Commonality Analysis for Lattice Boltzmann Systems

Peter Michalski,
Department of Computing and Software
McMaster University

April 1, 2020

Abstract

This report presents a commonality analysis for Lattice Boltzmann systems using a commonality analysis template based on Smith and Chen (2004). The document reviews both the methodology of commonality analysis and the details of Lattice Boltzmann systems. The commonality analysis itself consists of the following: *i*) terminology and definitions; *ii*) commonalities, or features that are common to all potential family members; *iii*) variabilities, or features and characteristics that may vary among family members; and, *iv*) parameters of variation, or the potential values that can be assigned to the variabilities. The documentation of the above items for Lattice Boltzmann systems is clarified by decomposing each item into subsections on Lattice Boltzmann methods, input, output, nonfunctional requirements, and, where appropriate, system constraints.

Keywords: Commonality analysis, Lattice Boltzmann methods, program family.

Contents

1 Introduction

Lattice Boltzmann systems are well suited to development as a program family because they fit Parnass definition of a program family: a set of programs whose common properties are so extensive that it is advantageous to study the common properties of the programs before analyzing individual members (Parnas, 1976). Developing such systems as a program family is advantageous because Lattice Boltzmann systems share many common features, or commonalities. Furthermore, when there are differences between systems, the variabilities between them can be systematically considered. The purpose of this document is to record these commonalities and variabilities and show the relationships between them, and thus facilitate the specification and development of Lattice Boltzmann program family members. This document will be valuable in all future phases of system development and analysis. For instance, the requirements documentation for any Lattice Boltzmann systems will use the commonality analysis, since the requirements should refine the commonalities, which are shared requirements of all such systems. Moreover, the design of any future system will use this documentation to facilitate consideration of the variabilities, so that likely changes can be made to the system with a minimal amount of work.

The scope of the commonality analysis presented here can be considered both from the point of view of Lattice Boltzmann systems and from the point of view of software engineering methodologies. From the Lattice Boltzmann system perspective, the starting point for the current document is a commonality analysis. With respect to software engineering methodologies, the scope of the current report is restricted to informal methods, with the intention that the informal requirements will form a starting point for later development and refinement by formal methods.

The first section below provides an overview of the program family of Lattice Boltzmann systems, by reviewing what is involved in a commonality analysis and the basics of Lattice Boltzmann systems. After this, the basic terminology and definitions necessary for understanding the remainder of the document are provided. The definitions include terms used in describing a commonality analysis and terms that are used in defining the characteristics and properties of Lattice Boltzmann methods. The next three sections consist of the lists of commonalities, variabilities and parameters of variation, respectively. These three sections form the heart of the documentation and include an extensive set of cross-references to demonstrate the relationships between the different items. The final section provides information on unresolved issues.

2 Overview

This section provides both an overview of the process of commonality analysis and of Lattice Boltzmann methods. The subsection on commonality analysis briefly introduces this topic, along with references that can be searched for further information. The subsection on Lattice Boltzmann methods outlines the basics of Lattice Boltzmann methods and of the scope of the commonality analysis.

2.1 Commonality Analysis

In some situations it is advantageous to develop a collection of related software products as a program family. The idea is that if the software products are similar enough, then it should be possible to predict what the products have in common, what differs between them and then reuse the common aspects and thus support rapid development of the family members. The idea of program families was introduced by Dijkstra (1972) and later investigated by Parnas (1976, 1979). More recently, Weiss (1997, 1998); Ardis and Weiss (1997) has considered the concept of a program family in the context of what he terms Family oriented Abstraction, Specification and Translation (FAST) (Cuka and Weiss, 1997; Weiss and Lai, 1999).

In the approach advocated by Weiss, the first step is a commonality analysis. This analysis consists of systematically identifying and documenting the commonalities that all program family members share, the variabilities between family members and the terminology used in describing the family. A commonality analysis provides a systematic way of gaining confidence that a family is worth building and of deciding what the family members will be. A commonality analysis document provides the following benefits Weiss (1997, 1998):

1. A starting point for the design of a domain specific languages (DSL): Once a DSL, or application modelling language, is developed the program family members can be rapidly generated by specifying a given family member using the language.

2. A basis for a common design for all family members: When the software engineers come to designing the individual family members, they can take advantage of the commonalities to reuse code. Moreover, the variabilities can be considered in the design so that they can be easily accommodated. One approach to the design may be to decompose the system into components that can each be customized by specification of values for its various parameters, where the parameters correspond to the parameters of variation identified in the commonality analysis document.

3. A historical reference: This document records the important issues concerning the scope and the nature of the family (as well as some unsolved issues) to facilitate the involvement of the participants in maintaining and evolving the family.

4. A basis for re-engineering a domain: Existing projects may not have been developed using software engineering methodologies. The projects can be systematically reor-

ganized and redesigned with the aid of a commonality analysis to unify the existing products.

5. A basic training reference for new software developers: This document provides the necessary basic information for a new team-member to understand the family.

The next section will show that the above uses of the commonality analysis document will be beneficial for the development of a family of Lattice Boltzmann programs. The commonality analysis will benefit all subsequent stages in the software development process. For instance, as mentioned in the introduction, the commonalities will act as requirements that will be the starting point for writing a software requirements specification. The commonalities will be refined into specific requirements by fixing the value of their associated variabilities. The change in the values of the variabilities then corresponds to the change from one program family member to another.

As commonalities and variabilities are requirements, they should express What functionalities and qualities the system should have, and not mention How these requirements are to be accomplished. That is, the commonalities and variabilities should not involve design decisions. The design decisions will be made after the requirements for a family member have been specified. The one exception to this is system constraints, which are requirements that explicitly make design decisions.

Besides the What versus How test, there are other tests that can be used to review commonalities and variabilities, as proposed by Weiss (1997). One such test is the what is ruled out test. This test determines if a commonality or variability actually makes a decision because if no alternatives are ruled out then no decision has really been made. Another test is the negation test. If the negation of a decision represents a position that someone might argue for, then the original decision is likely to be meaningful. For instance, the statement that the software should be reliable has a negation that no one would likely argue for and thus the statement does not represent a good characterization of a goal for the system.

In Weiss (1997) the stages of a commonality analysis are described in a systematic way. The stages include the following: prepare, plan, analyze, quantify and review. The stages are completed through the aid of a moderator and a series of meeting and preliminary documents and documentation reviews. The approach adopted here is to revise the original commonality analysis document produced by Chen (2003) and then make the new document available to others for review. The new document will be maintained in a concurrent versioning system repository so that multiple authors can work on it, and more importantly, so that the documentations revision history can be tracked and the documentation can be rolled back to an earlier version if necessary.

2.2 Lattice Boltzmann Systems

Lattice Boltzmann Methods (LBM) are a family of fluid dynamics algorithms for simulating single-phase and multiphase fluid flows, often incorporating additional physical complexities (Chen and Doolen, 1998). They consider the behaviours of a collection of particles as a

single unit at the mesoscopic scale. These methods predict the positional probability of a collection of particles moving through a lattice structure. Off the shelf (OTS) Lattice Boltzmann Solvers (LBS) allow for a range of fluid and physical model input parameters, computational parameters, and output parameters.

As LBS model fluid dynamics within a boundary using a predefined lattice structure, the methods rely on a two step calculation process. The first process is streaming, where the particles move along the lattice via links. The second process is collision, where energy and momentum is transferred among particles that collide (Bao and Meskas, 2011). There are many standardized lattice models - individual solvers within the family might only use a subset of them. The LBM uses the initial parameters of the fluid to find the probability of where along the lattice linkages a group of particles are most likely to travel. It then moves the particles into the next node, and transfers the energy and momentum if a collision occurs. Then the process repeats for the duration of the modeling instance.

3 Terminology and Definitions

This section is divided into two subsections. The first discusses the terminology that comes from the software engineering field, while the second presents the definitions used in Lattice Boltzmann solvers. Common acronyms are also listed in this section. The lists are not intended to be read sequentially, but rather to be consulted for reference purposes; therefore, the terms are ordered alphabetically, with the consequence that some terms that appear early in the list depend on the definitions of later terms.

3.1 Software Engineering Related Definitions and Acronyms

Commonality: A requirement or goal common to all family members.

Goal: Goals capture, at different levels of abstraction, the various objectives the system under consideration should achieve. Van Lamsweerde (2001)

Program Family: A set of programs that are analyzed and designed together starting from the initial stages of the software development life-cycle.

Requirements: A software requirement is: *i*) a condition or capability needed by a user to solve a problem or achieve an objective; *ii*) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document; or, *iii*) a documented representation of a condition or capability as in the above two definitions. Thayer and Dorfman (2000)

Variability: A requirement or goal that varies between family members.

3.2 Lattice Boltzmann Related Definitions and Acronyms

1D 1-Dimensional

2D: 2-Dimensional

3D: 3-Dimensional

LBM: Lattice Boltzmann Methods

LBS: Lattice Boltzmann Solvers

Velocity Directions: The number of links connecting to each lattice node in the chosen model from neighbouring nodes. All nodes in a chosen lattice model will have the same number of links. A single link will connect between two adjacent nodes.

4 Commonalities

This section lists all the common features among all the potential family members. The commonalities are organized using the following abstraction of the system, which can be used to describe all Lattice Boltzmann systems: input information, then generate the simulation and finally output the results. Section 4.1 describes the commonalities for the simulation step. Section 4.2 highlights the input information that is required for all Lattice Boltzmann systems. The next section, Section 4.3, shows the common features for the output of Lattice Boltzmann systems, such as the requirement that mesh information be written to files. (Although the output information could simply be written to the computers memory, in all practical applications it is desirable to have a persistent record of the output that was created.) The final section covers qualities of the system that cannot be classified as input, simulation generation or output. These commonalities are termed nonfunctional requirements of the system. For instance, all systems will have the goal that the response time to a users request is small enough to allow the user to focus on his/her problem and to maintain his/her train of thought, without being distracted by excessive waiting time. The commonality in this case is refined by a later variability because the specific requirement on the response time will depend on the intended usage of the mesh generating system.

Each commonality below uses the same structure. All of the commonalities are assigned a unique item number, which takes the form of a natural number with the prefix "C". Following this, a description of the commonality is provided along with a list of related variabilities, which are given as hyperlinks that allow navigation of the document to the text describing the variability. Finally, each commonality ends with a summary of the history, including the date of creation and any dates of modification, along with a brief description of the modification. If necessary, a previous version of the document can be obtained by using the concurrent versioning system where the files are stored.

4.1 Lattice Boltzmann Method Solvers

Item Number	C1

Description	A lattice discretizes a computational domain into a finite number of points. All LBS discretize the computational domain using a regular, evenly spaced grid.
Related Variability	V6 V7
History	Created April 1, 2020

Item Number	C2

Description	All Lattice Boltzmann versions use a collision operator which concerns collisions between particles.
Related Variability	V6 V7
History	Created April 1, 2020

Item Number	C3

Description	All Lattice Boltzmann versions user a probability density function to give the probability that fluid has moved into a specific domain.
Related Variability	V5
History	Created April 1, 2020

Item Number	C4

Description	All Lattice Boltzmann versions discretize velocity into a finite number of directions.
Related Variability	V6
History	Created April 1, 2020

Item Number	C5

Description	All Lattice Boltzmann versions update the velocity distribution during each LBM iteration.
Related Variability	V1
History	Created April 1, 2020

Item Number	C6

Description	Every Lattice Boltzmann version uses an equilibrium distribution function to capture the probability distribution of particles.
Related Variability	V3 V??
History	Created April 1, 2020

Item Number	C7

Description	Every Lattice Boltzmann version uses a Boltzmann transport equation to describe the statistical behaviour of a system that does not have collisions.
Related Variability	V8
History	Created April 1, 2020

4.2 Input

Item Number	C8

Description	The LBS require fluid, model, and boundary information for the problem.
Related Variability	V9 V10 V11
History	Created April 1, 2020

4.3 Output

Item Number	C9

Description	LBS write fluid parameter predictions to memory.
Related Variability	V12 V13
History	Created April 1, 2020

4.4 Nonfunctional Requirements

Item Number	C10

Description	LBS provides the precision required for the particular problems it is intended to solve.
Related Variability	V17
History	Created April 1, 2020

Item Number	C11

Description	LBS provides the accuracy required for the particular problems it is intended to solve.
Related Variability	V18
History	Created April 1, 2020

Item Number	C12

Description	The response time is small enough to allow the user to focus on their problem without being distracted by excessive waiting times.
Related Variability	V19
History	Created April 1, 2020

Item Number	C13

Description	LBS will be as portable to other operating systems as required by the users.
Related Variability	V20
History	Created April 1, 2020

Item Number	C14

Description	LBS shall be efficiently easy to use.
Related Variability	V21
History	Created April 1, 2020

5 Variabilities

This section provides a list of characteristics that may vary among family members. As in Section 4, the first three subsections on variabilities are organized into the following sublists: Mesh Generation, Input and Output. The final two subsections list variabilities that can be characterized as system constraints and as nonfunctional requirements.

As for the commonalities, each variability is labelled with a unique item number. In this case the numbers are prepended with the letter "V". The other four headings provided for each variability are: Description, Related Commonality, Related Parameter and History. The related commonalities and parameters are given as a set of identifiers that respectively refer back to the previous section on commonalities or refer forward to the next section on parameters of variation.

5.1 Lattice Boltzmann Method Solvers

Item Number	V1

Description	Different relaxation times are used to update the velocity distribution.
Related Commonality	C5
Related Parameter	P1
History	Created April 1, 2020

Item Number	V2

Description	LBS may use a framework for parallel processing of the model.
Related Commonality	None
Related Parameter	P2
History	Created April 1, 2020

Item Number	V3

Description	Different versions of an equilibrium distribution function can capture the probability distribution of particles.
Related Commonality	C6
Related Parameter	P3
History	Created April 1, 2020

Item Number	V4

Description	Storage patters for distribution function can vary.
Related Commonality	C6
Related Parameter	P4
History	Created April 1, 2020

Item Number	V5

Description	The number of dimensions in the lattice of the model can vary.
Related Commonality	C3
Related Parameter	P5
History	Created April 1, 2020

Item Number	V6

Description	The number of velocity directions in the lattice of the model can vary.
Related Commonality	C1 C2 C4
Related Parameter	P6
History	Created April 1, 2020

Item Number	V7

Description	Various collision operators can be used.
Related Commonality	C1 C2
Related Parameter	P7
History	Created April 1, 2020

Item Number	V8

Description	Various Transport Equations can be used to describe the statistical behaviour of the system
Related Commonality	C7
Related Parameter	P8
History	Created April 1, 2020

5.2 Input

Item Number	V9

Description	The input interface can vary between LBS.
Related Commonality	C8
Related Parameter	P9
History	Created April 1, 2020

Item Number	V10

Description	The number of fluids allowed in the LBS.
Related Commonality	C8
Related Parameter	P10
History	Created April 1, 2020

Item Number	V11

Description	The type of fluid parameters.
Related Commonality	C8
Related Parameter	P11
History	Created April 1, 2020

5.3 Output

Item Number	V12

Description	Visual presentation of the prediction.
Related Commonality	C9
Related Parameter	P12
History	Created April 1, 2020

Item Number	V13

Description	Format of prediction information.
Related Commonality	C9
Related Parameter	P13
History	Created April 1, 2020

5.4 System Constraints

Item Number	V14

Description	Hardware which processes the calculations
Related Commonality	None
Related Parameter	P14
History	Created April 1, 2020

Item Number	V15

Description	Operating systems on which LBS runs.
Related Commonality	None
Related Parameter	P15
History	Created April 1, 2020

Item Number	V16

Description	Amount of storage and memory needed for the LBS.
Related Commonality	None
Related Parameter	P16
History	Created April 1, 2020

5.5 Nonfunctional Requirements

Item Number	V17

Description	The precision needed for each input and output.
Related Commonality	C10
Related Parameter	P17
History	Created April 1, 2020

Item Number	V18

Description	The required accuracy for the output.
Related Commonality	C11
Related Parameter	P18
History	Created April 1, 2020

Item Number	V19

Description	The response time required for user interaction with the system varies.
Related Commonality	C12
Related Parameter	P19
History	Created April 1, 2020

Item Number	V20

Description	The operating systems on which a LBS can run.
Related Commonality	C13
Related Parameter	P20
History	Created April 1, 2020

Item Number	V21

Description	The ease with which LBS can effectively be run varies.
Related Commonality	C14
Related Parameter	P21
History	Created April 1, 2020

6 Parameters of Variation

This section specifies the parameters of variation for the variabilities listed in Section 5. They are organized into the same five subcategories as employed previously: Mesh Generation, Input, Output, System Constraints, Nonfunctional Requirements.

Each parameter of variation is given a unique identifier of the form P followed by a natural number. The corresponding variability is listed and a hyperlink is provided that allows navigation back to the appropriate item in Section 5. The final entry for each parameter of variation is the binding time, which is the time in the software lifecycle when the variability is fixed. The binding time could be during specification, or during building of the system (compile time), or during execution of the system (run time). It is possible to have a mixture of binding times. For instance, a parameter of variation could have a binding time of specification or building to represent that the parameter could be set at specification time, or it could be postponed until the given family member is built. The choice of postponing the decision until the build would be associated with the presence of a domain specific language that would allow postponing decisions on the values of the parameter of variation.

6.1 Lattice Boltzmann Method Solvers

Item Number	P1

Corresponding Variability	V1
Range of Parameters	Single relaxation rate, two relaxation-time methods, multi-relaxation -time can be used.
Binding Time	Build Time

Item Number	P2

Corresponding Variability	V2
Range of Parameters	If parallelism is used, commonly OpenMP, OpenCL, CUDA, or MPI are used.
Binding Time	Build Time

Item Number	P3

Corresponding Variability	V2
Range of Parameters	Equilibrium can be approximated up to different orders in incompressible or compressible versions.
Binding Time	Build Time

Item Number	P4

Corresponding Variability	V4
Range of Parameters	Single and multiple array storage patterns can be used.
Binding Time	Build Time

Item Number	P5

Corresponding Variability	V5
Range of Parameters	LBS models can have up to 3 dimensions.
Binding Time	Build Time or Run Time

Item Number	P6

Corresponding Variability	V6
Range of Parameters	One dimensional models include options of 2, 3, and 5 velocity directions. Two dimensional models include options of 9, 13, and 15 velocity directions. Three dimensional models include options of 15, 19, and 27 velocity directions.
Binding Time	Build Time or Run Time

Item Number	P7

Corresponding Variability	V7
Range of Parameters	SRT, TRT, BGK collision operators.
Binding Time	Build Time

Item Number	P8

Corresponding Variability	V8
Range of Parameters	Collision and collision free transport equations.
Binding Time	Build Time

6.2 Input

Item Number	P9

Corresponding Variability	V9
Range of Parameters	Input can be graphical, text or file.
Binding Time	Build Time

Item Number	P10

Corresponding Variability	V10
Range of Parameters	LBS can model one or more fluids.
Binding Time	Build Time

Item Number	P11

Corresponding Variability	V11
Range of Parameters	LBS fluid parameters include Reynolds Number, density, viscosity, time, pressure, force, direction, relaxation rate, turbulence.
Binding Time	Build Time

6.3 Output

Item Number	P12

Corresponding Variability	V12
Range of Parameters	LBS can provide 1D, 2D, and 3D rendering of the model.
Binding Time	Build Time or Run Time

Item Number	P13

Corresponding Variability	V13
Range of Parameters	LBS prediction information is output in either text or binary format.
Binding Time	Build Time

6.4 System Constraints

Item Number	P14

Corresponding Variability	V14
Range of Parameters	The LBS model can be calculated on the CPU or GPU.
Binding Time	Build Time

Item Number	P15

Corresponding Variability	V15
Range of Parameters	LBS can be run on Windows, MacOS, or Linux versions.
Binding Time	Build Time

Item Number	P16

Corresponding Variability	V16
Range of Parameters	The amount of memory and storage varies between LBS. LBS sizes begin in the low gigabytes.
Binding Time	Build Time

6.5 Nonfunctional Requirements

Item Number	P17

Corresponding Variability	V17
Range of Parameters	Each input and each output will have a specified the precision with which the real number is stored and/or calculated.
Binding Time	Build Time or Run Time

Item Number	P18

Corresponding Variability	V18
Range of Parameters	The tolerance allowed for each output produced by the system so that the system produces usable results.
Binding Time	Build Time or Run Time

Item Number	P19

Corresponding Variability	V19
Range of Parameters	The maximum amount of time that the user is expected to wait for the system to produce a result ranges between systems and models from near instantaneous to several minutes or longer.
Binding Time	Specification

Item Number	P20

Corresponding Variability	V20
Range of Parameters	The ease with which LBS can run on Windows, MacOS, and Linux system varies. Some LBS can run on all three, while some can run on multiple or only one operating system.
Binding Time	Specification

Item Number	P21

Corresponding Variability	V21
Range of Parameters	Some LBS require programming skills and domain expert knowledge while some LBS can be run without such skills.
Binding Time	Specification

7 Issues

References

Mark A Ardis and David M Weiss. Defining families: the commonality analysis (tutorial). In *Proceedings of the 19th international conference on Software engineering*, pages 649–650, 1997.

Yuanxun Bill Bao and Justin Meskas. Lattice boltzmann method for fluid simulations. *Department of Mathematics, Courant Institute of Mathematical Sciences, New York University*, page 44, 2011.

Chien-Hsien Chen. A software engineering approach to developing mesh generators, 2003. Masters thesis, McMaster University, Hamilton, Ontario, Canada.

Shiyi Chen and Gary D Doolen. Lattice boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364, 1998.

David A Cuka and David M Weiss. Specifying executable commands: An example of fast domain engineering. *Submitted to IEEE Transactions on Software Engineering*, pages 1–12, 1997. URL http://www.research.avayalabs.com/user/weiss/Publications.html.

E. W. Dijkstra. *Structured programming*. Academic Press Ltd., 1972. chapter Notes on Structured Programming.

David Lorge Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, (1):1–9, 1976.

David Lorge Parnas. Designing software for ease of extension and contraction. *IEEE transactions on software engineering*, (2):128–138, 1979.

W. Spencer Smith and Chien-Hsien Chen. Commonality and requirements analysis for mesh generating software. In F. Maurer and G. Ruhe, editors, *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2004)*, pages 384–387, Banff, Alberta, 2004.

Richard H Thayer and Merlin Dorfman. Ieee recommended practice for software requirements specifications. *IEEE Computer Society, Washington, DC, USA, 2nd ed. edition*, 2000.

Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings fifth ieee international symposium on requirements engineering*, pages 249–262. IEEE, 2001.

David M Weiss. Defining families: The commonality analysis. *submitted to IEEE Transactions on Software Engineering*, 1997. URL http://www.research.avayalabs.com/user/weiss/Publications.html.

David M Weiss. Commonality analysis: A systematic process for defining families. In *International Workshop on Architectural Reasoning for Embedded Systems*, pages 214–222. Springer, 1998. URL citeseer.ist.psu.edu/13585.html.

David M Weiss and Chi Tau Robert Lai. *Software product-line engineering: a family-based software development process*, volume 12. Addison-Wesley Reading, 1999.

8 Appendix