SIMULATION MODELING OF BODY WEIGHT DYNAMICS AND WEB GAME DEVELOPMENT

by

Mert Nuhoğlu

B.S. in Industrial Engineering, Bogazici University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2009

SIMULATION MODELING OF BODY WEIGHT DYNAMICS AND WEB GAME DEVELOPMENT

APPROVED BY:

Prof. Yaman Barlas (Thesis Supervisor)	.....
Assoc. Prof. Cengizhan Öztürk	.....
Assist. Prof. Aybek Korugan	.....

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor, Prof. Yaman Barlas, for his guidance during this research. I am thankful to him especially for his challenging inquiries that forced me to question the methods that I was using unconsciously. I believe that all the challenging discussions that we did throughout this research study enriched my scientific perspective deeply.

I am thankful to Assoc. Prof. Cengizhan Öztürk and Assist. Prof. Aybek Korugan for their valuable comments as members of my thesis committee.

I wish to thank to Kevin D. Hall for sharing his simulation model with me and for his help. I really appreciate the value of his help.

I wish to express my deepest gratitude to my parents, Asiye and Zafer Nuhoğlu. Without their support I could not afford to sustain this research.

I wish to thank to my wife, Hasret Nuhoğlu, for her friendship in everything I live. Although the subjects in this study are very far from her expertise, discussing the research problems with her was very valuable for me.

Also, I sincerely want to thank to the engineers that made Google. I believe that Google became an extension of our minds.

Finally, I wish to thank to all my sincere friends and to all the artists and intellectuals of the world for making our lives pleasant.

ABSTRACT

SIMULATION MODELING OF BODY WEIGHT DYNAMICS AND WEB GAME DEVELOPMENT

In this thesis, we first build a valid simulation model of the body weight dynamics system of a healthy, adult person. The model simulates the long term dynamics of the body weight, based upon Hall's body weight simulation model [1].

We perform extensive validation testing of Hall's model by using system dynamics approach to evaluate the validity of the model's causal structure. The validation tests show that the model has realistic behaviors under various dieting regimes. There are a few modification areas in which our model differs from Hall's original model. Firstly, our model has some simplifications that highlight the causality relationships more clearly, while maintaining the validity. Secondly, our model incorporates a hypothesized process called secondary oxidation to make the system produce valid behavior under very high energy expenditures. Thirdly, we render the model generic for any healthy, adult person, and we deduce the formulas of metabolic parameters to customize the generic model to the specific, simulated person.

In the second part of the thesis, we develop a software (Mashap) that automatically generates and runs a web based simulation game for any system dynamics model. Our software has some important advantages over the existing ones: First, it is an open platform on which new tools can be built. Second, it is built for generating web games for all system dynamics modeling software. We finally use the game generator to develop a web based game for our modified body weight dynamics model. The game generator will enable the system dynamics researchers to build web based gaming environments for their own models.

ÖZET

VÜCUT AĞIRLIĞI BENZETİM MODELİ VE WEB OYUNU GELİŞTİRİLMESİ

Bu tezin birinci kısmında, sağlıklı bir yetişkinin vücut ağırlığı dinamiğinin geçerlenmiş bir benzetim modeli geliştirilmiştir. Bu model, insan vücut ağırlığının uzun dönemdeki değişimini benzetmektedir ve büyük oranda Hall'un vücut ağırlığı benzetim modeli temel alınarak geliştirilmiştir [1].

Bu çalışma, Hall'un modelinin nedensel yapısının geçerliliğini göstermek için, sistem dinamiği yaklaşımının geçerleme testlerini kapsamlı bir şekilde kullanmıştır. Uygulanan geçerleme testleri, farklı diyet rejimleri altında modelin gerçekçi bir şekilde çalıştığını göstermektedir. Tezdeki benzetim modeli, Hall'un modelinden birkaç yerde ayrılmaktadır: Birincisi, nedensellik ilişkilerini vurgulayan bazı sadeleştirmeler yapılmıştır. İkincisi, aşırı enerji harcama durumunda, sistemin geçerli davranış üretmesini sağlayan ikincil oksitlenme adı verilen varsayımsal bir süreç eklenmiştir. Üçüncüsü, değiştirilmiş modelin herhangi bir sağlıklı yetişkine uyarlanabilmesi için, benzetim yapılan kişiye özgün metabolik özellikleri veren genel formüller türetilmiştir.

Tezin ikinci kısmında, herhangi bir sistem dinamiği modelinden web tabanlı bir benzetim oyunu üreten ve çalıştıran bir yazılım geliştirilmiştir. Yazılım, varolan yazılımlara birkaç bakımdan üstündür. Birincisi, üzerinde yeni araçların geliştirilebileceği açık bir platform sunmaktadır. İkincisi, bu yazılım, diğer tüm sistem dinamiği model kurma yazılımları için web oyunları üretebilecek şekilde geliştirilmiştir. Bu oyun kurucusu kullanılarak, Hall'un değiştirilmiş vücut ağırlığı benzetim modeli için web tabanlı bir oyun üretilmiştir. Oyun kurucusu yazılımı, sistem dinamiği araştırmacılarına, kendi modelleri için, web tabanlı benzetim oyunları üretme imkanı sunmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$act_{eng,bw}$	Activity energy per body weight
bm	Bone mass
bw	Body weight
c	Carbohydrate
cal_c	Caloric density of carbohydrate
cal_f	Caloric density of fat
cal_p	Caloric density of protein
ci	Carbohydrate intake
ciw	Constant intracellular water
cox	Carbohydrate oxidation
d_c	Glycogenolysis
d_f	Lipolysis
d_p	Proteolysis
$d_{f,mol,b}$	Baseline molar lipolysis
dnl	De Novo Lipogenesis
ecw	Extracellular water
$eff_{act,pox}$	Effect of physical activity on protein oxidation
$eff_{ci,cox}$	Effect of carbohydrate intake on carbohydrate oxidation
eff_{ci,d_f}	Effect of carbohydrate intake on lipolysis
$eff_{dg,cox}$	Effect of glycogenolysis on carbohydrate oxidation
$eff_{pi,pox}$	Effect of protein intake on protein oxidation
f	Fat
f_c	Fraction of carbohydrate oxidation
f_f	Fraction of fat oxidation
f_p	Fraction of protein oxidation
$f_{ecw,tw}$	Fraction of extracellular water to total water
$f_{icw,ecw}$	Fraction of intracellular water to extracellular water
$f_{tw,bw}$	Fraction of water to body weight
$f_{icw,cm}$	Fraction of intracellular water to cell mass

fi	Fat intake
fox	Fat oxidation
gng_f	Gluconeogenesis from fat
gng_p	Gluconeogenesis from protein
h_c	Carbohydrate hydration coefficient
h_p	Protein hydration coefficient
ics	Intracellular solids
icw	Intracellular water
m_g	Mass of glycerol
m_{cell}	Mass of body cells
m_{lean}	Mass of lean tissues
m_{tg}	Mass of triglyceride
mbc	Metabolism of body cells
mc	Metabolism of conversions
mei	Metabolic energy intake
mt	Metabolism of turnovers
$n_{\Delta ci}$	Normalized change in carbohydrate intake
$n_{\Delta pi}$	Normalized change in protein intake
n_{d_c}	Normalized glycogenolysis
n_{d_f}	Normalized lipolysis
n_{d_p}	Normalized proteolysis
ox	Total oxidation
p	Protein
pae	Physical activity energy
pox	Protein oxidation
rmr	Resting metabolic rate
s_{ci}	Sensitivity of carbohydrate oxidation to carbohydrate intake changes
s_{pae}	Sensitivity of protein oxidation to physical activity
$s_{pi,ox}$	Sensitivity of protein oxidation to protein intake changes
tee	Total energy expenditure
tef	Thermic effect of food

w_f	Weighting of oxidation for lipolysis
w_g	Weighting of oxidation for glycogenolysis
w_{ci}	Weighting of oxidation for baseline carbohydrate intake
w_{pi}	Weighting of oxidation for baseline protein intake
Δci	Change in carbohydrate intake
Δpi	Change in protein intake
Γ_c	Coefficient for effect of carbohydrate intake on gluconeogenesis from protein
Γ_p	Coefficient for effect of protein intake on gluconeogenesis from protein
a	Actual (primary)
b	Baseline
Δ	Change
n	Normalized
s	Secondary

1. INTRODUCTION AND LITERATURE SURVEY

The body weight dynamics is mainly determined by the interaction of two components: energy intake and energy expenditure. But the interaction between these two components is not a simple mechanism as some people believe. General public's view of body weight dynamics is that in order to lose weight you have to eat less and exercise more. The opposite is true for weight gain: eat more and do less exercise. But this world view does not reflect the complexity of the reality adequately.

Fighting with obesity is a crucial problem in the modern world. Only in USA, it is estimated that more than 100 billion dollars per year is spent on obesity treatments ([2]). The health costs of obesity and psychological problems caused by it are not included in this estimated cost. Most people have a simple and defective world view for how to lose weight. This misapprehension increases the psychological problems of obesity even deeper. People that try to lose weight try very hard, but they don't succeed. The failure in this fight causes the people to lose their self confidence.

This research has two aims:

1. Development of a body weight simulation model for gaming purposes
2. Development of a game engine that will automatically generate and run a web game for any system dynamics model

The final product will be a model-based gaming tool, called Body Weight Websim, that will allow people to track the changes in their body weight in the long term. The tool should be useful for a person to get a deeper understanding of the body weight dynamics and to build a better dieting and exercising regime for himself. Researchers may also use the simulation tool in order to find out easy to implement ways where the person loses weight, and the metabolism rate of the person remains stable.

Chapter 4 describes the body weight simulation model developed in the research.

The aim of the body weight simulation model is to model long term dynamics of the body weight of the human realistically. The model is a modified version of Hall's body weight simulation model ([1]).

Second part of this research contains development of a general, web based game engine for any system dynamics model. The game engine, called Mashap, is able to generate web user interfaces and run any system dynamics model developed in Vensim. Support for other software can be added easily to Mashap. We use Mashap in this study to develop a web based game for the body weight dynamics model. The aim of the tool is to provide a web based game building environment for system dynamics researchers around the world.

Currently, there are several software that run system dynamics simulation models on the web, but current software have several constraints that necessitate development of new tools. The primary constraint of the current software is that they are commercially licensed, closed source products. This is a hinder to building new tools upon the current ones. Open source or free software allows people to build innovative tools upon existing software that the original programmers could not imagine or afford on their own. This need is the fundamental rationale for the development of the general web based game engine in this research. The detailed comparison of Mashap and the available model building software in the market is described in Section 5.5.

There are three types of studies that contain mathematical models related to the dynamics of the body weight:

1. Short term models for glucose, insulin, or food intake metabolism
2. Mathematical analysis of the general dynamics of the body weight
3. Long term models for the dynamics of the body weight

The aim of the short term models is to understand the dynamic changes of metabolites in the digestion and absorption of the food. Their time frame changes from a few minutes to a few hours. An example of such a model is the model devel-

oped by Vicini [3]. Vicini's model is a simulation model of the glucose-insulin control system during meals. The model is useful for short term decisions such as when and how much drugs to give to diabetes patients. The model simulates the dynamics of the variables during a time frame of a few hours. Short term models of energy metabolism are plentiful. But they are not suitable for understanding the change of the body weight in the long term. Therefore, this kind of models is not relevant to this thesis study.

The aim of the models for mathematical analysis of the general dynamics of the body weight is to do equilibrium, stability, and phase plane analysis. An article of this kind is jointly written by Hall and Chow [4]. They study the dynamics of the body weight change in a very general way. The model in this study is not used to build a simulation model. Instead, the purpose is to analyze the differential equations governing body weight change analytically by doing stability and equilibrium analysis. These types of models contain a lot of simplifications. Therefore, they are not suitable to build a realistic game that simulates the long term body weight dynamics.

The models that study the long term body weight dynamics are suitable for the purposes of this study. There are two important models of this type in the literature:

1. Abdelhamid's model published as "Modeling the dynamics of human energy regulation" [2]
2. Hall's model published as "Computational model of in vivo human energy metabolism during semi starvation and refeeding" [1]

Abdelhamid's article does not publish the equations used. Although the diagrams that summarize the model and the results of the simulations are very detailed and realistic, Abdelhamid's model is not useful for this study since the mathematical relationships between the variables are not given.

Hall's model aims to build a general simulation model for the change of the body weight in long term. The model integrates nutrition, metabolism, body composition,

and physical activity. The composition of the body is an internal variable of the model. The model assumes that the rates of daily food intake and physical activity are external variables since their levels are determined by the decision of the person. The time unit of the model is one day. So, the dynamics of the metabolic activities within the daily rhythm of the body are not relevant to the model.

The values of the parameters in Hall's model are based on the Minnesota Starvation Experiment's data [5]. The Minnesota Experiment is an experiment done during 1944-1945 in Minnesota, USA. Its aim was to find out better ways of rehabilitation for the large masses of Europeans who had lived under hard starvation for a few years. The experiment consisted of three phases:

Control phase: The aim of this phase was to build baseline data where the body weight of the subjects were maintained in stability. It lasted 12 weeks and in this period 3200 kcal per day were given to the subjects.

- Starvation phase: The aim in this phase was to reduce the body weight by 25%. It lasted 24 weeks. During this period 1570 kcal per day were given to the subjects.
- Rehabilitation phase: The aim of this phase was to rehabilitate rapidly and safely. 2366 kcal per day were given to the subjects.
- Total number of subjects in the experiment were 36. Detailed data of feeding, body weight and body composition were recorded during the experiment.

The model contains three macronutrient stocks (compartments): fat, carbohydrate, and protein. Moreover there is one more stock in the model, a conceptual variable called adaptive thermogenesis. There are four external inputs to the model: daily caloric intakes of fat, carbohydrate, and protein are food intake variables. Daily level of physical activity is the fourth external input to the model. The model consists of three macronutrient stocks plus adaptive thermogenesis stock, and flows (fluxes) between them (Figure 1.1).

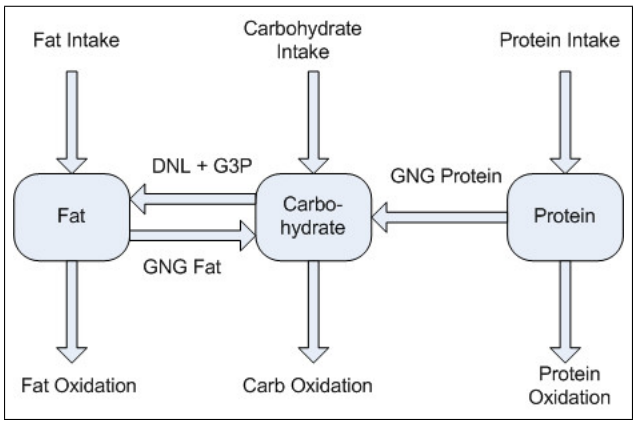

Figure 1.1. Stocks and flows of Hall's model. DNL denotes de novo lipogenesis rate, G3P denotes glycerol 3-phoshphate synthesis rate, GNG denotes gluconeogenesis rate. (Adopted from Hall [1])

Hall's model is the basis of this thesis study. In this thesis, we modify certain parts Hall's original model. From this modified model, we produce the gaming environment called Body Weight Websim. Hall's modified model is described in detail in Chapter 4. Body Weight Websim is described in Section 5.4.

2. PROBLEM DESCRIPTION AND RESEARCH OBJECTIVE

Obesity is treated as one of the primary health problems in the modern world. Obese people may try hard to lose weight, but usually they don't succeed in the long term. The difficulty of losing weight and maintaining the new body weight is caused by the negative feedback loops in the human energy regulation that complicate changes in body weight. In general, the negative feedback loops in the body weight regulation system counteract to the changes in the body weight. If the person gains weight, the regulation mechanisms try to limit the gain. If the person loses weight, the same mechanisms try to limit the loss. These strong negative feedback mechanisms help the mankind to adapt to a wide range of environments and living conditions.

There is no way to cut the counteracting forces of the negative feedback mechanisms in the energy regulation system. It is better for the dieting person to be aware of the limits of her body and how these limits constrain her regime. Understanding the negative feedback loops can help the people to put more realistic goals for weight losing diets.

The aim of this study is to develop an interactive simulation model for the long term body weight dynamics. The simulation model can be helpful to people who try to manage their body weights and to dietitians who try to find better dieting regimes for their patients.

Hall's current model of the body weight dynamics is the basis of this study. The first step in this study is to validate Hall's model extensively to be sure that it reflects the dynamics of the body weight regulation system realistically and to make any necessary modifications. Next step is to develop an interactive game that lets the people to experiment possible scenarios of dieting regimes. The game is designed to be calibrated for any healthy, adult person. This flexibility of customization requires some modifications in Hall's model for initializing some of the metabolic parameters.

Metabolic parameters are calibrated for the person who is simulated in the game.

Another outcome of this study is a general software platform that allows the model builders to produce web based, interactive, dynamic simulation games for any model that they develop in popular model building tools such as Vensim, Stella, or Powersim. This software platform will be used to build the game for the body weight dynamics model as a proof of the concept.

3. RESEARCH METHODOLOGY

The aim of this thesis study is to build a valid model of body weight dynamics of a healthy, adult person. The model is designed to be customized for the metabolic characteristics of the person that is simulated in the game. There are strong non-linear, negative feedback loops that regulate the system of the body weight dynamics. System dynamics is a methodology that is specifically useful to model and understand systems that have inherent non-linear feedback loops. Therefore, system dynamics approach is very appropriate for the model building, validating, and analysis of the body weight dynamics system. Although Hall's model is used as the basis of this study, system dynamics approach helps in improvements of the model by validating the model extensively, simplifying the model, and making the model more robust to the changes in external conditions.

System dynamics approach is a holistic approach that does not separate components of a system to analyze them in isolation of the interactions from other components. Analyzing the components in isolation is not suitable for understanding complex, dynamic systems because the behavior of the components in isolation can be totally different from their behavior in interaction with other components of the system. System dynamics approach focuses on causal explanation of the dynamic behavior that the system produces. This requires to approach the system from the point of view of the inherent feedback loops in the system.

In this study, validation takes substantial amount of effort. Understanding and validating Hall's model requires a systemic and methodological approach. We follow the validation procedures of system dynamics methodology explained in Barlas (1996) [6]. Validation of complex systems consists of two stages: First, the validity of the structure is examined. Next, if the model passes these first tests, then behavior patterns of the system are examined.

Validation study examines the system not only in normal conditions. It is essential

that the model works logically under extreme conditions too. If the model does not work logically in extreme conditions, then it is very probable that the model contains some non-causal structure that is probably built by using some statistical approach such as data fitting.

Another aim of this thesis is to develop a game engine that automatically generates and runs a web based simulation game for any system dynamics model. We use the game engine to develop a web based game for Hall's modified body weight dynamics model. The game engine enables the system dynamics researchers to build web based gaming environments for their own models.

The software is developed by using the object oriented development methodology. Domain driven design and design patterns are the main design techniques used in this thesis. These design techniques are effective for designing the software according to the object oriented development methodology. Domain driven design is a design technique that is used for the distribution of responsibilities of the objects to write the code cleanly and maintainable. Design patterns are general reusable solutions to commonly occurring problems in software design.

4. BODY WEIGHT MODEL

This chapter describes the body weight simulation model in detail. The aim is to model the long term dynamics of the body weight of the human. The model in this thesis is mostly based upon Hall's body weight simulation model [1]. This thesis study tests the validity of each equation in Hall's model. Our model has some simplifications and modifications over Hall's model.

The body weight dynamics is mainly determined by the interaction of two components: energy intake and energy expenditure. But the interaction between these two components is not a simple mechanism. There are several negative feedback loops in the human body weight regulation that counteract to the changes in the body weight.

4.1. Overview of the Model

The simplest form of the model is described in Figure 4.1. The stock "substance" is the aggregate representation of all the material that constitutes the body. There is one aggregated inflow as "food intake" and one aggregated outflow as "oxidation". Naturally, this is an oversimplification of the material exchange of the body. Apart of oxidation, there is also some amount of material outflow through excretory system. Actually the oxidation outflow contains excretory outflow implicitly because some parts of the protein molecules that are broken down and thrown away through excretory system are oxidized.

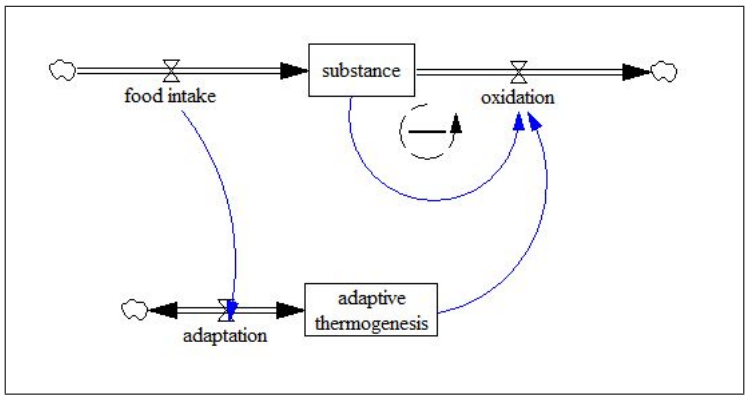

Figure 4.1. Simplified Model Overview

Our model contains three stocks that disaggregate the body substance. They are the stocks of protein, carbohydrate and fat. There are lots of other materials in the human body. At least 70% of the human body consists of water alone. There are also some organic materials that are not included in these three stocks such as nucleotides or hormones. The amounts of these materials are all closely correlated with the amounts of the three stocks in the model. Therefore our model defines the amounts of these materials as functions of the three material stocks in the model.

A conceptually different stock from the material stocks is the adaptive thermogenesis stock. This stock variable represents the ability of the body to adapt itself to the changes in the food intake. There is some controversy around this concept in the literature of energy metabolism. There is some published research that suggests that there is no adaptation of the body at all. But there are also several research results suggesting the existence of the adaptation.

The fundamental physiological idea behind adaptive thermogenesis is that the body cannot leave energy regulation to the willpower of the person only, because lack of energy resources in the long term will certainly bring the body to a death. Therefore, there must be some powerful enough control mechanisms that prevent death caused by the long term energy deficiency. Actually, there are some physiological explanations on how adaptation occurs: One explanation is that all physiological activities that require energy inside the body are diminished to a lower amount during starvation.

For example, active membrane transportation or cell division occurs more slowly and less frequently. This can be done by releasing and deploying less protein molecules that carry the physiological activities requiring energy.

So it seems reasonable that there exists a regulation mechanism that adapts body's physiological activities to the energy stock and inflow. During starvation all the activities are slowed down, and during energy excess all the activities are accelerated. But of course there must be a physical limit to the adaptation capacity of the body.

Another controversy occurs around the triggering cause of the adaptive thermogenesis. One possibility is that the changes in the energy intake trigger adaptation. Another possibility is that the changes in the energy stock trigger adaptation of the body. There is not enough evidence in the literature that proves one of the arguments. In our model, we adopt the causal relationship between the energy intake and adaptive thermogenesis as assumed in Hall's model.

In the most simplified overview of the model, there is one fundamental feedback loop: The level of the body substance increases the amount of oxidation, and this causes a relative decrease in the level of substance. So, this is a negative feedback loop. In reality, there are also some feedback loops that control the amount of the food intake desire and capacity of a person, but the boundaries of our model do not include these factors. If we had taken them into the model too, then our model could not be used by some person to test the effectiveness of a dieting habit, because dieting habit would then be an internal variable; the person could not change dieting habit freely in the game, and explore the results.

4.1.1. Main Sectors of the Model

Figure 4.2 shows the causal relationship of the whole model in aggregated terms. There are 12 important conceptual variables shown in this diagram. The stocks and their flows are the same as in Figure 4.1. This diagram shows additionally the intermediate variables that control the flows.

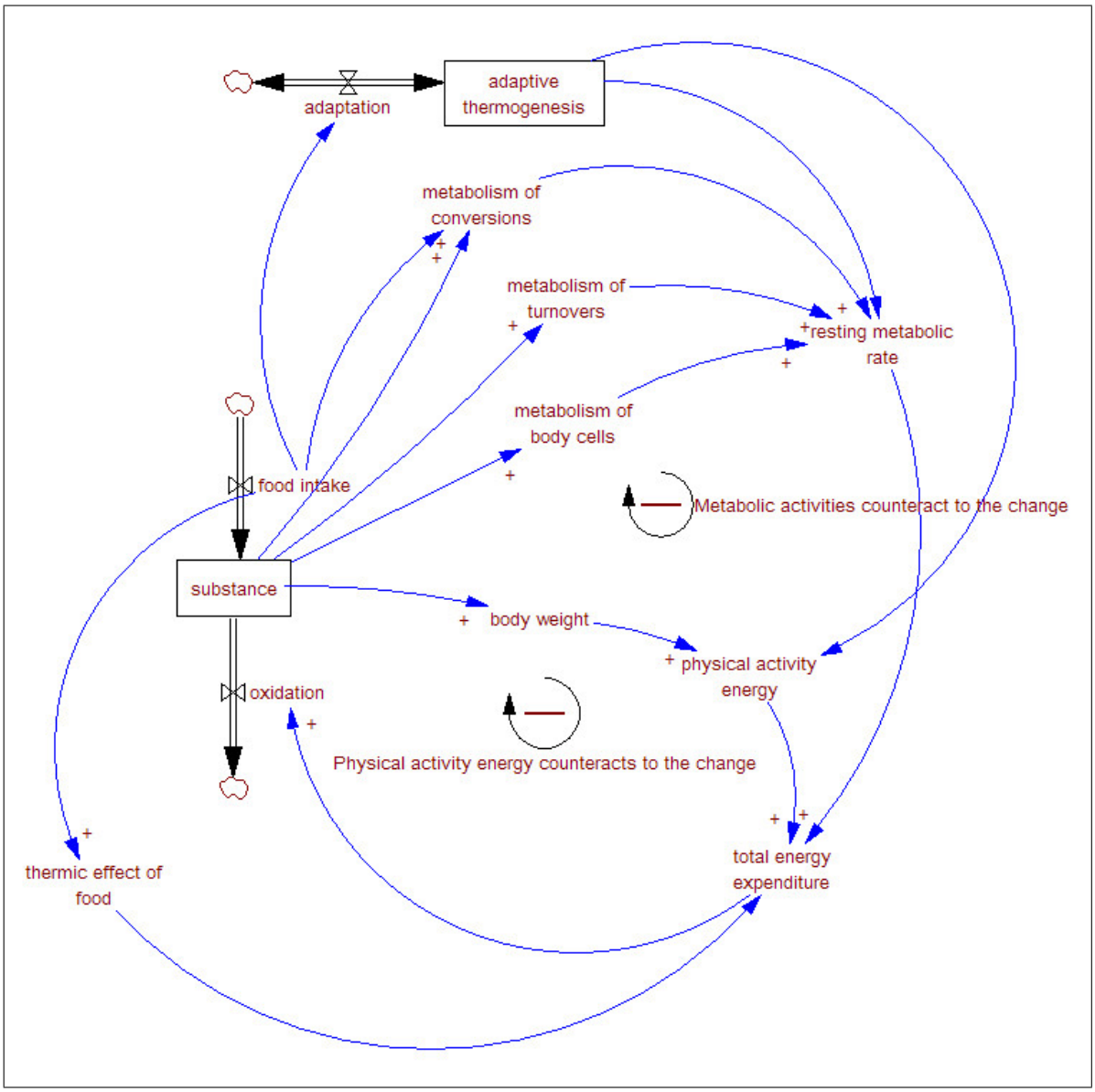

Figure 4.2. Overview of the main sectors of the model. Substance denotes aggregate energy stores (protein, fat, carbohydrate).

The food intake is an external variable in this model. It is assumed to be determined by the person freely. This assumption is actually not realistic as explained in the previous section. The food intake is controlled by the capacity of a person, and the capacity of the person is effected by the food intake rate and some other variables not included in the model. Nevertheless, leaving food intake as a constant external

variable is acceptable for the purpose of our model since we want our model to test the effectiveness of a dieting habit in terms of weight loss or gain.

The Figure 4.2 is actually a detailed view of the link between substance (energy stores) and oxidation rates in Figure 4.1. The amount of body substance is a self limiting stock variable. An increase in the body substance triggers mechanisms that increase the outflow of the body substance, therefore it limits its own growth. Self limiting growth in this system occurs through two negative feedback loops:

- Metabolic activities counteract the changes in the body substance
- Physical activity energy expenditure counteracts the changes in the body substance

The first loop is more important for an ordinary person since the share of resting metabolic rate takes around 60-70% of the total energy expenditure for a normal person. In this loop, the body substance effects the resting metabolic rate over three factors: the metabolism of conversions, the metabolism of turnovers, and the metabolism of body cells. The metabolism of conversions covers all the metabolic activities that involve conversion of an energy macronutrient (carbohydrate, fat, or protein) to another one. The metabolism of turnovers covers all the energy requiring metabolic activities that construct or deconstruct macronutrients from their building stones. The metabolism of body cells covers all the energy requiring metabolic activities to maintain living conditions such as active membrane transportation, cellular division. Actually there is a partial overlap between these three variables. Metabolism of conversions and turnovers are actually parts of metabolism of body cells, but it is more suitable for the purpose of our model to break down metabolism of body cells into three parts. The reason behind this separation is that the effect of nutrition and body substance differs between these three components.

Metabolism of turnovers is dependent only on the current stocks of macronutrients. Stocks of macronutrients (carbohydrate, fat, and protein) are shown as aggregated in one body substance stock in Figure 4.2. There is a direct linear proportional

causal relationship between the substance stock and the metabolism of turnovers. The level of the body substance directly effects the amount of macronutrients to be constructed or deconstructed. The reason behind the positive linear relationship between the amount of nutrient turnover and nutrient stock is based on two assumptions. The first assumption is that the body tries to recover the amount of macronutrients deconstructed. Namely, the constructed amount of macronutrients is more or less equal to the amount of deconstructed amount. The second assumption is that the amount of macronutrients deconstructed is a linear function of the current stock of macronutrients. One component of this assumption is the deconstruction rate of protein. The model is based on the assumption that the amount of protein deconstruction is dependent on the average life time of a protein molecule. The model assumes that the average life time for protein molecules remains stable. The validity of these and other assumptions in the model are open to further research. Most of these assumptions are actually not based on specific proofs of physiological research because these questions have not been studied in sufficient detail by researchers yet.

The metabolism of conversions is dependent on two types of variables: stocks and inflows of the macronutrients. The relationship is positive again. When there is more food intake or body substance, then there is more conversion of one macronutrient to another one.

The metabolism of body cells is dependent on the current stocks of the macronutrients only. The relationship is again positive. Actually, the metabolism of body cells is dependent mostly on one stock of macronutrient only: protein. Fat and carbohydrate stocks have comparatively very little effect on metabolism of body cells. This is due to the storage mechanism of fat and carbohydrate. Fat and carbohydrate molecules are stored in the body in a very efficient way. Most of the fat in the body is stored in fat cells (adipocytes). 90% of the content of these specialized cells is fat. Storing fat requires very little metabolic activity in the body. Therefore the contribution of the fat stock to the metabolism of body cells is very small. The contribution of carbohydrate stock to metabolism of body cells is also very small since the amount of the stored carbohydrate inside the body at a any time is very little (around 400 g). So, the

metabolism of body cells is mostly dependent on the level of protein stock. This is very reasonable since most of the metabolic activities inside the living organism is done by protein molecules. Apart from proteins there are also substances like nucleotides and some lipids that have some functions in the body metabolism. The model assumes that the amount of these other substances does not change for a healthy adult person. This assumption is mostly valid in real life. The body composition of these substances is maintained more or less stable in a healthy, adult person.

Resting metabolic rate is the energy requirement of an awaken person in resting conditions. It is not to be confused with the basal metabolic rate. Basal metabolic rate is the energy expenditure rate of the body during sleep. But resting metabolic rate is more than basal metabolic rate. Resting metabolic rate contains energy expenditure during basic movement activities like sitting, walking a few steps, and eating. Approximately, resting energy expenditure makes up 60-70 percent of the total energy expenditure.

Resting metabolic rate is the sum of three metabolism components: metabolism of conversions, turnovers, and body cells. Adaptive thermogenesis is the fourth factor that effects resting metabolic rate. Adaptive thermogenesis is a counteracting mechanism of the body against the change in the energy intake. The model assumes that when there is a shortage of energy intake, then the body slows down all metabolic activities. When there is an excess of energy intake, then the body increases all metabolic activities.

For a normal person, 20-30% of daily energy expenditure is due to the physical activities. The physical energy expenditure is directly dependent on the body weight due to the dependence of kinetic energy on mass: $E\alpha m$. So, there is another negative feedback loop that limits the self growth of the body substance. An increase in the body substance stock will eventually cause its own growth to slow down due to the increase in the physical energy expenditure rate.

The thermic effect of food is the third component of energy expenditure shown in Figure 4.2. Approximately 10% of energy expenditure in a healthy, adult person is

due to the thermic effect of food. This is the energy expenditure spent for digestion and absorption of the foods. Thermic effect of food is a linear function of food intake. In our model, thermic effect of food is not involved in any feedback loop. Anyway, the decrease in energy intake is partially compensated by the decrease in thermic effect of food. This is not a feedback loop in our model because the food intake is exogenous, but it is another counteracting force to the change in the food intake of a person.

The thermic effect of food, resting metabolic rate, and physical activity expenditure sum up to the total energy expenditure of a person. The total energy expenditure has a linear positive effect on the oxidation rate of the body since oxidation is the only way for the body to provide the required amount of energy expenditure. Actually, the body can also generate energy by anaerobic respiration, but this is not relevant for our model due to a few reasons. Firstly, anaerobic respiration has a very small share in total oxidation. Secondly, anaerobic respiration occurs exceptionally rarely only when there is not sufficient oxygen respiration. And whenever this happens, it can last for only a few minutes.

Oxidation is an outflow of the body substance stock since during oxidation stocks of macronutrients are broken down and thrown away out of the body through exhalation.

Figure 4.3 and Figure 4.4 show the stock flow diagrams of the whole model in detail. Actually, these diagrams still do not show constant parameters that have effects on the variables. Understanding the model as a whole in one step is not easy. Therefore, we will go over parts of the diagram.

Figure 4.2 is a simplified representation of the model in aggregated terms. In the simplified diagram, variables represent important sectors of the actual model. The next sections cover the variables metabolism of conversions, metabolism of turnovers, metabolism of body cells, body weight, oxidation, physical activity energy, and adaptive thermogenesis step by step as important sectors of the model. The variables total

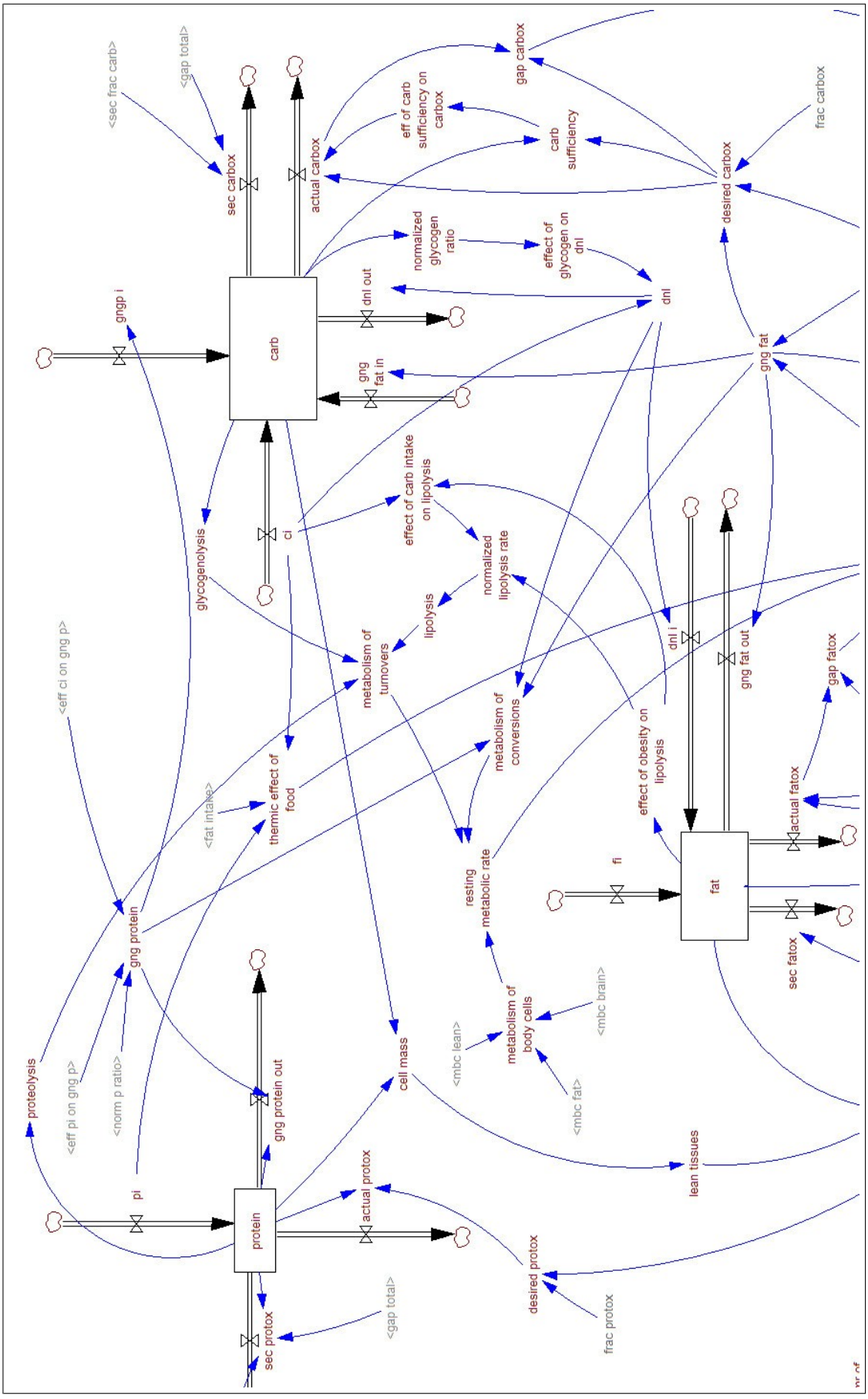

Figure 4.3. Whole Model Part 1

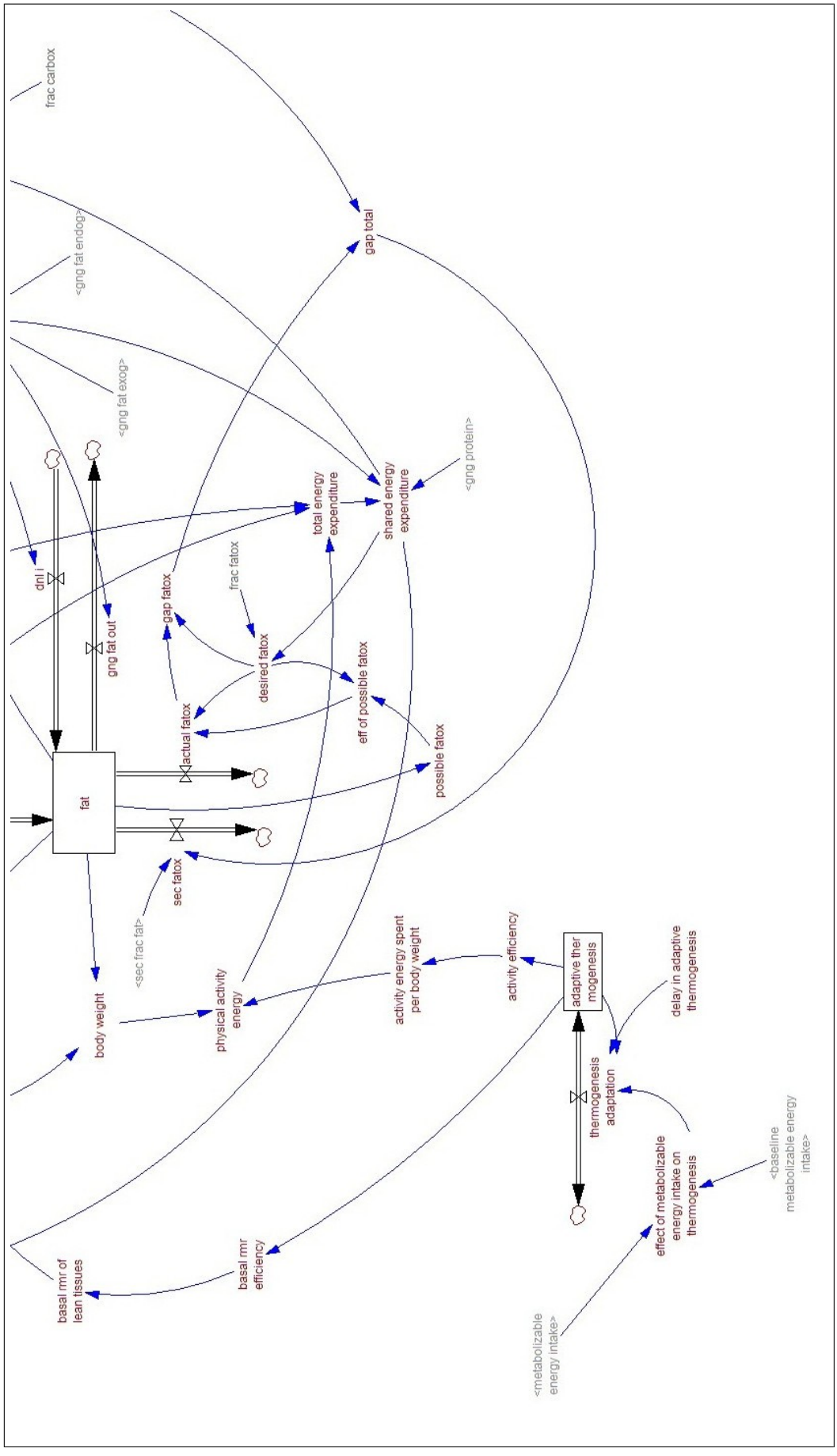

Figure 4.4. Whole Model Part 2

energy expenditure, resting metabolic rate, thermic effect of food, and food intake are not covered as separate sectors because their functions are pretty simple.

4.2. Description of the Model

The body weight simulation model is a general model for the body weight change in the long term. The model integrates nutrition, metabolism, body composition and physical activity. The model does not only track and calculate the body weight; it also calculates the composition of it. The model takes food intake and physical activity as external inputs. Time unit of the model is one day. So the metabolic activity details within the daily rhythm of the body are not relevant to the model.

Body weight simulation model in this research is a modified version of Hall's model [1]. Hall's original model is based on the Minnesota Starvation Experiment's data [5]. The model contains three macronutrient stocks (compartments): fat, carbohydrate (mostly glycogen) and protein. Moreover there is one more stock in the model, a conceptual variable called adaptive thermogenesis. There are four external inputs to the model. Daily caloric intakes of fat, carbohydrate, and protein are food intake variables. Daily level of physical activity is the fourth external input to the model. The model consists of three macronutrient stocks plus adaptive thermogenesis stock, and flows between them. The simplified diagram of Hall's model is shown in Figure 1.1.

Basic differential equations in our modified model are given in Equation 4.1.

$$\begin{aligned} \frac{dc}{dt} &= (ci + gng_p + gng_f - dnl - cox_a - cox_s)/cal_c \\ \frac{df}{dt} &= (fi + dnl - gng_f - fox_a - fox_s)/cal_f \\ \frac{dp}{dt} &= (pi - gng_p - pox_a - pox_s)/cal_p \end{aligned} \tag{4.1}$$

The stocks in the model, f, c, p denote fat, glycogen (carbohydrate), and protein masses in the body. The symbols fi, ci, pi are energy intake variables: fat intake,

carbohydrate intake, and protein intake. The variables fox, cox, and pox are oxidation rates of the corresponding masses. The indexes a and s denote the primary and secondary oxidations, respectively. The variable dnl is the fat synthesis rate from glucose. The variables gng_f and gng_p are gluconeogenesis rates, that is glucose synthesis rates, from fat and protein, respectively. The parameters cal_c, cal_f, cal_p denote caloric densities of carbohydrate, fat, and protein. Hall's original model is given below in Equation 4.2.

$$\begin{aligned} \frac{dc}{dt} &= (ci + gng_p + gng_f - dnl - g3p - cox)/cal_c \\ \frac{df}{dt} &= (3m_{ffa} * fi/m_{tg} + dnl - fox)/cal_f \\ \frac{dp}{dt} &= (pi - gng_p - pox)/cal_p \end{aligned} \tag{4.2}$$

The differential equations in Equation 4.1 differ from Hall's original model given in Equation 4.2 in a few places: Firstly, our model does not contain $g3p$ which denotes glycerol 3-phosphate synthesis. We left this variable out because of the complexity of its formula and relative indifference of the model to this variable. Secondly, our model adds fi directly whereas Hall's model includes $3m_{ffa} * fi/m_{tg}$. We simplified this formula because first there is very small quantitative difference between two formulas and second we could not find out the rationale behind Hall's formula. Third difference between two models is that Hall's model does not include gng_f in df/dt equation whereas our model includes it. We don't know why Hall excluded this variable. Fourth difference between two models is that our model contains the variables cox_s, fox_s, pox_s in excess. These variables denote a hypothetical process that we called secondary oxidation to represent many omitted processes in the body. We need this hypothetical process in order to make the model to work realistically when the stocks of fat or protein are too low to generate the desired oxidation rates.

4.2.1. Resting Metabolic Rate

The resting metabolic rate is the energy requirement of an awaken person in resting conditions. It contains energy expenditure during basic movement activities like sitting, walking a few steps and eating. Approximately, resting energy expenditure makes up to 60-70 percent of total energy expenditure.

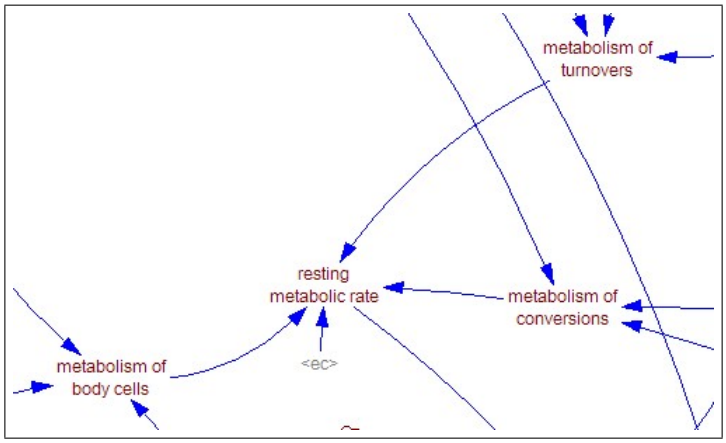

Figure 4.5. Causal Diagram of Resting Metabolic Rate

The equation for resting metabolic rate is given in Equation 4.3 (adopted from Hall's model except the arbitrary variable ec that exists in Hall's model which is excluded from our model). In this equation, resting metabolic rate consists of three terms: metabolism of body cells, metabolism of turnovers, metabolism of conversions.

$$rmr = mbc + mt + mc \tag{4.3}$$

In Section 4.1.1, we say that the adaptive thermogenesis is the fourth factor that effects the resting metabolic rate apart from the summation terms. In mathematical model, this effect is contained in the metabolism of body cells which is the major component of the resting metabolic rate.

As stated in Section 4.1.1, the metabolism of conversions covers all the metabolic

activities that involve conversion of an energy macronutrient (carbohydrate, fat or protein) to another one. The metabolism of turnovers includes all the energy requiring metabolic activities that construct or deconstruct macronutrients from their building stones. The metabolism of body cells includes all the energy requiring metabolic activities to maintain living conditions such as active membrane transportation or cellular division. There is a partial overlap between these three variables. Metabolism of conversions and turnovers are actually parts of the metabolism of body cells. But due to the benefits in modeling, the model separates these two variables from the metabolism of body cells.

The metabolism of body cells covers all the metabolic activities inside the body cells. In most of the cells, there is also some amount of conversion or turnover of macronutrients. So, metabolism of body cells actually contain in itself metabolism of turnovers and conversions. In order for these two concepts to be totally independent of each other, the metabolism of body cells should be defined in such a way that it does not include any metabolic conversion or turnover activity. This is very difficult due to data measurement problems in micro scale. Therefore our model accepts double counting as a bearable side effect of having the metabolism of conversions and turnover variables as separate entities from the metabolism of body cells.

4.2.2. Metabolism of Conversions Sector

Figure 4.2 shows the metabolism of conversions as a single aggregate variable. This is actually a simplification of the variables in our model. Figure 4.6 is a more detailed view of the model. In this diagram, conversions between the stocks of macronutrients are clearly shown. There are three flow variables representing the conversions: dnl, gng fat, and gng protein. The variable dnl is De Novo Lipogenesis which means synthesis of fat from carbohydrate. The variable gng is gluconeogenesis which means generation of glucose from non-carbohydrate substances. There are two forms of gng: gng protein and gng fat. The variable gng protein is the conversion of protein molecules into carbohydrates, and gng fat is the conversion of fat into carbohydrate.

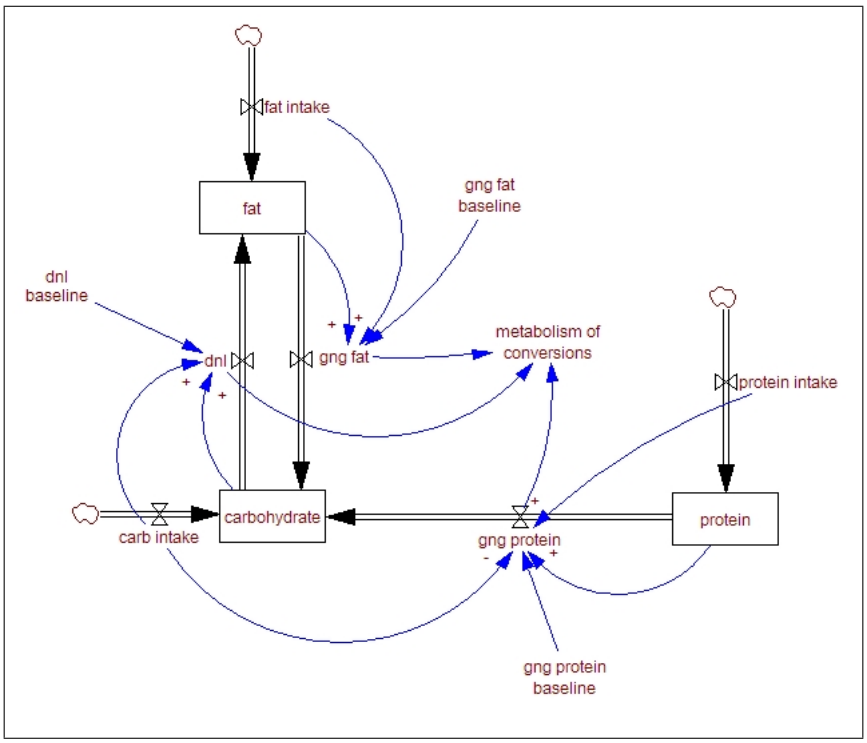

Figure 4.6. Simplified stock flow diagram of conversions sector

All the conversion flows have the same causality structure with the exception of gng protein. The amount of conversion from one macronutrient stock to another is a positive linear function of the origin stock and food intake. For example, gng fat is the flow from fat stock to carbohydrate stock. The amount of this flow is linearly proportional to amounts of fat stock and fat intake inflow. The reason behind this causal model is that when there is more macronutrient then there will be more conversion of it to another form of macronutrient. The model assumes that the relationship is linear because there is no evidence to suggest a non-linear relationship. The baseline values are the values of the variables that determine the metabolic characteristics of the person at time zero. In reality, baseline values change depending on the unique characteristics of the person. Our model does not calibrate all of the baseline values based on them. It presumes constant average baseline values based on the physiology literature. Nonetheless, some of the baseline values are calibrated for the simulated person. They are described in detail in Section 4.3.

All conversion variables are positive, linear functions of the stock and food intake

variables of the macronutrient under consideration. The only exception to this causal structure is gng protein. The variable gng protein depends on carbohydrate intake in addition to its own stock (protein) and food intake (protein intake). Carbohydrate intake has a protein sparing effect. Therefore carbohydrate intake has a negative effect on gng protein [7].

Figure 4.3 shows detailed causality structure of the conversion variables. This has some differences from the simplified diagram in Figure 4.6. Figure 4.7 is extracted out of Figure 4.3 where only variables directly effecting metabolism of conversions are shown. This diagram is provided in order to highlight the correspondence of simplified stock flow diagram in Figure 4.6 with the actual stock flow diagram of the model in Figure 4.3.

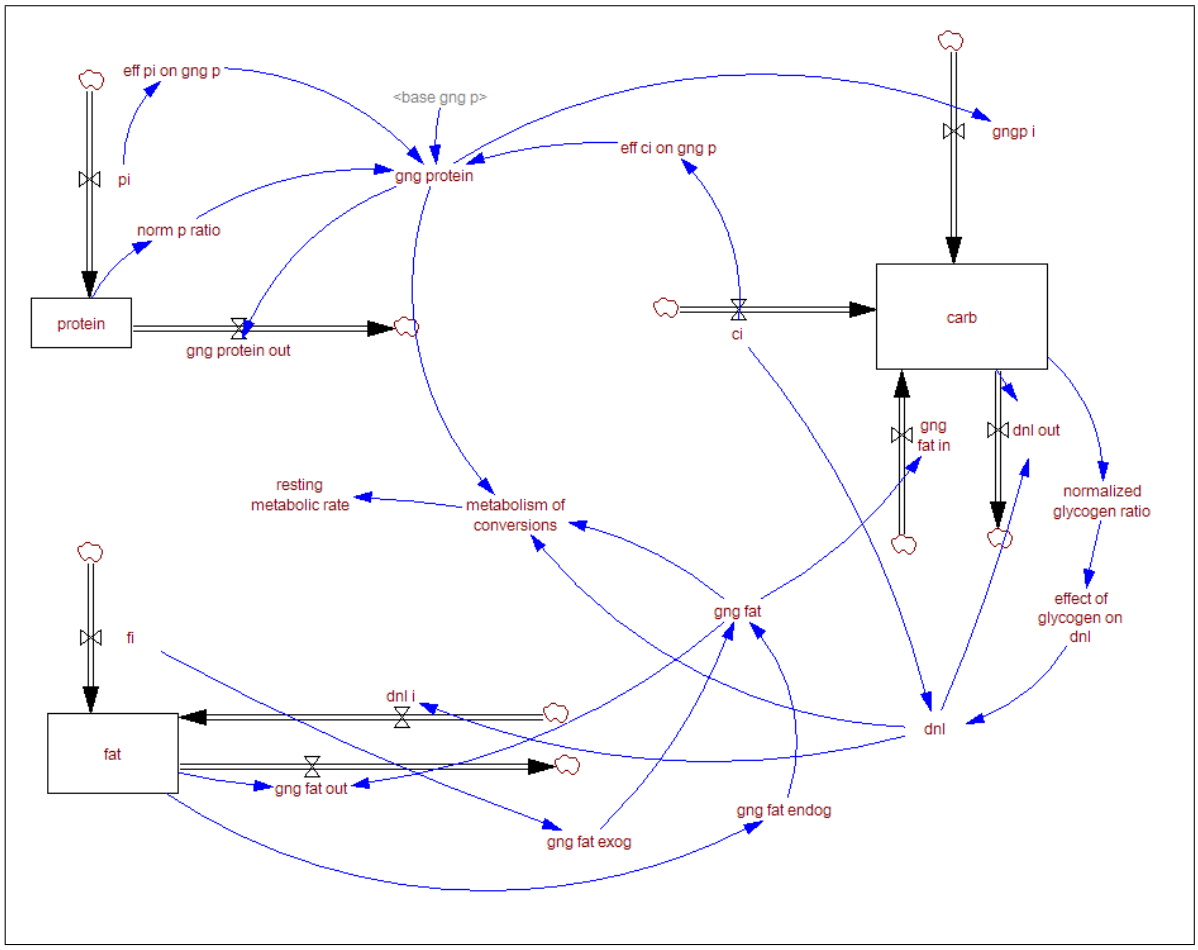

Figure 4.7. Conversions sector in detail

An important difference between the simplified version and the actual version of the stock flow model is that conversion flows between the stocks are disconnected from either the origin or destination stock. For example, gng fat is a flow from the fat stock to the carbohydrate stock in our model. But in actual model, there is an outflow "gng fat out" coming out of fat stock and an inflow "gng fat in" going into carb stock. We have to disconnect conversion flows in actual model because the units of the stocks are not equivalent. Carb stock has the unit "gram of carbohydrate", fat stock has the unit "gram of fat", and protein stock has the unit "gram of protein". Therefore, the fat amount coming out of fat stock through gng fat out cannot be the same amount of grams accumulating into carbohydrate stock through gng fat in.

The equations for the variables in the conversion sector are given in Table 4.1 in alphabetical order.

Table 4.1. Code for the variables in the conversions sector (Equations are adopted from Hall's model. Formulations of the variables look different but explicit mathematical forms are the same.)

```
dnl=ci * effect_of_glycogen_on_dnl * cal_dens_c
dnl_i=dnl / cal_dens_f
dnl_out=min( dnl / cal_dens_c , carb / time_step )
eff_ci_on_gng_p=coef_for_eff_ci_on_gng_p * normalized_change_in_
    carbohydrate_intake
eff_pi_on_gng_p=coef_for_eff_pi_on_gng_p * normalized_change_in_
    protein_intake
effect_of_glycogen_on_dnl=normalized_glycogen_ratio ** hill_dnl / (
    k_dnl ** hill_dnl + normalized_glycogen_ratio ** hill_dnl )
gng_fat=gng_fat_endog + gng_fat_exog
gng_fat_endog=baseline_gng_fat_endog * effect_of_carb_intake_on_
    lipolysis * effect_of_obesity_on_lipolysis
gng_fat_exog=exog_glycerol_per_kcal_fat_intake * fat_intake * cal_
    dens_c
gng_fat_in=gng_fat / cal_dens_c
gng_fat_out=min( gng_fat / cal_dens_f , fat / time_step )
gng_protein=max( 0.0 , base_gng_p * ( norm_p_ratio - eff_ci_on_gng_p
     + eff_pi_on_gng_p ) )
gng_protein_out=max( 0.0 , gng_protein / cal_dens_p)
gngp_i=gng_protein / cal_dens_c
metabolism_of_conversions=( 1.0 - efficiency_dnl ) * dnl + ( 1.0-
    efficiency_gng ) * ( gng_fat + gng_protein )
norm_p_ratio=protein / baseline_protein
normalized_glycogen_ratio=carb / base_carb
```

The final outcome of the conversion sector is the metabolism of conversions. This is defined in Equation 4.4 (Adopted from Hall's model)

$$mc = (1.0 - eff_{dnl}) \cdot dnl + (1.0 - eff_{gng}) \cdot (gng_f + gng_p) \tag{4.4}$$

The metabolism of conversions is the energy expenditure rate during the metabolic activities of macronutrient conversions. There are three macronutrient conversions: dnl (carbohydrate to fat), gng fat (fat to carbohydrate) and gng protein (protein to carbohydrate). The efficiency of dnl is 0.8 ([8]). The efficiency of gng fat and gng protein is 0.8 ([9]).

The variables that gng protein depend on are defined in Table 4.2.

Table 4.2. Code for the variables that gng protein depend on (Adopted from Hall's model)

```
gng_protein=max( 0.0 , base_gng_p * ( norm_p_ratio - eff_ci_on_gng_p
    + eff_pi_on_gng_p ) )
base_gng_p=100.0
norm_p_ratio=protein / baseline_protein
eff_ci_on_gng_p=coef_for_eff_ci_on_gng_p * normalized_change_in_
   carbohydrate_intake
normalized_change_in_carbohydrate_intake=change_in_carbohydrate_
   intake / baseline_carbohydrate_intake
eff_pi_on_gng_p=coef_for_eff_pi_on_gng_p * normalized_change_in_
   protein_intake
normalized_change_in_protein_intake=change_in_protein_intake /
   baseline_protein_intake
```

A more condensed mathematical definition for gng protein is given in Equation 4.5 (adopted from Hall [1]).

$$gng_p = gng_{p,b}\left[\frac{p}{p_b} - \Gamma_c\left(\frac{\Delta ci}{ci_b}\right) + \Gamma_p\left(\frac{\Delta pi}{pi_b}\right)\right] \tag{4.5}$$

The mapping between symbolic names in condensed mathematical form and long names in detailed form is given in Table 4.3.

Table 4.3. Mapping of symbols and long names for Equation 4.5

symbol	long name
gng_p	gng_protein
$gng_{p,b}$	base_gng_p
p	protein
p_b	baseline_protein
Γ_c	coef_of_eff_ci_on_gng_p
Δci	change_in_carbohydrate_intake
ci_b	baseline_carbohydrate_intake
Γ_p	coef_of_eff_pi_on_gng_p
Δpi	change_in_protein_intake
pi_b	baseline_protein_intake
$\frac{\Delta pi}{pi_b}$	normalized_change_in_protein_intake
$\frac{\Delta ci}{ci_b}$	normalized_change_in_carbohydrate_intake

There are two special parameters: Γ_c and Γ_p. These parameters are constant values that represent coefficients of the effects of carbohydrate and protein intakes on gng protein. Hall determines their values by solving Equation 4.5 using two sets of data. Our model uses these values.

Hall estimates the value for baseline gng protein (base_gng_p) as 100 kcal/day. This estimation is based on a measurement study with multiple isotopic tracer methodology of Nurjhan 1995 [10].

The condensed mathematical definition for gng fat in symbolic notation is given in Equation 4.6 (adopted from Hall [1]).

$$gng_f = fi\left(\frac{cal_c m_g}{cal_f m_{tg}}\right) + gng_{f,end,b} \cdot eff_{ci,d_f} \cdot eff_{obes,d_f} \tag{4.6}$$

The mapping between symbolic names in condensed mathematical form defined in

Equation 4.6 and long names in detailed form defined in Table 4.1 is given in Table 4.4.

Table 4.4. Mapping of symbols and long names for Equation 4.6

gng_f	gng_fat
fi	fat_intake
d_f	lipolysis
$gng_{f,end,b}$	baseline_gng_fat_endog
eff_{ci,d_f}	effect_of_carb_intake_on_lipolysis
eff_{obes,d_f}	effect_of_obesity_on_lipolysis

The condensed mathematical definition for gng fat in symbolic notation is given in Equation 4.7 (adopted from Hall [1]).

$$dnl = \frac{ci(c/c_b)^{hill_{dnl}}}{(c/c_b)^{hill_{dnl}} + k_{dnl}^{hill_{dnl}}} \tag{4.7}$$

The function of dnl is formulated as a hill function of c. The parameters $hill_{dnl}$ and k_{dnl} do not have any physiological meaning. Their goal is to make the function have a sigmoidal shape.

The mapping between symbolic names in condensed mathematical form defined in Equation 4.7 and long names in detailed form defined in Table 4.1 is given in Table 4.5.

Table 4.5. Mapping of symbols and long names for Equation 4.7

dnl	dnl
ci	carbohydrate_intake
c	carb
c_b	base_carb
$hill_{dnl}$	hill_dnl
k_{dnl}	k_dnl

4.2.3. Metabolism of Turnovers Sector

The metabolism of turnovers contains the energy expenditure for all the energy requiring metabolic activities that construct or deconstruct macronutrients from their building stones. Deconstruction (catabolism) of fat and carbohydrate does not require energy expenditure. Synthesis (construction) of all macronutrients and deconstruction of protein require energy expenditure.

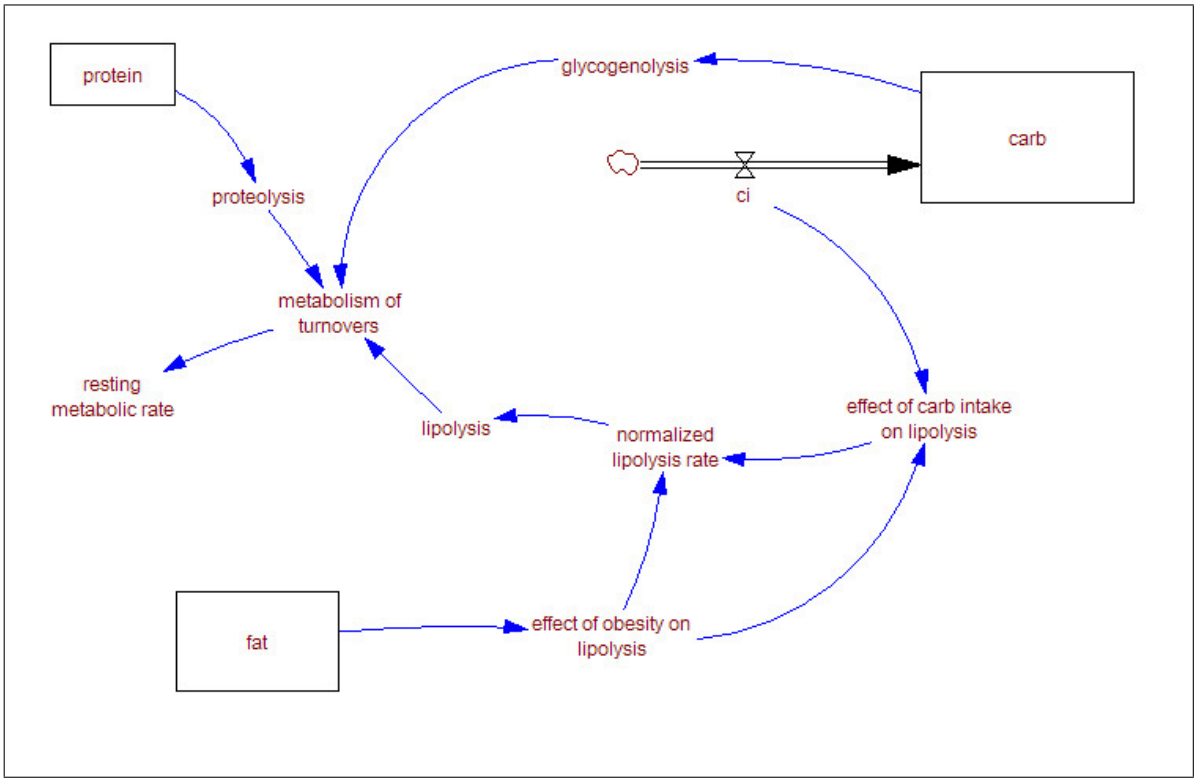

Figure 4.8. Metabolism of turnovers sector in detail

The equations for the variables in the metabolism of turnovers sector are given in Table 4.6. The only difference from Hall's model is that our model assumes that the daily amount of synthesized macronutrients is equal to the daily rate of deconstructed macronutrients. This assumption is true mostly when the stock levels of macronutrients don't change. But when there is a change in the stock levels, this assumption is not valid. However, the error is very low compared to the total amount of synthesis.

If we leave the assumption out, then the model becomes very complex to analyze and understand. Therefore, we prefer keeping the assumption.

Table 4.6. Code for the variables in the metabolism of turnovers sector (Equations are adopted from Hall's model. Formulations of the variables are different but explicit mathematical forms are the same.)

```
metabolism_of_turnovers=( deg_cost_p + dep_cost_p ) * proteolysis +
   dep_cost_fat * lipolysis +dep_cost_c * glycogenolysis
glycogenolysis=base_deg_c * carb / base_carb
lipolysis= baseline_lipolysis * effect_of_carb_intake_on_lipolysis *
    effect_of_obesity_on_lipolysis
baseline_lipolysis = base_molar_lipolysis * mass_tg
effect_of_carb_intake_on_lipolysis=1.0 + ( ( lipol_max - lipol_min )
     * exp( -k_lip * ci * cal_dens_c / baseline_carbohydrate_intake
   )+ lipol_min - 1.0 )/ max( 1.0 , effect_of_obesity_on_lipolysis
   )
effect_of_obesity_on_lipolysis=(fat / baseline_fat)**(2.0/3.0)
baseline_fat=baseline_bodyweight - baseline_lean_mass
proteolysis=baseline_proteolysis * protein / baseline_protein
baseline_protein=cell_mass_b - icw_b - intracel_solids - base_carb
baseline_proteolysis=molar_baseline_proteolysis * aminoacid_mass
```

The parameter `deg_cost_p` denotes energy cost of degradation of protein in kcal/g. The parameters `dep_cost_c`, `dep_cost_fat`, and `dep_cost_p` denote energy costs in kcal/g of deposition (synthesis) of carbohydrate, fat, and protein. The values of these parameters are determined by Hall by using biochemical pathways [1]. The parameter `cal_dens_c` denotes caloric density of carbohydrate in kcal/g. The parameters `mass_tg` and `aminoacid_mass` denote molar masses of triglyceride and aminoacids in g/mol.

The parameters `lipol_max`, `lipol_min` are coefficients that are related to the effect of carbohydrate intake on lipolysis. The parameter `base_deg_c` denotes baseline glycogen degradation rate. Hall determines its value by assuming that 70% of total glycogenolysis comes from hepatic glycogenolysis and 30% from skeletal muscle, and by

using hepatic glycogenolysis rate found by Magnusson, 1994 [11]. The parameter `base_carb` denotes baseline carbohydrate stock. The parameter `base_molar_lipolysis` denotes baseline molar lipolysis rate. Its value is found by Jensen, 1999 [12]. The parameter `molar_baseline_proteolysis` denotes baseline molar proteolysis rate. Its value is found by Wagenmakers, 1999 [13].

The condensed mathematical definition for proteolysis (d_p) in symbolic notation is given in Equation 4.8 (adopted from Hall [1]).

$$d_p = d_{p,b}\frac{p}{p_b} \tag{4.8}$$

The mapping between the symbolic names in condensed mathematical form defined in Equation 4.8 and the long names in detailed form defined in Table 4.6 is given in Table 4.7.

Table 4.7. Mapping of symbols and long names for Equation 4.8

d_p	`proteolysis`
$d_{p,b}$	`baseline_proteolysis`

The condensed mathematical definition for lipolysis (d_f) in symbolic notation is given in Equation 4.9 (adopted from Hall [1]).

$$d_f = d_{f,b} \cdot eff_{obes,d_f} \cdot eff_{ci,d_f} \tag{4.9}$$

The mapping between the symbolic names in the condensed mathematical form defined in Equation 4.9 and the long names in the detailed form defined in Table 4.6 is given in Table 4.8.

Table 4.8. Mapping of symbols and long names for Equation 4.9

d_f	`lipolysis`
eff_{obes,d_f}	`effect_of_obesity_on_lipolysis`
eff_{ci,d_f}	`effect_of_carb_intake_on_lipolysis`

The condensed mathematical definition for glycogenolysis (d_c) in symbolic notation is given in Equation 4.10 (adopted from Hall [1]).

$$d_c = d_{c,b} \cdot \frac{c}{c_b} \tag{4.10}$$

The mapping between the symbolic names in the condensed mathematical form defined in Equation 4.10 and the long names in the detailed form defined in Table 4.6 is given in Table 4.9.

Table 4.9. Mapping of symbols and long names for Equation 4.10

d_c	`glycogenolysis`
$d_{c,b}$	`base_deg_c`

4.2.4. Metabolism of Body Cells Sector

The equations for the variables in the metabolism of body cells sector are given in Table 4.10.

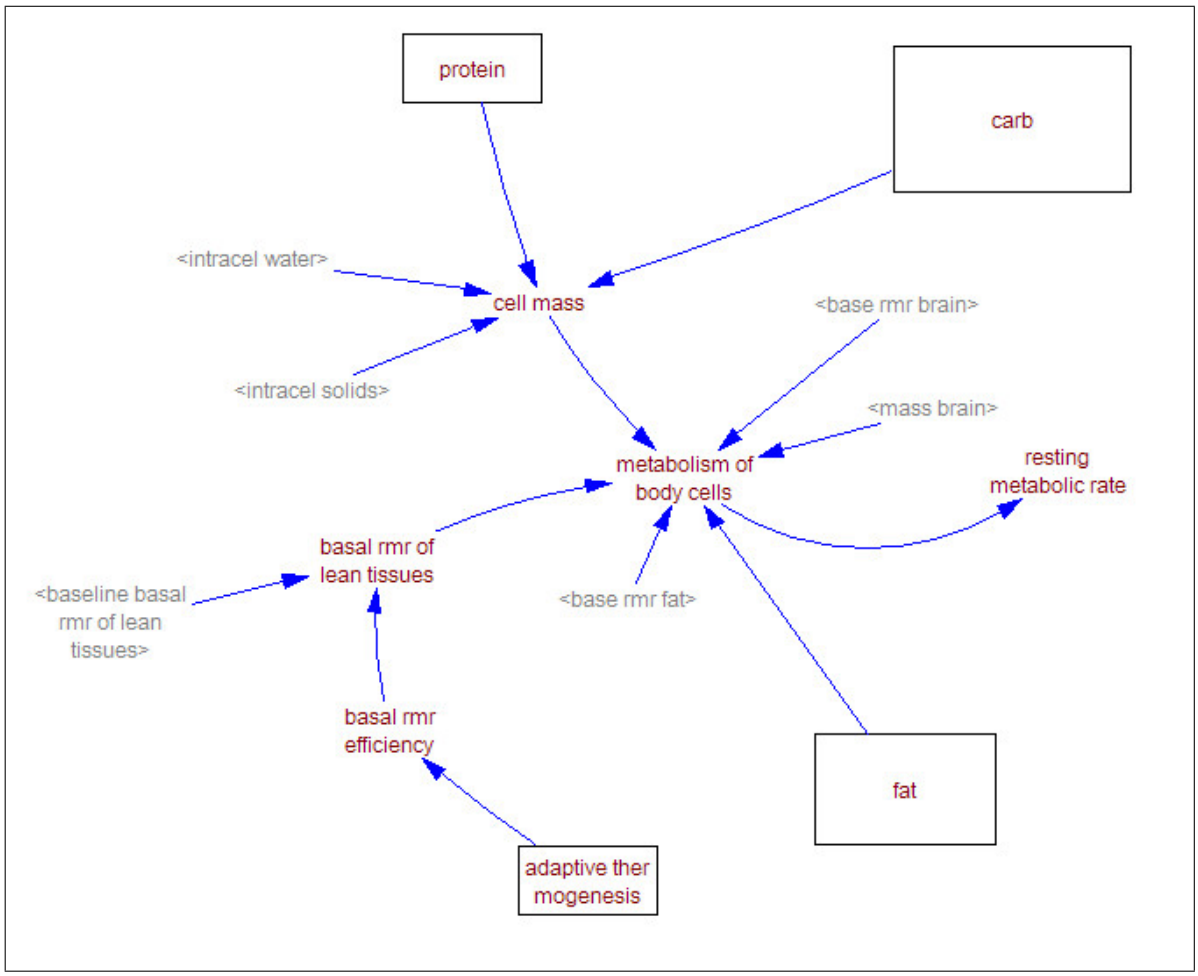

Figure 4.9. Metabolism of body cells sector in detail

Table 4.10. Code for the variables in the metabolism of body cells sector (Equations are adopted from Hall's model. Formulations of the variables look different but explicit mathematical forms are the same.)

```
metabolism_of_body_cells=mbc_lean + mbc_brain + mbc_fat
mbc_lean=basal_rmr_of_lean_tissues * ( cell_mass_b - mass_brain )
mbc_fat=base_rmr_fat * baseline_fat
mbc_brain=base_rmr_brain * mass_brain
basal_rmr_of_lean_tissues=baseline_basal_rmr_of_lean_tissues * basal
    _rmr_efficiency
basal_rmr_efficiency=1.0 + ( 1.0 - thermogenesis_effect_on_pae_vs_
    rmr ) * adaptive_thermogenesis
cell_mass=intracel_solids + carb + protein + intracel_water
ciw=fiew * baseline_ecw - ( glycogen_hydration_coefficient * c_init
    + protein_hydration_coefficient* p_init)
baseline_ecw=fw * fetw * baseline_bodyweight
```

The parameters `base_rmr_brain`, `base_rmr_fat`, and `baseline_basal_rmr_of_lean_tissues` are metabolic rates of brain, fat, and lean tissues in $kcal/(kg \cdot day)$. The parameter `mass_brain` denotes the mass of the brain in gram. The parameter `intracel_solids` denotes the mass of intracellular solids such as nucleic acids. The parameter `thermogenesis_effect_on_pae_vs_rmr` denotes the proportion of adaptive thermogenesis that is allocated to the change in physical activity energy. The remaining portion of the adaptive thermogenesis is allocated to the change in the resting metabolic rate. The value of this parameters is calculated from the measured resting metabolic rate data from the Minnesota Experiment [1]. The parameters `glycogen_hydration_coefficient` and `protein_hydration_coefficient` denote amount of water per each gram of carbohydrate and protein stored in the body.

The parameter `f_{tw,bw}` denotes the ratio of water to the total body weight. The parameter `fetw` denotes the ratio of extracellular water to total water inside the body. The variable `fiew` denotes the ratio of intracellular water to extracellular water.

The parameter `baseline_bodyweight` is the baseline body weight. The parameter `p_init` is the initial value of the protein stock. The values of these parameters are customized for each game player in the game version of the model.

4.2.5. Body Weight Sector

The equations for the variables in the body weight sector are given in Table 4.11.

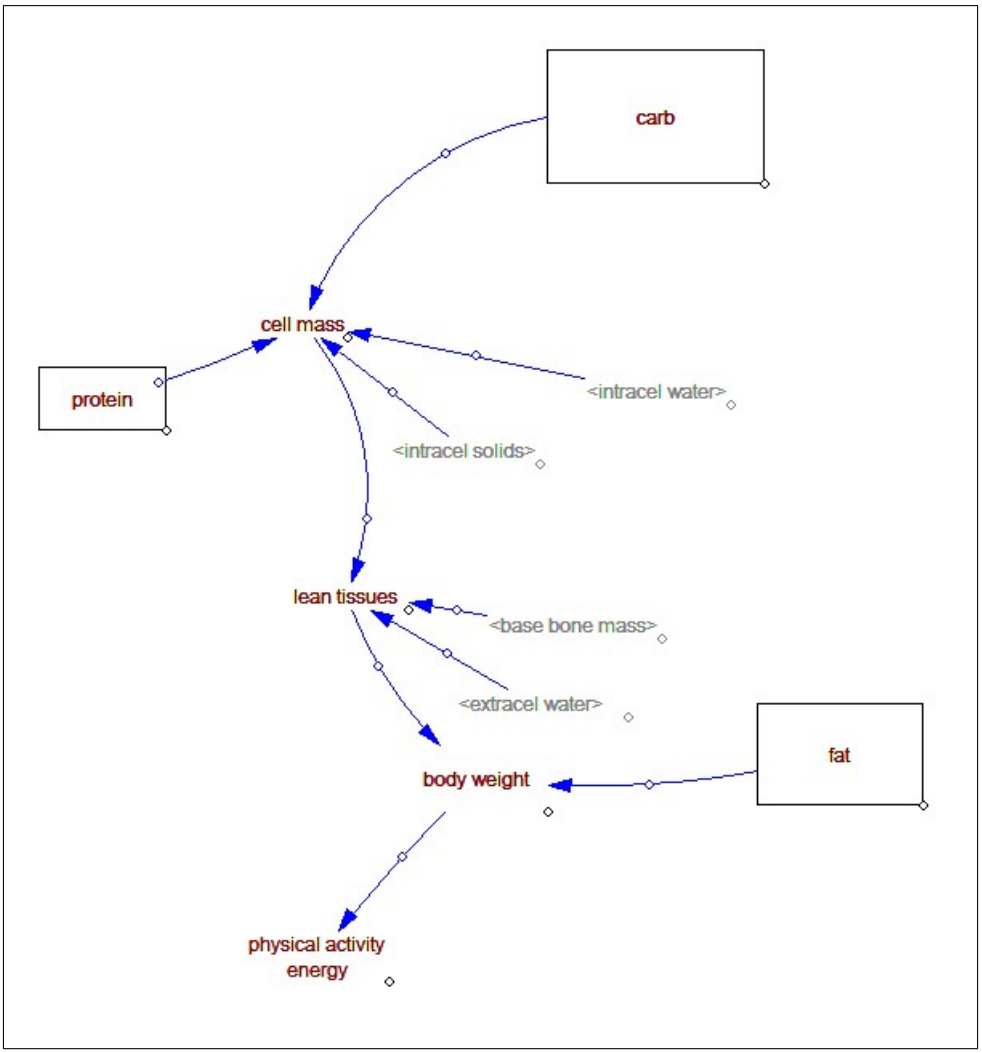

Figure 4.10. Body weight sector in detail

Table 4.11. Code for the variables in the body weight sector (Equations are adopted from Hall's model. Formulations of the variables look different but explicit mathematical forms are the same.)

```
body_weight=fat + lean_tissues
lean_tissues=base_bone_mass + extracel_water + cell_mass
base_bone_mass=0.04 * baseline_bodyweight
baseline_bodyweight=70000.0
cell_mass=intracel_solids + carb + protein + intracel_water
intracel_solids=3967.28
intracel_water=glycogen_hydration_coefficient * carb + protein_
   hydration_coefficient * protein+ ciw
glycogen_hydration_coefficient=2.7
protein_hydration_coefficient=2.0
ciw=fiew * baseline_ecw - ( glycogen_hydration_coefficient * c_init
   + protein_hydration_coefficient* p_init)
fiew=(1.0-fetw)/fetw
baseline_ecw=fw * fetw * baseline_bodyweight
extracel_water=baseline_ecw
baseline_ecw=fw * fetw * baseline_bodyweight
```

The parameter `ciw` denotes the amount of the constant intracellular water. This parameter is used to hold the equality of the body weight calculated from the components with the body weight given as an external parameter. The parameter `baseline_ecw` denotes the baseline amount of the extracellular water. The parameter `extracel_water` denotes the current amount of the extracellular water. In this model, we assumed them to be equal. Hall used measured data from the Minnesota Experiment for the value of the current level of the extracellular water. The parameter `base_bone_mass` denotes the baseline bone mass in grams.

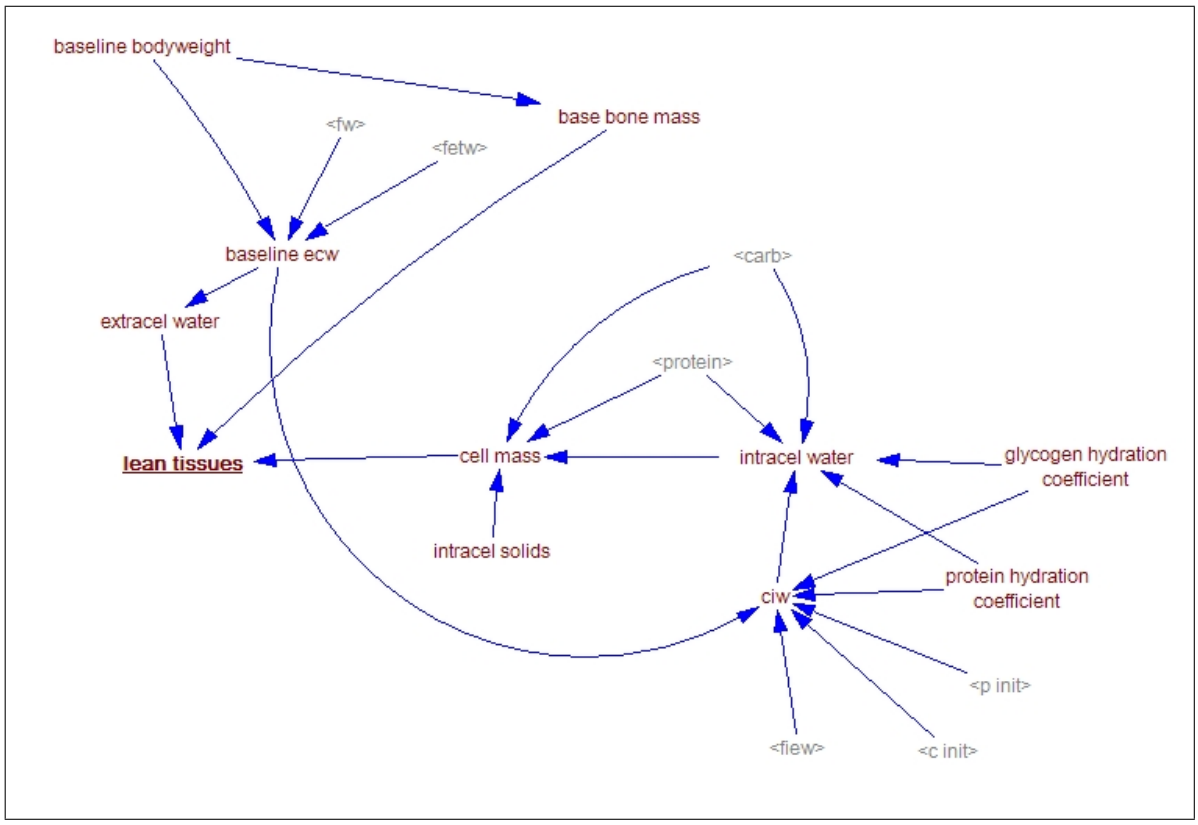

Figure 4.11. Dependencies of the lean tissues variable

The equations for the variables that depend on the lean tissues are given in Table 4.12.

Table 4.12. Code for the variables that lean tissues depend on (Adopted from Hall's model)

```
lean_tissues=base_bone_mass + extracel_water + cell_mass
base_bone_mass=0.04 * baseline_bodyweight
extracel_water=baseline_ecw
cell_mass=intracel_solids + carb + protein + intracel_water
baseline_ecw=fw * fetw * baseline_bodyweight
fw=7.0/10.0
fetw=3.0/8.0
intracel_solids=3967.28
intracel_water=glycogen_hydration_coefficient * carb + protein_
   hydration_coefficient * protein+ ciw
glycogen_hydration_coefficient=2.7
protein_hydration_coefficient=2.0
ciw=fiew * baseline_ecw - ( glycogen_hydration_coefficient * c_init
   + protein_hydration_coefficient* p_init)
fiew=(1.0-fetw)/fetw
```

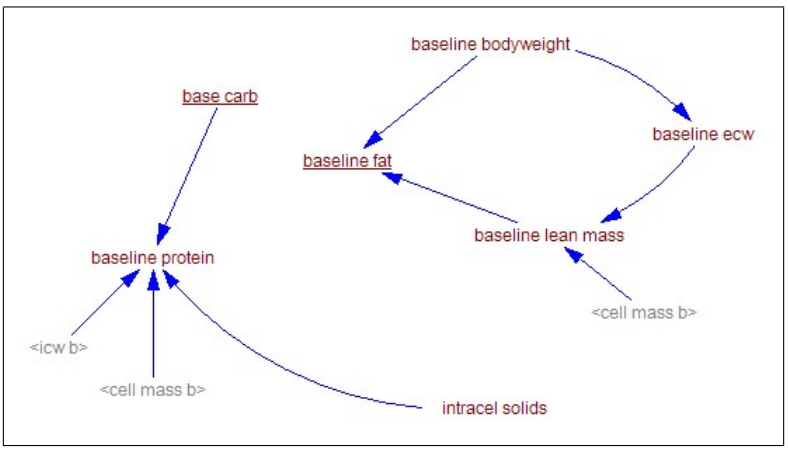

Figure 4.12. Dependencies of baseline stock values

The equations for the variables that the baseline stock parameters, `baseline_protein`, `baseline_fat`, `base_carb`, depend on are given in Table 4.13.

Table 4.13. Code for the variables that baseline stock parameters depend on (Adopted from Hall's model)

```
baseline_protein=cell_mass_b - icw_b - intracel_solids - base_carb
base_carb=400.0
intracel_solids=3967.28
cell_mass_b=1.0/fwcm * icw_b
icw_b=fiew * baseline_ecw
baseline_fat=baseline_bodyweight - baseline_lean_mass
baseline_bodyweight=70000.0
baseline_lean_mass=base_bone_mass + baseline_ecw + cell_mass_b
baseline_ecw=fw * fetw * baseline_bodyweight
```

4.2.6. Oxidation Sector

Figure 4.13 shows that the oxidation outflows have almost the same causality structure. They depend on the origin stock and food intake variables. The fat oxidation is the only exception. It depends only on the fat stock not on the fat intake [14], [15]. The oxidation functions of the relationship shown in Figure 4.13 are all positive and linear. The reasoning behind this causality structure is similar to the reasoning with the conversion flows. When there is more substance in the stocks or inflows, then the body tries to use more of them.

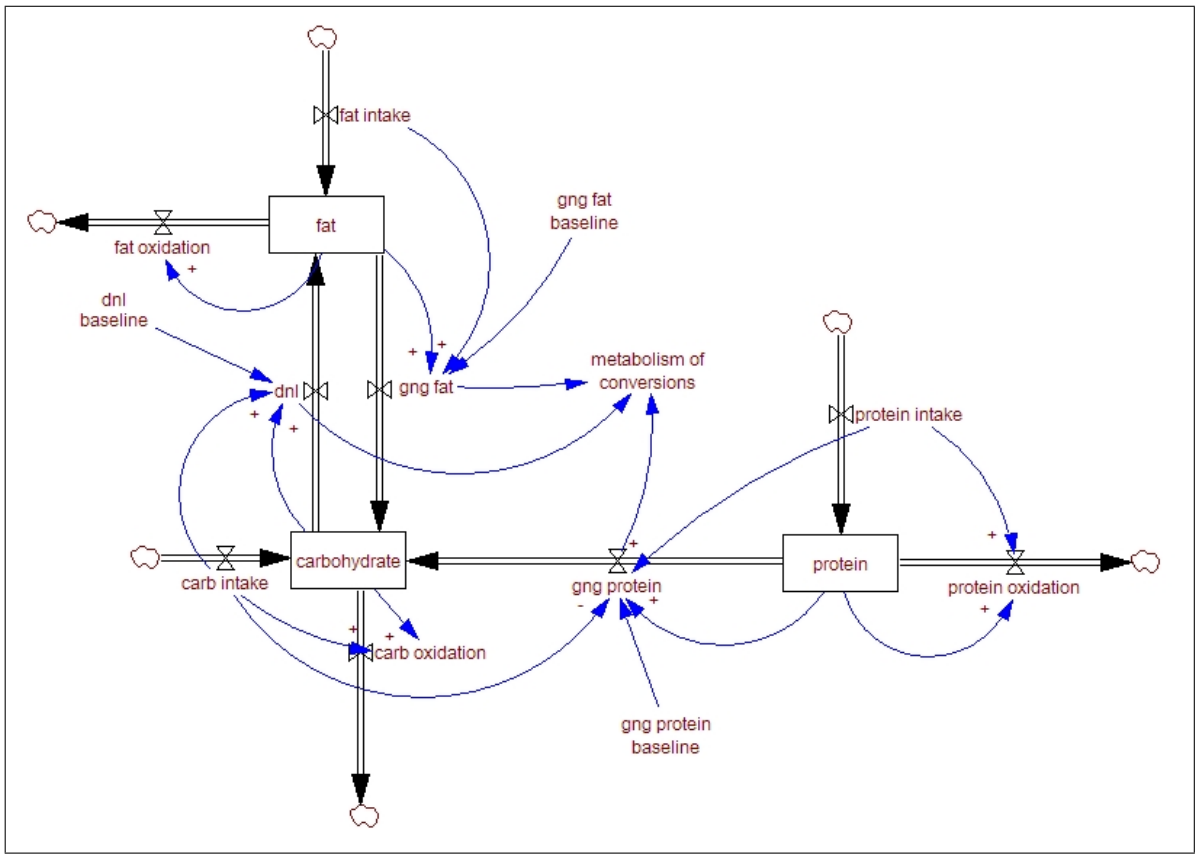

Figure 4.13. Simplified stock flow diagram of oxidation sector

Figure 4.13 is actually a simplified representation of the causality structure of the oxidation variables in our model. This stock flow diagram does not show the secondary oxidation variables used in the model. These will be explained in Section 4.2.6.

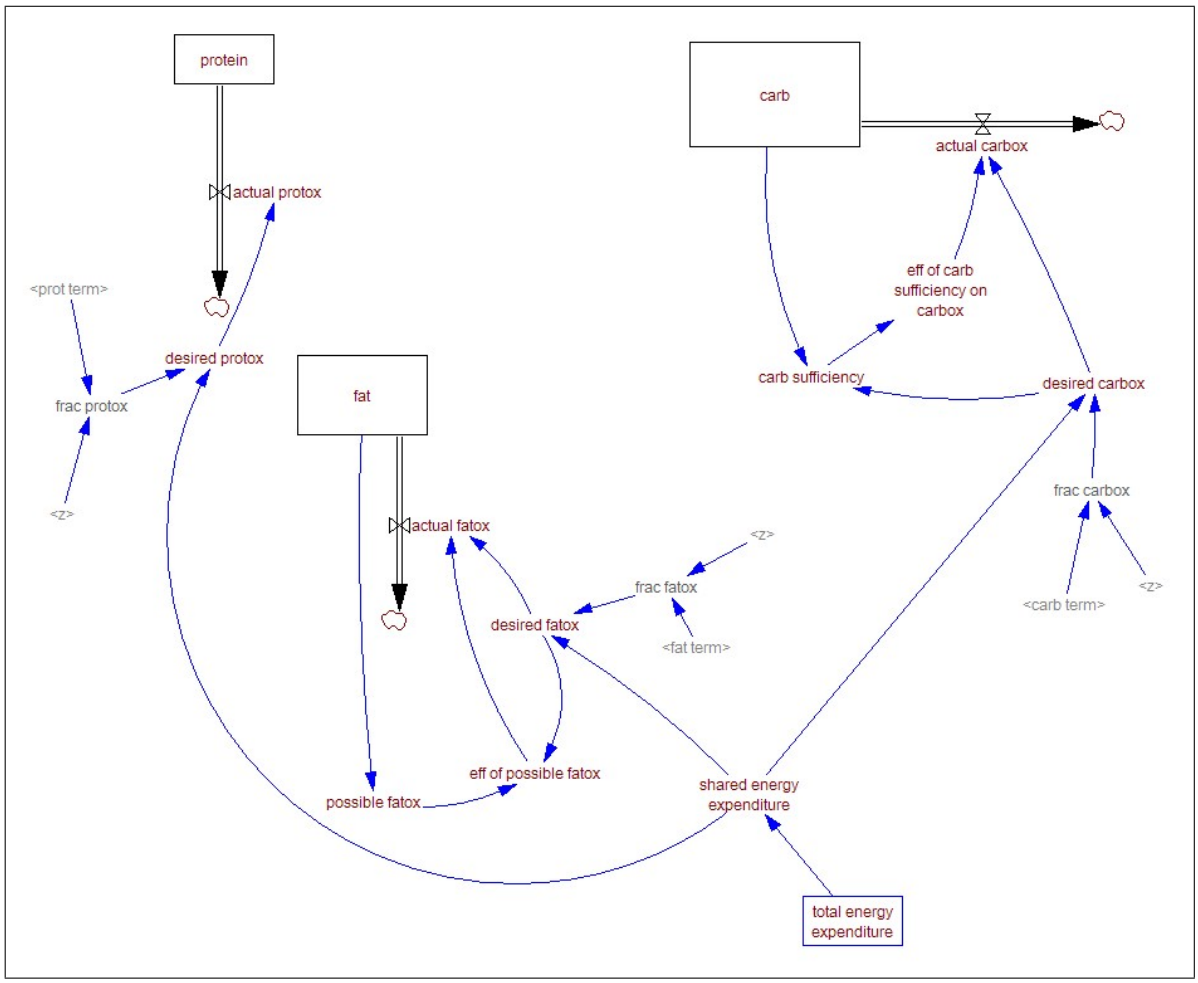

Figure 4.14. Primary oxidation subsector in detail

The condensed mathematical definition for the fraction of protein oxidation (f_p) in symbolic notation is given in Equation 4.11 (adopted from Hall [1]).

$$f_p = \frac{(n_{d_p} + max(0.0, eff_{pi,pox})) \cdot eff_{act,pox}}{z} \tag{4.11}$$

The effect of protein intake on protein oxidation, $eff_{pi,pox}$, is defined in Equation 4.12 (adopted from Hall [1]).

$$eff_{pi,pox} = w_{pi} \cdot (1.0 + s_{pi,ox} \cdot n_{\Delta pi}) \tag{4.12}$$

The effect of physical activity on protein oxidation ($eff_{act,pox}$) is defined in Equation 4.13 (adopted from Hall [1]).

$$eff_{act,pox} = sa \cdot e^{-\ln(sa) \cdot act_{eng,bw}/act_{eng,bw,b}} \tag{4.13}$$

The mapping between the symbolic names in the condensed mathematical form defined in Equation 4.11, 4.12, 4.13 and the long names in the detailed form defined in Table 4.17 is given in Table 4.14.

Table 4.14. Mapping of symbols and long names for Equation 4.11

f_p	`frac_protox`
n_{d_p}	`normalized_proteolysis`
$eff_{pi,pox}$	`eff_of_prot_intake_on_prot_oxidation`
$eff_{act,pox}$	`eff_of_physical_activity_on_prot_oxidation`
z	`z`
w_{pi}	`weighting_of_oxidation_for_basal_pi`
$s_{pi,ox}$	`sensitivity_of_oxidation_to_pi_changes`
$n_{\Delta pi}$	`normalized_change_in_protein_intake`
s_{pae}	`sensitivity_to_physical_activity`
$act_{eng,bw}$	`activity_energy_per_body_weight`
$act_{eng,bw,b}$	`baseline_activity_energy_per_body_weight`

The condensed mathematical definition for the fraction of fat oxidation (f_f) in symbolic notation is given in Equation 4.14 (adopted from Hall [1]).

$$f_f = \frac{w_f \cdot eff_{ci,d_f} \cdot eff_{obes,d_f}}{z} \tag{4.14}$$

The mapping between symbolic names in condensed mathematical form defined in Equation 4.14 and long names in detailed form defined in Table 4.17 is given in

Table 4.15.

Table 4.15. Mapping of symbols and long names for Equation 4.14

f_f	`frac_fatox`
w_f	`weighting_of_oxidation_for_lipolysis`

The fraction of carbohydrate oxidation (f_c) is defined in Equation 4.15 (adopted from Hall [1]).

$$f_c = \frac{eff_{dg,cox} + max(0.0, eff_{ci,cox}) \cdot \frac{c}{0.1+c}}{z} \tag{4.15}$$

The effect of glycogenolysis on carbohydrate oxidation ($eff_{dg,cox}$) is defined in Equation 4.16 (adopted from Hall [1]).

$$eff_{dg,cox} = w_g \cdot n_{d_c} \tag{4.16}$$

The effect of carbohydrate intake on carbohydrate oxidation ($eff_{ci,cox}$) is defined in Equation 4.17 (adopted from Hall [1]).

$eff_{ci,cox}$ is defined in Equation 4.17

$$eff_{ci,cox} = w_{ci} \cdot (1.0 + s_{ci} \cdot n_{\Delta ci}) \tag{4.17}$$

The mapping between symbolic names in condensed mathematical form defined in Equation 4.15, 4.16, 4.17 and long names in detailed form defined in Table 4.17 is given in Table 4.16.

Table 4.16. Mapping of symbols and long names for Equation 4.15

f_c	frac_carbox
$eff_{dg,cox}$	eff_of_glycogenolysis_on_carb_oxidation
$eff_{ci,cox}$	eff_of_carb_intake_on_carb_oxidation
w_g	weighting_of_oxidation_for_glycogenolysis
w_{ci}	weighting_of_oxidation_for_basal_ci
s_{ci}	sensitivity_of_oxidation_to_ci_changes
n_{d_c}	normalized_glycogenolysis
$n_{\Delta ci}$	normalized_change_in_carbohydrate_intake

The equations for the variables in the primary oxidation subsector are given in Table 4.17.

Table 4.17. Code for the variables in the primary oxidation sector (Some equations are adopted from Hall's model.)

```
shared_energy_expenditure=total_energy_expenditure - gng_protein -
   gng_fat
desired_fatox=frac_fatox * shared_energy_expenditure
actual_fatox=desired_fatox / cal_dens_f * eff_of_possible_fatox
frac_fatox=fat_term / z
fat_term=max( 0.0 , weighting_of_oxidation_for_lipolysis * effect_of
   _carb_intake_on_lipolysis * effect_of_obesity_on_lipolysis )
possible_fatox=fat / time_step * cal_dens_f
actual_carbox=desired_carbox / cal_dens_c * eff_of_carb_sufficiency_
   on_carbox
carb_sufficiency=carb * cal_dens_c / desired_carbox
desired_carbox=frac_carbox * shared_energy_expenditure + gng_fat +
   gng_protein
frac_carbox=carb_term / z
carb_term=eff_of_glycogenolysis_on_carb_oxidation + max( 0.0 , eff_
   of_carb_intake_on_carb_oxidation* ( carb / (0.1 + carb) )  )
z=carb_term + fat_term +prot_term
actual_protox=desired_protox / cal_dens_p
desired_protox=frac_protox * shared_energy_expenditure
frac_protox=prot_term / z
prot_term=( normalized_proteolysis + max( 0.0, eff_of_prot_intake_on
   _prot_oxidation )  ) * eff_of_physical_activity_on_prot_
   oxidation
z=carb_term + fat_term +prot_term
```

Our model contains the variables cox_s, fox_s, pox_s in addition to Hall's original model. These variables denote a hypothetical process that we call secondary oxidation. We need this hypothetical process in order to make the model to work realistically when the stocks of fat or protein are so low that they cannot meet the desired oxidation rates.

If these variables are not included, then the model behaves improperly under extreme conditions. For example, Figure 4.15 shows a simulation run with very high

level of physical activity. The total energy expenditure rate is around 7000 kcal/day whereas the total energy intake is around 3500 kcal/day. In real life, the person would lose weight under these conditions very rapidly. But the person in the simulation gains weight very rapidly due to the increase in the protein stock. This is not a valid behavior.

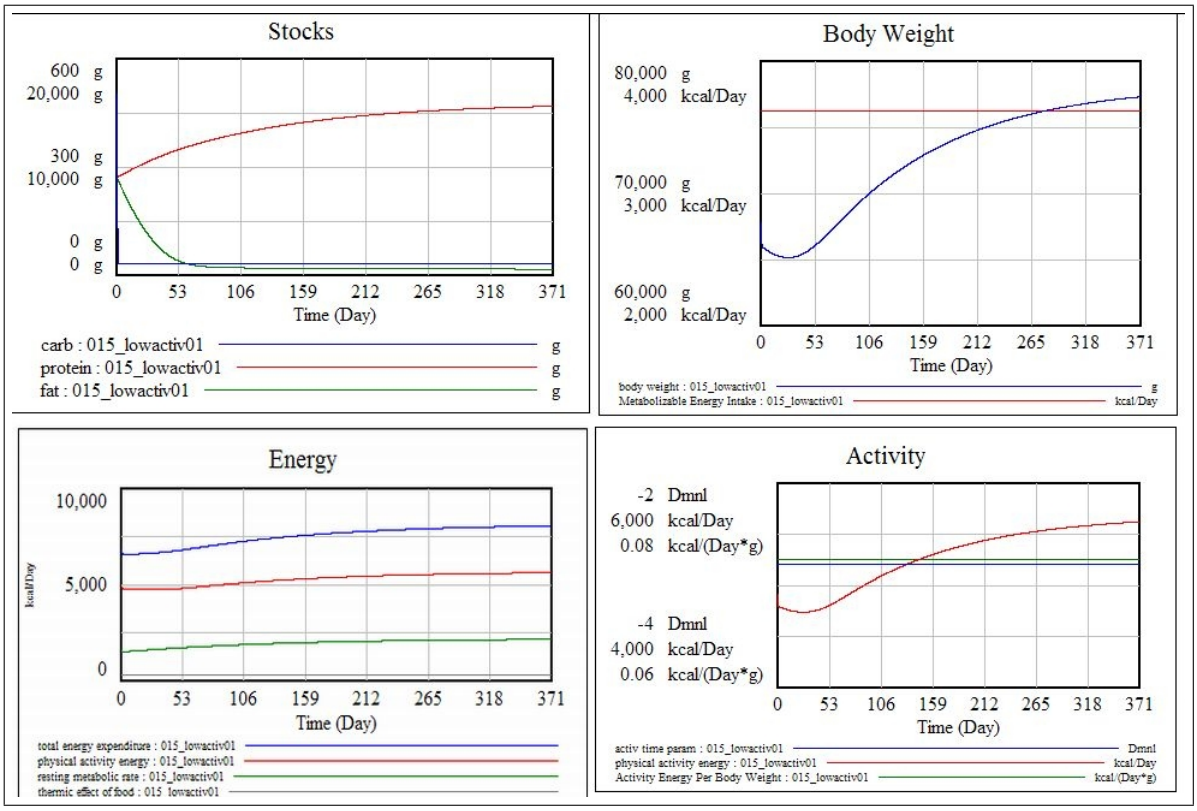

Figure 4.15. The model produces wrong behavior under extremely high activity level if the secondary oxidation rates are not included

The reason behind this wrong behavior is that the formulas for oxidation rates of the stocks does not take into account when the level of one or more of the stocks cannot meet the desired oxidation rate. In this case, the stocks of fat and carbohydrate are depleted. Therefore, their stocks cannot meet the desired oxidation rates that the model calculated for carbohydrate and fat. But the person gains weight because the desired oxidation rate for protein does not substitute when the other stocks cannot meet their desired oxidation rates.

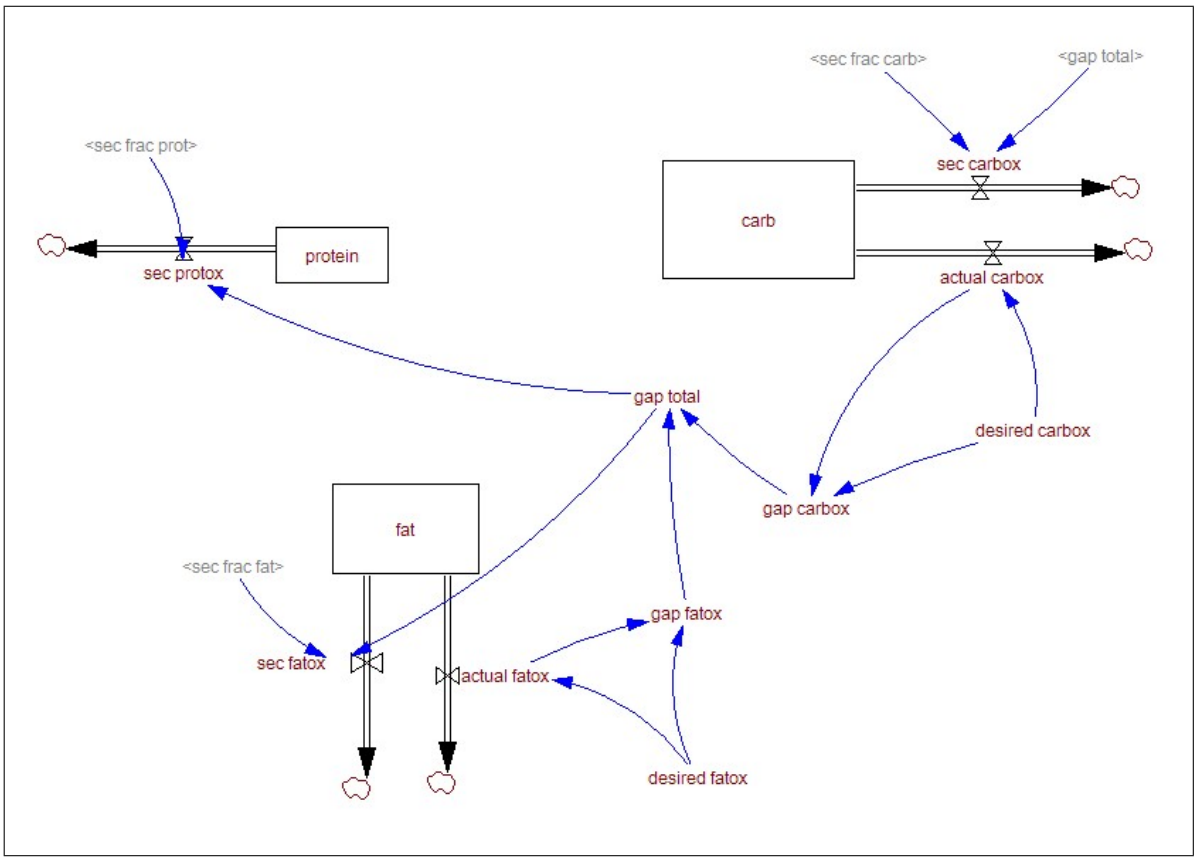

Figure 4.16. Secondary oxidation subsector in detail

The equations for the variables in the secondary oxidation subsector are given in Table 4.18 and Table 4.19. The formulas for the secondary oxidation sector handle the case when one of the stocks cannot meet its desired oxidation rate.

Table 4.18. Code for the variables in the secondary oxidation sector

```
sec_protox=sec_frac_prot * gap_total / cal_dens_p
sec_frac_prot=prot_term / y
y=prot_term + is_sufficient_carb * carb_term + is_sufficient_fat *
   fat_term
gap_total=gap_carbox + gap_fatox
is_sufficient_fat=1.0  if (gap_fatox < 0.1 ) else (0.0 )
is_sufficient_carb=1.0 if (gap_carbox < 0.1 ) else (0.0 )
gap_fatox=desired_fatox - actual_fatox * cal_dens_f
actual_fatox=desired_fatox / cal_dens_f * eff_of_possible_fatox
desired_fatox=frac_fatox * shared_energy_expenditure
sec_frac_fat=is_sufficient_fat * fat_term / y
gap_carbox=desired_carbox - actual_carbox * cal_dens_c
sec_carbox=sec_frac_carb * gap_total / cal_dens_c
sec_frac_carb=is_sufficient_carb * carb_term / y
desired_carbox=frac_carbox * shared_energy_expenditure + gng_fat +
   gng_protein
```

Table 4.19. Code for the variables that secondary oxidation fractions depend on

```
sec_frac_prot=prot_term / y
sec_frac_carb=is_sufficient_carb * carb_term / y
sec_frac_fat=is_sufficient_fat * fat_term / y
y=prot_term + is_sufficient_carb * carb_term + is_sufficient_fat *
   fat_term
is_sufficient_carb=1.0 if (gap_carbox < 0.1 ) else (0.0 )
is_sufficient_fat=1.0  if (gap_fatox < 0.1 ) else (0.0 )
gap_carbox=desired_carbox - actual_carbox * cal_dens_c
gap_fatox=desired_fatox - actual_fatox * cal_dens_f
```

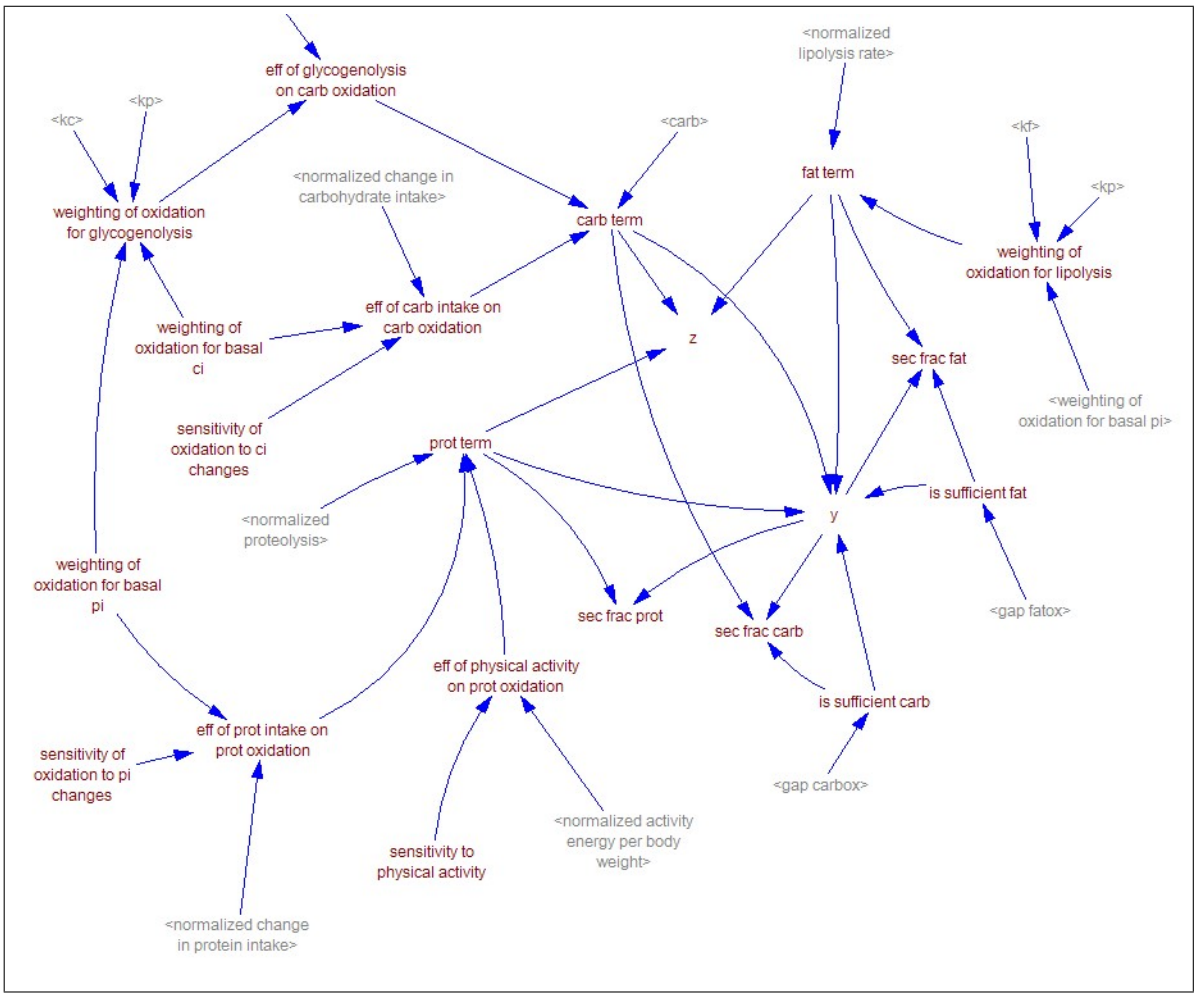

Figure 4.17. Oxidation fractions subsectorin detail

Table 4.20. Code for the variables that initial oxidation fraction parameters depend on (Adopted from Hall's model)

```
z=carb_term + fat_term +prot_term
prot_term=( normalized_proteolysis + max( 0.0, eff_of_prot_intake_on
    _prot_oxidation )  ) * eff_of_physical_activity_on_prot_
    oxidation
eff_of_physical_activity_on_prot_oxidation=sensitivity_to_physical_
    activity * exp( - log(sensitivity_to_physical_activity) *
    normalized_activity_energy_per_body_weight )
normalized_activity_energy_per_body_weight=activity_energy_per_body_
    weight / baseline_activity_energy_per_body_weight
eff_of_prot_intake_on_prot_oxidation=weighting_of_oxidation_for_
    basal_pi * ( 1.0 + sensitivity_of_oxidation_to_pi_changes*
    normalized_change_in_protein_intake )
carb_term=eff_of_glycogenolysis_on_carb_oxidation + max( 0.0 , eff_
    of_carb_intake_on_carb_oxidation* ( carb / (0.1 + carb) )  )
eff_of_carb_intake_on_carb_oxidation=weighting_of_oxidation_for_
    basal_ci * ( 1.0 + sensitivity_of_oxidation_to_ci_changes*
    normalized_change_in_carbohydrate_intake )
eff_of_glycogenolysis_on_carb_oxidation=weighting_of_oxidation_for_
    glycogenolysis * normalized_glycogenolysis
normalized_change_in_carbohydrate_intake=change_in_carbohydrate_
    intake / baseline_carbohydrate_intake
fat_term=max( 0.0 , weighting_of_oxidation_for_lipolysis * effect_of
    _carb_intake_on_lipolysis * effect_of_obesity_on_lipolysis)
```

4.2.7. Physical Activity Energy Sector

The equations for the variables in the physical activity energy sector are given in Table 4.21.

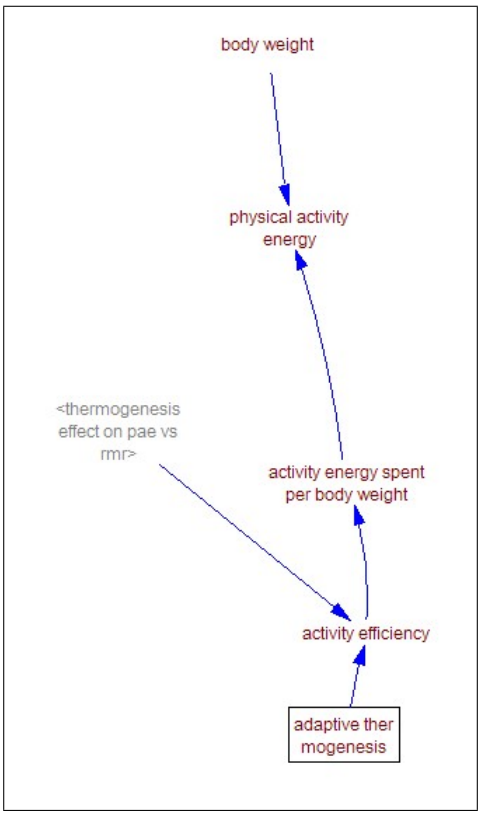

Figure 4.18. Physical activity energy sector in detail

Table 4.21. Code for the variables in the physical activity energy sector (Equations are adopted from Hall's model. Formulations of the variables look different but explicit mathematical forms are the same.)

```
physical_activity_energy=activity_energy_spent_per_body_weight *
    body_weight
activity_energy_spent_per_body_weight=activity_energy_per_body_
    weight * activity_efficiency
activity_efficiency=1.0 + thermogenesis_effect_on_pae_vs_rmr *
    adaptive_thermogenesis
thermogenesis_effect_on_pae_vs_rmr=0.52
```

4.2.8. Adaptive Thermogenesis Sector

The equations for the variables in the adaptive thermogenesis subsector are given in Table 4.22.

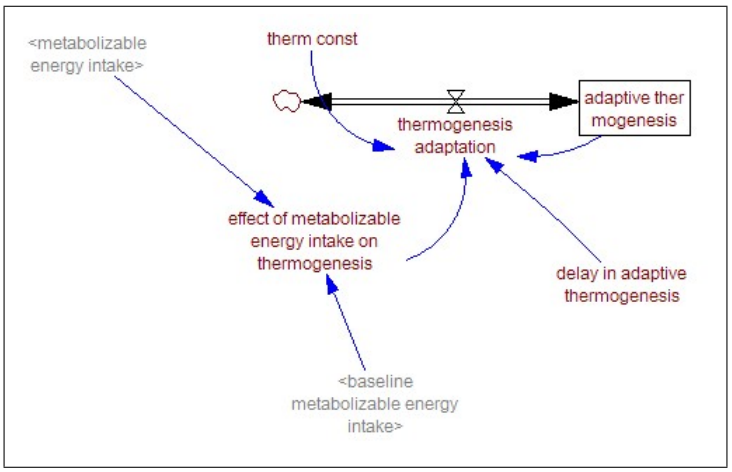

Figure 4.19. Adaptive thermogenesis sector in detail

Table 4.22. Code for variables in the adaptive thermogenesis sector (Equations are adopted from Hall's model. Formulations of the variables look different but explicit mathematical forms are the same.)

```
thermogenesis_adaptation=( therm_const * effect_of_metabolizable_
    energy_intake_on_thermogenesis - adaptive_thermogenesis) / delay
    _in_adaptive_thermogenesis
therm_const=0.8
delay_in_adaptive_thermogenesis=7.0
effect_of_metabolizable_energy_intake_on_thermogenesis=(
    metabolizable_energy_intake - baseline_metabolizable_energy_
    intake ) / baseline_metabolizable_energy_intake
baseline_metabolizable_energy_intake=baseline_carbohydrate_intake +
    baseline_fat_intake +baseline_protein_intake
metabolizable_energy_intake=carbohydrate_intake + fat_intake +
    protein_intake
```

4.2.9. Thermic Effect of Food

The equations for the variables that depend on thermic effect of food are given in Table 4.23.

Table 4.23. Code for the variables that thermic effect of food depends on (Adopted from Hall's model)

```
thermic_effect_of_food=tef_c * cal_dens_c * ci + tef_f * fat_intake
   + tef_p * cal_dens_p * pi
tef_p=0.25
tef_f=0.025
tef_c=0.075
cal_dens_c=4.18
cal_dens_p=4.7
cal_dens_f=9.44
```

4.3. Customization of the Model

We aim the body weight simulation game to be useful for general public to manage and test their dieting habits. To achieve this goal, the model should be customized for the game player. Namely, the parameters in the model should be calibrated for each player, so that they reflect the metabolic characteristics of the game player.

Everybody has different metabolic characteristics. One person eats a lot and does not gain weight whereas another person eats less and gains weight. These differences are caused by different metabolic activity levels of the people.

Customization of the model consists of two steps:

1. Putting the system into energy/weight balance specific to the player
2. Initializing parameters reflecting body characteristics of the player

Putting the system into energy/weight balance assures that the simulation stays stable for any game player given his baseline (initial) conditions. Baseline conditions are baseline values of some of the parameters. In current model, they are baseline body weight, protein intake, carbohydrate intake, and fat intake. Currently, *the model assumes that the person is in weight balance in real life initially.* This assumption can

be released in future versions of the model.

When energy balance is maintained, then total energy expenditure tee is equal to the total energy intake mei. The term mei is the abbreviation for metabolizable energy intake. The variable tee is defined in Equation 4.18 (Adopted from Hall).

$$tee = pae + rmr + tef \tag{4.18}$$

The variable rmr is defined in Equation 4.19 (Adopted from Hall except the variable ec existing in Hall's model).

$$rmr = mbc + mt + mc \tag{4.19}$$

At energy balance, the total energy expenditure is equal to the metabolic energy intake. We assume that the physical activity energy expenditure can vary largely from one person to another one. So, we deduce the baseline physical activity energy expenditure from the baseline values of the total energy intake, resting metabolic rate, and thermic effect of food:

$$\begin{aligned} mei_b &= tee_b \\ &= pae_b + tef_b + rmr_b \\ &\Longrightarrow \\ pae_b &= mei_b - (tef_b + rmr_b) \end{aligned} \tag{4.20}$$

Since the physical activity expenditure is dependent on the variable $act_{eng,bw}$, activity energy per body weight, the baseline value of this variable should be calibrated according to the baseline physical activity expenditure:

$$act_{eng,bw,b} = \frac{pae_b}{bw_b} \tag{4.21}$$

Energy balance occurs when there is no change in stocks, thus differential equations are in balance:

$$\frac{dc}{dt} = 0, \frac{df}{dt} = 0, \frac{dp}{dt} = 0 \tag{4.22}$$

The definitions of the differential equations are given in Equation 4.1. Putting them into Equation 4.22 gives:

$$\begin{aligned} ci + gng_f + gng_p - cox_a - dnl - cox_s &= 0 \\ fi + dnl - gng_f - fox_a - fox_s &= 0 \\ pi - gng_p - pox_a - pox_s &= 0 \end{aligned} \tag{4.23}$$

Secondary oxidation is zero when energy balance is maintained:

$$cox_{s,0} = fox_{s,0} = pox_{s,0} = 0 \tag{4.24}$$

Note that, in the equations in this section the index 0 of a variable, such as x_0, denotes the value of the variable x at time 0. The index b denotes baseline. The baseline conditions are assumed to be the initial conditions. Therefore, the indexes 0 and b can be used interchangeably in the equations.

So, Equation 4.23 becomes:

$$\begin{aligned} cox_{a,0} &= ci_0 + gng_{f,0} + gng_{p,0} - dnl_0 \\ fox_{a,0} &= fi_0 + dnl_0 - gng_{f,0} \\ pox_{a,0} &= pi_0 - gng_{p,0} \end{aligned} \tag{4.25}$$

Equation 4.25 describes the constraints on primary oxidation variables in energy balance conditions. Definitions of primary oxidation variables are given in Equa-

tion 4.26.

$$cox_a = f_c \cdot see + gng_p + gng_f \tag{4.26a}$$

$$fox_a = f_f \cdot see \tag{4.26b}$$

$$pox_a = f_p \cdot see \tag{4.26c}$$

We need to put the definitions of primary oxidation variables in Equation 4.26 into the constraints in Equation 4.25 such that we can extract the definitions of the metabolic parameters in terms of the exogenous constants explicitly. To do this, we need to substitute all the dependent variables until we reach independent, exogenous variables.

Now, we go over each equation in 4.26. Let's start with 4.26c. f_p is defined in Equation 4.27:

$$f_p = \frac{prot\_term}{z} \tag{4.27}$$

$prot\_term$ is defined in Equation 4.28:

$$prot\_term = n_{d_p} + max(eff_{pi,pox}, 0) \cdot eff_{act,pox} \tag{4.28}$$

The variables that $prot\_term$ depend on are functions of other variables.

$$n_{d_p} = \frac{d_p}{d_{p,b}} \tag{4.29}$$

$d_{p,0}$ denotes baseline value of d_p. As explained in Section 4.3, our model assumes that the person is in weight balance in real life. Thus $d_{p,b} = d_{p,0}$. So Equation 4.29 becomes $n_{d_p} = 1$ in baseline.

$prot\_term$ depends also on $eff_{pi,pox}$. Its definition is given in Equation 4.30.

$$eff_{pi,pox} = w_{pi} \cdot (1.0 + s_{pi,ox} \cdot n_{\Delta pi}) \tag{4.30}$$

$n_{\Delta pi}$ is defined in Equation 4.31

$$n_{\Delta pi} = \frac{\Delta pi}{pi_b} \tag{4.31}$$

Baseline value of Δpi is zero: $dpi_0 = 0$ since $pi_0 = pi_b$. Thus $ndpi_0 = 0$ too. Putting this into Equation 4.30 results in $eff_{pi,pox,0} = w_{pi}$

$eff_{act,pox}$ is defined in Equation 4.32

$$eff_{act,pox} = sa \cdot e^{-\ln(sa) \cdot n_{acteng_{bw}}} \tag{4.32}$$

Definition of $n_{acteng_{bw}}$ is in Equation 4.33

$$n_{acteng_{bw}} = \frac{act_{eng,bw}}{act_{eng,bw,b}} \tag{4.33}$$

Since $act_{eng,bw,0} = act_{eng,bw,b}$, $n_{acteng_{bw,0}} = 1$. So putting this into Equation 4.32 results in Equation 4.34

$$eff_{act,pox,0} = sa \cdot e^{-\ln(sa)} = 1 \tag{4.34}$$

Putting $eff_{act,pox,0} = 1$, $n_{d_p} = 1$, and $eff_{pi,pox,0} = w_{pi}$ into Equation 4.28 gives $prot\_term_0 = w_{pi} + 1$. Putting this term into Equation 4.27 gives:

$$f_{p,0} = \frac{1 + w_{pi}}{z_0} \tag{4.35}$$

Now $f_{p,0}$ in Equation (4.26c) is described in terms of independent, exogenous variables. We need to do describe $f_{f,0}$ and $f_{c,0}$ in the same way.

f_f is defined in Equation 4.36.

$$f_f = \frac{fat\_term}{z} \tag{4.36}$$

fat_term is defined in Equation 4.37:

$$fat\_term = w_f \cdot n_{d_f} \tag{4.37}$$

n_{d_f} is defined in Equation 4.38:

$$n_{d_f} = \frac{d_f}{d_{f,b}} \tag{4.38}$$

$$d_{f,0} = d_{f,b}$$

Therefore $n_{d_f,0} = 1$ in baseline conditions. Putting this into Equation 4.37 gives

$fat\_term_0 = w_f$. And therefore Equation 4.36 gives:

$$f_{f,0} = \frac{w_f}{z_0} \tag{4.39}$$

Now $f_{p,0}$ and $f_{f,0}$ in Equation (4.26) is described in terms of independent, exogenous variables. We need to describe $f_{c,0}$ in the same way.

f_c is defined in Equation 4.40.

$$f_c = \frac{carb\_term}{z} \tag{4.40}$$

$carb\_term$ is defined in Equation 4.41:

$$carb\_term = eff_{dg,cox} + max(0.0, eff_{ci,cox} \cdot (\frac{c}{0.1 + c})) \tag{4.41}$$

$0.1 + c \simeq c \Longrightarrow \frac{c}{0.1+c} \simeq 1$

$eff_{dg,cox}$ is defined in Equation 4.42:

$$eff_{dg,cox} = w_g \cdot n_{d_c} \tag{4.42}$$

n_{d_c} is defined in Equation 4.43:

$$n_{d_c} = \frac{d_c}{d_{c,b}} \tag{4.43}$$

$d_{c,0} = d_{c,b}$ in baseline conditions. Thus $n_{d_{c,0}} = 1$ in baseline. Putting this into

Equation 4.42 gives:

$$eff_{dg,cox} = w_g \tag{4.44}$$

$eff_{ci,cox}$ is defined in Equation 4.45

$$eff_{ci,cox} = w_{ci} \cdot (1.0 + s_{ci} \cdot n_{\Delta ci}) \tag{4.45}$$

The term $n_{\Delta ci}$ is defined in Equation 4.46:

$$n_{\Delta ci} = \frac{\Delta ci}{ci_b} \tag{4.46}$$

$\Delta ci = 0$ in baseline. Therefore $n_{\Delta ci,0} = 0$ too.

Putting $n_{\Delta ci,0} = 0$ into Equation 4.45 gives:

$$eff_{ci,cox,0} = w_{ci} \tag{4.47}$$

Putting $eff_{ci,cox,0} = w_{ci}$ and $eff_{dg,cox} = w_g$ into Equation 4.41 gives:

$$carb\_term_0 = w_g + w_{ci} \tag{4.48}$$

$$f_{c,0} = \frac{w_g + w_{ci}}{z} \tag{4.49}$$

Now, we have definitions of all the oxidation fractions $f_{c,0}, f_{f,0}, f_{p,0}$ in terms of

exogenous constants except z. z is defined in Equation 4.50

$$z = carb\_term + fat\_term + prot\_term \tag{4.50}$$

Putting values in baseline conditions results in Equation 4.51:

$$z_0 = (w_g + w_{ci}) + (1 + w_{pi}) + (w_f) \tag{4.51}$$

Now, since 4.25 = 4.26, we need to equate balance constraints in Equation 4.25 and definitions of primary oxidation variables in Equation 4.26. This results in Equation 4.53:

$$\begin{aligned}
cox_{a,0} &= f_{c,0} \cdot see_0 + gng_{p,0} + gng_{f,0} \\
fox_{a,0} &= f_{f,0} \cdot see_0 \\
pox_{a,0} &= f_{p,0} \cdot see_0 \\
cox_{a,0} &= ci_0 + gng_{f,0} + gng_{p,0} - dnl_0 \\
fox_{a,0} &= fi_0 + dnl_0 - gng_{f,0} \\
pox_{a,0} &= pi_0 - gng_{p,0} \\
&\Longrightarrow
\end{aligned} \tag{4.52}$$

$$\begin{aligned}
f_{c,0} \cdot see_0 + gng_{p,0} + gng_{f,0} &= ci_0 + gng_{f,0} + gng_{p,0} - dnl_0 \\
f_{f,0} \cdot see_0 &= fi_0 + dnl_0 - gng_{f,0} \\
f_{p,0} \cdot see_0 &= pi_0 - gng_{p,0} \\
&\Longrightarrow
\end{aligned} \tag{4.53}$$

$$
\begin{aligned}
f_{c,0} \cdot see_0 &= ci_0 - dnl_0 \\
f_{f,0} \cdot see_0 &= fi_0 + dnl_0 - gng_{f,0} \\
f_{p,0} \cdot see_0 &= pi_0 - gng_{p,0} \\
&\Longrightarrow
\end{aligned}
\tag{4.54}
$$

$$
\begin{aligned}
f_{c,0} &= \frac{ci_0 - dnl_0}{see_0} \\
f_{f,0} &= \frac{fi_0 + dnl_0 - gng_{f,0}}{see_0} \\
f_{p,0} &= \frac{pi_0 - gng_{p,0}}{see_0}
\end{aligned}
\tag{4.55}
$$

see is defined in Equation 4.56

$$see = tee - gng_p - gng_f \tag{4.56}$$

$tee_0 = mei_b$ since the body is in energy equilibrium during baseline. Thus $see_0 = mei_b - gng_{f,0} - gng_{p,0}$. Putting this into Equation 4.55 gives Equation 4.57:

$$
\begin{aligned}
f_{c,0} &= \frac{ci_0 - dnl_0}{mei_b - gng_{f,0} - gng_{p,0}} \\
f_{f,0} &= \frac{fi_0 + dnl_0 - gng_{f,0}}{mei_b - gng_{f,0} - gng_{p,0}} \\
f_{p,0} &= \frac{pi_0 - gng_{p,0}}{mei_b - gng_{f,0} - gng_{p,0}}
\end{aligned}
\tag{4.57}
$$

Now, let's call $f_{x,0}$ terms as k_x. The terms in right hand sides in Equation 4.57 are all values at baseline conditions. They are either given, as defined in the model such as $gng_{f,b}$, or input by the game player, such as ci_0. So, k_c, k_f, k_p variables are

functions of constant values as given in Equation 4.58.

$$\begin{aligned} k_c &= \frac{ci_0 - dnl_0}{mei_b - gng_{f,0} - gng_{p,0}} \\ k_f &= \frac{fi_0 + dnl_0 - gng_{f,0}}{mei_b - gng_{f,0} - gng_{p,0}} \\ k_p &= \frac{pi_0 - gng_{p,0}}{mei_b - gng_{f,0} - gng_{p,0}} \end{aligned} \tag{4.58}$$

Equations 4.35, 4.39, 4.49 are constraints on f_f, f_p, f_c in baseline conditions where energy balance is maintained.

$$\begin{aligned} k_c &= \frac{w_g + w_{ci}}{(1 + w_{pi}) + (w_f) + (w_g + w_{ci})} \\ k_f &= \frac{w_f}{(1 + w_{pi}) + (w_f) + (w_g + w_{ci})} \\ k_p &= \frac{1 + w_{pi}}{(1 + w_{pi}) + (w_f) + (w_g + w_{ci})} \end{aligned} \tag{4.59}$$

Now, since k_c, k_f, k_p are constant values defined in Equation 4.58, we have three equations and four unknowns w_g, w_{ci}, w_f, w_{pi} in Equation 4.59. By assigning arbitrary values to two of the unknowns, we can extract explicit formulas for the remaining two unknowns. By using substitution method we obtain the solutions for w_f and w_g in Equation 4.60:

$$\begin{aligned} w_f &= (1.0 + w_{pi}) \cdot \frac{k_f}{k_p} \\ w_g &= \frac{k_c}{k_p} \cdot (1.0 + w_{pi}) - w_{ci} \end{aligned} \tag{4.60}$$

Now, we need to find initial values of the stocks that keep the system in equilibrium. Given the baseline body weight, we can extract the baseline values of the stocks. Body weight is defined in Equation 4.61.

$$bw = f + m_{lean} \tag{4.61}$$

The symbol f denotes the stock fat. The symbol m_{lean} denotes the mass of lean tissues. The mass of lean tissues is defined in Equation 4.62.

$$m_{lean} = bm_b + ecw + m_{cell} \tag{4.62}$$

The symbol bm_b denotes baseline value of the bone mass. The model assumes that the bone mass does not change for a healthy, adult person. The symbol ecw is the mass of extracellular water in grams. The symbol m_{cell} is the total cell mass in the body. The model assumes ecw constant for the purpose of simplicity. Total cell mass in the body is defined in Equation 4.63.

$$m_{cell} = ics + c + p + icw \tag{4.63}$$

The symbol ics denotes intracellular solids which is assumed to be constant. The symbols c and p are levels of the stocks of carbohydrate and protein. The symbol icw is the mass of the intracellular water in the body. It is defined in Equation 4.64.

$$icw = h_c \cdot c + h_p \cdot p + ciw \tag{4.64}$$

The symbols h_c and h_p denote constant hydration coefficients for carbohydrate and protein. The symbol ciw is the mass of a hypothetical parameter called constant intracellular water. This parameter is used to hold the equality of the body weight calculated from the components with the body weight given as an external parameter.

Now, we relax the constraint that ecw is constant for the moment where we measured baseline body weight to define ecw_b as a function of bw_b. Assuming that fraction of water to body weight $f_{tw,bw}$ and fraction of extracellular water to total water $f_{ecw,tw}$ are constant, we can formulate baseline extracellular water as a function

of baseline bodyweight as in Equation 4.65:

$$ecw_b = f_{tw,bw} \cdot f_{ecw,tw} \cdot bw_b \tag{4.65}$$

When ecw_b is given, we can deduce the value of the parameter icw_b assuming that the fraction of intracellular water to extracellular water $f_{icw,ecw}$ is constant:

$$icw_b = f_{icw,ecw} \cdot ecw_b \tag{4.66}$$

Note that Equation 4.64 defines icw as a function of c and p whereas Equation 4.66 defines icw as a function of ecw_b which is a function of bw_b. So, given the exogenous parameter bw_b, we can deduce icw_b.

Now, we can reverse the dependency relationship between icw and m_{cell}. Equation 4.63 defines m_{cell} as a function of icw. But since we can deduce icw_b from the given value of bw_b, it is possible to define $m_{cell,b}$ as a function of icw_b by assuming that the fraction of intracellular water to cell mass $f_{icw,cm}$ is constant:

$$m_{cell,b} = 1.0/f_{icw,cm} \cdot icw_b \tag{4.67}$$

From the formula of the mass of lean tissues in Equation 4.62 is the formula of the baseline mass of lean tissues given in Equation 4.68 deduced.

$$m_{lean,b} = bm_b + ecw_b + m_{cell,b} \tag{4.68}$$

By reversing the relationship between body weight and fat mass defined in Equa-

tion 4.61, the formula of the baseline fat mass is deduced:

$$f_b = bw_b - m_{lean,b} \tag{4.69}$$

By reversing the relationship between cell mass and protein mass defined in Equation 4.63, the formula for the baseline protein mass is deduced:

$$p_b = m_{cell,b} - icw_b - ics - c_b \tag{4.70}$$

Now, we have two formulas for some of the variables at baseline condition. For example, icw is normally defined according to Equation 4.64, but at baseline it is defined also with Equation 4.66. Both formulas should give the same value at baseline. Therefore, we need to put a constraint that ensures the equality of both formulas. Since both formulas of icw must produce the same value at baseline, we can extract ciw to define it as a function of variables that are constant or that depend ultimately on exogenous parameter bw_b:

$$\begin{aligned} icw &= h_c \cdot c + h_p \cdot p + ciw \\ icw_b &= f_{icw,ecw} \cdot ecw_b \\ \implies fiew \cdot ecw_b &= h_c \cdot c_b + h_p \cdot p_b + ciw \\ \implies ciw &= f_{icw,ecw} \cdot ecw_b - (h_c \cdot c_b + h_p \cdot p_b) \end{aligned} \tag{4.71}$$

This section customizes Hall's modified model such that it becomes a generic model for anybody. The modifications in Hall's original model were explained in Section 4.2 and in this section. They can be summarized under a few items. First, there are some simplifications that we make to increase the understandability of the model while maintaining its validity. Second, our model adds a hypothesized process called secondary oxidation to make the system produce valid behavior under very high energy expenditure rates. Third, we make the modified model generic to apply it to any

healthy, adult person and we deduce the formulas for some metabolic parameters to customize the generic model to fit the specific, simulated person.

4.4. Validation and Analysis of the Model

This section describes the tests used to demonstrate the validity of the model. The model is simulated with Vensim software. Time unit is day. Time step for the simulation is 1/16 days which is tested to be sufficiently small. The time horizon is kept as 1000 days in order to see the behavior in the long run.

4.4.1. Reference Behavior of the Model

The reference run corresponds to the equilibrium conditions. The baseline values for daily carbohydrate intake, protein intake, and fat intake are set to 1500, 500, and 1000 kcal/day, respectively. The baseline body weight is 85 kg. During the simulation, the person takes same amounts of food intakes to maintain her stocks stable. So, the inputs of daily carbohydrate, protein, and fat intakes are constant at 1500, 500, and 1000 kcal/day, respectively. In this run, all stocks stabilize in their baseline levels when their initial values are set to the calculated baseline levels. The result of the simulation run is shown in Figure 4.20

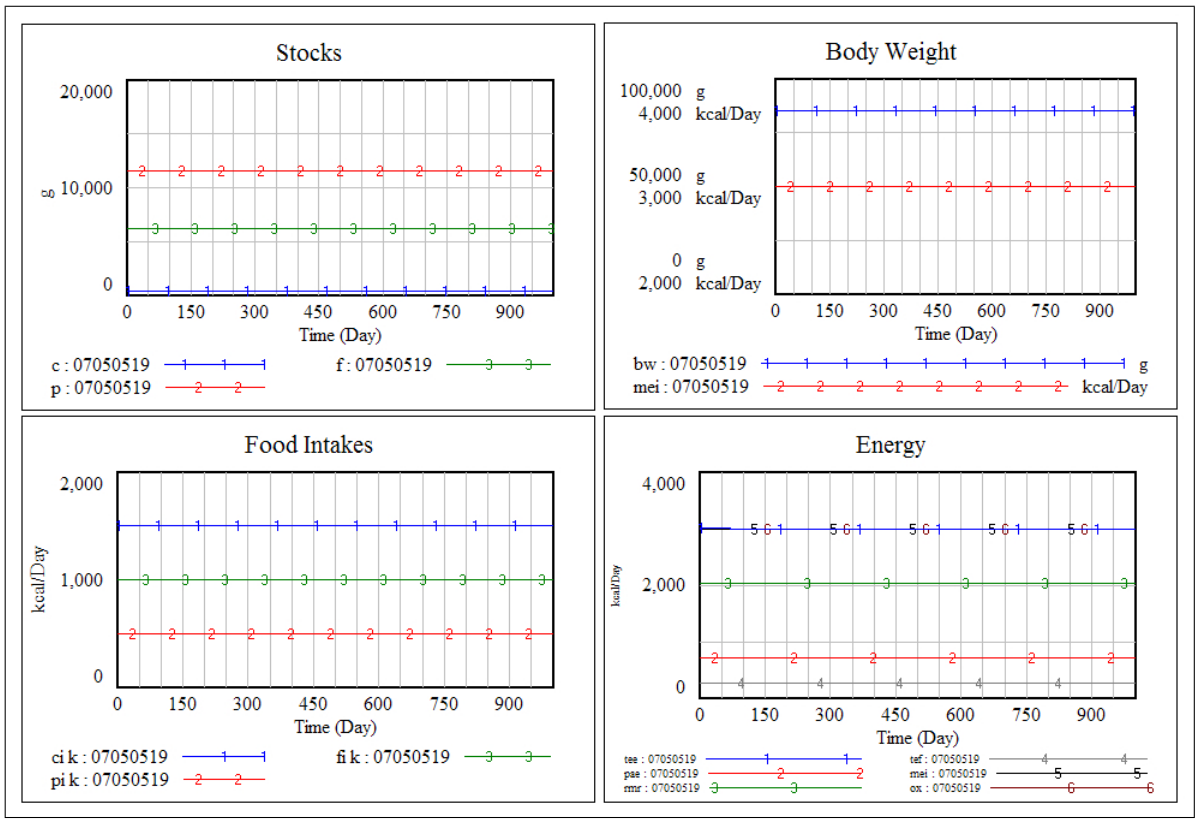

Figure 4.20. Equilibria of important variables in the reference run. (Food intake values during the simulation run are maintained constant at the baseline values)

4.4.2. Validation of the Model

This section discusses validity of the model. Validity in system dynamics model consists of two main parts: structure validity and behavior validity. System dynamics models try to explain the causal description of the processes. Therefore, system dynamics models are treated as valid only if they have acceptable structures representing the real system, and they can reproduce the dynamical behavior patterns of the reality. A formal validation process as described in Barlas (1996) [6] is followed in order to detect structural flaws in the model.

Studying model validity starts with model building and it is distributed through all phases. Since the underlying model in this thesis was developed essentially by Hall, we need to test validity of the model by studying and understanding Hall's model. We

study each equation one by one to test certain validity criteria by using direct tests of structure confirmation, parameter confirmation, direct extreme condition testing, and dimensional consistency. These activities are already described in the previous sections.

Structure-oriented behavior tests test the validity of a system as a whole. The tests are done by performing simulation runs of some characteristic scenarios that might occur in real life. Moreover, structure-oriented behavior tests involve testing the system under extreme conditions to reveal any non-causal structure that is hidden under the complexity of the model [6].

This section describes structure oriented behavior tests performed on the model. The scenarios used for validating the model are described in summary in Table 4.24. In this table, ci, pi, fi columns denote food intake input variables. The column ai denotes activity level index. The column mei denotes total metabolizable energy intake, namely the sum of food intake input variables. Different baseline values of food intake variables and body weight are experimented to test the model for different people. The columns ci_b, pi_b, fi_b denote baseline food intake variables. All values are in unit of kcal/day except bw_b. bw_b is given in grams.

Table 4.24. Parameter values used in validation scenarios

Scenario	ai	ci	pi	fi	mei	ci_b	pi_b	fi_b	bw_b
Reference run	0	1500	500	1000	3000	1500	500	1000	85000
Overweight	0	2500	500	1000	3000	2500	500	1000	135000
Underweight	0	1500	500	1000	3000	1500	500	1000	53500
Increased food intake	0	1750	600	1300	3650	1500	500	1000	85000
Decreased food intake	0	1250	350	750	2350	1500	500	1000	85000
Decreased activity	10	1500	500	1000	3000	1500	500	1000	85000
Increased activity	75	1500	500	1000	3000	1500	500	1000	85000
Increased act. and fi	75	1750	600	1000	3000	1500	500	1000	85000
Extreme activity up	-135	1500	500	1000	3000	1500	500	1000	85000
Extreme activity down	0	1500	500	1000	3000	1500	500	1000	85000
Extreme pi up	0	1500	2000	1000	4500	1500	500	1000	85000
Extreme pi down	0	1500	0	1000	2500	1500	500	1000	85000
Extreme fi up	0	1500	500	8000	10000	1500	500	1000	85000
Extreme fi down	0	1500	500	0	2000	1500	500	1000	85000
Extreme ci up	0	10000	500	1000	11500	1500	500	1000	85000
Extreme ci down	0	0	500	1000	1500	1500	500	1000	85000

4.4.2.1. Experiments with Different Metabolic Characteristics. Here, we run the simulation with equilibrium conditions for an overweight person and for an underweight person. The model maintains the stocks of these simulated people stable. This proves that the model customizes itself to the baseline conditions of anyone given that the simulated person is in equilibrium conditions.

Overweight Person: In this experiment, the conditions of an overweight person in equilibrium are simulated. The overweight person has a baseline body weight of 135 kg. All the other parameters except the carbohydrate intake are maintained as in the reference run. The carbohydrate intake is increased from 1500 kcal/day to 2500 kcal/day to cover up the additional, required energy expenditure of the overweight person. The dynamics of the simulation is shown in Figure 4.21.

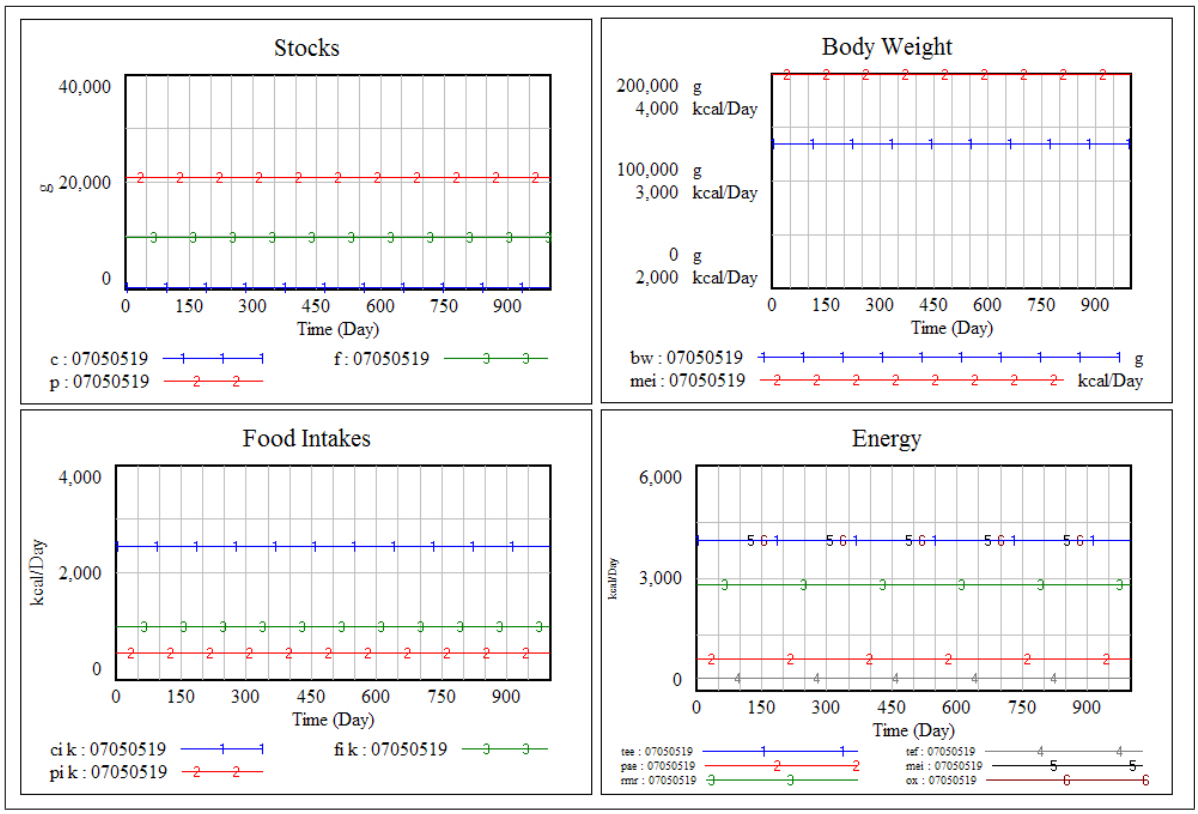

Figure 4.21. Equilibria of important variables in the case of overweight person. (Food intake values are maintained stable at the baseline values)

The person is in dynamic equilibrium in this run. This is due to setting food intake values during the simulation run equal to the baseline food intake values.

Underweight Person: In this experiment, the conditions of an underweight person are simulated. The underweight person has a baseline body weight of 54 kg. All the other variables are maintained from the reference run. The dynamics of the simulation is shown in Figure 4.22

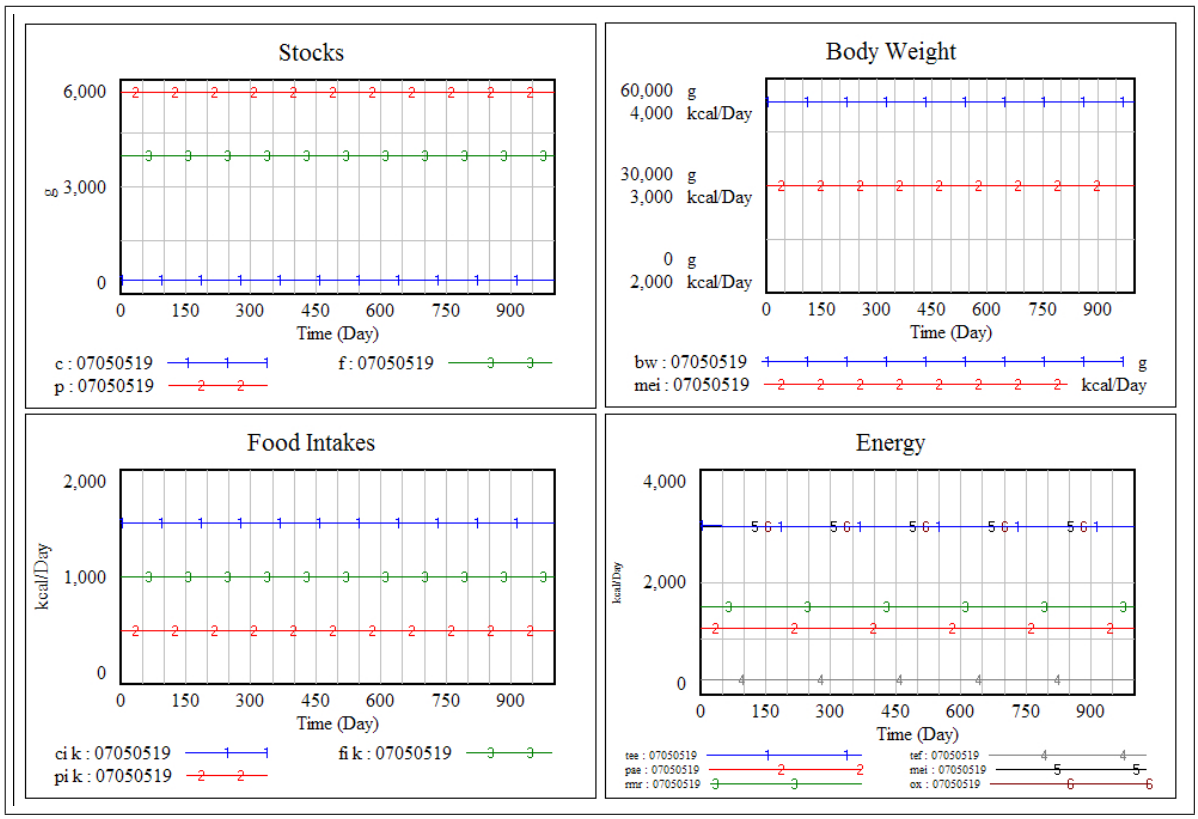

Figure 4.22. Equilibria of important variables in the case of underweight person. (Food intake values are maintained stable at the baseline values)

The person is in dynamic equilibrium in this run. This is due to setting food intake values during the simulation run equal to the baseline food intake values.

4.4.2.2. Experiments with Different Food Intakes. *Increased Daily Food Intake:* In this experiment, daily food intake is increased as a step function at day 50 from the baseline level to a higher level, see Table 4.24. The dynamics of the simulation is shown in Figure 4.23

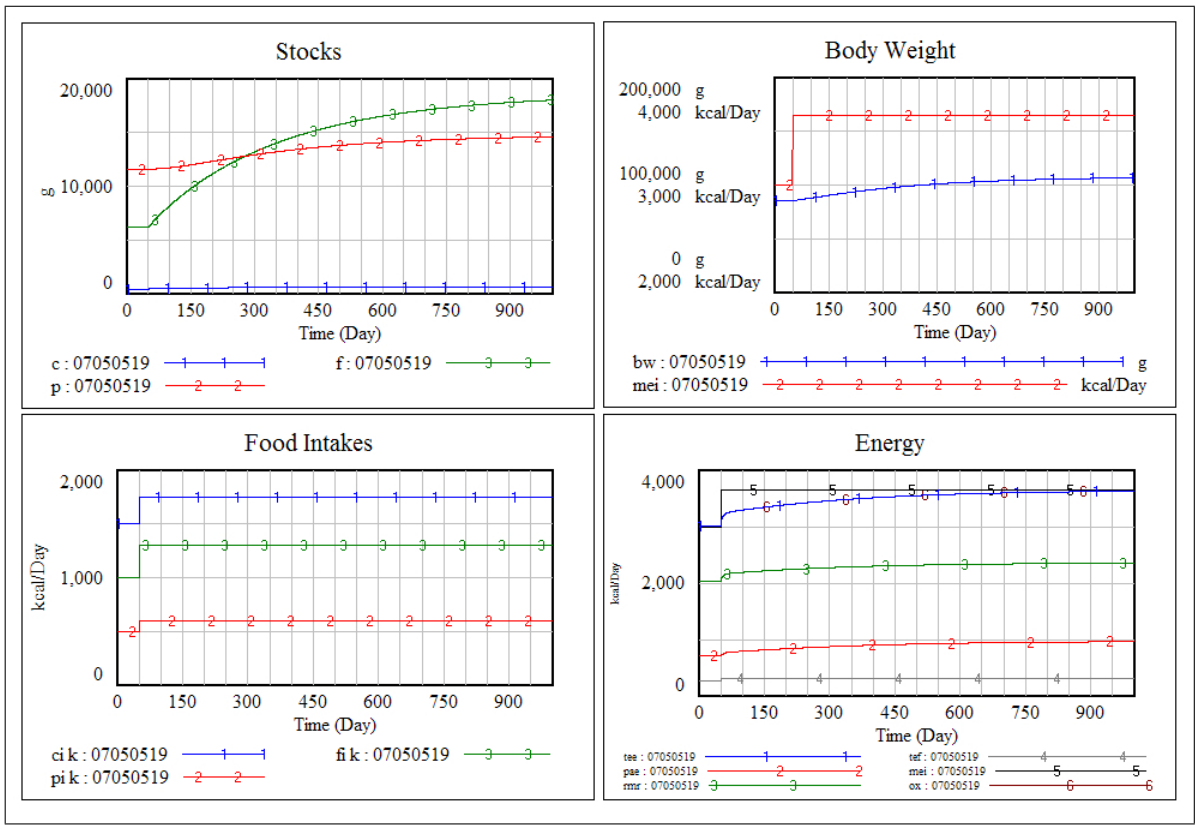

Figure 4.23. Dynamics of important variables in the case of increased daily food intake. (Food intake values are increased at day 50 from the baseline level to a higher level)

The person is in dynamic equilibrium until day 50. From day 50 on excess food intake is given. So $mei > tee$ (daily energy intake becomes higher than daily energy expenditure). Therefore, the person starts gaining weight. Because of the two main negative feedback loops in the model, weight gaining process follows goal seeking behavior. The pattern of the weight gaining is realistic because a person who tries to gain weight can increase his body weight easily at first. While the weight increases, gaining more weight becomes more difficult. The person experiences a plateau effect. From a causal point of view, the plateau effect is the symptom of the counteracting forces of the resting metabolic rate and physical activity expenditure against the change in the body weight.

Decreased Daily Food Intake: In this experiment, daily food intake is decreased

as a step function at day 50 from the baseline level to a lower level, see Table 4.24. The dynamics of the simulation is shown in Figure 4.24

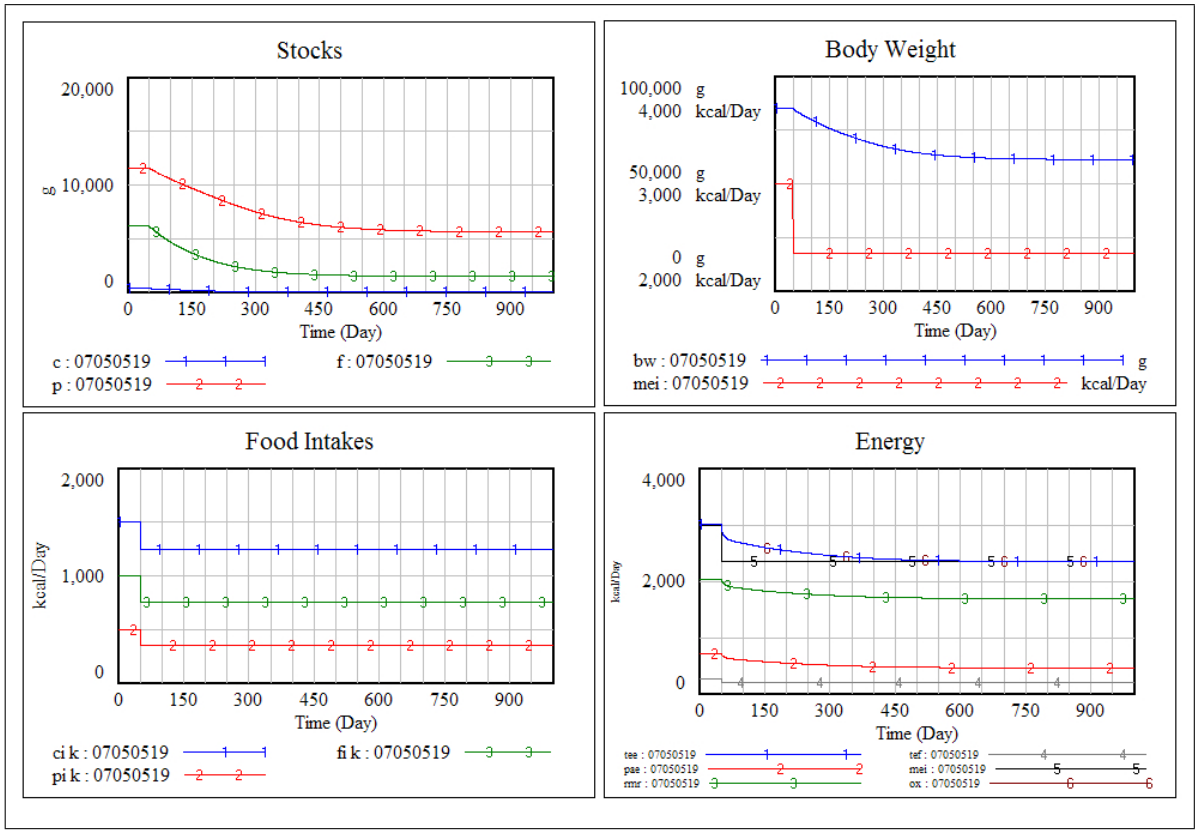

Figure 4.24. Dynamics of important variables in the case of decreased daily food intake. (Food intake values are decreased at day 50 from the baseline level to a lower level)

The person is in dynamic equilibrium until day 50. From day 50 on, food intake is decreased. So $mei < tee$ (daily energy intake becomes lower than daily energy expenditure). Therefore the person starts losing weight. Because of the two main negative feedback loops in the model, weight losing process follows goal seeking behavior. The pattern of the weight loss is realistic because of the same reasons as the previous experiment. A person who tries to lose weight can increase his body weight easily at first. While the weight decreases, losing more weight becomes more difficult. The person experiences a plateau effect. The reason behind the plateau effect is again the

two negative feedback loops of resting metabolic rate and physical activity expenditure that counteract to the change in the body weight.

4.4.2.3. Experiments with Different Activity Levels. *Decreased Daily Activity Level:* In this experiment, daily activity level is decreased as a step function at day 50 from the baseline level to a lower level, see Table 4.24. The dynamics of the simulation is shown in Figure 4.25

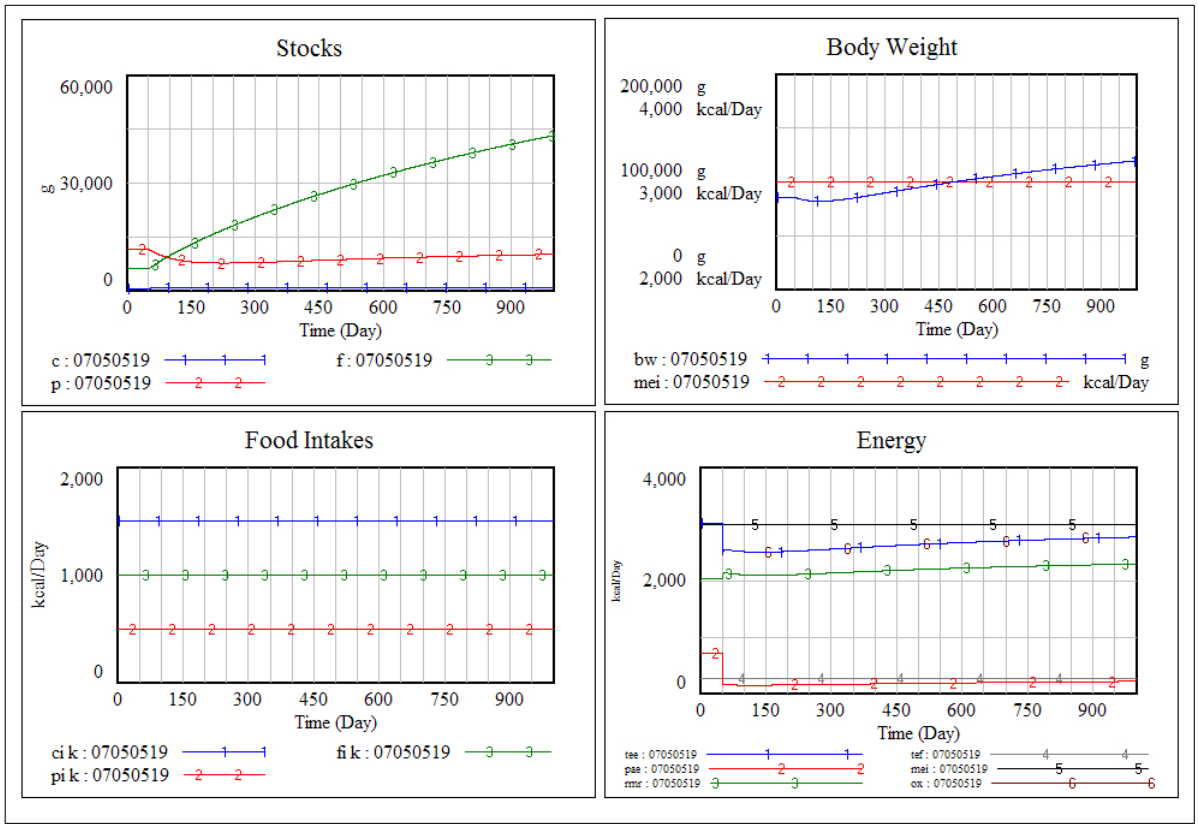

Figure 4.25. Dynamics of important variables in the case of decreased daily activity level. (Daily activity levels are decreased at day 50 from the baseline level to a lower level)

The person is in dynamic equilibrium until day 50. From day 50 on physical activity level is decreased. This causes physical activity energy expenditure to fall instantly. So $mei > tee$ (daily energy intake becomes higher than daily energy expen-

diture). Therefore the person starts gaining weight. Because of the two main negative feedback loops in the model, weight gaining process follows goal seeking behavior. This is realistic because of the same reasons as the experiment "Decreased Daily Food Intake". The two negative feedback loops of resting metabolic rate and physical activity expenditure counteract to the increase in the body weight which is triggered by the decrease of the physical activity expenditure.

Increased Daily Activity Level: In this experiment, daily activity level is set to a higher level than the reference run, see Table 4.24. The dynamics of the simulation is shown in Figure 4.26

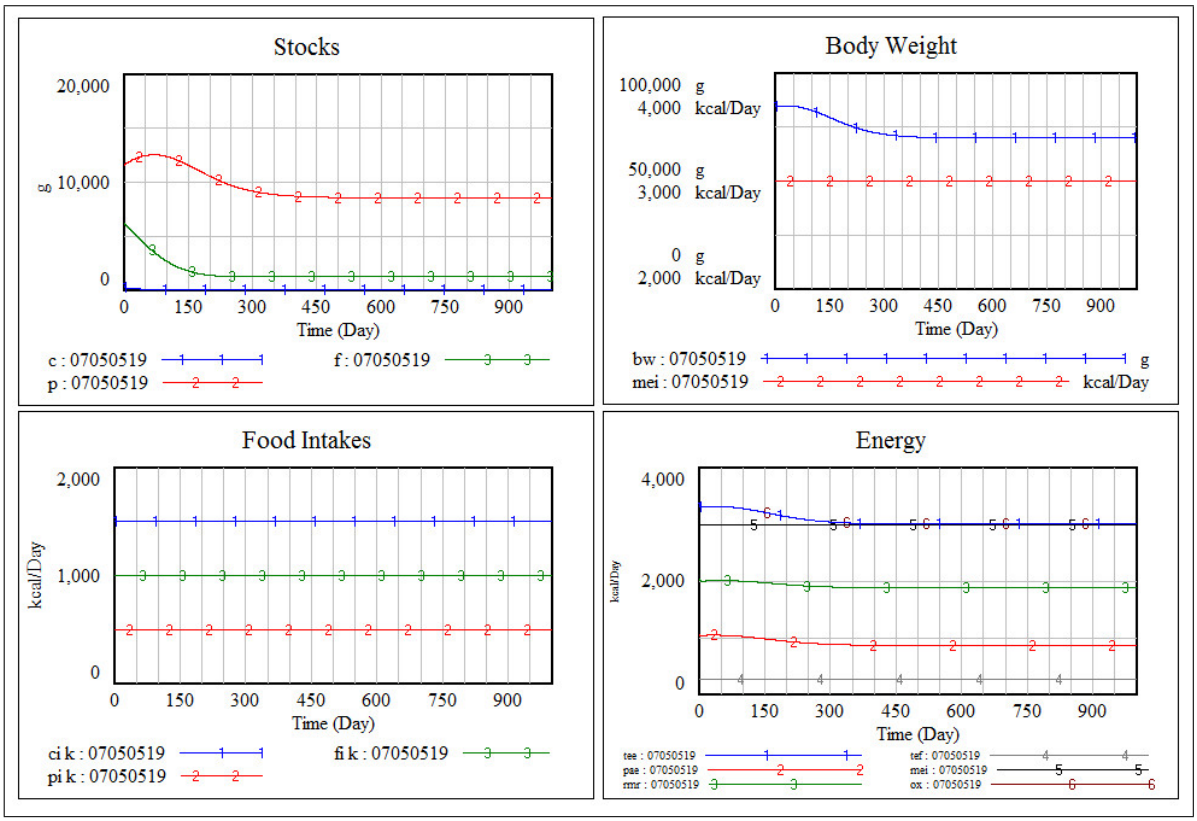

Figure 4.26. Dynamics of important variables in the case of increased daily activity level. (Daily activity level is set to a higher level than the reference run)

The person does more activity than the reference run. In the reference run, the daily physical activity energy expenditure is around 700 kcal/day. In this run, it is

around 1050 kcal/day. The daily food intake is not sufficient to cover up the daily energy expenditure $mei < tee$. For some time, until day 50 approximately, the body weight stays in dynamic equilibrium. This dynamic equilibrium is the result of the increase in protein stock and decrease in fat stock. Because of the deficiency in the energy intake, the body spends its fat stocks to cover up the gap. In the meantime, the protein stock grows because of the increased physical activity. However, after around day 50, the rate of decrease of the fat stock decelerates, and the protein oxidation rate starts to increase. Then, the body weight begins to decrease in goal seeking behavior because of the losses in the protein and fat stocks. The simulation results are realistic because it shows the protein sparing effect of the fat stock.

Increased Daily Activity Level and Food Intake: In this experiment, daily activity level and food intake rates are set to a higher level than the reference run, see Table 4.24. The dynamics of the simulation is shown in Figure 4.26

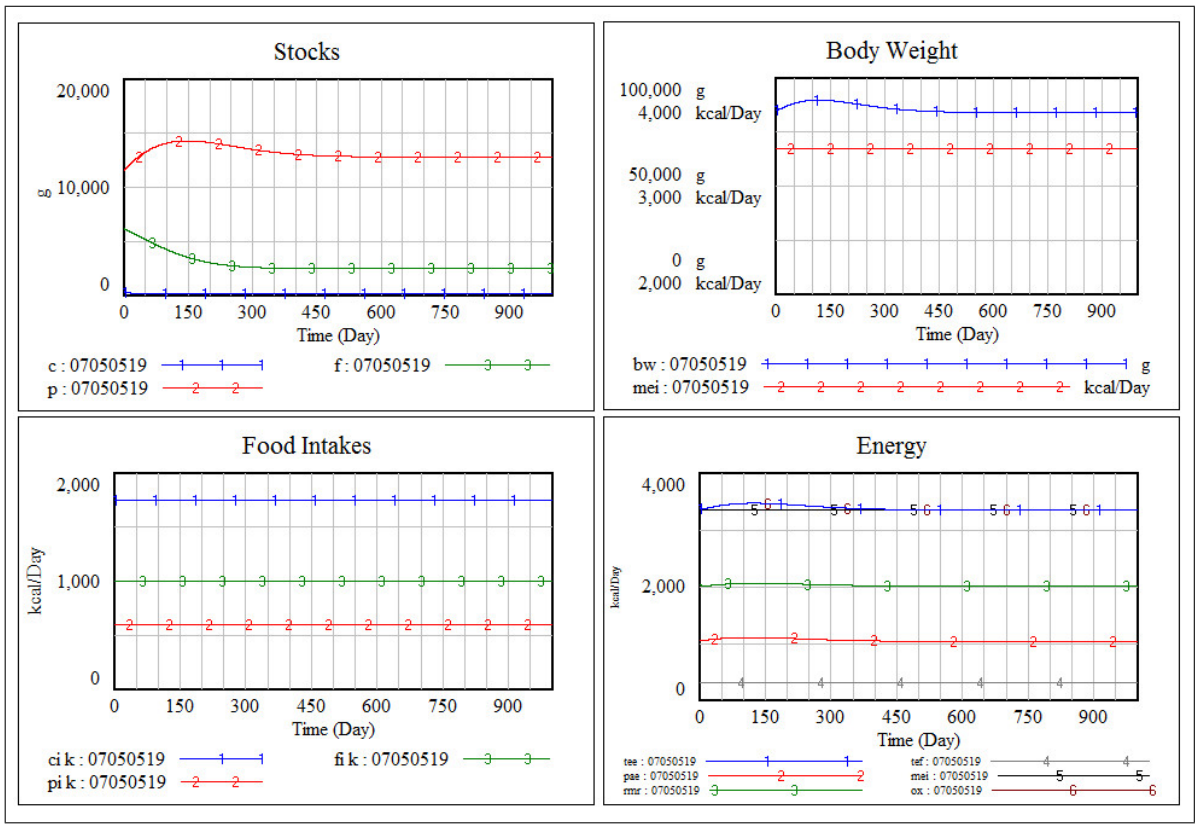

Figure 4.27. Dynamics of important variables in the case of increased daily activity level and food intake. (Daily activity level and food intake rates are set to a higher level than the reference run)

This experiment is a variant of the previous one. In the previous experiment, the daily activity level is increased with respect to the reference run, but the food intake rates are equal to the reference run. In this experiment, food intake rates are increased too. The energy deficiency due to increased physical activity is around 350 kcal/day in the previous run. In this run, there is the energy gap is covered up by the increased food intake rates. The carbohydrate and protein intake rates are increased by 250 and 100 kcal/day, respectively.

The simulation starts where energy intake is equal to the energy expenditure rates. The protein stock grows, while the fat stock decreases. The growth in the protein stock is due to the additional physical activity. The growth in the protein stock causes the body weight to increase because of the multiplicative effect of the

protein stock in the lean tissues. High metabolic activity of the lean tissues increases the resting metabolic rate. Growing body weight increases the physical activity energy expenditure. So, the growth in the body weight is balanced by the increasing energy expenditure which causes the body weight to return back to its original level after a while.

Extreme Increase in Daily Activity Level: In this experiment, daily activity level is set to an extremely high value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.28

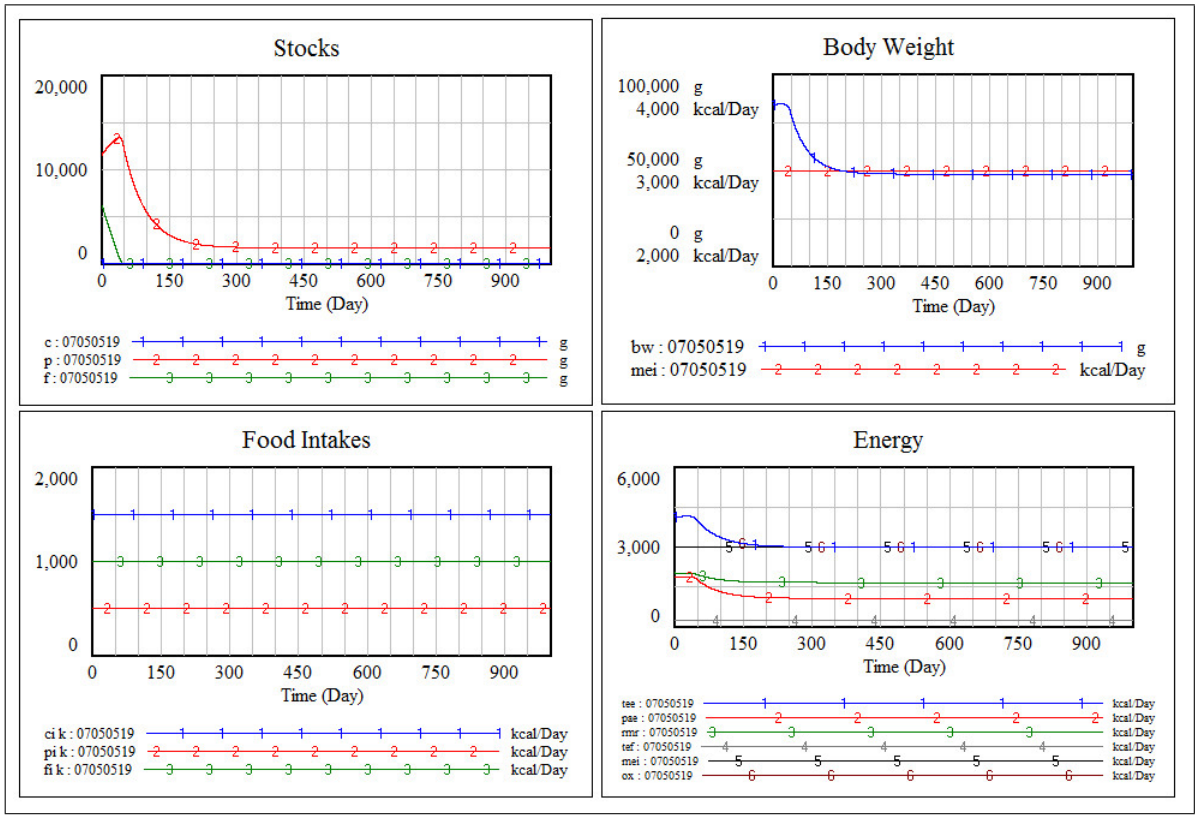

Figure 4.28. Dynamics of important variables in the case of extreme increase in daily activity level

The person does a lot of activity. This causes the daily physical activity energy expenditure to be very high relative to other components of energy expenditure. The daily food intake is not sufficient to cover up the daily energy expenditure $mei < tee$. There-

fore, the person loses weight very rapidly. Because of the two main negative feedback loops in the model, weight losing process follows goal seeking behavior. The behavioral pattern is realistic because the two negative feedback loops of resting metabolic rate and physical activity expenditure counteract to the rapid rate of decrease in the body weight. They slow down the rate of change of the body weight.

The high level of physical activity causes the physical activity expenditure to become very high. This causes the oxidation rates of the energy stores to become very high. This causes a very rapid rate of decrease in the energy stocks and body weight. The decrease in the protein stock decreases resting metabolic rate. The decrease in the body weight decreases the physical activity expenditure which decreases the oxidation rates of energy stores. The decreases in the oxidation rates cause the rates of decline of the energy stocks to decrease. In conclusion, the rate of decline of the body weight slows down.

Extreme Decrease in Daily Activity Level: In this experiment, daily activity level is set to an extremely low value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.29

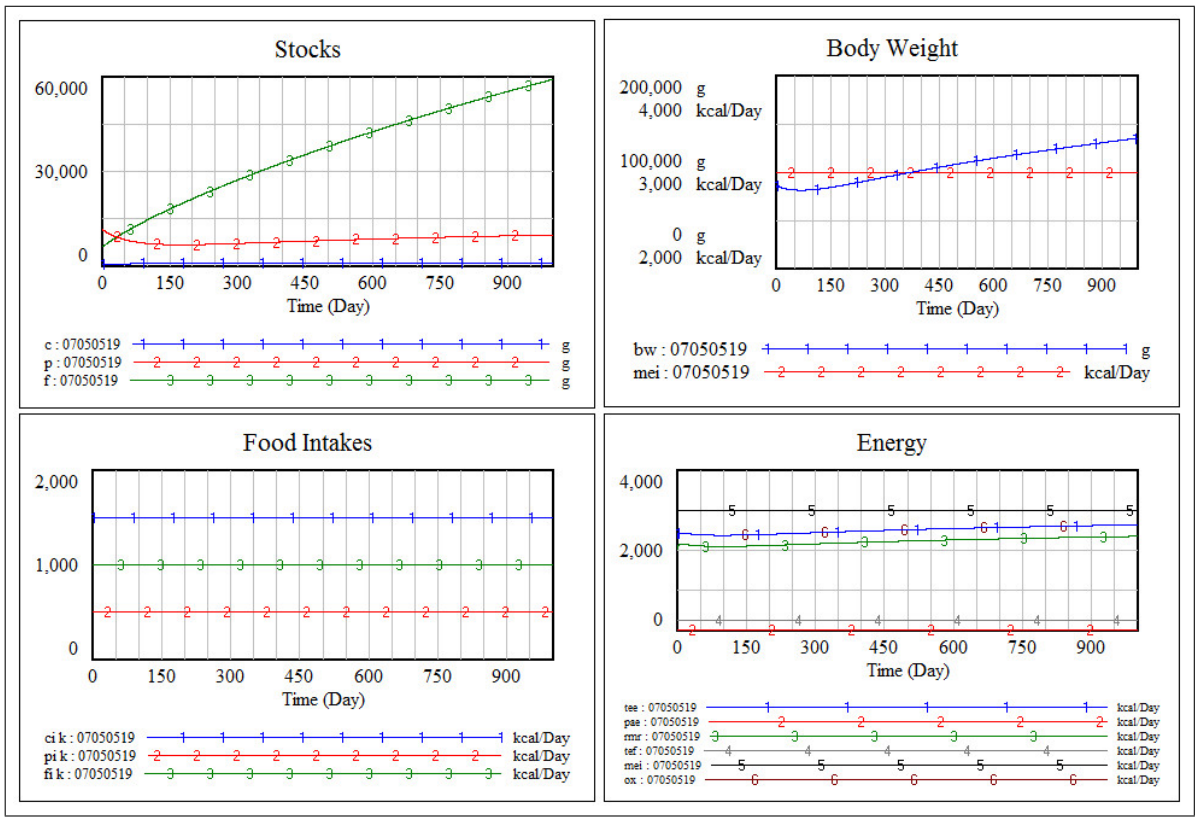

Figure 4.29. Dynamics of important variables in the case of extreme decrease in daily activity level

The person does no activity at all. This causes daily physical activity energy expenditure to be very low relative to other components of energy expenditure. Daily food intake is in excess relative to daily energy expenditure $mei > tee$. Therefore the person gains weight very rapidly. Because of the two main negative feedback loops in the model, weight gaining process follows goal seeking behavior. The behavioral pattern is realistic because the two negative feedback loops of resting metabolic rate and physical activity expenditure counteract to the rate of increase in the body weight. They slow down the rate of change of the body weight.

The low level of physical activity causes the physical activity expenditure to become very low. This causes the oxidation rates of energy stores to become very low. This causes a high level of net inflow to the energy stocks which causes the fat stock and body weight to increase. The increase in the body weight increases the physical

activity expenditure which increases the oxidation rate of fat stock. The increases in the oxidation rates cause the rate of growth of the fat stock to decrease. In conclusion, the rate of growth of the body weight slows down.

4.4.2.4. Experiments with Different Protein Intakes. *Extreme Increase in Daily Protein Intake:* In this experiment, daily protein intake is set to an extremely high value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.30

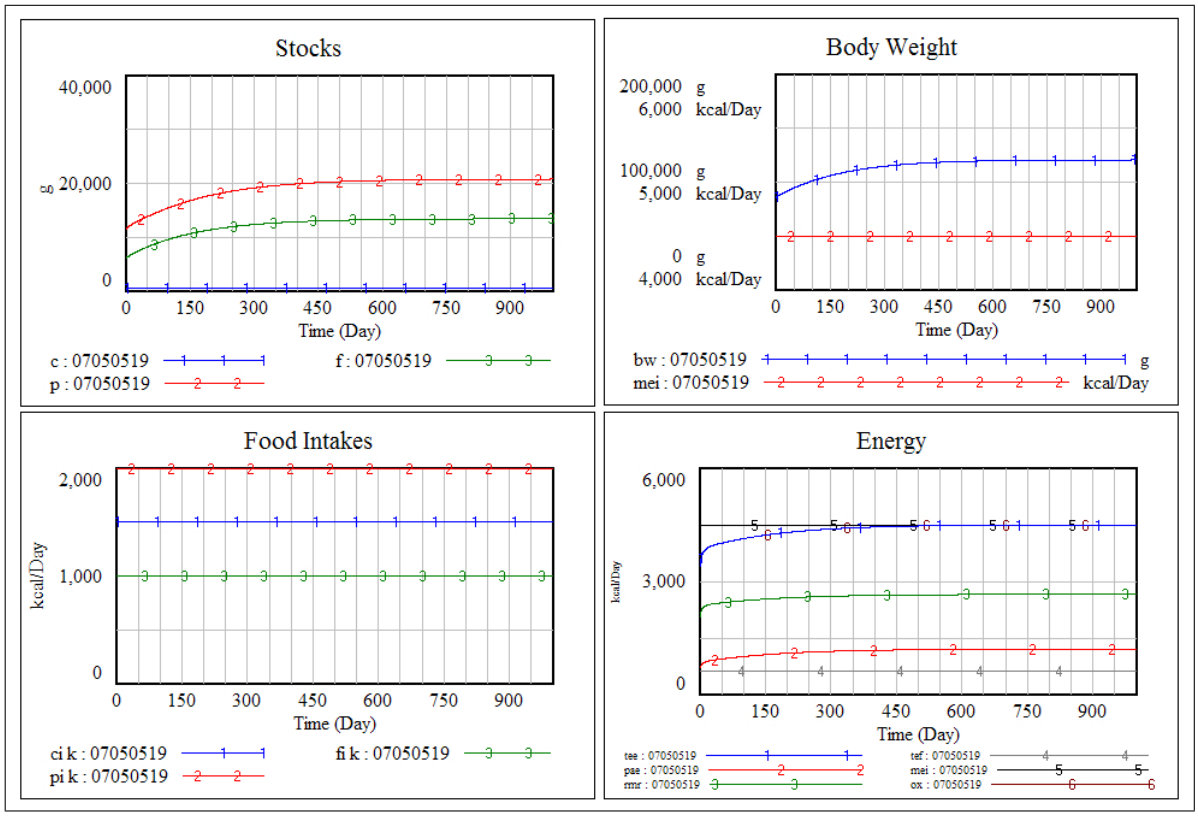

Figure 4.30. Dynamics of important variables in the case of extreme increase in protein intake

The person eats a lot of protein. This causes daily energy intake (mei) to be much higher than daily energy expenditure $mei > tee$. Therefore the person gains weight. Because of the two main negative feedback loops in the model, weight gain process follows goal seeking behavior.

Note that, some of the gained weight goes to the fat stock and some goes to the protein stock. This is partially realistic behavior. The protein stock should grow when there is more muscle activity and enough protein intake. In this scenario, there is some additional muscle activity due to the increase of the body weight. As the body weight increases, the muscles work harder to carry their normal work load. But the amount of the growth of the protein stock in this experiment may be higher than the amount of the growth in the reality.

Extreme Decrease in Daily Protein Intake: In this experiment, daily protein intake is set to an extremely low value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.31

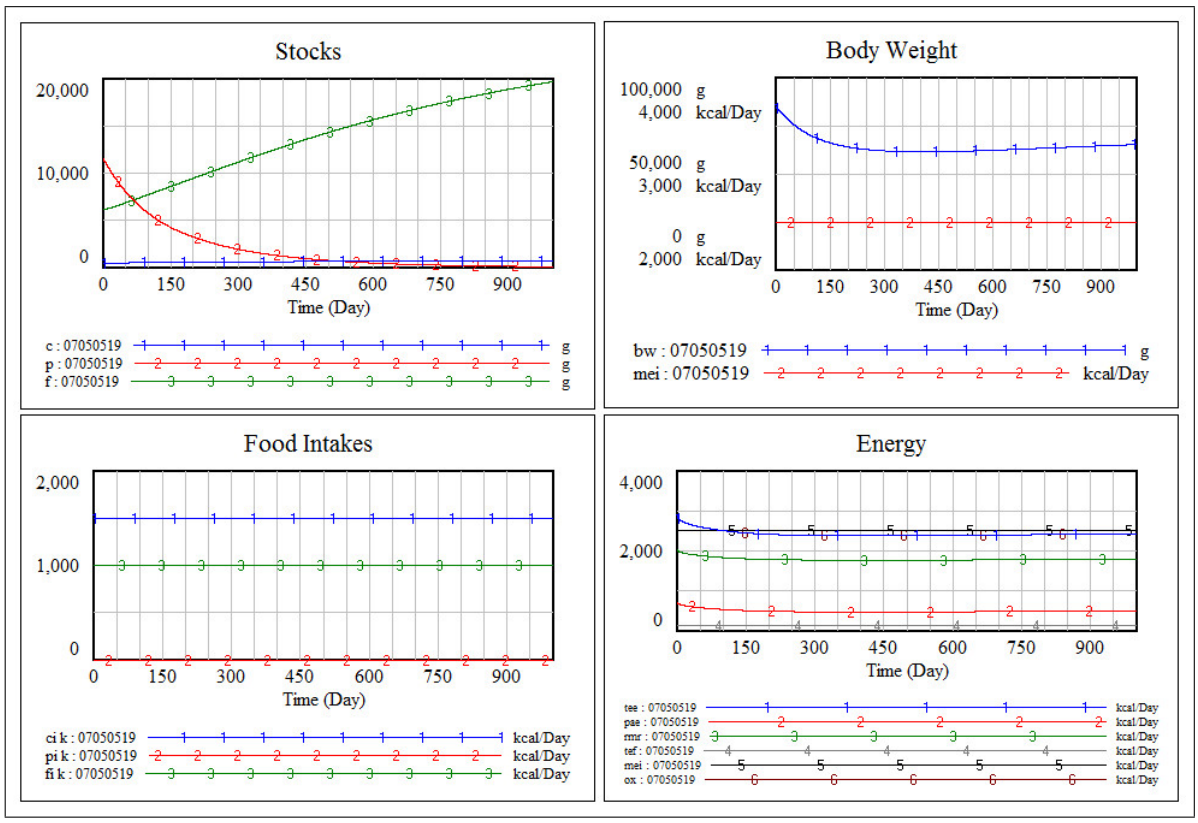

Figure 4.31. Dynamics of important variables in the case of extreme decrease in protein intake

The person eats no protein at all. This causes daily energy intake (*mei*) to be

slightly less than daily energy expenditure $mei < tee$. Therefore the person loses weight for the some time. Most of the lost body weight is due to the loss in the protein stock. Since the protein stock is the main substance in lean tissues and lean tissues have very high metabolic rate, the resting metabolic energy expenditure (rmr) substantially. Moreover physical activity energy expenditure decreases in this period too due to the loss in body weight. As a result, total energy expenditure (tee) decreases substantially for some time. Due to the significant reduction in total energy expenditure, daily energy intake becomes higher than daily energy expenditure ($mei > tee$). Then body weight recovers itself slowly by storing excess energy as fat in the fat stock.

The pattern of behavior in this scenario is partially realistic. The realistic part is the loss of body weight due to the decrease in resting metabolic rate because lean tissues are responsible for the major part of the resting metabolic rate. But there is a limit to the loss in the protein stock in reality which is not reflected in the model. After certain amount of loss of protein, the organism will certainly die because proteins are essential building stones of the body that are responsible for nearly every life process. So, there is a lower limit for the loss in the protein stock in real life which is not present in the model.

4.4.2.5. Experiments with Different Fat Intakes. *Extreme Increase in Daily Fat Intake:* In this experiment, daily fat intake is set to an extremely high value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.32

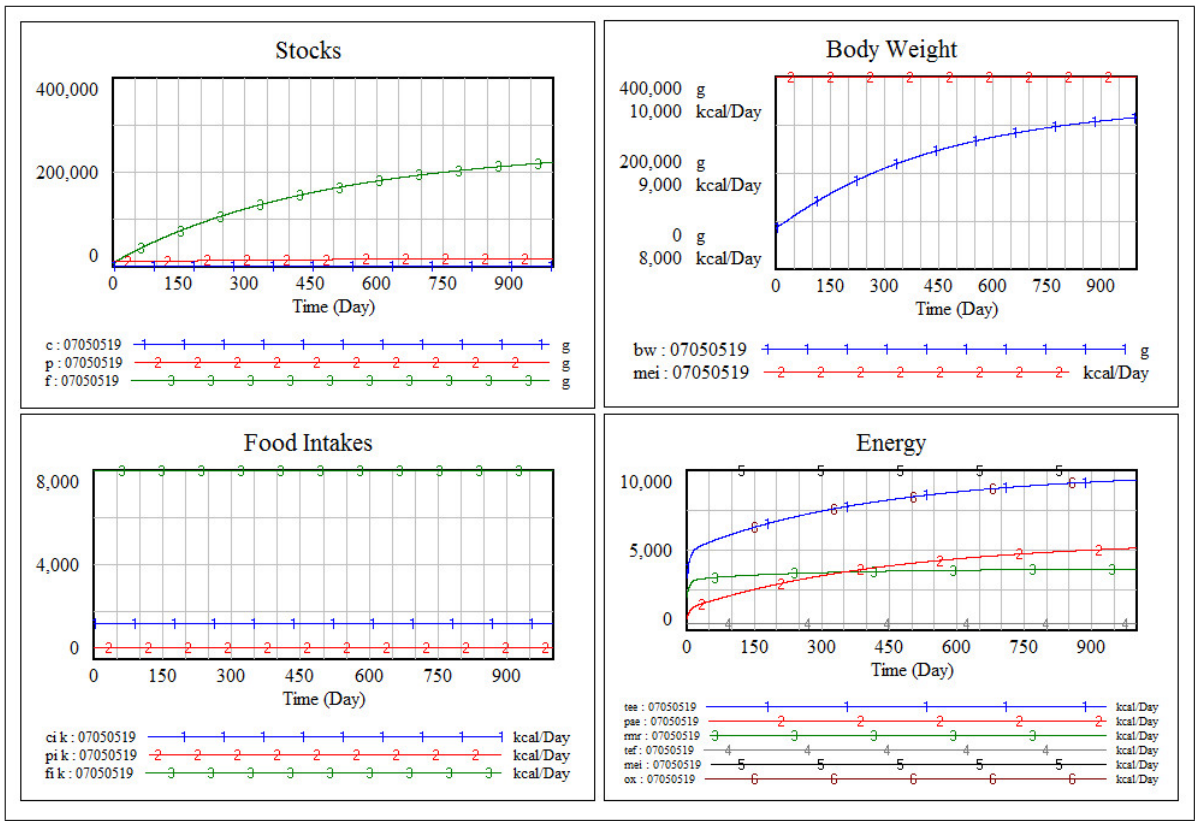

Figure 4.32. Dynamics of important variables in the case of extreme increase in fat intake

The person eats a lot of fat. This causes daily energy intake (*mei*) to be much higher than daily energy expenditure $mei > tee$. Therefore the person gains weight very rapidly. Because of the two main negative feedback loops in the model, weight gain process follows goal seeking behavior.

Note that all of the gained weight goes to the fat stock not to the protein stock. This is realistic behavior. The protein stock should grow only when there is more muscle activity. In this scenario there is some additional muscle activity due to increase in body weight. But the amount of protein intake constrains the protein stock from growing. Therefore all of the gained weight goes to the fat stock.

Extreme Decrease in Daily Fat Intake: In this experiment, daily fat intake is set to an extremely low value, see Table 4.24. The dynamics of the simulation is shown in

Figure 4.33

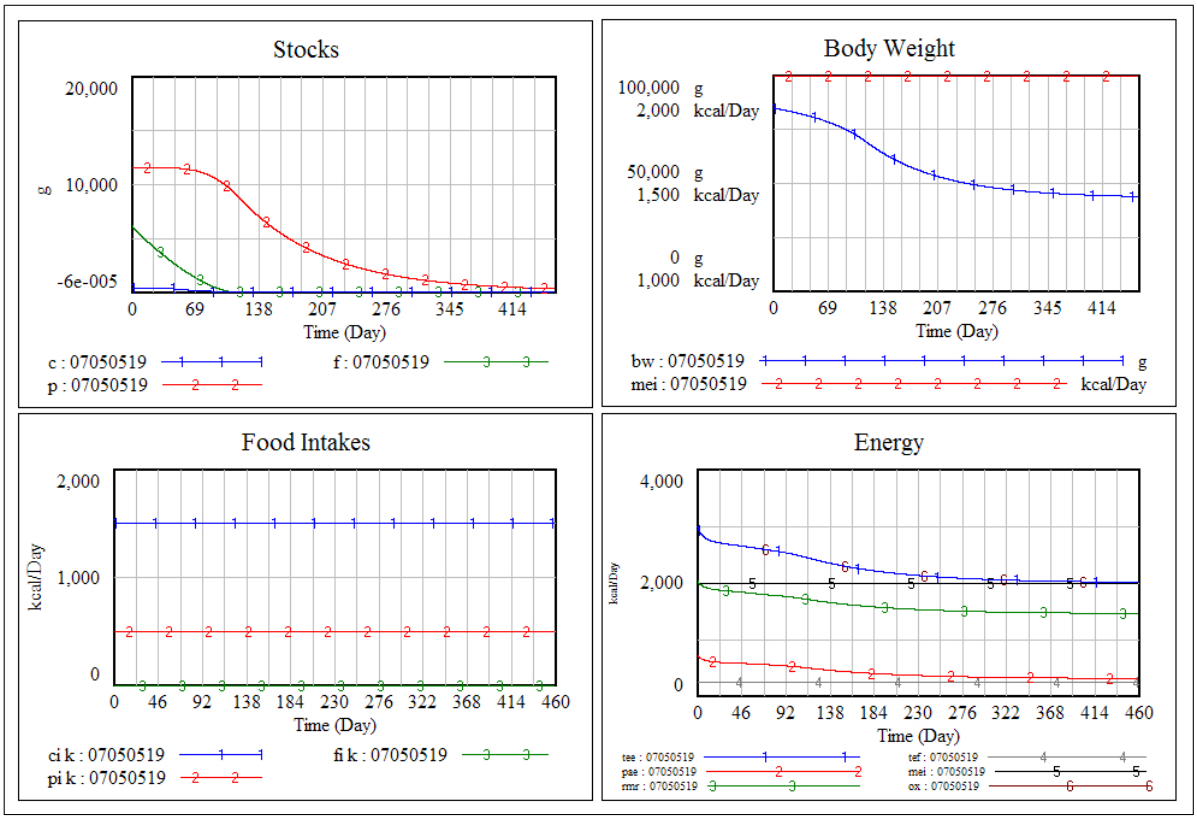

Figure 4.33. Dynamics of important variables in the case of extreme decrease in fat intake

The person eats no fat at all. This causes daily energy intake (mei) to be much less than daily energy expenditure $mei < tee$. Therefore the person loses weight for the some time until they become equal ($mei = tee$). Most of the lost body weight is due to the loss in the fat stock. Since the fat stock has very low metabolic rate, the resting metabolic energy expenditure (rmr) does not change substantially. But after a while (around day 100), the protein stock starts decreasing as well. This is due to the lack of energy intake to cover daily energy expenditure. When the fat stock becomes very low, then the body has to burn some of its protein stock. The loss in the protein stock increases the rate of the fall in resting metabolic rate since lean tissues have very high level of metabolic rate. After a while, energy expenditure becomes equal to the energy intake. Then the body weight and levels of the stocks stay stable.

The behavior of the carbohydrate stock shown in Figure 4.33 shows also the carbohydrate sparing effect of the fat stocks. The carbohydrate stock stays nearly stable for the beginning of the simulation run. When the fat stock becomes very low, then the body spends its carbohydrate stock.

The pattern of behavior in this scenario is partially realistic. It shows the protein sparing effect of the fat stocks when there is sufficient fat reserve. The rate of the decrease in the resting metabolic rate becomes accelerated after the body starts losing from the protein stocks. This is also a realistic behavior of the model. The unrealistic part of the behavior is that the organism does not die in the simulation. In real life, after certain amount of protein loss, the organism cannot sustain its vital activities, and it dies.

Note that, although all the macronutrient stocks of the body vanish, the body weight has a lower limit. This is due to the assumption of constant extracellular water *ecw* and some amount of constant intracellular water *ciw* in the model. Figure 4.34 shows the dynamics of the variables that make up the body weight.

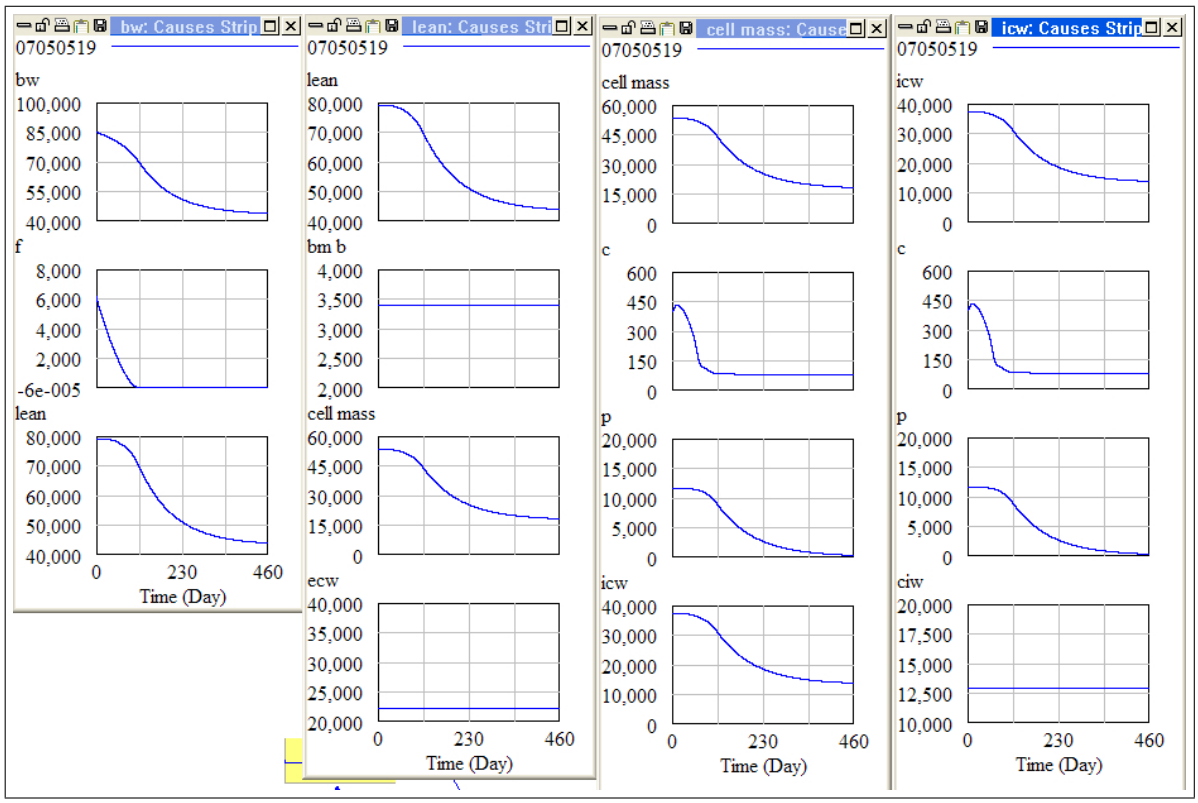

Figure 4.34. Components of body weight: Constant extracellular water *ecw* and constant part of the intracellular water *ciw* cause the body weight to have a lower limit when other stocks vanish.

4.4.2.6. Experiments with Different Carbohydrate Intakes. *Extreme Increase in Daily Carbohydrate Intake* In this experiment, daily carbohydrate intake is set to an extremely high value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.35

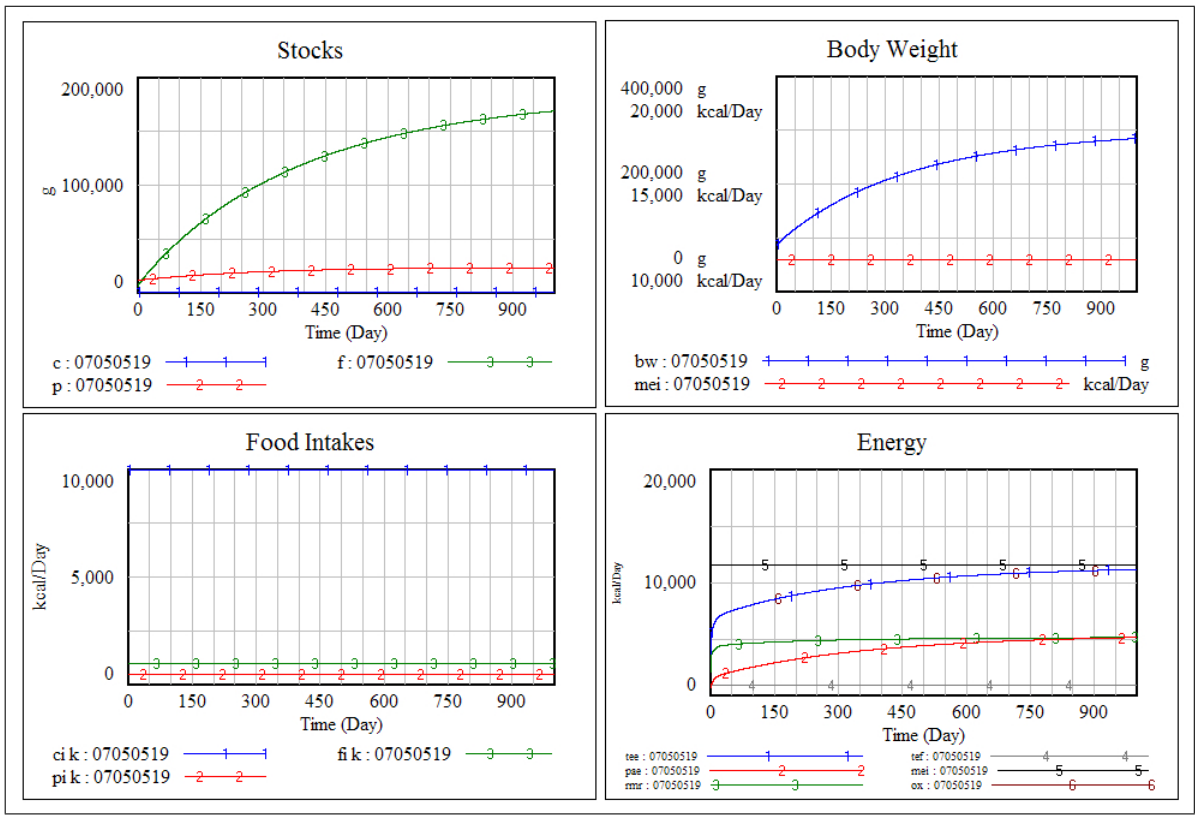

Figure 4.35. Dynamics of important variables in the case of extreme increase in carbohydrate intake

The dynamics of the system in this scenario is very similar to the experiment "Extreme Increase in Daily Fat Intake" explained in Section 4.4.2.5. The person eats a lot of carbohydrate. This causes daily energy intake (*mei*) to be much higher than daily energy expenditure $mei > tee$. As a result, the body weight increases very rapidly. Because of the two main negative feedback loops in the model, weight gain process follows goal seeking behavior.

Note that most of the gained weight goes to the fat stock not to the protein stock. This is realistic behavior. The protein stock should grow only when there is more muscle activity. In this scenario there is some additional muscle activity due to increase in body weight. Here the behavior of the system differs from the experiment "Extreme Increase in Daily Fat Intake". In the previous experiment, there is no increase in the protein stock at all; all excess energy is stored as fat. But in this experiment,

there is some increase in the protein stock though most of the excess energy is stored as fat. The reason behind the difference is the constraining effect of the protein inflow. In the previous experiment, the protein intake cannot supply excess amount of the protein inflow required for growth. In this experiment, although the protein intake is the same as the previous experiment, there is less protein outflow due to the decrease of gng protein. The decrease in gng protein is due to the negative effect of carbohydrate intake on gng protein. So consequently, there is a growth in the net protein inflow of the protein stock.

Extreme Decrease in Daily Carbohydrate Intake: In this experiment, daily carbohydrate intake is set to an extremely low value, see Table 4.24. The dynamics of the simulation is shown in Figure 4.36

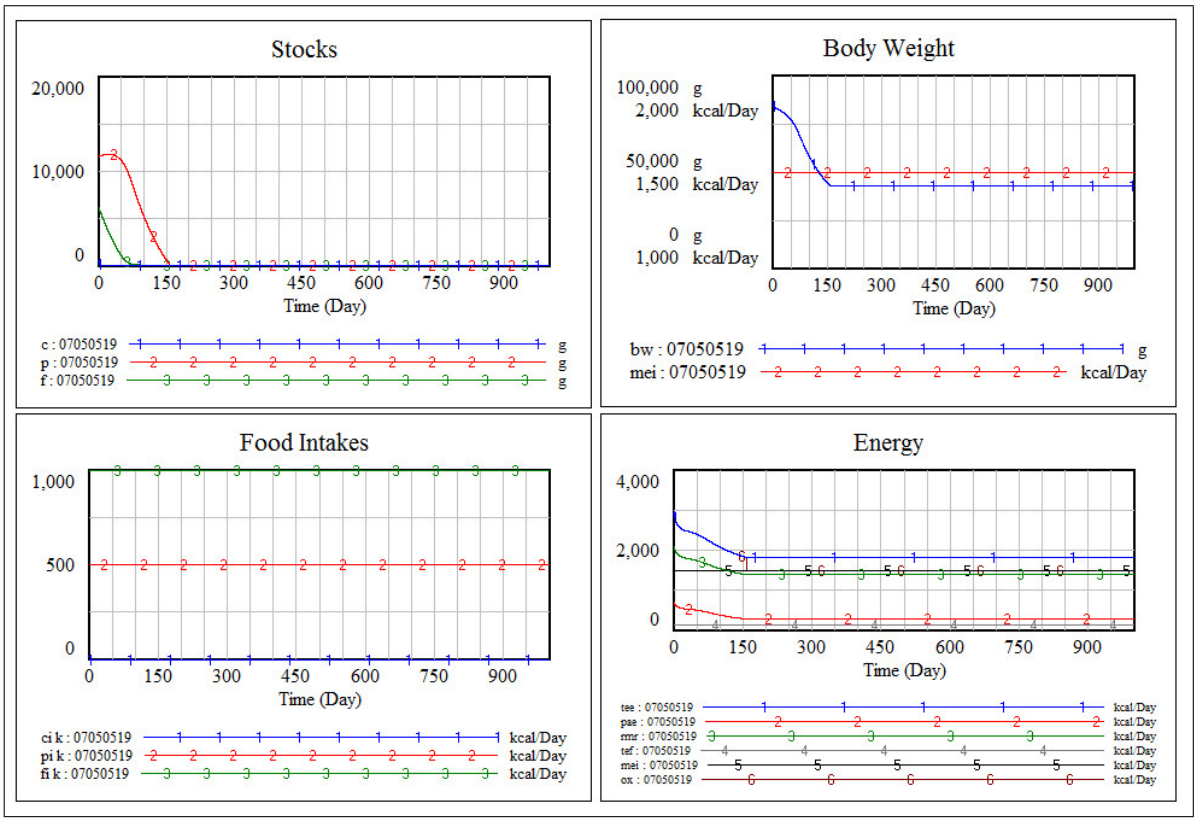

Figure 4.36. Dynamics of important variables in the case of extreme decrease in carbohydrate intake

This scenario is very similar to "Extreme Decrease in Daily Fat Intake" described in Section 4.4.2.5. The person eats no carbohydrate at all. This causes daily energy intake (mei) to be much less than daily energy expenditure $mei < tee$. Therefore the person loses weight for the some time until they become equal ($mei = tee$). Most of the lost body weight is due to the loss of the fat stock. Since the fat stock has very low metabolic rate, the resting metabolic energy expenditure (rmr) does not change substantially. But after a while (around day 50) protein stock starts decreasing as well. This is due to the lack of energy intake to cover daily energy expenditure. When fat stock becomes very low, then the body has to burn some of its protein stock. The loss of protein stock increases the rate of the fall in resting metabolic rate since lean tissues have very high level of metabolic rate. After a while, energy expenditure becomes equal to the energy intake. Then the body weight and levels of the stocks stay stable.

One small difference in the behavior of the system between this scenario and "Extreme Decrease in Daily Fat Intake" is the behavior of the protein stock. In the previous experiment, the protein stock has a lighter rate of change. It stays nearly stable until day 100. Then it starts decreasing with a very low rate. In this scenario, the protein stock changes in a sharper rate. The reason behind the difference in the rate of change of protein stock is that carbohydrate intake falls from 1500 kcal/day to 0 in this scenario whereas fat intake falls from 500 kcal/day to 0. So, the amount of loss in energy intake is higher in this scenario.

The pattern of behavior in this scenario is realistic. It shows the protein sparing effect of fat stocks when there is sufficient fat reserve. The decrease of the resting metabolic rate becomes accelerated after the body starts losing protein stocks is also a realistic behavior of the model. There is a lower limit for the body weight in this simulation run because of the same reasons as explained in Section 4.4.2.5.

4.4.2.7. Experiments with Adaptive Thermogenesis. *Leaving Adaptive Thermogenesis Out:* In this experiment, the effect of adaptive thermogenesis on physical activity energy expenditure and resting metabolic rate is left out.

The existence and effect of the adaptive thermogenesis is a controversial topic. To see the effects of the adaptive thermogenesis on the dynamics of the body weight, we leave the adaptive thermogenesis out of the model, and run the experiments given in Table 4.24.

Adaptive thermogenesis represents the ability of the body to adapt itself to the changes in the food intake. During starvation all the activities are slowed down, and during energy excess all the activities are accelerated by the effect of the adaptive thermogenesis. By leaving adaptive thermogenesis out, the model is supposed to produce the same behavior pattern. However, the variables can reach their goals more rapidly, and the amplitude of the change triggered by the difference of energy intake and expenditure becomes larger.

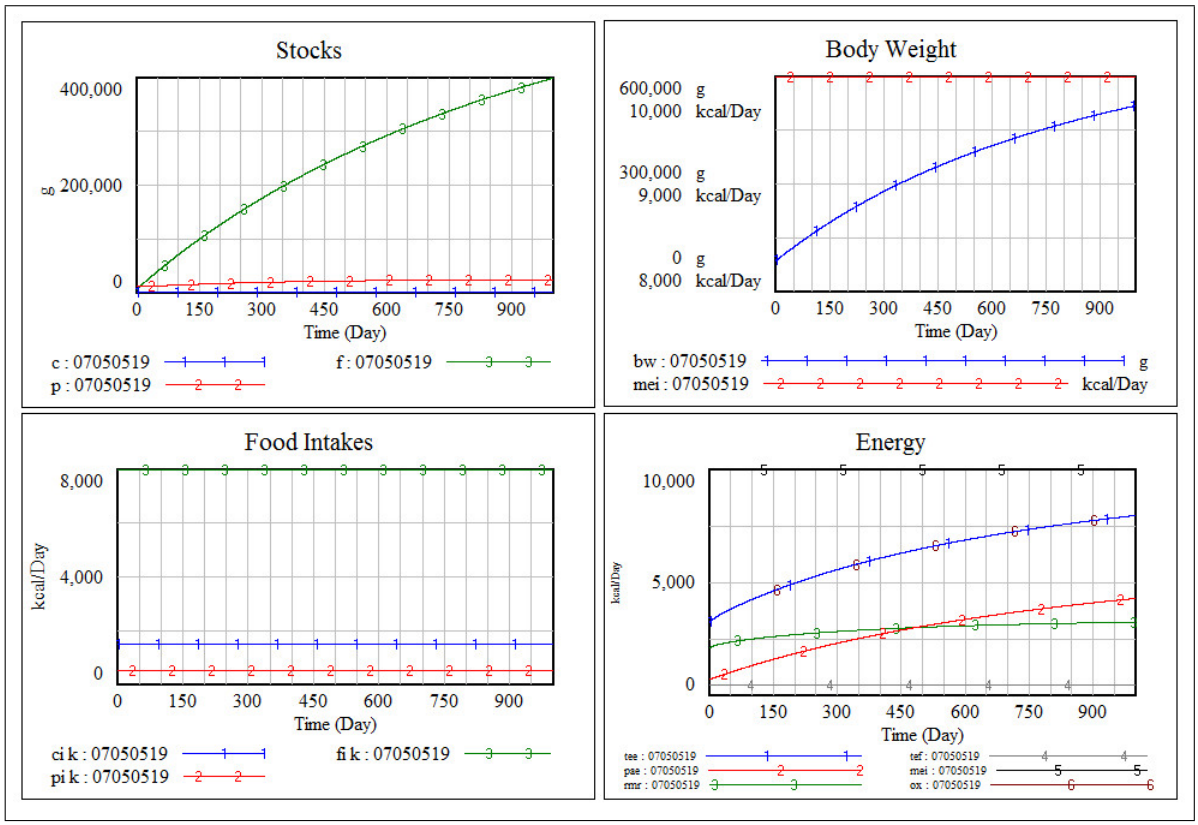

Figure 4.37. Dynamics of important variables in the case of leaving adaptive thermogenesis out and extreme increase in fat intake

Figure 4.37 shows the dynamics of important variables in the case of extreme increase in fat intake where the adaptive thermogenesis is left out. The comparison of the dynamics in Figure 4.37 with the dynamics in Figure 4.32 shows that inclusion of the adaptive thermogenesis variable damps the growth of the body weight.

Figure 4.38 shows the dynamics of important variables in the case of decreasing activity level where the adaptive thermogenesis is left out. The comparison of the dynamics in Figure 4.37 with the dynamics in Figure 4.25 shows that inclusion of the adaptive thermogenesis variable has no effect on body weight in this experiment.

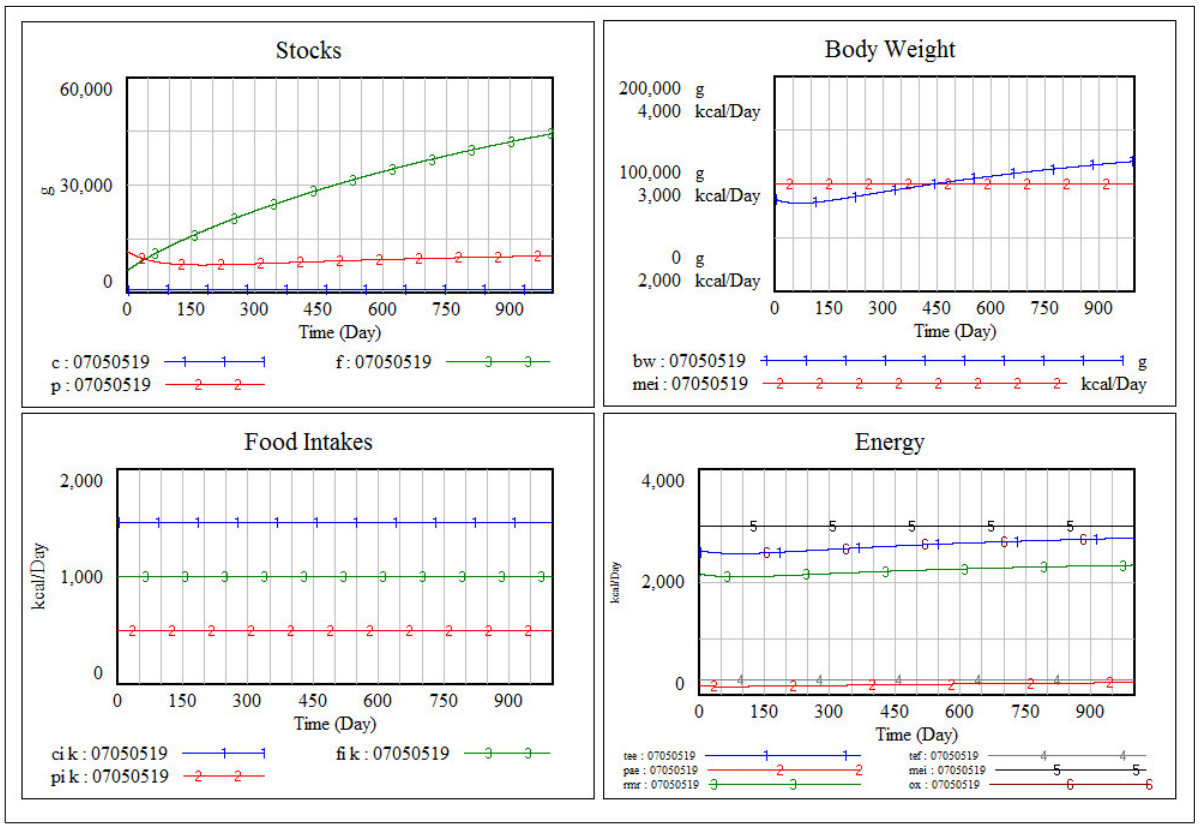

Figure 4.38. Dynamics of important variables in the case of leaving adaptive thermogenesis out and decreasing activity level

In conclusion, inclusion of the adaptive thermogenesis damps the dynamics of the stock variables when the dynamics is the consequence of the change of the food intake rates. However, inclusion of the adaptive thermogenesis has no effect on the dynamics of the stock variables when the dynamics is the consequence of the change

of the activity level.

5. WEB GAME APPLICATION DEVELOPMENT

The second part of this research project is the development of a web application that produces and runs simulation games for system dynamics models. The application (called Mashap) is designed to run any system dynamics model developed in popular model building tools such as Vensim or Stella. The application has three basic features:

1. Conversion of models built in a system dynamics modeling software into Python syntax
2. Running simulation of system dynamics models
3. Automatic user interface generation to construct web simulation games

Currently, Mashap supports only Vensim and Mapsys as input files. Support for Stella and Powersim are planned. Mashap can run simulation by itself and it can build a gaming environment for simulation models automatically.

Mashap can produce web based, simulation games for any system dynamics model. In this research project, we use Mashap to develop a web based game for the body weight dynamics model as a proof of concept application. Mashap produces the web based game automatically without requiring any additional work. We expect that system dynamics researchers will use Mashap to build web gaming environments for their own models.

5.1. Rationale for the Web Application

Currently Vensim, Stella, Forio and some other software run system dynamics simulation models on web. But these tools have a very important constraint: They are commercially licensed, closed source products.

Commercial licensing of software does not provide enough flexibility to extend existing programs. This might not be a problem for most of the software users. But

for innovative, pioneering communities, the requirements space of users extend to new frontiers continuously. It is crucial that new tools are developed in parallel to research ideas.

In the open source world, platforms are more important than end software. Programmers other than the original authors of the software can build their own tools upon the platform that have very different aims and value composition than the original tool.

Another benefit of open source software is that the maintenance and support of the software can become easier and more rapid. This actually depends on the acceptance of the open source software by other programmers. If it is accepted, then the burden of maintaining and supporting the software is distributed over a wide community of programmers.

Web based games have several important advantages over desktop games:

1. Access to general public is much easier for web applications. Anyone who has access to a browser can play the game.
2. Access to domain experts for validation, testing, dissemination, or model development is much easier.
3. Maintenance of the software is much less costly for programmers. There is no burden for updates, patches, or support for different operating systems.

An additional benefit of web based game engine Mashap is that it can become a base for group modeling. We assume that having effective tools for group modeling is crucial for group modeling to realize its actual potential. Current software are not designed for collaborative model development. Desktop software is not suitable for collaboration by its nature. Mashap is not developed for collaboration either, but it is possible to build collaborative features upon Mashap.

We developed Mashap on python platform. Python is a dynamically typed programming language. Dynamically typed languages have higher expression power than

languages that have static typing like Java, C, or C Sharp. Although statically typed languages have important advantages in many areas, dynamic languages increase the productivity of the programmer in areas like mathematical programming or web applications that require lots of text manipulation.

5.2. Requirements

There are two distinct types of users for Mashap:

- Model builders
- Game players

Model builders and game players have different goals. We assumed that model builders have the following goals and expectations in using a web based game engine like Mashap:

- To make the simulation available to a lot of people easily
- To continue using their accustomed tools such as Vensim or Stella for model building
- To verify the correctness of the simulations
- To get the web game up instantly without spending a lot of labor for customization or configuration of the software
- To access to the run data of the games of the end users
- To experiment the same model with different parameters or other factors

We assumed that game players have the following goals and expectations in using a web based simulation game built upon Mashap:

- To see the effects of their decisions instantly
- To get convinced of the validity of the game
- To learn something about a complex system or to develop an effective strategy to deal with the problems

- To see effects of their current behavior in long term
- To socialize over games

The first four goals are actually valid for any system dynamics game, not only for web games. To socialize over games is a goal specific to web games. Now, people play games not only for the enjoyment they get from the game itself; many popular games have some kind of socializing feature. Game players like to compare themselves with other people or to enjoy playing games together with their friends. Mashap currently does not have any feature to satisfy socialization requirements, but it has the infrastructure to support socialization features.

5.2.1. Use Case Scenarios

Functional requirements are described as use case scenarios. There are two important use cases for Mashap:

1. Model builder uploads the model and prepares the game.
2. Game player plays the game.

Use Case: Uploading the model and preparing the game. The primary actor of the use case is the model builder.

Normal flow:

1. The user uploads the model file of Vensim to the system (Mashap).
2. The system verifies the syntax of the model, converts the model to its own language, and presents the equations of the model to the user for him to confirm them.
3. The user confirms the equations and enters the game title and description.
4. The system asks the user the parameters that are to be set by the game player at the beginning of the game. For this purpose, system presents to the user the

list of constant variables in the model.

5. The user selects and submits the variables to be set initially by the game player at the beginning of the game.
6. The system asks the length of the time between two decision steps, total game length. User determines these parameters.
7. The system asks the decision variables of the game. For this purpose, system presents to the user the list of constant variables in the model.
8. The user selects and submits the decision variables.
9. The system asks the variables that are going to be presented as output in the user interface of the game. The user determines and submits those variables.
10. The system asks game management settings such as email address of the model builder and notification preferences.
11. The system presents to the user the link to access the produced game.

Use Case: Playing the game. The primary actor of the use case is the game player.

Normal flow:

1. The system presents the introduction of the game to the game player and asks for the values of the initial variables.
2. The game player states the values of the initial variables. The steps between 3 and 5 repeat until the end of the game.
3. The system presents the game user interface and the current state of the game in graphics. The system asks the values of the decision variables.
4. The game player states the values of the decision variables for the current period.
5. The system simulates the system until the next decision step.

5.2.2. Domain Model

In this section, we will explain the domain model beneath the software. Domain model is an internal representation of the target domain. In our case, the target domain covers system dynamics models, simulations, and games. The domain model shown in Figure 5.1 is an abstraction of the domain that expresses what we know about the relationships between the concepts of the domain. [16]

Historically, there are lots of different notations to express domain knowledge in object oriented analysis methodologies. UML is the most common standard notation currently. The domain model of the Mashap software shown in Figure 5.1 is not drawn according to a specific notation because we think that the notation is not important. The idea that the diagram is intended to express should be clear to the domain experts who are usually not trained in any diagramming notation. The diagram shows domain objects and their relationships for system dynamics models, simulations, and games.

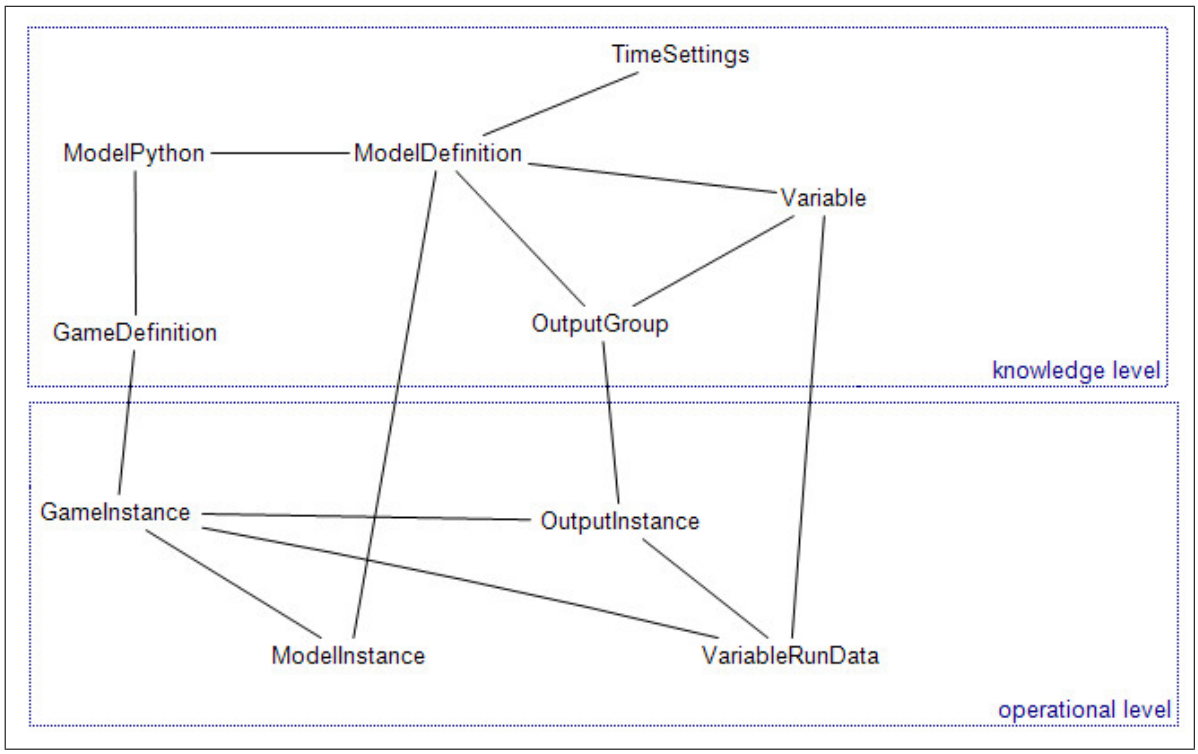

Figure 5.1. Domain model of the Mashap software

There are two layers in the domain model: knowledge level and operational level. At the operational level the model expresses the events that occur during the operation of the domain. At the knowledge level the model expresses the general rules that govern this structure. Instances of the concepts in the knowledge level constrain the configuration of instances in the operational level [17]. In the model shown in Figure 5.1 the instances of OutputInstance are constrained by the relationship between OutputGroup and Variable.

5.2.3. User Interface

The user interface of a game depends on the decision variables and the output graphics. Since every system dynamics model has different variables, every simulation game has a different user interface. In popular system dynamics software such as Vensim or Stella, the model builder develops a unique user interface for each game. This method is an effective way to customize every game for its own, unique requirements.

But custom development of user interfaces for every simulation game is not easy to implement for web based games.

Customization of user interfaces by the model builder has several technical requirements that will increase the work load for the development of the software immensely. Moreover, this feature puts a lot of work load on the model builder as well. The model builder will have to learn a lot of additional technical details and maybe a new programming language too as it is the case for Forio. Model building is a very difficult job already. We do not want to make the job of the model builder any more difficult. There is a trade off between usability and customization. There is value in customization, but we value usability more than customization.

Actually, there is another choice that has the advantages of usability and customization together: automatic generation of the game user interface from the current Vensim or Stella file. This feature does not put any work load on the model builder, but it requires some additional programming work.

So currently, Mashap does not require the model builder to develop a custom user interface for the simulation game. It generates the user interface automatically by taking some inputs from the model builder as described in Section 5.2.1 in the use case "uploading the model and preparing the game". The user interface of a game depends totally on the choices of the model builder in this use case. Therefore the user interface of a game is built dynamically by Mashap during run time of the software.

Table 5.1 presents the views (screens) of the user interface, required inputs from the user, and the corresponding step numbers in the use case "uploading the model and preparing the game" described in Section 5.2.1.

Table 5.1. Mapping between views, inputs from the user, and use case steps for use case "uploading the model and preparing the game". (The view names are the same in the code of the Mashap software.)

Screen (view method in the code)	Inputs	Use case step
upload_model	model_file	1
verify_model	game_description	3
define_initial_params	initial_parameters	5
define_time_settings	time_settings	6
define_decision_params	decision_parameters	7
define_outputs		9
define_new_output_group	output_group	9
define_game_management_settings	management_settings	10
access_data		11

Table 5.2 presents the views (screens) of the user interface, required inputs from the user, and the corresponding step numbers in the use case "playing the game" described in Section 5.2.1.

Table 5.2. Mapping between views, inputs from the user, and use case steps for use case "playing the game". (The view names are the same in the code of the Mashap software.)

Screen (view method in the code)	Inputs	Use case step
introduce_game		1
input_initial_values	values for initial_parameters	2
game_basic	values for decision_parameters	4
game_results		

5.3. Design

This section describes the design ideas of the Mashap software. The intent of this section is to document the software to the programmers for them to maintain or extend the software easily. The design documentation covers the modules of the simulation and the web user interface of Mashap. There are some other assisting tools and software developed or used during the development of Mashap. But their usage is documented in this section.

Mashap is the name of the software that lets the model builders to prepare a web based simulation game from the models developed in popular system dynamics model building tools such as Vensim. All the code and other documents that make the Mashap project are available from `http://code.google.com/p/mashap/` The project is available to contributions of other people through open source Apache 2.0 license.

5.3.1. System Overview

Mashap consists of three basic functionalities:

1. Conversion of models built in Vensim into other formats
2. Simulation of system dynamics models
3. Automatic web user interface generation

The system currently supports conversion of Vensim files into Python format. Support for import from Mapsys and export to Matlab and Excel are partially completed. Support for import from Stella and Powersim model files are planned.

Simulation of system dynamics models are done by a numerical integration solver implementing 4th order Runge-Kutta method for solving the initial value problem $\frac{d\vec{y}}{dt} = F(t, \vec{y})$ and $\vec{y}(0) = \vec{\alpha}$.

Automatic generation of web user interface of the game is done according to the decisions made by the model builder in the preparation of the game. The model builder tells the system three decisions:

1. Initial parameters: The variables of the model that will be initialized at the beginning of the game
2. Decision variables: The variables that will be fed into the system
3. Output groups: The outputs of the model at each decision step to be produced by the system

The system builds the game user interface according to these three decisions automatically.

5.3.2. Design Issues

Assumptions and Dependencies. There are several assumptions that guided the design decisions of Mashap:

1. Model builders won't prefer to learn another model development tool instead of the popular software such as Vensim or Stella.
2. Model builders won't prefer to learn any special language or configuration format for game preparation.
3. Game players will possibly prefer to enter their decisions inside the game in various ways such as by a batch file prepared in Excel or by some convenient graphical tools.
4. Some modules of Mashap can possibly be used as an infrastructure for development of other software in some domains such as modeling, gaming, or validating.
5. The simulation engine of Mashap can possibly be transferred to client side due to capacity limits of server's computing power.
6. The database of Mashap can possibly be transferred from relational model to non-relational model such as Google's BigTable due to capacity limits.

One of the primary assumptions under the design decisions of Mashap is that model builders are accustomed to their current model development tools strongly and they would't prefer to learn another tool doing the same job in a different way. Therefore, we do not try to imitate any core functionalities of model development tools such as Vensim or Stella. Another implication of this assumption is that model builders won't like to enter the equations of the model by hand or any other method. They would like to use their current model building environment. Mashap will be used as a complementary tool that is going to be used for purposes that are not currently handled by popular software in an easy or standard way.

Another assumption about preferences of model builders is that they won't prefer to learn any special language or a cumbersome configuration format when preparing a game environment from their models. Forio is probably the mostly used software for web based simulation games in system dynamics domain. Forio requires the model builders to prepare the game by using a special language. This feature allows a lot of flexibility for the customization of the game user interface. We don't have any solid, empirical proof for our argument, but we assumed that usability is more important than flexibility.

Another assumption is that game players will possibly prefer to enter their decisions inside the game in various ways such as by a batch file prepared in Excel or by some convenient graphical tools. Web applications do not have the same interactivity level as desktop applications. Gaming environments prepared with Stella or Vensim has a lot of interaction with the game player. In web environment, the waiting time from submitting a decision until taking the response is so long that the users cannot get into the same flow of interaction as in desktop applications. Therefore, we assumed that the interaction between the user and the software can evolve to new methods. An alternative for the interaction method is entering the decisions as a batch in one step or in a few steps for the duration of simulation's run time. The batch data can be entered with an Excel file or it can be produced by the user on an interactive graphics that lies on the web application.

An assumption on the development of Mashap after this thesis study is that some modules of Mashap can possibly be used as an infrastructure for development of other software. Mashap is released under a free software license. Other programmers can extend the software for their own purposes. We guess that a group modeling tool in the style of social web applications or a tool that helps the researcher to perform an experiment with different parameters in a model can be built upon Mashap.

The assumptions above are all related to the possible changes in functionality of Mashap. There are some possible changes that involve the architecture of the software as well. One of them is that there can be severe capacity problems in the servers' computing power where Mashap is hosted because simulation requires a lot of computing power. The capacity problem might necessitate the transfer of the simulation engine from the server side to the client side. Therefore the architecture of Mashap should be decomposable into independently working modules. A simulation engine running on the web browser (client side), which is probably implemented in Javascript, should be able to communicate with Mashap's server components seamlessly.

Another assumption related to the capacity problems involve a possible change of the database from relational model to a non-relational model such as Google's BigTable. In web applications, the key problem is scalability. As a web application becomes more popular, the capacity constraints on the system gets harder to solve. On a relational database, adding more hardware resources does not add to the capacity of the system linearly. There are several constraints that limit the growth of the system [18]. Maintaining a web application that stores lots of data is very costly. So depending on the usage rate of Mashap, there might be a need to transfer the database of the system to a scalable model such as Google's BigTable.

Goals and Guidelines. There are several goals and guidelines that embody the design of the software. These goals include:

1. To keep all entities (such as class or function) small [19]

2. To use rich domain objects
3. To decide as late as possible when there is uncertainty [20]
4. Usability is more important than having lots of features

The four goals mentioned above are all interrelated. Keeping all entities small is a prerequisite for having rich domain objects. Also this practice helps to the design of a layered architecture. To decide as late as possible helps to limit the features in the application low, which helps to keep usability of the software high and to keep modules small.

In "Object Calisthenics", Jeff Bay tells that well written code is easy to understand, test, and maintain. To achieve these properties, there are code qualities that are commonly accepted: cohesion, loose coupling, zero duplication, encapsulation, testability, readability, and focus. To put these concepts into practice, Jeff Bay suggests several ways that we summarized as "to keep all entities small" [19]. In summary, these techniques help the programmers to eliminate the waste in the code.

Using rich domain objects is one of the goals that governed the design of Mashap. The opposite of this goal is an ineffective design pattern called "Anemic Domain Model". Anemic Domain Model is one of the most frequent design errors in software. Anemic domain objects are objects with names representing a concept from the domain space. But they don't have any responsibility of behavior. They are containers of data. This is contrary to the basic idea of object-oriented design, which is to combine data and behavior in order to make the data container (object) responsible for the business logic [16], [17].

To decide as late as possible is a principle in Lean Software Development. Lean Software Development suggests to make value flow at the pull of the user of the software. This implies to make the decisions in uncertain areas as late as possible. Lean Software Development claims that this principle helps to reduce waste in software development [20].

The fourth goal that governed the design of Mashap says that usability is more important than having lots of features. This is a controversial claim. But it is a choice we made in the design of the software. This goal is related to the minimalism and Unix philosophy in the programmers' communities.

5.3.3. Architectural Decisions and Strategies

The architectural decisions and strategies used in this thesis include the following:

1. Usage of Python platform
2. Usage of Django library as the web framework
3. Usage of Django-orm library as the data access framework
4. Design of service layer as the coordinator of business logic
5. Usage of domain driven design methodology for the distribution of responsibilities of the objects
6. To design in layered architecture

The rationale for using python platform is explained in Section 5.1. Django framework allows rapid development and clean design. It focuses on automating as much as possible. Django-orm library allows data access logic to be encapsulated in the domain objects. Design of service layer as the coordinator of business logic helps to keep presentation layer free of business logic and to keep domain layer focused on its own responsibility. Usage of domain driven design for the distribution of responsibilities of the objects helps to design the software cleanly and maintainable.

The sixth architectural decision mentioned above is to design the software in layered architecture. Layered architecture is one of the most common techniques that software designers use to decompose a software system. In layered architecture, each layer rests on a lower layer. The lower layer is independent of the upper layers. The upper layer is dependent on the lower layers [17].

5.3.4. System Architecture

Figure 5.2 shows the layers of the system architecture of Mashap. This diagram shows the core modules of Mashap. It does not include noncritical modules or external modules. Each box denotes one layer of the architecture. The modules that constitute the layer are put inside the box. The directed associations between the modules and layers denote the dependencies between them. The dependency of presentation layer on domain model is exceptional; therefore it is drawn as a dashed association.

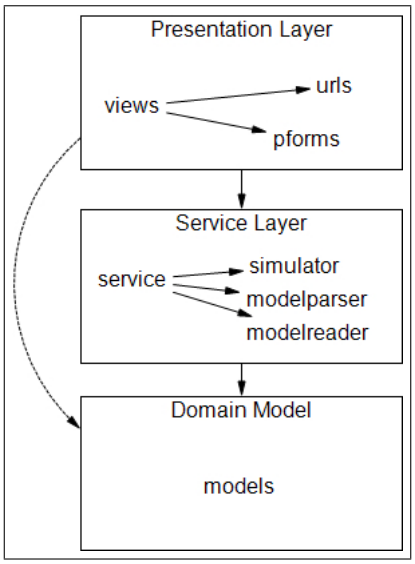

Figure 5.2. Layers of the system architecture of Mashap

The application consists of three basic layers. Presentation layer is responsible for everything related to the user interface such as presenting information to the user and getting user commands. Service layer is responsible for coordinating the business logic. It also contains processing logic that does not fit into domain objects such as performing the simulation or converting the model files from Vensim format to Python format. Domain model contains information about the domain. This is the core of the software. The state of the business objects is held here.

Presentation layer consists of three core modules: views, urls, and pforms. The module views is a thin layer that delegates the input received from the user to the

service layer and prepares the user interface that user gets. The module views delegates actual preparation of the user interface to pforms module and html templates. The module pforms produces form components in html pages. Form components consist of form widgets such as combo boxes or text boxes. The module pforms is responsible for checking submitted data against a set of validation rules and converting form data to the corresponding Python data types. The module urls is responsible for navigating between the web pages and views functions.

Nearly, every views function correspond to a web page. A views function consists of two parts: get and post. The post part is responsible for receiving the input of the user and delegating the control to the service layer. The get part is responsible for making the user interface, that the user will see, with the help of pforms module.

Service layer consists of four core modules and some small helper modules. The module service is a thin layer that is responsible for coordinating the business logic. It performs this with the help of the other three modules. The module simulator is responsible for performing the simulation. It delegates numerical integration to the helper module "run_kut4". The module modelreader reads a list of equations expressing a system dynamics model in Python syntax. The module reads the equations and builds the domain objects ModelPython and ModelDefinition. The module modelparser parses and converts a model file of Vensim into a list of equations in Python syntax.

Domain model consists of one module: models. The module models contains the definitions of the domain model objects shown in 5.1.

5.3.5. Policies and Tactics

This subsection describes design policies that do not affect the overall architecture of Mashap, but which affect the details of the implementation of Mashap. The policies used in the development of Mashap include the following:

1. Naming conventions and style guide
2. Topological ordering of equations
3. Outputting the equations with the help of strategy design pattern
4. Usage of delegation pattern to encapsulate subtypes of the objects
5. Uniform access principle and encapsulation of internal state of domain objects
6. Caching data in properties
7. Declarative expression of data queries
8. Usage of the method unicode for object description
9. Programming by convention practices
10. Readable and searchable url addresses
11. Usage of dynamic forms

Naming conventions and style guide. Throughout the Mashap code, we follow as much as possible the conventions described in PEP-8 Style Guide for Python Code. The goal in naming any entity is to make the entity self-explanatory without requiring any further comment on the code. Comments are written mostly to explain the rationale behind some part of the code.

Topological ordering of equations. The equations in Vensim's model file are in an arbitrary order. This is not suitable for simulation algorithm because the simulation algorithm advance step by step for each variable. All the dependent terms of a variable must be calculated before calculating the variable itself. Therefore the equations must be fed into the simulation algorithm in the order of dependencies. The ordering is done with the topological sorting algorithm. The pseudo code of the algorithm is given in Table 5.3.

Table 5.3. Pseudo code for topological sorting algorithm (Adopted from Cormen [21])

```
L = Empty list that will contain the sorted elements
S = Set of all nodes with no incoming edges
while S is non-empty do
    remove a node n from S
    insert n into L
    for each node m with an edge e from n to m do
        remove edge e from the graph
        if m has no other incoming edges then
            insert m into S
if graph has edges then
    output error message (graph has at least one cycle)
else
    output message (proposed topologically sorted order: L)
```

The variables in the model equations correspond to the nodes in the topological sorting algorithm. The algorithm is implemented by the topological_sorting method in graph module. The code of the algorithm is given in Table 5.4. The method returns the list of variables in the dependency order.

Table 5.4. Actual code implementing topological sorting algorithm

```
def topological_sorting(dependencies, effects):
    deps = copy.deepcopy(dependencies)
    L = []
    S = find_independent_vars(deps)
    while S != set([]):
        ind = S.pop()
        L.append(ind)
        for dep in effects[ind]:
            deps[dep].remove(ind)
            if not deps[dep]:
                S.add(dep)
    return L
```

Outputting the equations with the help of strategy design pattern. Outputting the equations is done by EqnsWriter object in models module. This object encapsulates the processing logic of output. It takes data from the containing object which is either a ModelInstance object or a ModelDefinition object. The strategy object does not know the container.

Usage of delegation pattern to encapsulate subtypes of the objects. There are several cases in the Mashap's code that requires the client object to distinguish the type of the server object. An example for this is the outputting the equations by EqnsWriter strategy object. The strategy object is unaware of the actual type of the server object (ModelInstance or ModelDefinition). This is a usage of polymorphism where one type can be used like another. ModelInstance and ModelDefinition types are actually from the same type hierarchy but they support the same interface that is used by client objects. To achieve polymorphism in that case, ModelInstance delegates the messages that it does not understand to ModelDefinition object that is stored as a data member of ModelInstance object.

Uniform access principle and encapsulation of internal state of domain objects. Uniform access principle was defined by Bertrand Meyer. It states "all services offered by a module should be available through a uniform notation, which does not betray whether they are implemented through storage or through computation." [22]. This principle encapsulates the internal state of an object from its client. This principle is followed by Mashap's code throughout domain objects by using properties. Client objects are decoupled from the internal state of the domain model objects.

Caching data in properties. Property access in some cases requires a lot of data manipulation throughout Mashap's code. Therefore we cache the data in a private member variable when property is called at least once.

Usage of the method unicode for object description. The method unicode is exclusively used for human readable description of a domain object.

Programming by convention practices. An important practice for programming by convention throughout Mashap code is the ordering of Variable objects in some container data structures. Variable objects are manipulated in several places where the ordering of the objects is critical to make the program work properly. Therefore the ordering of Variable objects in any container data structure follows always the same ordering rule: the objects are ordered alphabetically by their names.

Readable and searchable url addresses. To produce readable and searchable url addresses, we defined slugify method in GameDefinition class. This method produces a unique address string for every GameDefinition object using its actual name.

Usage of dynamic forms. As described in Section 5.3.1, the user interface of the game is produced automatically according to the decisions made by the model builder in the preparation of the game. The model builder gives the system three inputs: initial variables, decision variables, output groups. The system generates a dynamic user interface according to these choices. This functionality is the responsibility of pforms module.

The method that produces the game user interface at every decision step is shown in Table 5.5. The method firstly gets the decision variables that are input by the model builder. For each decision variable, the method produces a text box and binds this text box to the variable's name in the map fields.

Table 5.5. Code that produces the game user interface dynamically at every decision step

```
def get_game_basic_form(model_definition, game_instance):
    decision_variables = Variable.objects.filter(model=model_
        definition,is_decision=True)
    fields = {}
    for var in decision_variables:
        fields[var.name] = forms.FloatField()
    return type('GameBasicForm', (forms.BaseForm,), {'base_fields':
        fields})
```

5.4. Example Application of Mashap: Body Weight Websim

Mashap is designed to produce web based, interactive dynamic system simulation games for any system dynamics model. As the proof of concept application, we use Mashap to develop a web based game for the body weight dynamics model. The example game is called Body Weight Websim. The aim of Body Weight Websim is to allow users to explore the possible effects of nutrition and physical exercising on the long term dynamics of body weight of the human realistically. The model beneath Body Weight Websim is a modified version of Hall's body weight simulation model [1].

Mashap produces the web based game instantly from the input Vensim file that holds body weight dynamics model without requiring any additional work except entering decisions required for the customization of game user interface. The relationship between Vensim model file, Mashap, and Body Weight Websim is shown in Figure 5.3. Mashap is the software that produces simulation games from the input Vensim model file for body weight dynamics explained in Chapter 4. The model builder inputs her decisions to customize the game interface. Mashap produces from these inputs the Body Weight Websim game. The game players access to the Body Weight Websim game.

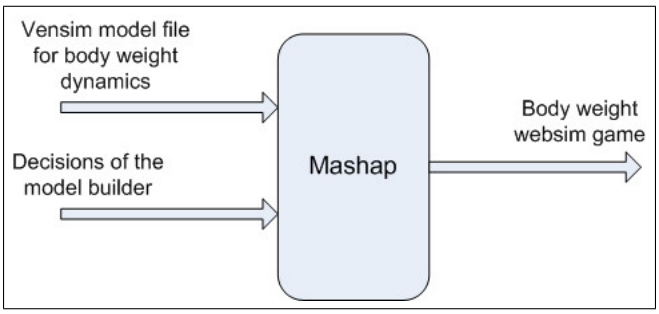

Figure 5.3. The relationship between Vensim model file, Mashap, and Body Weight Websim game

Mashap is a platform that is designed to host several websim games at the same time as shown in Figure 5.4. Also model builder can produce as many different versions from the same model file as she wishes. The Figure 5.4 shows that there are two versions of Body Weight Websim. Anytime the user inputs a different Vensim model file or set of decisions that customize the game, Mashap produces a new version of the game. This feature allows the researchers to experiment a simulation model with different sets of parameters.

Mashap produces a unique address for each game. Anybody who knows the address of the game can access and play the game. When a researcher designs an experiment with different sets of parameters, she can disseminate different url addresses for different versions of the same game to different sample groups.

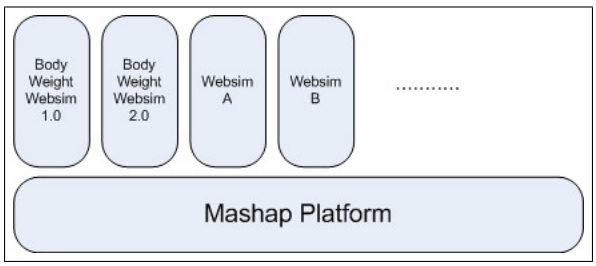

Figure 5.4. Mashap is a platform that can host several websim games.

To prepare the Body Weight Websim game on Mashap, the model builder follows the steps explained in the use case scenario "Uploading the model and preparing the

game" described in Section 5.2.1. The model builder inputs four different types of decisions according to the use case:

1. Initial parameters (step 5 in the use case)
2. Decision variables (step 8)
3. Output groups (step 9)
4. Time settings (step 6)

Initial Parameters. The idea of inputting initial parameters is to let the model customize itself for the specific game player by calibrating metabolic variables that depend on the initial parameters. In body weight dynamics model, there are four initial parameters: `baseline_bodyweight`, `ci_0`, `fi_0`, `pi_0`. The model calibrates several metabolic parameters that depend on these parameters such as `base_dnl` or weighting_of_oxidation_for_glycogenolysis. Moreover, the model calculates initial values of stock variables `fat` and `carb` based on the initial parameters.

Table 5.6. Initial parameters in Body Weight Websim

Variable's Description	Name in the Model
Baseline Body Weight	`baseline_bodyweight`
Baseline Carbohydrate Intake	`ci_0`
Baseline Fat Intake	`fi_0`
Baseline Protein Intake	`pi_0`

From a physiological point of view, initial parameters let the model adapt itself to the specific metabolic characteristics of the simulated person from whom the initial parameters are taken. The selection of initial parameters is a critical decision that depends on the model at hand. The model has to be designed taking the initial parameters into consideration because initial parameters can effect the initialization of stocks and several characteristic features of the model. For the body weight dynamics model, the calibration formulas of the metabolic variables that depend on the initial

parameters are described in Section 4.3.

In Body Weight Websim, the initial parameters are `baseline_bodyweight`, `ci_0`, `fi_0`, `pi_0`. The rationale behind this selection is that these parameters are easy to measure. Measuring these parameters does not require any special tool or expertise. Simplicity of measurement is critical for the acceptance of Body Weight Websim by general public because the aim of this application is to help common people to manage their body weight in real life. Therefore, the model under Body Weight Websim cannot depend on finely measured observations.

The parameters `ci_0`, `fi_0`, and `pi_0` are daily food intake rates of the simulated person at the beginning of the simulation. As stated in Section 4.3, the model assumes that the simulated person is in weight balance in real life. Therefore, initial daily food intake rates `ci_0`, `fi_0`, and `pi_0` are assumed to maintain the current body weight of the simulated person stable.

Decision Variables. The decision variables are the parameters of which values the game player assigns at every game step. The game player manages the system under the game by using the decision variables. In Body Weight Websim, there are four decision variables, as given in Table 5.7: `activity_level`, `ci_input`, `pi_input`, `fi_input`.

Table 5.7. Decision variables in Body Weight Websim

Parameter's Description	Name in the Model
Activity Level Index	`activity_level`
Daily Carbohydrate Intake Rate	`ci_input`
Daily Fat Intake Rate	`fi_input`
Daily Protein Intake Rate	`pi_input`

The parameter `activity_level` is a dimensionless index variable. It denotes the activity level of the simulated person. It can have negative or positive values. The

value it takes does not correspond to a certain physical energy expenditure rate. The actual physical energy expenditure rate depends on the body weight of the simulated person. The parameter `activity_level` determines uniquely the variable `activity_energy_per_body_weight` which indirectly effects `physical_activity_energy`.

The parameters `ci_input`, `pi_input`, `fi_input` are daily food intake rates of the person in simulation. They correspond to the real daily food intake rates of the simulated person. The rationale behind the selection of these decision variables is that a person has control over these decision variables in her real life. So, the simulated person can make experiments to explore possible effects of nutrition and physical exercising. She can manage and change her nutrition and physical exercising habits in her real life if she finds any benefit in simulation experiments.

An output group consists of a group of variables. Mashap draws the graphics of the dynamics of the variables put into output groups during simulation at every game step.

Time settings consist of time related settings of the simulation. There are four such settings in Mashap: time between decision steps, time step, total game time, and record period. Time between decision steps is the length of the simulation time between two consecutive game steps. Time step is the size of dt used in numerical integration algorithm. This value does not have any real life meaning. Total game time is the length of the game in simulation time. Record period is the length of time between two points where Mashap records the values of the variables in output groups into the database.

All time settings are given in the same time unit. The time unit depends on the model. In Body Weight Websim, the time unit is one day. The time settings except time step have no effect on the dynamics of simulation. Time step can have an effect on the simulation. But this effect is a side effect of the numerical integration algorithm used in simulation engine of Mashap.

The user interface of the Body Weight Websim that the game player interacts with is shown in Figure 5.5. The other screens of the game are shown in 7.

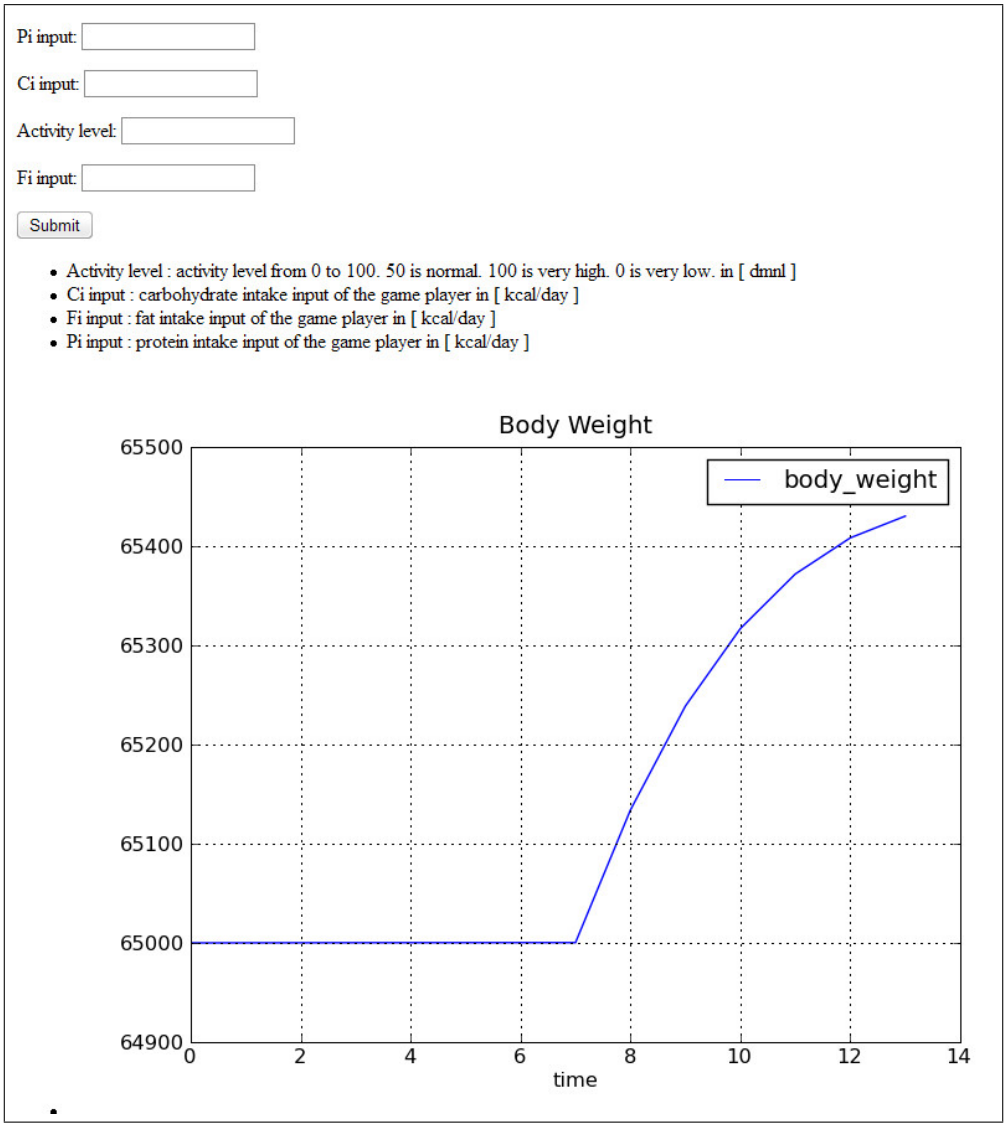

Figure 5.5. User interface of the Body Weight Websim

5.5. Comparison to Current Software

There are several popular software in system dynamics domain. This section covers comparison of Mashap with Stella (and Netsim upon it), Vensim, Powersim, and Forio.

All these tools are commercial. They have model building and game environment building features bundled. Vensim and Powersim also offer software development kits. So, it is possible to develop new tools based on Vensim or Powersim.

Forio is the only software among these four software that is exclusively developed for web based simulation development. But all of these four software support web based simulations in some way. Stella and Forio support more natively since they have features that simplify building web user interface. Powersim and Vensim support web based interfaces implicitly. They offer open application programming interfaces through which programmers can build web based simulation games.

Mashap is designed as a complementary tool to the current, popular model development tools not as a directly competing tool. In terms of supported features, Mashap is far ulterior than these four tools. But Mashap has some unique features that can make Mashap useful in some use cases:

The cost of development of a web based simulation game is much lower in Mashap than all of the four software when having a simple, automatically generated user interface for a game is enough. The server side software Netsim has very high price. Powersim and Vensim are cheaper in terms of price but they don't support web interfaces out of the box. They require custom development for the web interface. So there is additional development cost when this path is followed. Forio serves its products with different price segments. Some of them are pretty accessible to anybody. But they lack from many of the features that higher segment versions of Forio support.

Web user interface development is much easier in Mashap than Forio. Forio has many features that allow the game developers to customize the user interface finely. Netsim is very easy to use too. It does not require the model builder to develop web interface in a special language. Netsim builds a flash based web user interface from a game developed in Stella.

Primary constraint of current software is that they are commercially licensed, closed source products. Commercial licensing is a hinder on the universal access to software. But there are much more important, implicit consequences of closed source software. Closed source software prevent people to build new tools based on current software seriously. This is actually more important than access to software. Most software tools are pretty cheap today. It is easy to find the funds to buy them. But building something new upon existing software is the real burden of closed source software.

Normally this might not be a problem if the domain space of the requirements of the users were explored extensively. But in science communities this is rarely possible. Scientific research extends to new frontiers continuously. It is crucial that new tools are developed in parallel to research.

Open source or free software allows the people to build very innovative tools upon existing software that the original programmers could not imagine or afford on their own. This need is the fundamental rationale for development of a general web based game engine in this research.

Mashap is written in Python language. This is a benefit for some use cases: Firstly, since Python is dynamically typed language, we believe that it increases the productivity of the programmer in areas like mathematical programming or web applications. Secondly, Google Appengine supports running Python applications on its cloud computing infrastructure. This infrastructure offers several services that simplify web application development, hosting, and scaling. Scalability is one of the most difficult bottlenecks for maintenance of web based applications. Google Appengine infrastructure abstracts the programmers from the scalability problems substantially.

6. CONCLUSIONS

Dynamics of energy metabolism and body weight has been modeled in a lot of research studies. Some of the models focus on short term energy metabolism, some focus on long term body weight dynamics, and some focus on general dynamical analysis of body weight change. Our model is a modified version of Hall's model that focuses on long term body weight dynamics. It has a systemic view which takes food intake, physical exercising, adaptive thermogenesis, and body composition into account.

We modify and validate Hall's model in this thesis. Some of the formulas of the metabolic parameters of the model are modified such that the model can be calibrated to the specific metabolic characteristics of any adult, healthy person. The modified model is found to be realistic and robust under various test scenarios. Then, an interactive web game, Body Weight Websim, that lets the people experiment possible scenarios of dieting regimes is developed.

To produce Body Weight Websim, we first develop a generic, simulation platform called Mashap. Mashap is designed to run any system dynamics model developed in popular model building tools such as Vensim or Stella. Mashap can produce web based, simulation games from any system dynamics model automatically without requiring any additional work for configuration.

There are two important use cases for Mashap. First, model builder uploads the model and prepares the game. Second, a user/player plays the game. In preparation of the game, the model builder specifies four different sets of decisions: initial parameters, decision variables, output groups, and time settings. Mashap customizes and produces the user interface of the game by using these inputs.

Body Weight Websim is an example game built with Mashap. The model behind Body Weight Websim is the modified simulation model of Hall for body weight dynamics. The initial variables chosen for Body Weight Websim are the baseline bodyweight

and the baseline daily food intake rates. Initial parameters are estimated by assuming to yield a body weight in equilibrium. By using this assumption, Body Weight Websim calibrates the values of some metabolic parameters and initial stock values for the specific player.

The decision variables in Body Weight Websim are the activity level index and the daily food intake rates: carbohydrate intake, fat intake, and protein intake. Any healthy adult person can manage his body weight by controlling these variables.

Future Development Areas. Currently, some of the metabolic parameters are taken as they are in Hall's model. There are two problems that may require a better approach. Firstly, some of these metabolic parameters are specific to the Minnesota Experiment data. Secondly, some of these metabolic parameters are actually dependent on the baseline conditions of the body. Adjusting these parameters at the beginning of the game, by using more general data and formulas can improve the validity of the body weight dynamics model.

The current model assumes that the simulated person is in energy and weight equilibrium at the beginning of the simulation. This is probably not the case for most of the potential users of Body Weight Websim. So, the assumption of the initial weight equilibrium can be left out in future versions of the model.

On the software side, there may be several improvement areas as well. The support for Stella, Powersim model files can be added. The partial capability of exporting of the model's equations to Matlab format can be completed. There are several improvement potentials in the usability of the software. Data entry can be made through more convenient ways such as by uploading an Excel file or by using a graphical tool.

Scalability of Mashap is one of the important technical requirements. To achieve higher scalability without incurring high amounts of server costs, there are some solution alternatives: transferring the simulation engine to the client side and using a non-relational database such as Google's BigTable.

Lastly, the software can be improved by implementing innovative, collaborative use cases. Current popular, modeling and simulation software are not designed for collaborative model development. Mashap can serve as a platform for developing collaborative tools upon it.

7. APPENDIX A. MASHAP AND BODY WEIGHT WEBSIM USER GUIDE

7.1. User Guide for Mashap

This section explains the use case "uploading the model and preparing the game" performed by the model builder. The steps of the use case is explained in Section 5.2.1.

Model builder uploads the model file of Vensim to Mashap in Model Upload Screen shown in Figure 7.2. Mashap verifies the syntax of the model file and converts the model to a format that it can manipulate. Then, the system presents the equations in its own format to the model builder for ensuring that the model builder uploaded the right model file. The model builder can read the equations if she wishes. Then, she enters the game title and description in Model Verification and Description Screen shown in Figure 7.2. The system presents to the user the list of constant variables in the model. The model builder selects and submits the variables to be set initially by the game player at the beginning of the game in Initial Parameters Definition Screen shown in Figure 7.3. In Body Weight Websim, the initial parameters are `baseline_bodyweight`, `pi_0`, `ci_0`, `fi_0` as explained in Section 5.4.

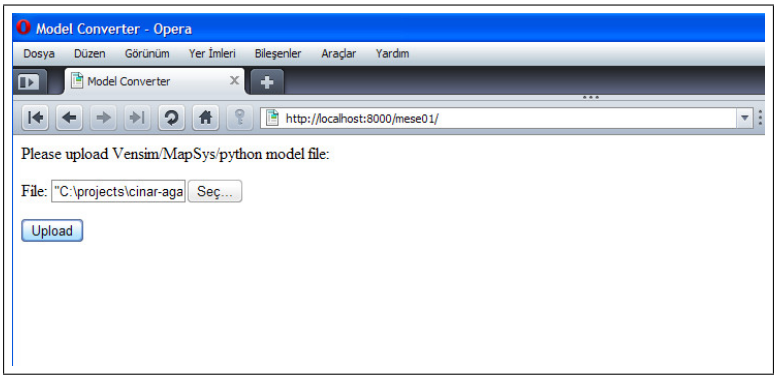

Figure 7.1. Model Upload Screen

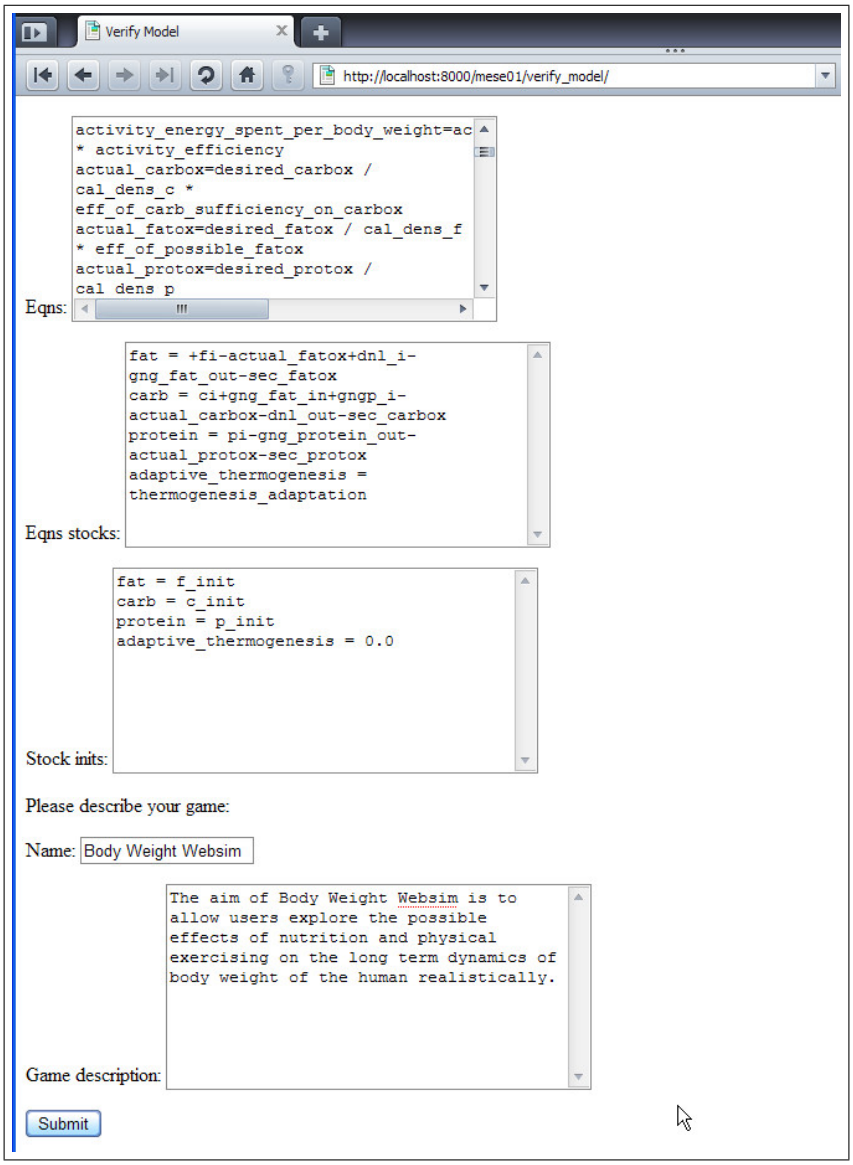

Figure 7.2. Model Verification and Description Screen

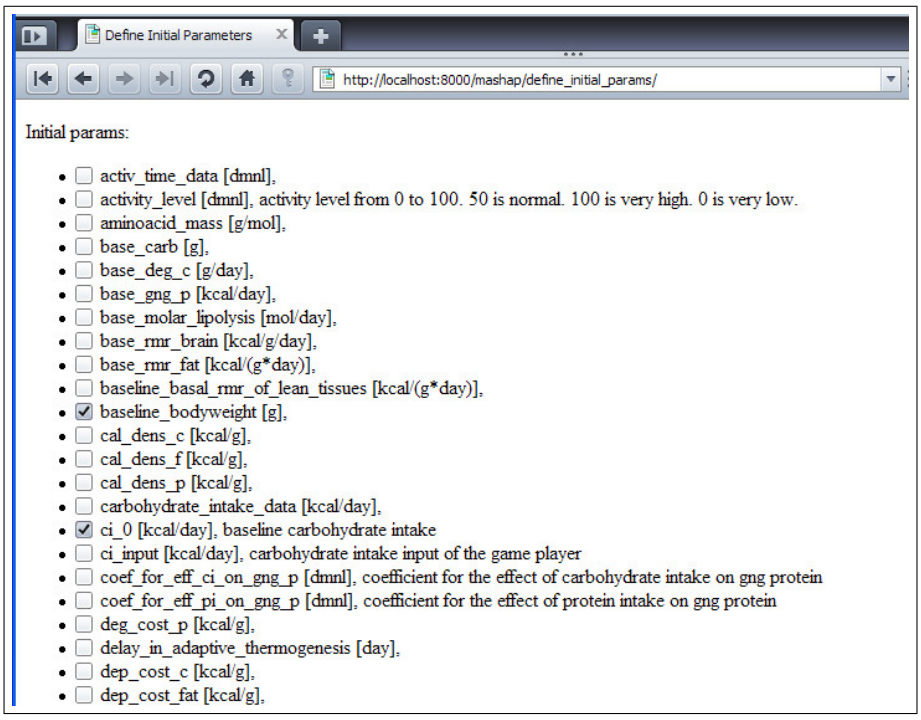

Figure 7.3. Initial Parameters Definition Screen

Next, Mashap asks the time related settings of the game to the model builder in Time Settings Definition Screen shown in Figure 7.4. Next, Mashap asks the decision variables of the game in Decision Variables Definition Screen shown in Figure 7.5. The system presents the list of constant variables in the model. The model builder selects the decision variables out of the list. In Body Weight Websim, the decision variables are `pi_input`, `ci_input`, `fi_input`, and `activity_level` as explained in Section 5.4.

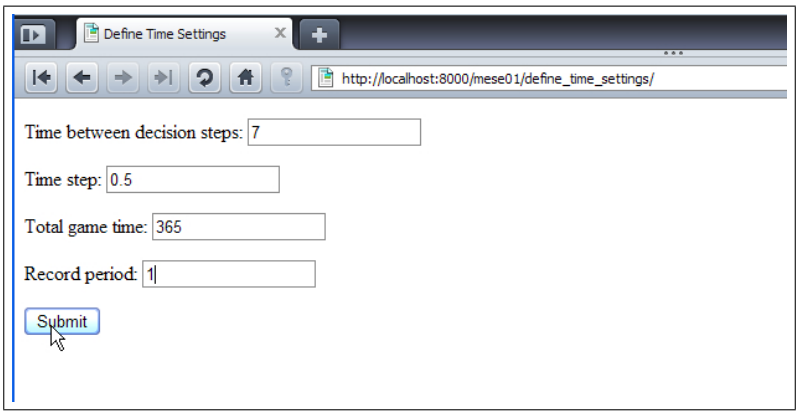

Figure 7.4. Time Settings Definition Screen

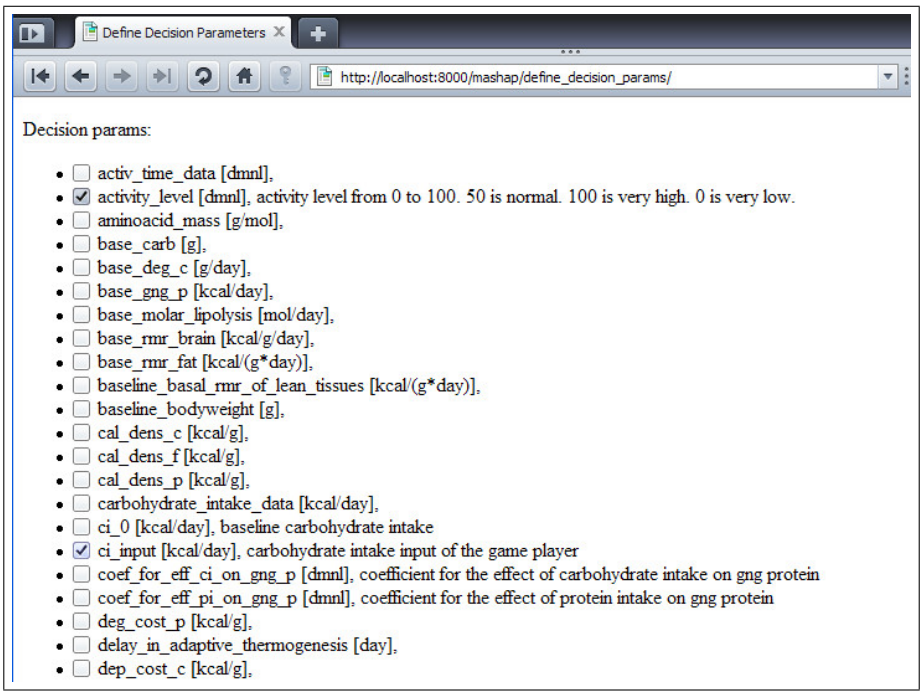

Figure 7.5. Decision Variables Definition Screen

Next step is to define the output graphics to be shown to the game player. Definition of output groups occurs in two screens. Firstly, the model builder clicks the link "New Output Group" in the Output Groups Screen shown in Figure 7.6. Then, Mashap presents the list of all the variables of the model to the model builder in the Output Group Definition Screen shown in Figure 7.7. The model builder can repeat this process as much as she wishes. In our version of Body Weight Vensim, we produce two output groups: one for the body weight, the other for the stocks as shown in Figure 7.8.

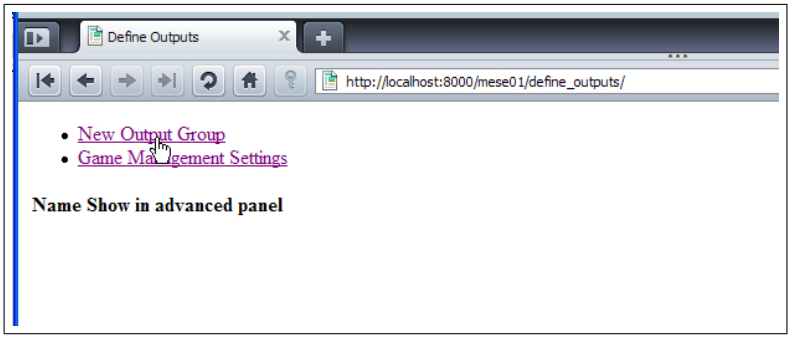

Figure 7.6. Output Groups Screen

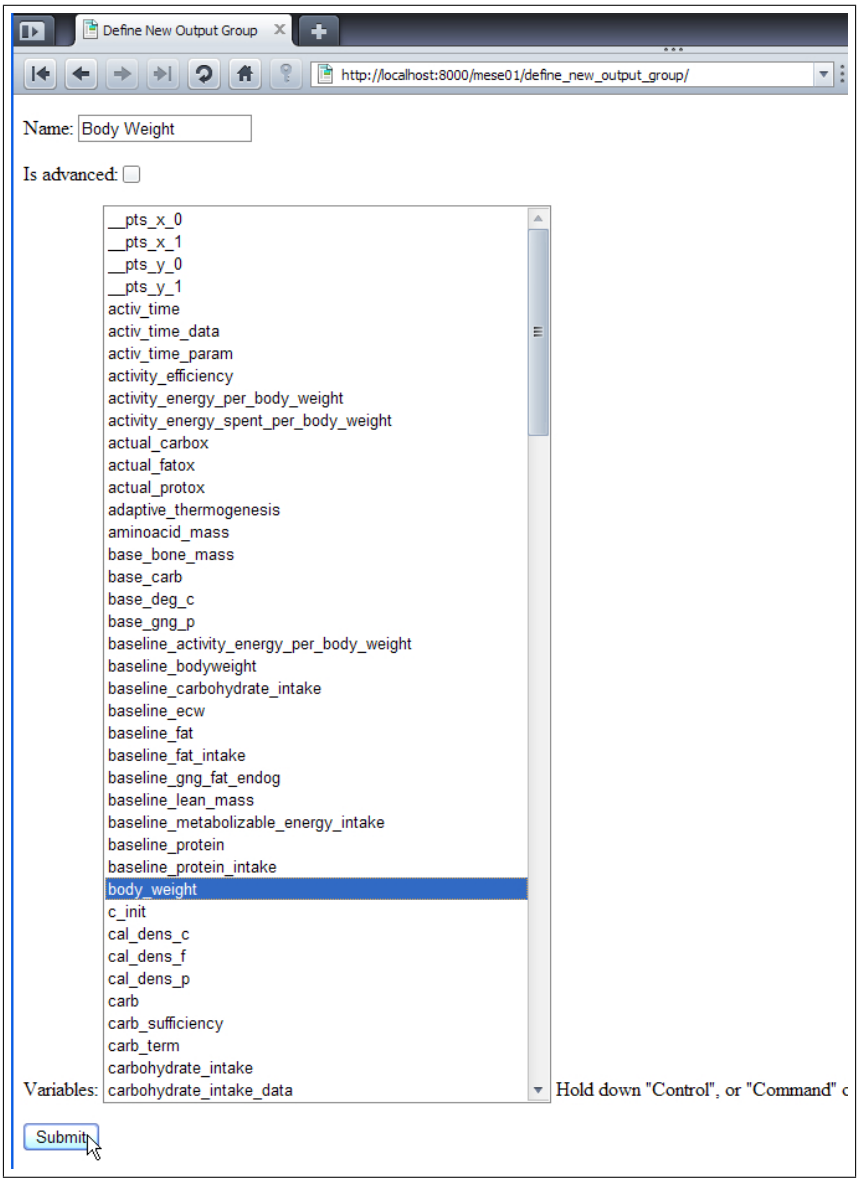

Figure 7.7. Output Group Definition Screen

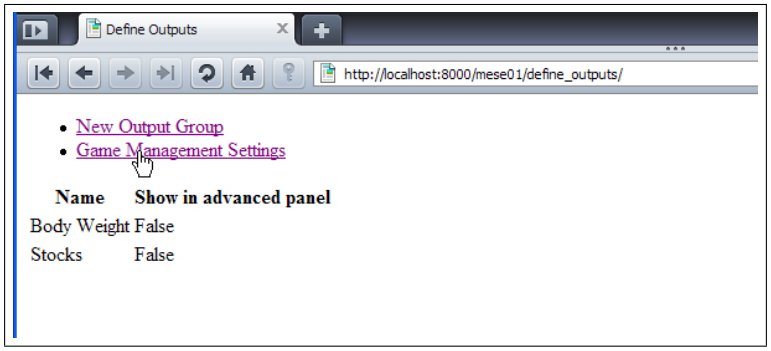

Figure 7.8. Output Groups Screen (after the definition of the output groups)

The system asks settings related to the game management such as the email ad-

dress of the model builder. Email address has to be given in the right format. The Game Management Settings Screen is shown in Figure 7.9. After submitting these settings, the use case completes. The system produces the address of the web game, and presents it to the model builder as shown in Figure 7.10. The model builder can distribute the game link to anybody for them to access the produced game.

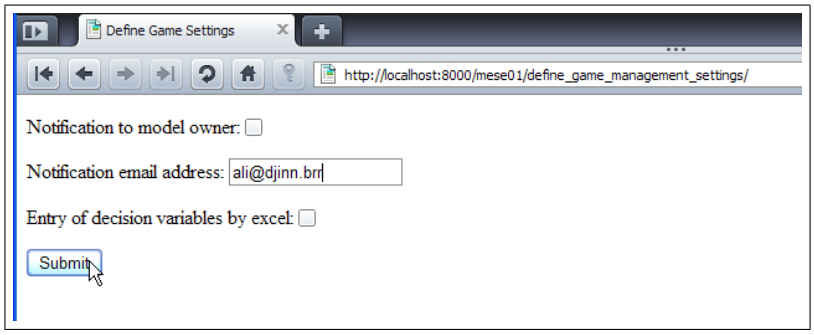

Figure 7.9. Game Management Settings Screen

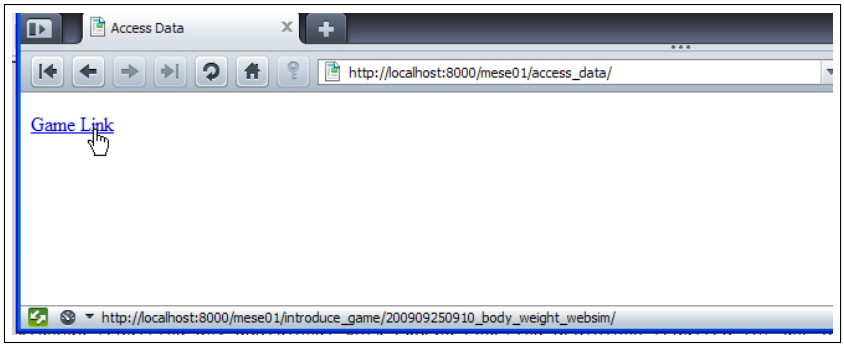

Figure 7.10. Game Access Link Screen

7.2. User Guide for Body Weight Websim

This section explains the use case "playing the game" performed by the game player. The steps of the use case is explained in Section 5.2.1.

The game player opens his web browser and enters the access link of the game. In the first screen, the system presents the introduction of the game to the game player as shown in Figure 7.11. Then, Game Customization Screen follows, Figure 7.12. In this screen, the game player states the values of the initial variables of Body Weight

Websim. Then, the system presents the actual game as shown in Figure 7.13. The game player states the values of the decision variables for the first period. The system takes these inputs, simulates the system until the next decision step, and presents the results of the simulation to the game player, see Figure 7.14.

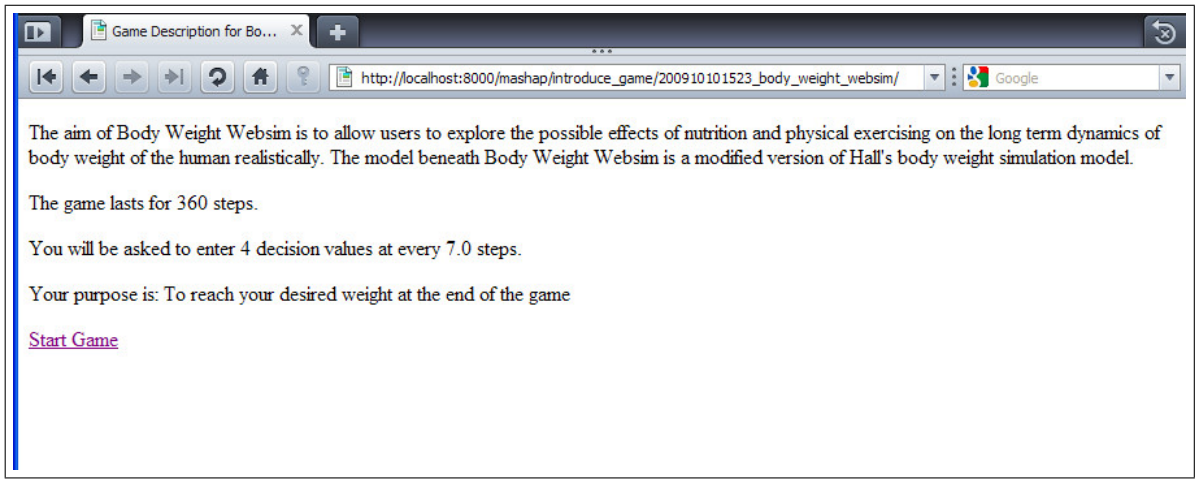

Figure 7.11. Game Introduction Screen

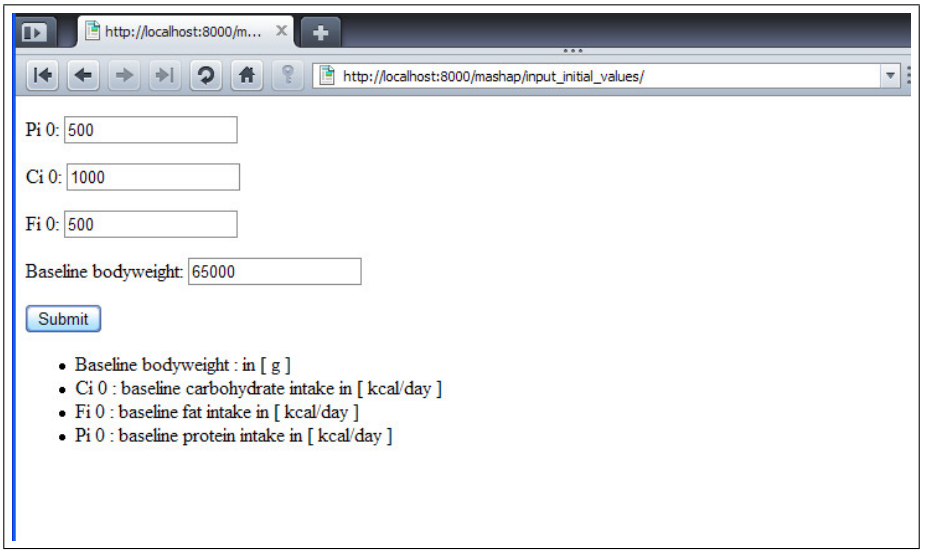

Figure 7.12. Game Customization Screen

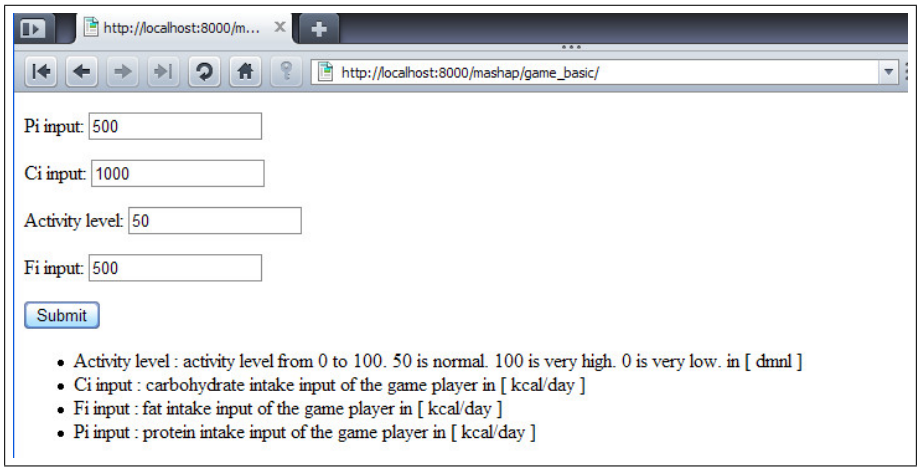

Figure 7.13. Initial Game Screen

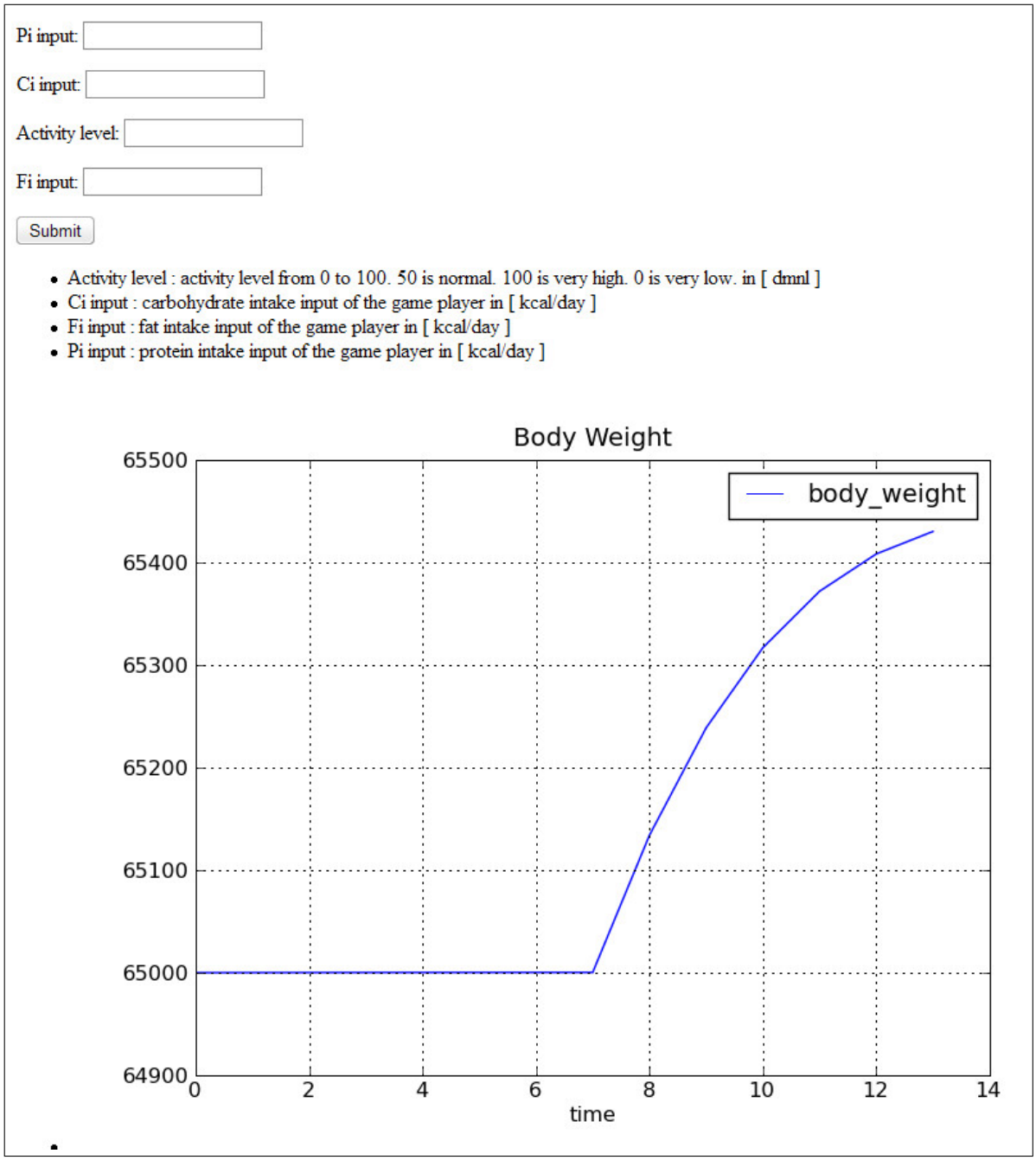

Figure 7.14. Game screen during a game

8. APPENDIX B. EQUATIONS OF THE BODY WEIGHT MODEL

The equations in this section are given in Python format. The equations in Vensim format are available from http://code.google.com/p/mashap/

```
__pts_x_0 = array([__x for __x,__y in table_eff_of_carb_sufficiency_on
    _carbox])
__pts_x_1 = array([__x for __x,__y in table_eff_of_possible_fatox])
__pts_y_0 = array([__y for __x,__y in table_eff_of_carb_sufficiency_on
    _carbox])
__pts_y_1 = array([__y for __x,__y in table_eff_of_possible_fatox])
activ_time=activ_time_data  if (read_from_data == 1.0) else (activity_
    index )
activ_time_data
activity_efficiency=1.0 + thermogenesis_effect_on_pae_vs_rmr *
    adaptive_thermogenesis
activity_energy_per_body_weight=baseline_activity_energy_per_body_
    weight * ( 1.0 - 0.652218 * activ_time )
activity_energy_spent_per_body_weight=activity_energy_per_body_weight
    * activity_efficiency
activity_index=1.5 - activity_level * (3.0 / 100.0)
activity_level=50.0
actual_carbox=desired_carbox / cal_dens_c * eff_of_carb_sufficiency_on
    _carbox
actual_fatox=desired_fatox / cal_dens_f * eff_of_possible_fatox
actual_protox=min( desired_protox / cal_dens_p,  protein / time_step )
aminoacid_mass=110.0
basal_rmr_efficiency=1.0 + ( 1.0 - thermogenesis_effect_on_pae_vs_rmr
    ) * adaptive_thermogenesis
basal_rmr_of_lean_tissues=baseline_basal_rmr_of_lean_tissues * basal_
    rmr_efficiency
base_bone_mass=0.04 * baseline_bodyweight
base_carb=400.0
base_deg_c=180.0
base_gng_p=100.0
base_molar_lipolysis=0.16
```

```
base_rmr_brain=0.24
base_rmr_fat=0.0045
baseline_activity_energy_per_body_weight=pae_b / baseline_bodyweight
baseline_basal_rmr_of_lean_tissues=0.0239374
baseline_bodyweight=85000.0
baseline_carbohydrate_intake=ci_0
baseline_ecw=fw * fetw * baseline_bodyweight
baseline_fat=baseline_bodyweight - baseline_lean_mass
baseline_fat_intake=fi_0
baseline_gng_fat_endog=base_molar_lipolysis * mass_of_glycerol * cal_
   dens_c
baseline_lean_mass=base_bone_mass + baseline_ecw + cell_mass_b
baseline_metabolizable_energy_intake=baseline_carbohydrate_intake +
   baseline_fat_intake +baseline_protein_intake
baseline_protein=cell_mass_b - icw_b - intracel_solids - base_carb
baseline_protein_intake=pi_0
baseline_proteolysis=molar_baseline_proteolysis * aminoacid_mass
body_weight=fat + lean_tissues
c_init=base_carb
cal_dens_c=4.18
cal_dens_f=9.44
cal_dens_p=4.7
carb_sufficiency=carb * cal_dens_c / desired_carbox
carb_term=eff_of_glycogenolysis_on_carb_oxidation + max( 0.0 , eff_of_
   carb_intake_on_carb_oxidation* ( carb / (0.1 + carb) )  )
carbohydrate_intake=carbohydrate_intake_data  if (read_from_data ==
   1.0) else (ci_input )
carbohydrate_intake_data
cell_mass=intracel_solids + carb + protein + intracel_water
cell_mass_b=1.0/fwcm * icw_b
change_in_carbohydrate_intake=carbohydrate_intake - baseline_
   carbohydrate_intake
change_in_protein_intake=protein_intake - baseline_protein_intake
ci  = carbohydrate_intake / cal_dens_c
ci_0=1500.0
ci_input=1500.0
ciw=fiew * baseline_ecw - ( glycogen_hydration_coefficient * c_init +
   protein_hydration_coefficient* p_init)
```

```
coef_for_eff_ci_on_gng_p=0.5
coef_for_eff_pi_on_gng_p=0.3
deg_cost_p=0.172727
delay_in_adaptive_thermogenesis=7.0
dep_cost_c=0.211111
dep_cost_fat=0.176744
dep_cost_p=0.863636
desired_carbox=frac_carbox * shared_energy_expenditure + gng_fat + gng
   _protein
desired_fatox=frac_fatox * shared_energy_expenditure
desired_protox=frac_protox * shared_energy_expenditure
df_b=base_molar_lipolysis * mass_tg
diff_bw_ox_and_tee=0.0 if (total_ox - total_energy_expenditure < 0.1)
   else (total_ox - total_energy_expenditure)
dnl=ci * effect_of_glycogen_on_dnl * cal_dens_c
dnl_b=ci_0 * eff_c_dnl_b
dnl_i=dnl / cal_dens_f
dnl_out=min( dnl / cal_dens_c , carb / time_step )
ec=baseline_metabolizable_energy_intake - (tef_b + pae_b + rmr_b )
eff_c_dnl_b=1.0 / ( k_dnl ** hill_dnl + 1.0 )
eff_ci_on_gng_p=coef_for_eff_ci_on_gng_p * normalized_change_in_
   carbohydrate_intake
eff_of_carb_intake_on_carb_oxidation=weighting_of_oxidation_for_basal_
   ci * ( 1.0 + sensitivity_of_oxidation_to_ci_changes *normalized_
   change_in_carbohydrate_intake )
eff_of_carb_sufficiency_on_carbox = numpy.interp(carb_sufficiency/
   oneday, __pts_x_0, __pts_y_0)
eff_of_glycogenolysis_on_carb_oxidation=weighting_of_oxidation_for_
   glycogenolysis * normalized_glycogenolysis
eff_of_physical_activity_on_prot_oxidation=sensitivity_to_physical_
   activity * exp( - log(sensitivity_to_physical_activity) *
   normalized_activity_energy_per_body_weight)
eff_of_possible_fatox = numpy.interp(possible_fatox / desired_fatox, _
   _pts_x_1, __pts_y_1)
eff_of_prot_intake_on_prot_oxidation=weighting_of_oxidation_for_basal_
   pi * ( 1.0 + sensitivity_of_oxidation_to_pi_changes *normalized_
   change_in_protein_intake )
eff_pi_on_gng_p=coef_for_eff_pi_on_gng_p * normalized_change_in_
```

```
   protein_intake
effect_of_carb_intake_on_lipolysis=1.0 + ( ( lipol_max - lipol_min )
   * exp( -k_lip * ci * cal_dens_c / baseline_carbohydrate_intake)+
   lipol_min - 1.0 )/ max( 1.0 , effect_of_obesity_on_lipolysis )
effect_of_glycogen_on_dnl=normalized_glycogen_ratio ** hill_dnl / ( k_
   dnl ** hill_dnl + normalized_glycogen_ratio ** hill_dnl )
effect_of_metabolizable_energy_intake_on_thermogenesis=( metabolizable
   _energy_intake - baseline_metabolizable_energy_intake ) / baseline
   _metabolizable_energy_intake
effect_of_obesity_on_lipolysis=(fat / baseline_fat)**(2.0/3.0)
efficiency_dnl=0.8
efficiency_gng=0.8
exog_glycerol_per_kcal_fat_intake=mass_of_glycerol / ( cal_dens_f *
   mass_tg )
extracel_water=baseline_ecw
f_init=baseline_fat
fat_intake=fat_intake_data  if (read_from_data == 1.0) else (fi_input
   )
fat_intake_data
fat_term=max( 0.0 , weighting_of_oxidation_for_lipolysis * normalized_
   lipolysis_rate)
fetw=3.0/8.0
fi=fat_intake / cal_dens_f
fi_0=1000.0
fi_input=1000.0
fiew=(1.0-fetw)/fetw
frac_carbox=carb_term / z
frac_fatox=fat_term / z
frac_protox=prot_term / z
fw=7.0/10.0
fwcm=7.0/10.0
gap_carbox=desired_carbox - actual_carbox * cal_dens_c
gap_fatox=desired_fatox - actual_fatox * cal_dens_f
gap_total=gap_carbox + gap_fatox
glycogen_hydration_coefficient=2.7
glycogenolysis=base_deg_c * carb / base_carb
gng_fat=gng_fat_endog + gng_fat_exog
gng_fat_endog=baseline_gng_fat_endog * normalized_lipolysis_rate
```

```
gng_fat_exog=exog_glycerol_per_kcal_fat_intake * fat_intake * cal_dens
   _c
gng_fat_in=gng_fat / cal_dens_c
gng_fat_out=min( gng_fat / cal_dens_f , fat / time_step )
gng_protein=max( 0.0 , base_gng_p * ( norm_p_ratio - eff_ci_on_gng_p +
    eff_pi_on_gng_p ) )
gng_protein_out=min( max( 0.0 , gng_protein / cal_dens_p), protein /
   time_step )
gngf_b=baseline_gng_fat_endog + gngf_ex_b
gngf_ex_b=exog_glycerol_per_kcal_fat_intake * fi_0 * cal_dens_c
gngp_i=gng_protein / cal_dens_c
hill_dnl=4.0
icw_b=fiew * baseline_ecw
intracel_solids=3967.28
intracel_water=glycogen_hydration_coefficient * carb + protein_
   hydration_coefficient * protein + ciw
is_sufficient_carb=1.0 if (gap_carbox < 0.1 ) else (0.0 )
is_sufficient_fat=1.0  if (gap_fatox < 0.1 ) else (0.0 )
k_dnl=2.0
k_lip=log( ( lipol_max - lipol_min ) /( 1.0 - lipol_min ))
kc=( ci_0 - dnl_b ) / see_b
kf=( fi_0 + dnl_b - gngf_b ) / see_b
kp=( pi_0 - base_gng_p ) / see_b
lean_tissues=base_bone_mass + extracel_water + cell_mass
lipol_max=3.1
lipol_min=0.9
lipolysis=base_molar_lipolysis * normalized_lipolysis_rate * mass_tg
mass_brain=1400.0
mass_ffa=( cal_dens_f * mass_tg - cal_dens_c * mass_of_glycerol ) / (
   3.0 * cal_dens_f)
mass_of_glycerol=92.0
mass_tg=860.0
mbc_b=mbc_lean_b + mbc_brain + mbc_fat
mbc_brain=base_rmr_brain * mass_brain
mbc_fat=base_rmr_fat * baseline_fat
mbc_lean=basal_rmr_of_lean_tissues * ( cell_mass_b - mass_brain )
mbc_lean_b=baseline_basal_rmr_of_lean_tissues * ( cell_mass_b - mass_
   brain )
```

```
mc_b=( 1.0 - efficiency_dnl ) * dnl_b + ( 1.0- efficiency_gng ) * (
   gngf_b + base_gng_p )
metabolism_of_body_cells=mbc_lean + mbc_brain + mbc_fat
metabolism_of_conversions=( 1.0 - efficiency_dnl ) * dnl + ( 1.0-
   efficiency_gng ) * ( gng_fat + gng_protein )
metabolism_of_turnovers=( deg_cost_p + dep_cost_p ) * proteolysis +dep
   _cost_fat * lipolysis +dep_cost_c * glycogenolysis
metabolizable_energy_intake=carbohydrate_intake + fat_intake + protein
   _intake
molar_baseline_proteolysis=2.73
molar_caloric_dens_glycerol=cal_dens_c * mass_of_glycerol
molar_caloric_dens_tg=cal_dens_f * mass_tg
mt_b=( deg_cost_p + dep_cost_p ) * baseline_proteolysis +dep_cost_fat
   * df_b +dep_cost_c * base_deg_c
norm_p_ratio=protein / baseline_protein
normalized_activity_energy_per_body_weight=activity_energy_per_body_
   weight / baseline_activity_energy_per_body_weight
normalized_change_in_carbohydrate_intake=change_in_carbohydrate_intake
    / baseline_carbohydrate_intake
normalized_change_in_protein_intake=change_in_protein_intake /
   baseline_protein_intake
normalized_glycogen_ratio=carb / base_carb
normalized_glycogenolysis=glycogenolysis / base_deg_c
normalized_lipolysis_rate=effect_of_carb_intake_on_lipolysis * effect_
   of_obesity_on_lipolysis
normalized_proteolysis=proteolysis / baseline_proteolysis
oneday=1.0
p_init=baseline_protein
pae_b=baseline_metabolizable_energy_intake - tef_b - rmr_b
physical_activity_energy=activity_energy_spent_per_body_weight * body_
   weight
pi  = protein_intake / cal_dens_p
pi_0=500.0
pi_input=500.0
possible_fatox=fat / time_step * cal_dens_f
pri_carbox_kcal=actual_carbox * cal_dens_c
pri_fatox_kcal=actual_fatox * cal_dens_f
pri_protox_kcal=actual_protox * cal_dens_p
```

```
prot_term=( normalized_proteolysis + max( 0.0, eff_of_prot_intake_on_
   prot_oxidation )  ) * eff_of_physical_activity_on_prot_oxidation
protein_fraction_of_cell_mass=0.2
protein_hydration_coefficient=2.0
protein_intake=protein_intake_data  if (read_from_data == 1.0) else (
   pi_input )
protein_intake_data
proteolysis=baseline_proteolysis * protein / baseline_protein
read_from_data=0.0
resting_metabolic_rate=metabolism_of_body_cells + metabolism_of_
   turnovers + metabolism_of_conversions + ec
rmr_b=mbc_b + mt_b + mc_b
sec_carbox=sec_frac_carb * gap_total / cal_dens_c
sec_carbox_kcal=sec_carbox * cal_dens_c
sec_fatox=sec_frac_fat * gap_total / cal_dens_f
sec_fatox_kcal=sec_fatox * cal_dens_f
sec_frac_carb=is_sufficient_carb * carb_term / y
sec_frac_fat=is_sufficient_fat * fat_term / y
sec_frac_prot=prot_term / y
sec_protox=min( sec_frac_prot * gap_total / cal_dens_p,  protein /
   time_step )
sec_protox_kcal=sec_protox * cal_dens_p
see_b=tee_b - base_gng_p - gngf_b
sensitivity_of_oxidation_to_ci_changes=0.761188
sensitivity_of_oxidation_to_pi_changes=15.0253
sensitivity_to_physical_activity=4.0
shared_energy_expenditure=total_energy_expenditure - gng_protein - gng
   _fat
table_eff_of_carb_sufficiency_on_carbox = [(0.0,0.0),(0.1,0.6)
   ,(0.2,0.85),(0.3,0.95),(0.4,1.0),(1.0,1.0)]
table_eff_of_possible_fatox = [(0.0,0.0),(0.4,0.4),(0.798165,0.627193)
   ,(1.30275,0.824561),(2.0,0.95),(3.0,1.0)]
tee_b=baseline_metabolizable_energy_intake
tef_b=tef_c * baseline_carbohydrate_intake + tef_f * baseline_fat_
   intake + tef_p * baseline_protein_intake
tef_c=0.075
tef_f=0.025
tef_p=0.25
```

```
therm_const=0.8
thermic_effect_of_food=tef_c * cal_dens_c * ci + tef_f * fat_intake +
   tef_p * cal_dens_p * pi
thermogenesis_adaptation=( therm_const * effect_of_metabolizable_
   energy_intake_on_thermogenesis - adaptive_thermogenesis) / delay_
   in_adaptive_thermogenesis
thermogenesis_effect_on_pae_vs_rmr=0.52
total_energy_expenditure=physical_activity_energy + resting_metabolic_
   rate + thermic_effect_of_food
total_ox=total_pri_ox + total_sec_ox
total_pri_ox=pri_carbox_kcal + pri_fatox_kcal + pri_protox_kcal
total_sec_ox=sec_carbox_kcal + sec_fatox_kcal + sec_protox_kcal
weighting_of_oxidation_for_basal_ci=3.31414
weighting_of_oxidation_for_basal_pi=0.1
weighting_of_oxidation_for_glycogenolysis=kc / kp * ( 1.0 + weighting_
   of_oxidation_for_basal_pi )- weighting_of_oxidation_for_basal_ci
weighting_of_oxidation_for_lipolysis=(1.0 + weighting_of_oxidation_for
   _basal_pi) * kf / kp
y=prot_term + is_sufficient_carb * carb_term + is_sufficient_fat * fat
   _term
z=carb_term + fat_term +prot_term
z_check_0=1.0+weighting_of_oxidation_for_basal_pi+weighting_of_
   oxidation_for_basal_ci+weighting_of_oxidation_for_glycogenolysis+
   weighting_of_oxidation_for_lipolysis
z_check_1=weighting_of_oxidation_for_lipolysis/kf
```

Formulas for the stocks:

```
fat = +fi-actual_fatox+dnl_i-gng_fat_out-sec_fatox
carb = ci+gng_fat_in+gngp_i-actual_carbox-dnl_out-sec_carbox
protein = pi-gng_protein_out-actual_protox-sec_protox
adaptive_thermogenesis = thermogenesis_adaptation
```

9. APPENDIX C. SMALL APPLICATIONS DEVELOPED DURING THE THESIS STUDY

Usage examples of the utilities are as follows:

```
>>> translate_vensim.convert(model='model_short_07_06_01.mdl',
    long2short=False)
  Outputs:
    model_long_07_06_01.mdl
>>> vensimparser.convert(file='model_short_07_06_01.mdl')
  Outputs:
    eqns_short_07_06_01.txt
    eqns_stocks_short_07_06_01.txt
    stock_inits_short_07_06_01.txt
>>> filename_eqns='eqns_short_07_06_01.txt'
>>> filename_eqns_stocks='eqns_stocks_short_07_06_01.txt'
>>> filename_translations='translations_names_symbols.csv'
>>> convert_to_latex.convert(filename_eqns=filename_eqns,filename_eqns
    _stocks=filename_eqns_stocks,filename_translations=filename_
    translations)
  Outputs:
    eqns_short_07_06_01_latex.txt
>>> excel_converter.convert(model='eqns_long_07_06_01.txt')
  Outputs:
    eqns_short_07_06_01_excel.csv
```

Table 9.1. Inputs, outputs, and descriptions of the utility scripts

Input	Output	Description	Script
Vensim file with short names	Vensim file with long names	Converts short names to long names	`translate_vensim.py`
Vensim file with long names	Vensim file with short names	Converts long names to short names	`translate_vensim.py`
Vensim file	Text of equations	Parses Vensim file and produces text of equations (Python format)	`vensimparser.py`
Text of equations	Text of equations for Latex	Converts the variable the names to their latex forms	`convert_to_latex.py`
Text of equations	Excel readable equations file	Converts the equations into Excel format	`excel_converter.py`

REFERENCES

1. Hall KD. "Computational model of in vivo human energy metabolism during semistarvation and refeeding". *Am J Physiol Endocrinol Metab*, Vol 291(1), July 2006.

2. Abdel-Hamid TK. "Modeling the dynamics of human energy regulation and its implications for obesity treatment". *System Dynamics Review*, Vol 18(4):pp 431-471, 2002.

3. Vicini P, Caumo A, and Cobelli C. "The hot ivgtt two-compartment minimal model: indexes of glucose effectiveness and insulin sensitivity". *The American journal of physiology*, Vol. 273(5 Pt 1), November 1997.

4. Chow CC, Hall KD. "The dynamics of human body weight change". *PLoS computational biology* Vol 4(3), 2008.

5. Keys A. "The Biology of Human Starvation". Minneapolis, MN: University of Minnesota, 1950

6. Barlas Y. . "Formal aspects of model validity and validation in system dynamics". *System Dynamics Review* Vol 12(3):pp 183-210, 1996.

7. Clore JN, Helm ST, and Blackard WG, "Loss of hepatic autoregulation after carbohydrate overfeeding in normal man", *J Clin Invest*, Vol. 96, 1995.

8. Flatt JP. "Conversion of carbohydrate to fat in adipose tissue: an energy-yielding and, therefore, self-limiting process". *Journal of lipid research*, Vol. 11(2):pp. 131-143, March 1970.

9. Blaxter K. L.. "Energy metabolism in animals and man." Cambridge University, 1989.

10. Nurjhan N, Bucci A, Perriello G, Stumvoll M, Dailey G, Bier DM, Toft I, Jenssen

TG, and Gerich JE, “Glutamine: a major gluconeogenic precursor and vehicle for interorgan carbon transport in man", *J Clin Invest*, Vol. 95, pp. 272-277, 1995.

11. Magnusson I, Rothman DL, Jucker B, Cline GW, Shulman RG, and Shulman GI. “Liver glycogen turnover in fed and fasted humans". *Am J Physiol Endocrinol Metab*, Vol. 266(5):pp. 796-803, May 1994.

12. Jensen M. “Regional glycerol and free fatty acid metabolism before and after meal ingestion". *Am J Physiol Endocrinol Metab*, Vol. 276(5):pp. 863-869, May 1999.

13. Wagenmakers AJ. “Tracers to investigate protein and amino acid metabolism in human subjects.". *The Proceedings of the Nutrition Society* Vol 58(4):pp 987-1000,1999.

14. Flatt JP, Ravussin E, Acheson KJ, and Jequier E. , “Effects of dietary fat on postprandial substrate oxidation and on carbohydrate and fat balances", *J Clin Invest*, Vol. 76, pp. 1019-1024, 1985.

15. Schutz Y, Flatt JP, and Jéquier E. “Failure of dietary fat intake to promote fat oxidation: a factor favoring the development of obesity". *The American journal of clinical nutrition*, Vol. 50(2):pp. 307-314, August 1989.

16. Evans E. “Domain-Driven Design: Tackling Complexity in the Heart of Software". Addison-Wesley Professional, 1. a. edition, August 2003.

17. Fowler M. “Analysis Patterns: Reusable Object Models". Addison-Wesley Professional, October 1996.

18. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Andrew Fikes, and Robert E. Gruber. “Bigtable: A distributed storage system for structured data". *ACM Trans. Comput. Syst.*, Vol. 26(2):pp. 1-26, June 2008.

19. ThoughtWorks Inc. “The ThoughtWorks Anthology: Essays on Software Technology and Innovation (Pragmatic Programmers)". Pragmatic Bookshelf, illustrated

edition, March 2008.

20. Poppendieck M and Poppendieck T. “Lean Software Development: An Agile Toolkit". Addison-Wesley Professional, May 2003.

21. Cormen TH, Leiserson CE, Rivest RL, and Stein C. “Introduction to Algorithms", Third Edition. The MIT Press, 3 edition, September 2009.

22. Meyer B. “Object-Oriented Software Construction" Prentice Hall PTR, 2nd edition, March 2000.

23. Linn T, Santosa B, Gronemeyer D, Aygen S, Scholz N, Busch M, and Bretzel RG, “Effect of long-term dietary protein intake on glucose metabolism in humans", *Diabetologia*, Vol. 43, pp. 1257-1265, 2000.

24. Appleton B. “A software design specification template."

25. Kiusalaas J. “Numerical Methods in Engineering with Python". Cambridge University Press, illustrated edition, July 2005.