage	8.8		
Total transportation	7.57	3	41.41
Infield transportation	6.79		
Biomass handling	3.24		

Table 5.7. The parts of the "Output" table to display other dependent variables.

	Central storage facility area (square meters)
Location-3	0
	Transportation Fleet requirement (number of trucks)
Flat_Bed_Trailer	34
	operating cost ($/Mg)
Packing	4.44
Transportation	5.94
	fixed cost ($/Mg)
Packing	0.2
Transportation	1.62
	capacity
Biorefinery	1249.5
	area (square meters)
Total on-farm storage	416889.08
Total silage pit	114396.92

Step 4 in Figure 5.42 is the method that creates tables to display ProfileVariable

objects. The information needed to create this kind of table also includes the title and the name of the ProfileVariable object to be displayed. The difference between the table describing ProfileVarialbes, as opposed to DepedentVariables, is that the ProfileVariable table only displays one ProfileVariable object. Figure 5.44 illustrates the necessary information in the properties file. The next two paragraphs explain the methods for drawing figures.

1	ResultOutput.ScheduleTable1=Transportation Fleet Schedule
2	ResultOutput.ScheduleTable1.Data=Transportation fleet schedule

Figure 5.45. The content of the properties file for creating tables displaying.

Step 6 in Figure 5.42 draws pie charts, which illustrate the fractions occupied by different categories within a DependentVariable. In Figure 5.43, the third line indicates that three pie charts should be created and displayed. The information for creating these three different pie charts is from the properties file (Appendix A), including the title and the data source for each pie chart. The DependentVariable object drawn in the first pie chart is assigned by the information in line 2 (Figure 5.46) after the equal sign. The order of the text assigns the drawn fields in the DependentVariable object is: first, specifying the name of the DependentVariable object and second, specifying the name of the drawn fields which follow the colon (:). The colon is used when not all fields in this DependentVariable object are counted and only the field names of the data after the colon are counted. As a result, the values of the field names within the DependentVariable object, “cost ($/Mg)”, which are counted in the PieChart1 are (1) Harvesting, (2) Raking, (3) Total packing, (4) Storage, (5) Total transportation, (6) Infield transportation, and (7) Biomass handling. The other two charts count all field values where the context structure after the equal sign has the name of the DependentVariable object. This is because there is

a need to explicitly write values included in the pie chart of the dependent variable. For example, for the pie chart (Figure 4.36), for cost the label is $/Mg where for "Total delivered" the label is the value 43.25 (Table 5.6). If all the fields need to be included, the syntax is like line 4 and 6 in Figure 5.46, which quotes the full name of the DependentVariable object.

```
1  ResultOutput.PieChart1=Cost ($/Mg)
2  ResultOutput.PieChart1.Data=cost ($/Mg):Harvesting,Raking,Total
   packing,Storage,Total transportation,Infield transportation,Biomass
   handling;
3  ResultOutput.PieChart2=Energy consumption (MJ/Mg)
4  ResultOutput.PieChart2.Data=energy consumption (MJ/Mg);
5  ResultOutput.PieChart3=Greenhouse gas emission (kg of CO2 eq/Mg)
6  ResultOutput.PieChart3.Data=greenhouse gas emission (kg of CO2
   eq/Mg);
```

Figure 5.46. The content of the properties file for creating pie charts.

Step 8 creates a chart for visualization of the trends of a ProfileVariable object. The information needed for creating profile charts is in Figure 5.47. This type of figure requires two additional attributes, XLabel and YLabel, for labeling the horizontal and vertical axes.

```
1  ResultOutput.Profile1=Transportation Fleet Schedule
2  ResultOutput.Profile1.XLabel=Day
3  ResultOutput.Profile1.YLabel=Number of Trucks
4  ResultOutput.Profile1.Data=Transportation fleet
   schedule
```

Figure 5.47. The content of the properties file for creating profile charts.

Figure 5.48 shows the sequence diagram of visualizing ResultOutput objects. This sequence diagram provides a more detailed explanation of the behaviors after the ResultOutput method showComponent is called. The visualization of a ResultOutput object is created by a ResultOutputReportCreator object by calling one of its functions,

CreateReport. The segment between arrow 1.1.1 and 1.1.2 conducts Steps 1 and 2 in Figure 5.42; the segment between the arrows 1.1.3 and 1.1.4 conducts Steps 3 and 4. The segment between the arrows 1.1.5 and 1.1.6 conducts steps 5 and 6 to generate the pie charts; the segment between the arrows 1.1.7 and 1.1.8 to generate the profile charts.

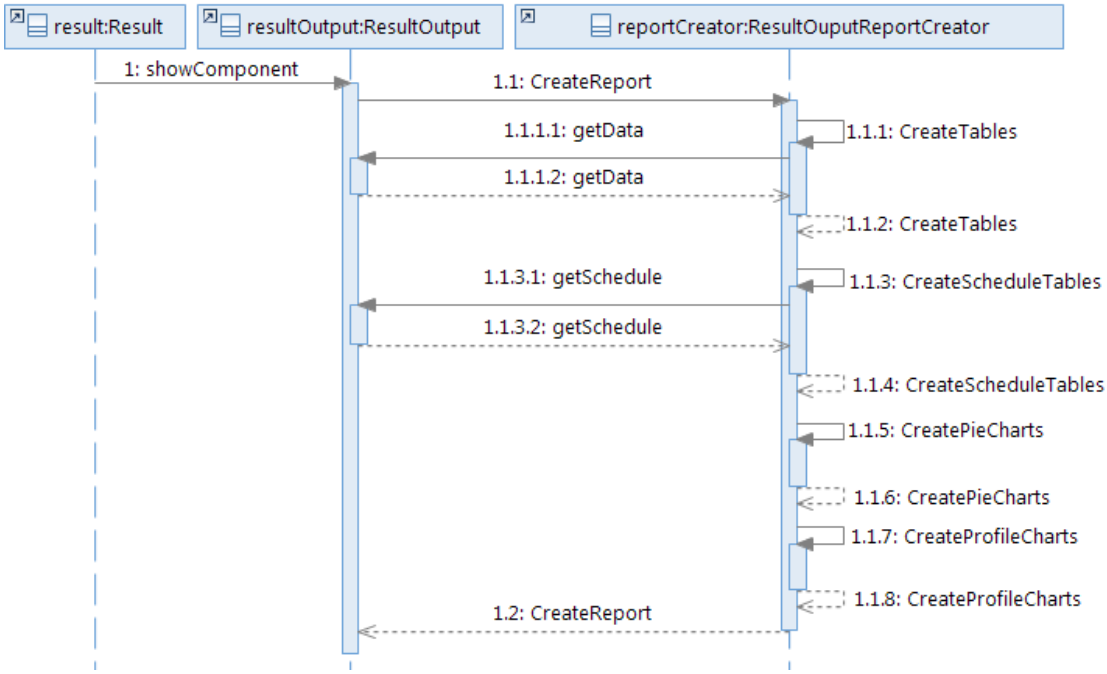


Figure 5.48. Sequence diagram of visualizing ResultOutput object.

In BPSys, the JFreeChart open-source library is utilized to create these charts. This library has a comprehensive set of drawing tools for generating various types of charts. BPSys is responsible for providing the data sets required by JFreeChart. The structure of the data set and the way to initialize the figure is follows the guidelines from the JFreeChart manual (Gilbert, 2009). After the steps from 1 to 9, the visualized components are created based on the default contents in the properties file (Figure 4.34 – Figure 4.39) so that every output generated by the same model has the same components. If there are other models utilized later, the properties file can be specially edited for the construction of tables, pie charts, and profile charts. The next section will discuss the procedure of visualizing the input data.

5.2.5.4 Visualizing ResultInput

The procedure for visualizing input data is shown in Figure 5.48. The first step is to check the array size of IndependentVariable objects. If it there is any input data, an empty table is created. Steps 5 to 7 add new rows for the three attributes of N IndependentVariable object in the array. The number of the added rows in step 7 is equal to the array size of the attribute "value" in the Nth IndependentVariable object.

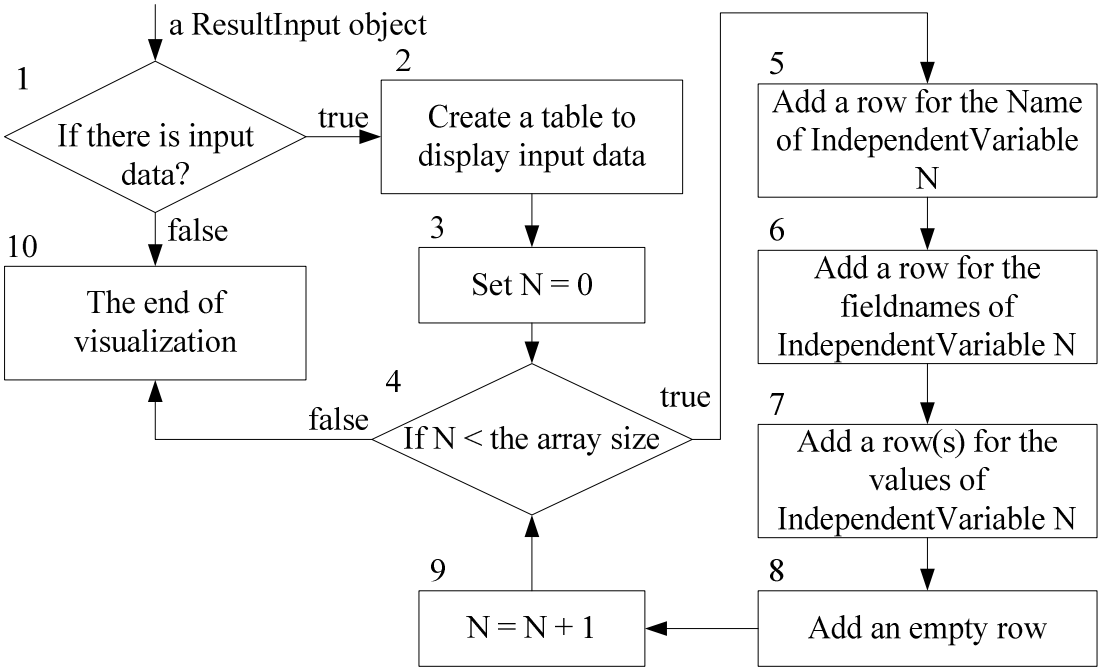


Figure 5.49. The flowchart to visualize a ResultInput object.

Through the previous four subsections (5.2.5.1 - 5.2.5.4), steps 2 – 5 in Figure 5.36 related to visualizing a result are introduced. The remaining step is to display the created visualized components in the user interface as Figure 4.34 – Figure 4.39. Figure 5.50 shows the sequence diagram of visualizing ResultInput object. The segment between the arrows 1 and 1.1 perform step 1 in Figure 5.49. The segment between the arrows 1.1 and 1.1.1 perform steps 2 to collect input data. The segment between 1.1.1 and 1.1.2 is responsible for steps 3 – 9.

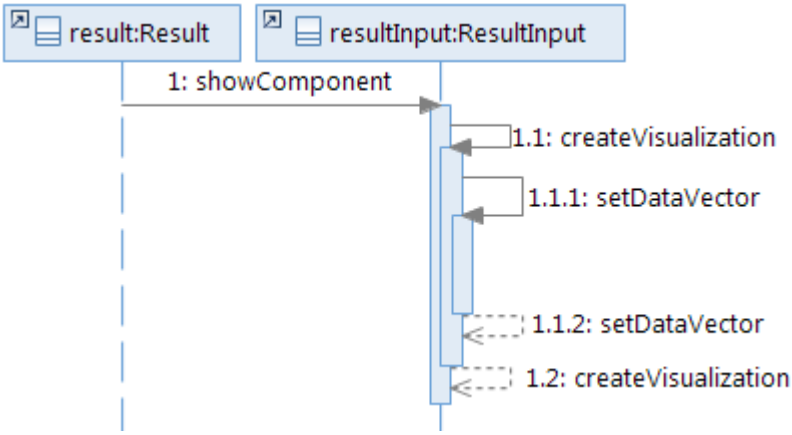


Figure 5.50. Sequence diagram of visualizing ResultInput object.

5.2.6 Comparing Results

The function of parametric analysis has been introduced in Section 4.4.9. The execution of this function via the graphical user interface has nine steps. The first three are to create a DataGroup object; the remaining steps create ParametricAnalysis object. The following two subsections focus on the procedures executed behind the user interface to respond to this user request.

5.2.6.1 Creating a DataGroup Object

The procedure for creating a DataGroup object is in Figure 5.51 and is initiated when the user presses the “Edit Grouped Data” menu item (Figure 4.49) and finishes the configuration (Figure 4.50). Step 1 is to find out which independent variable is changing throughout these results. Then the DataGroup object is created by step 2. After declaring the new object of the DataGroup, it is assigned to the Project object for further parametric analyses and displayed as a node (Figure 4.51) in step 3. When there is at least one DataGroup object in the project, users are able to do the parametric analysis.

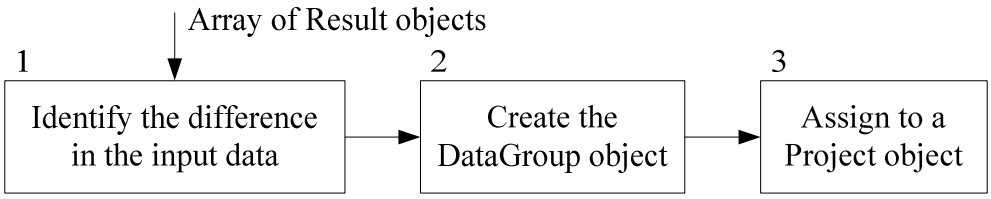


Figure 5.51. The flowchart to create a DataGroup object.

Figure 5.52 shows the sequence diagram of creating a DataGroup object. The array of Result objects is passed with the calling of the method, EditGroupedData. The segment between the arrows 1.1.1 and 1.1.2 conducts step 1 in Figure 5.51. Once this step is finished, the DataGroup object is created after the dashed arrow 1.1.2. The segment between the arrows 1.3 and 1.4 is to assign the created DataGroup object to the Project object as mentioned in step 3.

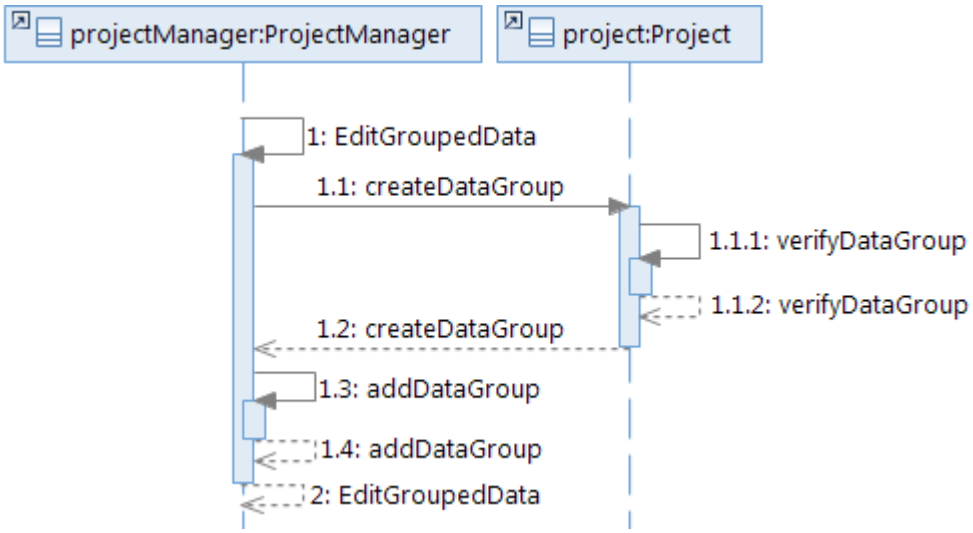


Figure 5.52. Sequence diagram of creating DataGroup object.

5.2.6.2 Comparing Model Results

The procedure of comparing results is shown in Figure 5.53. It is started when the user finishes the configurations mentioned in Section 4.4.9. The procedure will identify the configuration edited by the user and create the dataset for the plot.

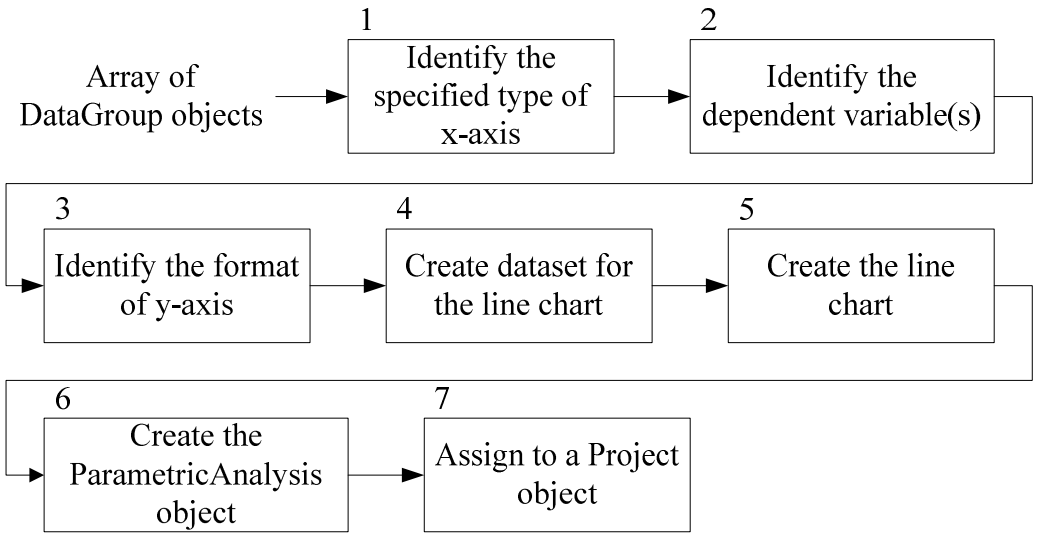


Figure 5.53. The flowchart to create a ParametricAnalysis object.

Step 1 identifies the method of creating the horizontal axis values. The possible formats include: the exact values of the independent variable, the difference in value from the first number, and the change ratios from the first number. Step 2 is to identify the selected dependent variables. If n dependent variables are chosen, there will be n legends in the created line chart. The values in each legend represent the values of each chosen dependent variable in the results created in the selected DataGroup object. Step 3 is to identify the method for creating the y-axis values, a procedure analogous to the creation of the vertical axis values. Step 4 will create the dataset that will generate the line chart in step 5. Step 6 creates the object of the ParametricAnalysis class by passing the chart and dataset to the constructor method and step 7 assigns the ParametricAnalysis object to the Project object.

Figure 5.54 shows the sequence diagram of comparing model results. The segment between the arrows 1.1 and 1.2 accomplishes steps 1 – 4 in Figure 5.53. Then the created ParametricAnalysis object is returned back to the Project object.

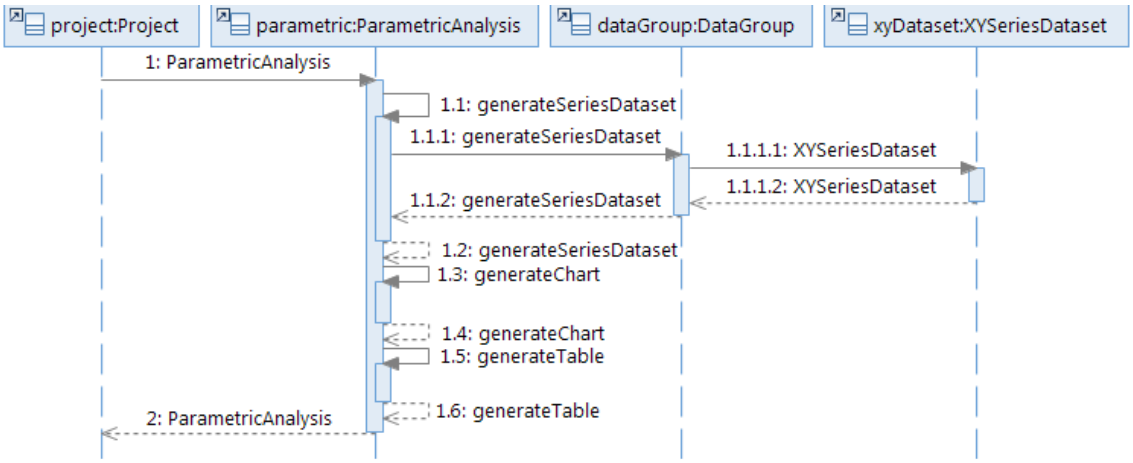


Figure 5.54. Sequence diagram of comparing model result.

Chapter 6. DSS Application Case Study

This chapter presents three applications of decision-support for switchgrass production on: optimizing equipment selection, comparing the impact of increased attribute values of a selected machine, and comparing the impacts of increased attribute values of various machines. The data used to conduct the analysis are generated by BioFeed (Shastri et al., 2009; 2010) with the scenario depicted in Figure 6.1. To prepare the results for analyzing a user-specific system, the user should be able to manipulate the equipment selections, attribute values, the user should be able to make use of the batch execution function, and the user should be able to select result sets and draw charts in BPSys. These procedures were discussed in Chapter 4.

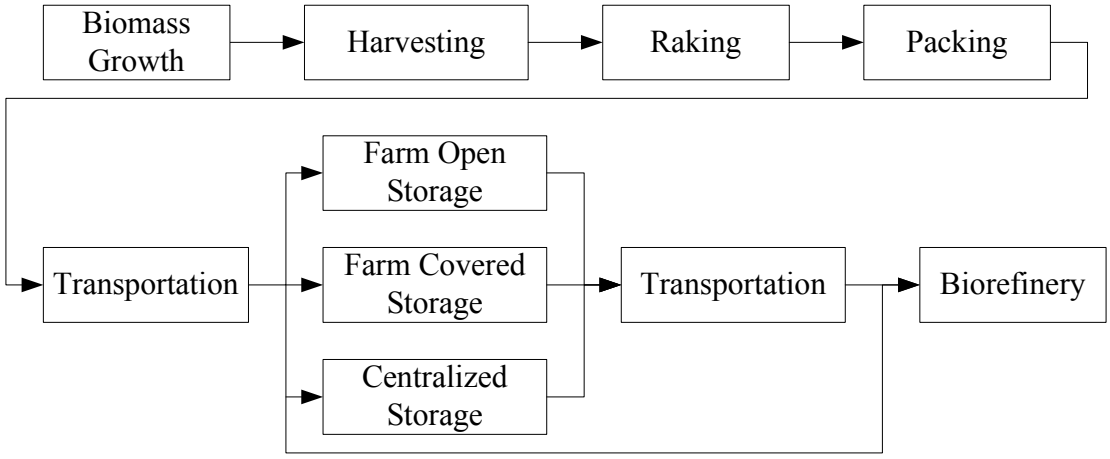


Figure 6.1. The scenario modeled by BioFeed (excluding Biorefinery).

6.1 Optimized Equipment Selection

This section demonstrates choosing equipment for the three storage options (Farm Open Storage, Farm Covered Storage, and Centralized Storage) and the minimization of total cost. In each case, one storage method is selected and the optimization is applied to each farm modeled in BioFeed. Table 6.1 lists the equipment available for each storage

task. The equipment used for the other tasks are listed in Table 6.2. There are two, four, and two choices for the three storage tasks. Consequently, there are sixteen different combinations for the storage equipment selections (Table 6.3). For the other four tasks, all available equipment are utilized as options to select from and no modification is made on the default equipment attributes. By comparing the sixteen results, the most optimized combination of storage tasks can be identified.

Table 6.1. The candidate equipment of the storage tasks.

Storage task		
Farm open storage	Farm covered storage	Centralized storage
Gravel_Pad Paved_Pad	Shed_NoWalls Shed_3Walls Enclosed_NoVentilation Enclosed_Ventilation	Enclosed_NoVentilation Enclosed_Ventilation

Table 6.2. The selected equipment for the tasks other than the storage.

Task	Equipment
Harvesting	Forage_Harvester_SP Mower_Conditioner
Raking	Type1
Packing	Round_Baler Square_Baler Square_Baler_PullType_Contractor Square_Baler_SP_Contractor
Transportation	Flat_Bed_Trailer Forage_Truck Truck_Box_Bulk Trailer_Bulk Trailer_Gooseneck Trailer_Flat_Bed_F20

Table 6.3. The sixteen combinations of the equipment for the storage tasks.

Comb'n	Farm open storage	Farm covered storage	Centralized storage
1	Gravel_Pad	Enclosed_NoVentilation	Enclosed_NoVentilation
2	Gravel_Pad	Enclosed_NoVentilation	Enclosed_Ventilation
3	Paved_Pad	Enclosed_NoVentilation	Enclosed_NoVentilation
4	Paved_Pad	Enclosed_NoVentilation	Enclosed_Ventilation
5	Gravel_Pad	Enclosed_Ventilation	Enclosed_NoVentilation
6	Gravel_Pad	Enclosed_Ventilation	Enclosed_Ventilation
7	Paved_Pad	Enclosed_Ventilation	Enclosed_NoVentilation
8	Paved_Pad	Enclosed_Ventilation	Enclosed_Ventilation
9	Gravel_Pad	Shed_3Walls	Enclosed_NoVentilation
10	Gravel_Pad	Shed_3Walls	Enclosed_Ventilation
11	Paved_Pad	Shed_3Walls	Enclosed_NoVentilation
12	Paved_Pad	Shed_3Walls	Enclosed_Ventilation
13	Gravel_Pad	Shed_NoWalls	Enclosed_NoVentilation
14	Gravel_Pad	Shed_NoWalls	Enclosed_Ventilation
15	Paved_Pad	Shed_NoWalls	Enclosed_NoVentilation
16	Paved_Pad	Shed_NoWalls	Enclosed_Ventilation

The costs of these sixteen cases range from \$37.33 to 41 Mg^{-1} in Figure 6.2. In Figure 6.2, there are four bars of different colors in each category representing the cost of storage, infield transportation, total transportation, and biomass handling, and a blue line in the figure displaying the total cost. It shows that the combinations 13 to 16 have the lowest costs for storage and infield transportation than the combinations 1 to 12. Although the combinations 13 to 16 have the higher costs of total transportation and biomass handling, they still have lower total cost than the combinations 1 to 12 by \$3 Mg^{-1}. In contrast to the total cost, the truck fleet size for the combinations is significantly different. The difference of the truck numbers between the combinations can be as many as 200 trucks (Figure 6.3). Although this difference does not cause a huge impact in the total cost, the management of the truck fleet might be an issue. Based on these results, a production system employing "Shed_NoWalls" in the "Farm covered storage" (combination 13 – 16 of Table 6.2) has lower production cost and a smaller truck fleet.

The combination 16 has the optimal cost, \$37.33 Mg^{-1}.

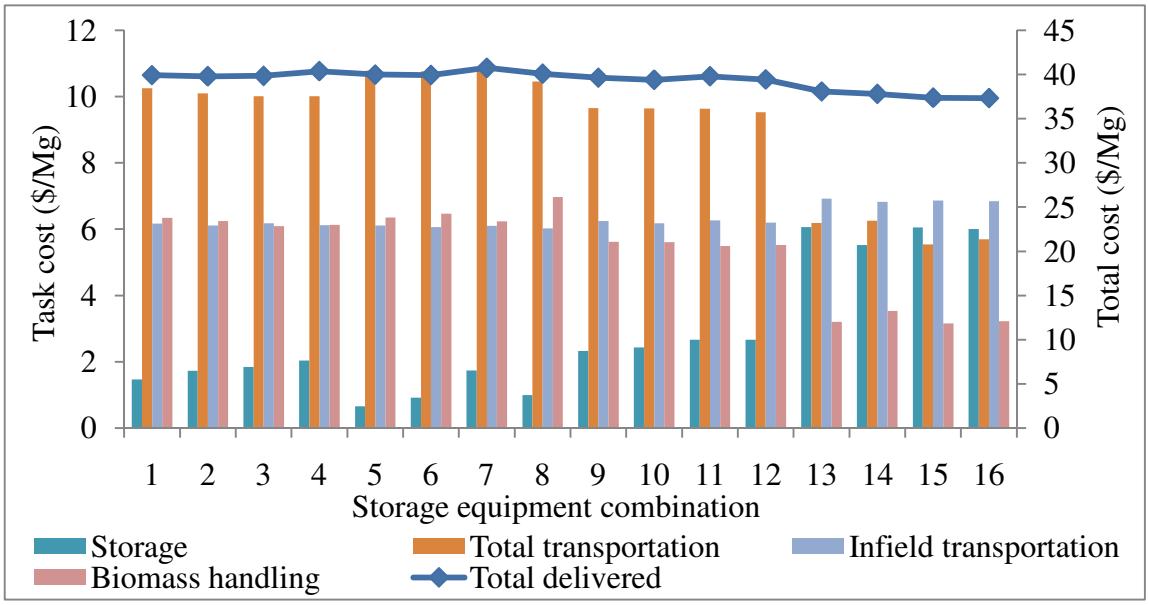


Figure 6.2. Cost for the sixteen storage equipment combinations.

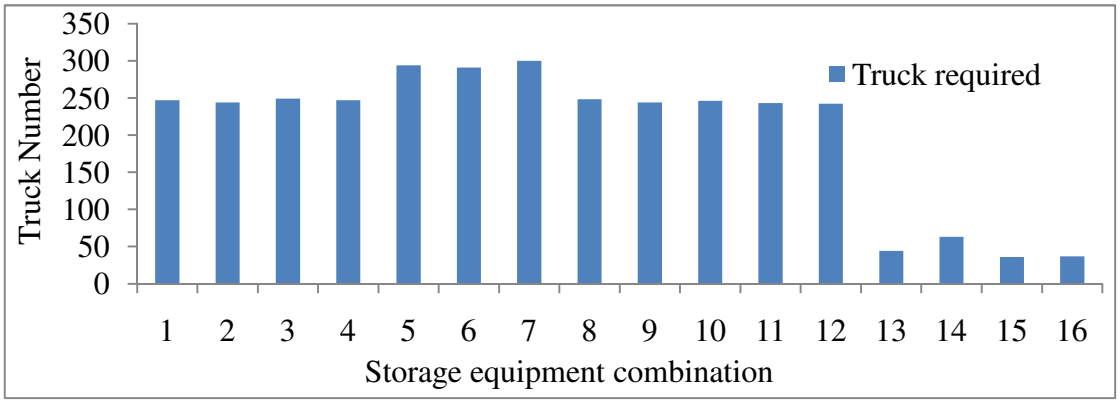


Figure 6.3. Trucks required for the different storage equipment combinations (Storage equipment combinations refer to Table 6.2).

6.2 Impacts of the Packing Density

The purpose of the analysis in this section is to measure the possible benefit from the technology improvement. The influence of the improved packing density provided by the "Square_Baler" is illustrated as an application. A series of the packing densities is generated from the default value, 0.144 (Mg/m^3), to 1 with increments of 0.05 (Mg/m^3)

providing nineteen values. The equipments used in the production system are listed in Table 6.4.

Table 6.4. The equipment used for the packing density.

Task	Equipment	Task	Equipment
Harvesting	Forage_Harvester_SP Mower_Conditioner	Transportation	Flat_Bed_Trailer Forage_Truck Truck_Box_Bulk Trailer_Bulk Trailer_Gooseneck Trailer_Flat_Bed_F20 Trailer_Flat_Bed_F40
Raking	Type1		
Packing	Square_Baler		
Farm open storage	Paved_Pad		
Farm covered storage	Shed_NoWalls		
Centralized storage	Enclosed_Ventilation		

The changing of packing density will affect the costs of three subsystems: storage cost, transportation cost, and biomass handling cost. These three indicators decrease differently while the packing density increases (Figure 6.4). The storage cost is decreased most from \$5 to \$1 Mg^{-1} and the trend shows that it can be decreased further. The decreased costs of the biomass handling and total transportation become stable when the packing density is larger than 0.394 (Mg/m^3). Overall, the total cost is decreased from \$37.58 to 30.11 Mg^{-1-1}. Figure 6.5 shows the ratio of the decreased cost to the initial cost. It can be concluded from the analysis that the increased packing density has a beneficial impact on the costs of storage, biomass handling, and transportation.

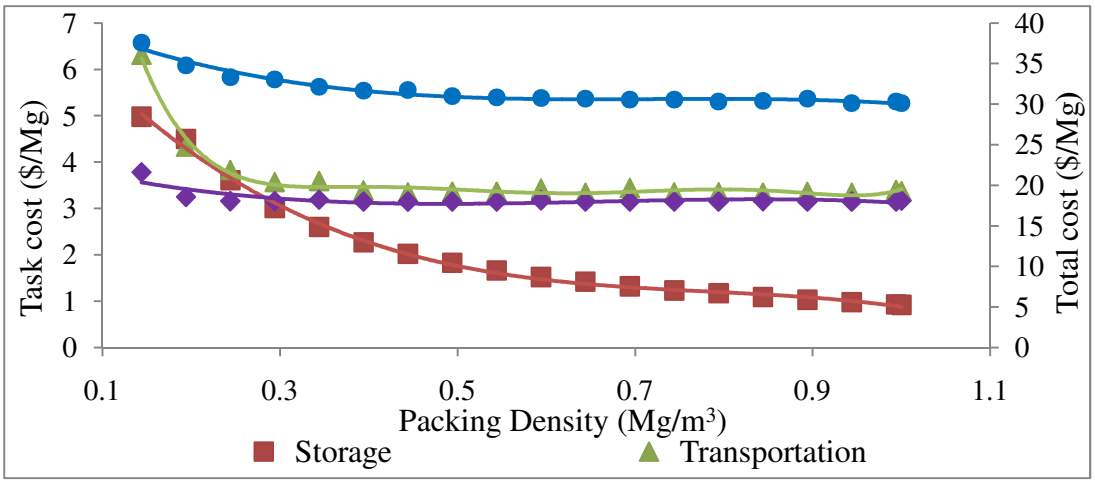


Figure 6.4. Total cost decrement with the increased packing density (Mg/m^3).

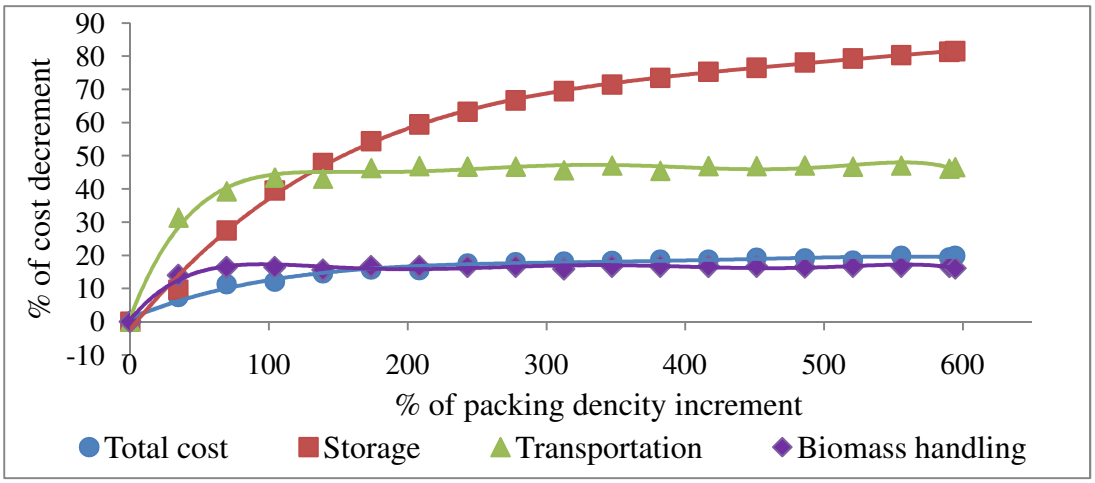


Figure 6.5. The ratio of the decreased percentage of total cost to the percentage of the packing density (Mg/m^3).

6.3 Impacts of Material Throughput

This section illustrates the application of identifying the equipment attribute with heaviest impact on the total cost. The result of this approach can guide the technology development to improve the identified factors with high impact. The equipment configuration for this study is listed in Table 6.5. Three units of equipment, the harvesting machine "Mower_Conditioner", the raking machine "Type1", and the packing machine "Square_Baler", were selected to measure the influence from one of their attributes on the total cost. The attribute "Throughput", with units of Mg/hr, common to all three

machines are selected to conduct this analysis. The throughput of these three machines was modified to generate the results while the other input data remained at the default setting. The values of the three Throughput attributes were changed from their default values to 100 by the increment of 5 Mg/hour. In total, sixty-one results were generated.

Table 6.5. Equipment used for studying the impacts of the attribute “Throughput”.

Task	Equipment
Harvesting	Mower_Conditioner
Raking	Type1
Packing	Square_Baler
Farm open storage	Paved_Pad
Farm covered storage	Shed_NoWalls
Centralized storage	Enclosed_Ventilation
Transportation	Trailer_Flat_Bed_F40

With the separately increasing throughput values of these three machines, Figure 6.6 shows that the improvement in the throughput of the Mower_Conditioner decreases the total cost most. Figure 6.7 shows the comparison between the ratio of increased value in the attribute Throughput to the default value and the ratio of the decreased value in the total cost to the initial value. Although the improvements of the harvesting machine Square_Baler and the raking machine Type1 caused the similar decrease in the total cost, the improvement for Square_Baler was slightly more than Type1. This can be observed more clearly in Figure 6.7 than Figure 6.6.

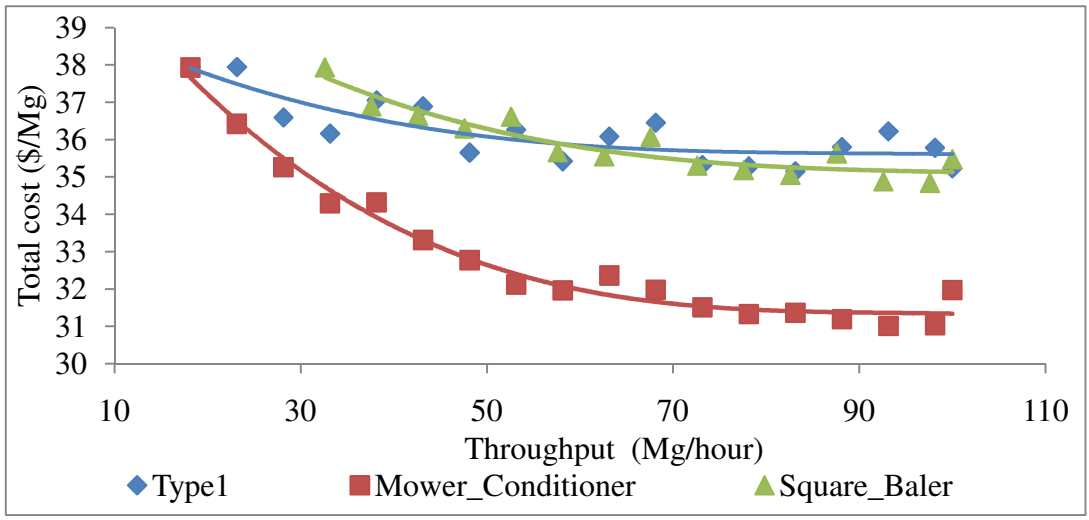


Figure 6.6. The corresponding total cost to the increased throughput (Mg/hour).

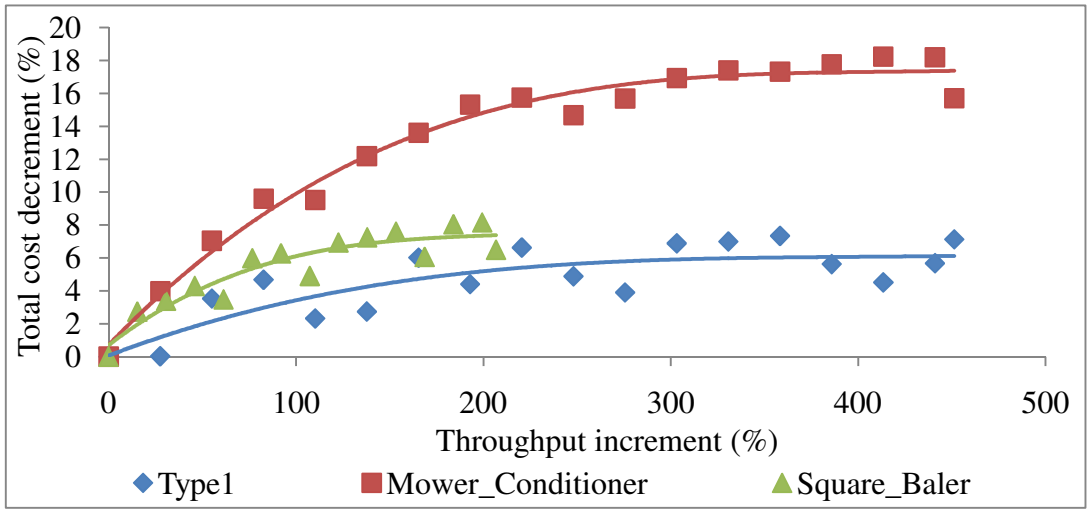


Figure 6.7. The percentage of the decreased total cost corresponding to the percentage of increased throughput (Mg/hour).

Chapter 7. Conclusions & Future Work

7.1 Conclusions

The proposed ConSEnT platform has been implemented as a Java-based web application. The utility of the application for biomass feedstock production analysis has been shown here. The design of the platform has been presented and can be extended for additional functionality in the future.

The web functionality of BPSys has been established with the connection between a client-side application run within a web browser and the server-side application. This enables the user to access the computational resources of ConSEnT, including retrieving model data, executing the model on the server, and receiving model results from the server. With the Internet connection, the research team is able to upload the latest research results into the database and allow user to incorporate this into analysis. These operations are performed in a user-friendly manner within the graphical user interface.

The functions of the user interface, BPSysClient, were formulated by the user story method, a software engineering technique for designing software applications. As a result, the user can not only generate a model, but also conduct parametric analysis to study the impacts of the equipment attributes and equipment combinations on the system performance. Parametric analysis can be useful for decision making at the system level. Conducting parametric analysis in BPSys answers two types of system level questions: selecting which equipment can optimize production cost and identification of technology attributes that bring the most significant improvement in the production cost. BPSys is able to assist the user in understanding the differences between system scenarios and

system designs impact system performance factors. User may also save their results and export them for use in other software tools if the capabilities are not sufficient within BPSys.

For future expansion, UML diagrams of the system design have been prepared. The Java™ classes prepared here can be readily improved and expanded. This includes the support of executing various kinds of modeling applications and programming languages. BPSys maximizes the usability and capability of the computer models and distributes the benefits of model worldwide. The users just have to make sure that Java Virtual Machine is installed on his/her machine. BPSys has been tested on the Macintosh, Window, and Ubuntu operating systems and found that it functioned properly in all cases.

7.2 Future Work

7.2.1 Effectiveness Test

An effectiveness test for the graphical user interface needs to be implemented. This will measure how effective the information and the charts to present the information in BPSys are for assisting the user to make decision. The outcome of the test is aimed at guiding the future improvements of the interface and models.

7.2.2 Decision Making

The outputs of BioFeed are greatly simplified in their presentation here. For example, the model provides farm operational data on a daily basis. This decision support system would be much more valuable if operational management information for could be provided in an effective manner. In order to provide more comprehensive information, the analytical tools for extracting daily basis information require improvements on computational speed and on decision rules targeted key variables, and an improved

mechanism for creating appropriate charts.

7.2.3 Off-line Model Execution

The time for solving the complex problems in BioFeed might be in excess of one hour, parametric analysis can easily exceed half a day. This function would allow users to launch a session on the server to execute the model and wait for the server to inform them when it was completed. The major benefit is that the user would not need to keep the web page opening and Internet connection active while the analysis was running.

7.2.4 Scenario Building

The current version of BioFeed is able to model various crop productions, which include miscanthus and energy cane, and different task combinations; however, crops are fixed (switchgrass) in BPSys and only the equipment selections and the attributes are currently changeable. Scenario building would allow users to design the scenario through BPSys by themselves. This capability can make BPSys more adaptable to different production systems and extend the usability.

References

Alderfasi, A. A. and D. C. Nielsen. 2001. Use of crop water stress index for monitoring water status and scheduling irrigation in wheat. *Agricultural Water Management* 47(1): 69-75.

Apache Software Foundation. 2010. Apache HTTP Server Version 2.0 Documentation. Available at: httpd.apache.org/docs/2.0/. Accessed 1 April 2011.

Ayoub, N., R. Martins, K. Wang, H. Seki and Y. Naka. 2007. Two levels decision system for efficient planning and implementation of bioenergy production. Energy Conversion and Management 48(3): 709-723.

Basham, B., B. Bates and K. Sierra. 2004. Head First Servlets and JSP. Sebastopol, CA : O'Reilly Media, Inc.

Bhargava, H. K., D. J. Power and D. Sun. 2007. Progress in Web-based decision support technologies. Decision Support Systems 43(4): 1083-1095.

Blackmer, T. M. and J. S. Schepers. 1996. Aerial photography to detect nitrogen stress in corn. Journal of Plant Physiology 148(3-4): 440-444.

Blackmer, T. M., J. S. Schepers, G. E. Varvel and G. E. Meyer. 1996. Analysis of aerial photography for nitrogen stress within corn fields. Agronomy Journal 88(5): 729-733.

Blanchard, B.S. and W.J. Fabrycky. 1990. Systems Engineering and Analysis. Prentice-Hall, Englewood Cliffs, NJ.

Browne, J. A. and A. Hunter. 1998. Logistics management and costs of biomass fuel supply. International Journal of Physical Distribution & Logistics Management 28(5): 463.

Cao, X. and W. Yu. 2010. Using content management system Joomla! to build a website for research institute needs. In 2010 International Conference on Management and Service Science, MASS 2010.

Cohn, M. 2004. User Stories Applied: For Agile Software Development. Amsterdam: Addison-Wesley Longman Publishing Co., Inc.

Coombs, K. 2009. Drupal done right. Library Journal 134(19): 30-32.

Cyert, R. M., W. R. Dill and J. G. March. 1958. The Role of Expectations in Business Decision Making. Administrative Science Quarterly 3: 307-340.

Cyert, R. M., H. A. Simon and D. B. Trow. 1956. Observation of a Business Decision. The Journal of Business 29(4, Human Aspects of Management): pp. 237-248.

Das, S., L. Girard, T. Green, L. Weitzman, A. Lewis-Bowen and T. Clark. 2009. Building biomedical web communities using a semantically aware content management system. Briefings in Bioinformatics 10(2): 129-138.

De La Torre Ugarte, D. G. and D. E. Ray. 2000. Biomass and bioenergy applications of the POLYSYS modeling framework. Biomass and Bioenergy 18(4): 291-308.

Domdouzis, K., L. Rodriguez, Y. Shastri, M.-C. Hu, A. C. Hansen and K. Ting. 2009. Systems informatics for biomass feedstock production engineering. ASABE Paper No. 096702. St. Joseph, Mich.: ASABE.

Dresselhaus, M. S. and I. L. Thomas. 2001. Alternative energy technologies. Nature 414(6861): 332.

Fowler, M. 2003. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. Boston, MA.: Addison-Wesley Publishing Co., Inc.

Frombo, F., R. Minciardi, M. Robba, F. Rosso and R. Sacile. 2009. Planning woody biomass logistics for energy production: A strategic decision model. Biomass and Bioenergy 33(3): 372-383.

Giampietro, M. and S. Ulgiati. 1997. Feasibility of large-scale biofuel production. Bioscience 47(9): 587-600.

Gilbert, D. 2009. The JFreeChart Class Library. Available at: http://www.jfree.org/jfreechart/. Accessed 29 April 2011.

Hanks, J. E. and J. L. Beck. 1998. Sensor-controlled hooded sprayer for row crops. Weed Technology 12(2): 308-314.

Hess, R., J., C. T. Wright and K. L. Kenney. 2007. Cellulosic biomass feedstocks and logistics for ethanol production. Biofuels, Bioproducts and Biorefining 1(3): 181-190.

Hoskinson, R. L., R. C. Rope and R. K. Fink. 2007. Using a decision support system to optimize production of agricultural crop residue Biofeedstock. Biomass and Bioenergy 31(4): 186-194.

Huisman, W., P. Venturi and J. Molenaar. 1997. Costs of supply chains of Miscanthus giganteus. Industrial Crops and Products 6(3-4): 353-366.

Irwin, S. H., G. D. Schnitkey, D. L. Good and P. N. Ellinger. 2004. The farmdoc project: This is still your father's extension program. American Journal of Agricultural Economics 86(3): 772-777.

Jones, J. W., G. Hoogenboom, C. H. Porter, K. J. Boote, W. D. Batchelor, L. A. Hunt, P. W. Wilkens, U. Singh, A. J. Gijsman and J. T. Ritchie. 2003. The DSSAT cropping system model. European Journal of Agronomy 18(3-4): 235-265.

Keating, B., G. Carberry, P.S. Hammer, M. Probert, M. Robertson, D. Holzworth, N. Huth, J. Hargreaves, H. Meinke, Z. Hochman, G. McLean, K. Verburg, V. Snow, J. Dimes, M. Silburn, E.Wang, S. Brown, K. Bristow, S. Asseng, S. Chapman, R. McCown, D. Freebairn and C. Smith. 2003. An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy 18: 267–288.

Kleijnen, S. and S. Raju. 2003. An open web services architecture. ACM Queue, 1(1):39–46.

Koh, L. P. and J. Ghazoul. 2008. Biofuels, biodiversity, and people: Understanding the conflicts and finding opportunities. Biological Conservation 141(10): 2450-2460.

Kropff, M. J., J. Bouma and J. W. Jones. 2001. Systems approaches for the design of sustainable agro-ecosystems. Agricultural Systems 70(2-3): 369-393.

Kumar, A. and S. Sokhansanj. 2007. Switchgrass (Panicum vigratum, L.) delivery to a biorefinery using integrated biomass supply analysis and logistics (IBSAL) model. Bioresource Technology 98(5): 1033–1044.

Kumar, A., S. Sokhansanj and P. C. Flynn. 2006. Development of a multicriteria assessment model for ranking biomass feedstock collection and transportation systems. Applied Biochemistry and Biotechnology 129(1-3): 71-87.

Lewandowski, I. and A. Heinz. 2003. Delayed harvest of miscanthus - Influences on biomass quantity and quality and environmental impacts of energy production. European Journal of Agronomy 19(1): 45-63.

McCown, R. L., G. L. Hammer, J. N. G. Hargreaves, D. P. Holzworth and D. M. Freebairn. 1996. APSIM: a novel software system for model development, model testing and simulation in agricultural systems research. Agricultural Systems 50(3): 255-271.

Mitchell, C. P. 1995. New cultural treatments and yield optimisation. Biomass and Bioenergy 9(1-5): 11-34.

Mitchell, C. P. 2000. Development of decision support systems for bioenergy applications. Biomass and Bioenergy 18(4): 265-278.

Mooney, S. D. and P. H. Baenziger. 2008. Extensible open source content management systems and frameworks: A solution for many needs of a bioinformatics group. Briefings in Bioinformatics 9(1): 69-74.

Moran, M. S., T. R. Clarke, Y. Inoue and A. Vidal. 1994. Estimating crop water deficit using the relation between surface-air temperature and spectral vegetation index. Remote Sensing of Environment 49(3): 246-263.

Mordani, R. 2010. JavaTM Servlet Specification 3.0. Available at: http://jcp.org/aboutJava/communityprocess/mrel/jsr315/index.html. Accessed 13 March 2011.

Nilsson, M. and T. Ziemke. 2007. Information fusion: A decision support perspective. In Proceedings of International Conference on Information Fusion. Quebec, Canada.

Perlack, R. D., L. L. Wright, A. F. Turhollow, R. L. Graham, B. J. Stokes and D. C. Erbach. 2005. Biomass as feedstock for bioenergy and bioproducts industry: The technical feasibility of a billion-ton annual supply. Technical report, Oak Ridge National Laboratory.

Pinter Jr., P. J., J. L. Hatfield, J. S. Schepers, E. M. Barnes, M. S. Moran, C. S. T. Daughtry and D. R. Upchurch. 2003. Remote sensing for crop management. Photogrammetric Engineering and Remote Sensing 69(6): 647-664.

Power, D.J. 2002. Decision Support Systems: Concepts and Resources for Managers. Westport, CT: Greenwood/Quorum Books.

Prochnow, A., M. Heiermann, M. Plöchl, T. Amon and P. J. Hobbs. 2009. Bioenergy from permanent grassland - A review: 2. Combustion. Bioresource technology 100(21): 4945-4954.

Rentizelas, A. A., A. J. Tolis and I. P. Tatsiopoulos. 2009. Logistics issues of biomass: The storage problem and the multi-biomass supply chain. Renewable and Sustainable Energy Reviews 13(4): 887-894.

Rodriguez, L. F., S. Kang and K.C. Ting. 2003. Top-level modeling of an ALS system utilizing object-oriented techniques. Advances in Space Research 31(7): 1811-1822.

Sagar, A. D. and S. Kartha. 2007. Bioenergy and Sustainable Development? Annual Review of Environment and Resources 32(1): 131-167.

Shastri, Y., K. Domdouzis, M.C. Hu, A. Hansen, L. Rodriguez and K.C. Ting. 2009. System level analysis of biomass feedstock production for bioenergy sector. ASABE Paper No. 095998. St. Joseph, Mich.: ASABE.

Shastri, Y., A. Hansen, L. Rodriguez and K.C. Ting. 2010. BioFeed optimization model enhancement and its application to Miscanthus production. ASABE Paper No. 1008488. St. Joseph, Mich.: ASABE.

Shim, J. P., M. Warkentin, J. F. Courtney, D. J. Power, R. Sharda and C. Carlsson. 2002. Past, present, and future of decision support technology. Decision Support Systems 33(2): 111-126.

Shklar, L. and R. Rosen. 2009. Web Application Architecture: Principles, Protocols and Practices, 2nd ed., John Wiley & Sons Ltd, Chichester, England.

Simon, H. A. 1959. Theories of Decision-Making in Economics and Behavioral Science. American Economic Review 49(3): 253.

Smeets, E. M. W., I. M. Lewandowski and A. P. C. Faaij. 2009. The economical and environmental performance of miscanthus and switchgrass production and supply chains in a European setting. Renewable and Sustainable Energy Reviews 13(6-7): 1230-1245.

Sokhansanj, S., A. Kumar and A. F. Turhollow. 2006. Development and implementation of integrated biomass supply analysis and logistics model (IBSAL). Biomass and Bioenergy 30(10): 838-847.

Sokhansanj, S., S. Mani, S. Tagore and A. F. Turhollow. 2010. Techno-economic analysis of using corn stover to supply heat and power to a corn ethanol plant – Part 1: Cost of feedstock supply logistics. Biomass and Bioenergy 34(1): 75-81.

Sokhansanj, S., A. Turhollow, J. Cushman and J. Cundiff. 2002. Engineering aspects of collecting corn stover for bioenergy. Biomass and Bioenergy 23(5): 347-355.

Steinberg, D.H. and D.W. Palmer. 2004. Extreme Software Engineering: A Hands-on Approach. Upper Saddle River, N.J.: Prentice-Hall, Inc.

Ting, K.C. 1997a. Automation and Systems Analysis. In Plant Production in Closed Ecosystems, 171-187. Goto, E., K. Kurata, M. Hayashi and S. Sase, ed. Norwell, Mass.: Kluwer Academic Publishers.

Ting, K.C. 1997b. Chapter 12: Systems Analysis, Integration, and Economic Feasibility. In Plant Production in Closed Ecosystems Robotics for Bio-Production Systems, 287-320. N. Kondo and K.C. Ting, ed. St. Joseph, Mich.: ASAE.

Ting, K.C. 2002. Concurrent science and engineering approach to decision support for controlled environment plant production. Acta Hort. (ISHS) 578: 35-43.

Ting, K.C. 2009. Engineering solutions for biomass feedstock production. Resource: Engineering and Technology for Sustainable World 16(3): 12-13. St. Joseph, Mich.: ASABE.

Ting, K.C., Fleisher, D. H., Rodriguez, L. F., 2003. Concurrent science and engineering for phytomation systems. Journal of Agricultural Meteorology 59: 93-101.

Torget, R., C. Hatzis, T. K. Hayward, T. N. Hsu and G. P. Philippidis. 1996. Optimization of reverse-flow, two-temperature, dilute-acid pretreatment to enhance biomass conversion to ethanol. Applied Biochemistry and Biotechnology 57/58: 85-101.

Torget, R., M. Himmel, J. D. Wright and K. Grohmann. 1988. Initial design of a dilute sulfuric acid pretreatment process for aspen wood chips. Applied Biochemistry and Biotechnology 17: 89-104.

U.S. Department of Energy. 2009. Emissions of greenhouse gases in the United States 2008. Washington, D.C.: Energy Information Administration, Office of Integrated Analysis and Forecasting.

U.S. Department of Energy. 2010. Annual Energy Review 2009. Washington, D.C.: Energy Information Administration, Office of Energy Markets and End Use.

van Ouweker, E. N., D. E. James, T. L. Richard and M. Liebman. 2003. A multi-model approach for sustainable agriculture in the US corn belt. ASAE Paper No. 033009. St. Joseph, Mich.: ASAE.

Wanjura, D. F. and D. R. Upchurch. 2002. Water status response of corn and cotton to altered irrigation. Irrigation Science 21(2): 45-55.

Yang, C., J. H. Everitt, J. M. Bradford and D. E. Escobar. 2000. Mapping grain sorghum growth and yield variations using airborne multispectral digital imagery. *Trans. ASAE* 43(6): 1927-1938.

Yergin, D. 2006. Ensuring Energy Security. Foreign Affairs 85(2): 69.

Zakhour, S., Scott Hommel, Jacob Royal, Isaac Rabinovitch, Tom Risser and Mark Hoeber. 2006. The Java Tutorial: A Short Course on the Basics, 4th Edition. Upper Saddle River, N.J.: Prentice Hall.

Zhang, Y. 2008. Reviving the carbohydrate economy via multi-product lignocellulose biorefineries. *Journal of Industrial Microbiology & Biotechnology* 35(5): 367-375.

Appendix A. The Format of Properties File

//The following contents are used to create the Scenario which named Switchgrass Production.

ModelName=Switchgrass Production

Model_id=1

Scenario_id=1

NumofTask=9

NumofBox=10

Task1.class=BiomassGrowth

Task2.class=Harvesting

Task2.editable=true

Task3.class=Raking

Task3.editable=true

Task4.class=Packing

Task4.editable=true

Task5.class=FarmOpenStorage

Task5.editable=true

Task6.class=FarmCoveredStorage

Task6.editable=true

Task7.class=CentralizedStorage

Task7.editable=true

Task8.class=Transportation

Task8.editable=true

Task9.class=Biorefinery

//The following contents are used to create the ScenarioPanel for visualizing the flow chart of the modeled scenario.

Box1.Task=Task1

Box2.Task=Task2

Box3.Task=Task3

Box4.Task=Task4

Box5.Task=Task8

Box6.Task=Task5

Box7.Task=Task6

Box8.Task=Task7

Box9.Task=Task8

Box10.Task=Task9

Box1.next=Box2

Box2.next=Box3

Box3.next=Box4

Box4.next=Box5

Box5.next=Box6,Box7,Box8,Box10

Box6.next=Box9

Box7.next=Box9

Box8.next=Box9

Box9.next=Box10

//The following contents are used to create the input file for executing BioFeed.

TableIO=ColummIdentifiers,Types,Table

harvesting.ColummIdentifiers=HarvesterAttributes.csv

harvesting.Types=HarvesterTypes.csv

harvesting.Table=Harvesting.csv

raking.ColummIdentifiers=RakeAttributes.csv

raking.Types=RakerTypes.csv

raking.Table=Raking.csv

transportation.ColummIdentifiers=TransportationAttributes.csv

transportation.Types=TransportationTypes.csv

transportation.Table=Transportation.csv

farmopenstorage.ColummIdentifiers=FarmOpenStorageAttributes.csv

farmopenstorage.Types=FarmOpenStorageTypes.csv

farmopenstorage.Table=FarmOpenStorage.csv

farmcoveredstorage.ColummIdentifiers=FarmCoveredStorageAttributes.csv

farmcoveredstorage.Types=FarmCoveredStorageTypes.csv

farmcoveredstorage.Table=FarmCoveredStorage.csv

centralizedstorage.ColummIdentifiers=CentralizedStorageAttributes.csv

centralizedstorage.Types=CentralizedStorageTypes.csv

centralizedstorage.Table=CentralizedStorage.csv

packing.ColummIdentifiers=PackingAttributes.csv

packing.Types=PackingTypes.csv

packing.Table=Packing.csv

//The following contents are used to create and visualize the Result.

ResultOutput.PieChartVariableKeys=greenhouse gas emission;energy consumption;Fleet requirement;operating cost;fixed cost;capacity;cost;area

ResultOutput.ProfileVariableKeys=schedule

ResultOutput.TableNum=1

ResultOutput.ScheduleTableNum=1

ResultOutput.Table1=Output

ResultOutput.ScheduleTable1=Transportation Fleet Schedule

ResultOutput.Table1.Data=cost ($/Mg),greenhouse gas emission (kg of CO2 eq/Mg),energy consumption (MJ/Mg);Central storage facility area (square meters);Transportation Fleet requirement (number of trucks);operating cost ($/Mg);fixed cost ($/Mg);capacity;area (square meters)

ResultOutput.ScheduleTable1.Data=Transportation fleet schedule

ResultOutput.PieChartNum=3

ResultOutput.PieChart1=Cost ($/Mg)

ResultOutput.PieChart2=Energy consumption (MJ/Mg)

ResultOutput.PieChart3=Greenhouse gas emission (kg of CO2 eq/Mg)

ResultOutput.PieChart1.Data=cost ($/Mg):Harvesting,Raking,Total packing,Storage,Total transportation,Infield transportation,Biomass handling;

ResultOutput.PieChart2.Data=energy consumption (MJ/Mg);

ResultOutput.PieChart3.Data=greenhouse gas emission (kg of CO2 eq/Mg);

ResultOutput.ProfileNum=1

ResultOutput.Profile1=Transportation Fleet Schedule

ResultOutput.Profile1.XLabel=Day

ResultOutput.Profile1.YLabel=Number of Trucks

ResultOutput.Profile1.Data=Transportation fleet schedule

Appendix B. Input File Contents of BioFeed

This section introduces the contents of three different input files (Section 5.2.3.1) for executing BioFeed. For each task which allows to be modified in the modeled scenario in BioFeed through the attribute table (Figure 4.14 andFigure 4.15), BPSys have to prepare the three input files. For illustration, the following contents illustrate the Types, ColumnIdentifiers, and Table input files for the packing task. If the user only selects the Round_Baler for the packing task like Figure 4.14, the input files are like the contents below.

//Types input file

Round_Baler

// ColumnIdentifiers input file

Name

PurchasePrice

OperatingCost

AnnualInterest

IHT

Throughput

Efficiency

Horsepower

FuelConsumption

PackingDensity

PackingVolume

RotaryPowerRequirement

TractorHorsepower

TractorOperatingCost

TractorFuelConsumption

TractorIHT

TractorAnnualInterest

BiomassLossFraction

//Table input file

Name,Round_Baler

PurchasePrice,44493.06

OperatingCost,23.10293

AnnualInterest,1769

IHT,1.06

Throughput,18.14

Efficiency,0.85

Horsepower,120

FuelConsumption,5.256

PackingDensity,0.128

PackingVolume,4

RotaryPowerRequirement,1.8

TractorHorsepower,120

TractorOperatingCost,29.11

TractorFuelConsumption,5.256

TractorIHT,1.07

TractorAnnualInterest,2771

BiomassLossFraction,0.05

Appendix C. Output File Contents of BioFeed

Biorefinery capacity = 1268.80

Total delivered cost = 37.69

Harvesting cost ($/Mg) = 8.42

Raking cost ($/Mg) = 3.05

Packing fixed cost ($/Mg) = 0.36

Packing operating cost ($/Mg) = 3.90

Total packing cost ($/Mg) = 4.25

Storage cost ($/Mg) = 4.41

Transportation fixed cost ($/Mg) = 1.60

Transportation operating cost ($/Mg) = 5.95

Total transportation cost ($/Mg) = 7.55

Infield transportation cost ($/Mg) = 6.84

Biomass handling cost ($/Mg) = 3.16

Harvesting energy consumption (MJ/Mg) = 49.99

Raking energy consumption (MJ/Mg) = 22.50

Packing energy consumption (MJ/Mg) = 78.58

Transportation energy consumption (MJ/Mg) = 41.35

Harvesting greenhouse gas emission (kg of CO2 eq/Mg) = 3.63

Raking greenhouse gas emission (kg of CO2 eq/Mg) = 1.63

Packing greenhouse gas emission (kg of CO2 eq/Mg) = 5.70

Transportation greenhouse gas emission (kg of CO2 eq/Mg) = 3.00

Central storage facility area (square meters)

Location-3 13.42

Total on-farm storage area (square meters) = 409535.67

Total silage pit area (square meters) = 0.00

Transportation Fleet requirement (number of trucks)

Type1 34.00

Type2 0.00

Transportation fleet schedule

	Type1	Type2
1	34.00	0.00
2	34.00	0.00
3	34.00	0.00
.	.	.
.	.	.
.	.	.
358	34.00	0.00
359	34.00	0.00
360	34.00	0.00