Conceptual Modeling Applied to Genomics: Challenges Faced in Data Loading

Matthijs van der Kroon

september 2011

Contents

Chapter 1

Introduction

Todays genomic domain evolves around insecurity: too many imprecise concepts, too much information to be properly managed. Considering that conceptualization is the most exclusive human characteristic, it makes full sense to try to conceptualize the principles that guide the essence of why humans are as we are. This question can of course be generalized to any species, but in this work we are especially interested in showing how conceptual modeling is strictly required to understand the "execution model" that human beings "implement". The main issue is to defend the idea that only by having an in-depth knowledge of the Conceptual Model that is associated to the Human Genome, can this Human Genome properly be understood. This kind of Model-Driven perspective of the Human Genome opens challenging possibilities, by looking at the individuals as implementation of that Conceptual Model, where different values associated to different modeling primitives will explain the diversity among individuals and the potential, unexpected variations together with their unwanted effects in terms of illnesses and susceptibilities to certain types of drugs.

Genomics, from an Information Systems point of view, is largely situated in the first phase of systems design, the analysis. Due to the youth of the genomics domain, many aspects of what is driving the mechanisms of life are still unknown. This work presents an inter-disciplinary approach, in which experience in Information Systems development is put to practice in genomics. Concretely speaking, by applying a conceptual modeling approach, fixing the present day knowledge about the human genome in a visual form. The genomic domain can be characterized by three distinct properties: large data quantities, high complexity and rapid evolution. The first property poses certain challenges on resources, like processing time and storage space. Processing the large linkage disequilibrium data files provided by the HapMap [1] resource (around $\sim$40 Gb compressed, $\sim$224 Gb uncompressed) for instance certainly counts as challenging, but is by no means impossible due to the regular structure. It is the high complexity, rapid evolution and the ambiguity that often results from those aspects that pose the real problems on the long run as they call for a stable and homogeneous structure, while at the same time allowing for efficient and easy

evolution of this same structure. No amount of processing power can compensate for a lack of structure, if what one is looking for simply has not been stored.

It is true that genomics is often not an exact science due to the immense complexity of nature and its processes. It is true that basic concepts like genes, alleles and mutations are frequently variable in their definition. Their exact denotation often depends on both context and position in time. A gene for instance can be defined as a locatable region of genomic sequence, corresponding to a unit of inheritance, which is associated with regulatory regions, transcribed regions, and or other functional sequence regions. However, the existence of splicing, in which gene expression changes at runtime, complicates this view as is very well described by [2]: "*in eukaryotes, the gene is, in most cases, not yet present at DNA-level. Rather, it is assembled by RNA processing*". Pearson [3] and Gerstein et al. [2] provide recent insights on the evolution of the gene concept.

But it is the central point of this work that for efficient research to take place, clear definitions of concepts and a common vocabulary are crucial. This is especially the case in research where worldwide various separate research groups are collaborating and exchanging information on a regular basis. Formally describing concepts, ruling out ambiguity and relating concepts to each other and their context is the main objective of model driven software development. Simply put, a conceptual model is a simplified representation of reality, devised for a certain purpose and seen from a certain point of view. The objective of a conceptual model is simulation of reality, it therefore needs to react to input in the same way as reality would. It is in this context, where a precise connection between the genomics domain and the Conceptual Modeling approach makes full sense.

Describing a system by means of conceptual models means viewing the world as consisting of objects that belong to different classes, have distinct properties, and are related to each other in various ways. This way of viewing a system provides a powerful representation and reasoning tool. Modeling a domain in term of concepts thus roughly has either one of two applications, or both. Either the models serve by use as a reasoning tool to gain a deeper understanding of the domain at hand, usually as part of the design of an Information System, or they guide the creation of a system which is ultimately directed at controlling or modifying that same domain. In the first application, the conceptual model serves as a visual representation of the domain, linking concepts and their respective behaviors while the latter resembles the blue print used in traditional building.

Not surprisingly, conceptual models and ontologies are closely intertwined. There is still a large debate about the use of ontologies in the bioinformatics domain. Section 2.3 will dig deeper into this subject, further refining the definitions of both conceptual models and ontologies and how they interconnect in the context of this work. In general, current efforts doing exactly what this work pretends to do, capturing the semantics of the Human genome in a formal manner, are almost all ontology based. The Gene Ontology [4] is considered

to be the de-facto standard here. It is however the contribution of this work to show how conceptual modeling techniques can improve on these efforts by providing a solid, well researched base which -due largely to its visual and comprehensive character- enables the engineer to capture these earlier mentioned semantics in close collaboration with domain experts. All this without losing the clear advantages and many promises that ontologies, as conceived by the bioinformatics domain, present while at the same time adding a few more.

By allowing for a minor paradigm-shift, the human genome (or any genome) can be considered an Information System, a natural and highly complex form maybe but an Information System nonetheless. Stated in a very simplified manner, data that is stored in DNA, undergoes recombination, processing and ultimately translation to proteins. It is the combination of these proteins and the influence from external factors (the environment) that define the way an individual looks and behaves. These characteristics map neatly to the generic characteristics usually associated to Information Systems, only replacing man-made hardware in the form of chips and circuits, by biological molecules and physics. As Chikofsky and Cross [5] state, reverse engineering is defined as the process of analyzing a subject system to (i) identify the systems components and their inter-relationships and (ii) create representations of the system in another form or at a higher level of abstraction. Following this philosophy, just like software can be reverse-engineered in order to apply after-market changes or facilitate maintenance [6], it might very well be possible to reverse engineer life itself and create a higher level of abstraction representation in the form of a conceptual model. In this case, after-market changes and maintenance include treatments and/or prevention of previously untreatable disorders and disease. Basically, said in Information Systems jargon, debug life itself.

The value of a conceptual model of the human genome is thus two-fold: first, it allows for a visual, and formal representation of the domain. By doing so, fixing a vocabulary and conceptual gamut from which scientists can draw in order to ensure communication takes place based on the same dictionary, using the same concepts. It deserves mentioning here, that the conceptual model as proposed in this work is thus expected to evolve along with the advancing understanding of the domain in time. This evolution capacity is an extra value in itself for a domain where knowledge is continuously being generated, and thus continuously subject to change. Only by having a well-defined, precise conceptual background the Conceptual Model- can this new knowledge be properly incorporated, be understood and be adequately managed.

Second, the conceptual model can be used to (semi-)automatically generate software. When seen as a Platform Independent Model (PIM) the Conceptual Schema of the Human Genome can be considered as the base from which various Platform Specific Models (PSM) can be deduced. Each of these PSMs can then be transformed into a functional implementation. The degree of functionality, and interference of the engineer depends on the level of expressiveness of the model. Current efforts, like OO-method [7], suggest that it is possible to generate fully functional Information Systems from nothing but models, in which low-level programming becomes redundant. It is well known that programming

languages have been subject to a constant evolution towards higher levels of abstraction. Model Driven Engineering might very well be the next step in this process, where an engineer is no longer expected to manually program the Information System, but rather model it adequately and then through a series of automated steps this model is then transformed (or compiled) into the final IS. Seen from this perspective it makes full sense to create an adequate PIM for the genomic domain, where rapid evolution makes it hard to keep up with the state of affairs. Indeed, ad-hoc programming solutions tend to not cope very well with change as they are hardly reusable, if at all. The documentation issue is also quite present; ad-hoc solutions, often poorly documented, are difficult to grasp for other engineers than the original creator. Model based solutions are expected to greatly reduce the problems associated to these issues by incorporating the body of knowledge as accumulated over the past decades in the Information Systems domain.

This document can be separated into three parts: the first being the theoretical introduction to the work, including problem statement, contribution, state of the art and the conceptual schema description. In this first part the more theoretical contribution of the work will be presented. The second part will report about how the practical application of the schema has worked out, detailing various data loading efforts. This latter effort has been done by taking a practical approach. Loading the database that came to be as a result of the conceptual schema with real data has uncovered flaws in the original schema which lead to modifications as will be discussed. The real data presented in this work deal with genetic mutations. Current genomic research is very much clinically focussed, attempting to uncover relations between genotype and phenotypic traits like disease and individual response to medical drugs. It therefore makes full sense to start validating the conceptual schema by applying it to the domain of genetic variation. The third part then describes in a summarized manner how these results have been interpreted and related to the context of the work.

This chapter will proceed to explain the domain, providing a short introduction to cell biology and genetics/genomics in section 1.1. The domain description will be separated in various subdomains: genome structure, the genotype to phenotype process, metabolic pathways and genetic variation. The problems issued by this work will be stated in section 1.2 as decomposed into the two subsections, data chaos and conceptual chaos. In section 1.3 the requirements of a possible solution to the earlier stated problem will be stated, while pointing out how this work contributes to it. In the following chapter 2 a detailed overview on current solutions that already exist in the domain will be given, as well as detailing how the contribution of this work significantly differs from those. Section 2.1 will proceed to provide context to conceptual modeling in general, as used as a tool in the creation of Information Systems. Section 2.2 then further specifies to the use of conceptual modeling in the domain of bioinformatics. Ontology based solutions, their shortcomings and major strongpoints will be discussing in section 2.3. Details on how the Conceptual Schema of the Human Genome is set up will be explained in chapter 3, in which all views of

the schema will be discussed in a separate section.

The first chapter of part two, 4, will describe the effort of loading the Human Gene Mutation Database (HGMD) database. The various sections describe HGMD as a data source (section 4.1), the problems that were encountered (section 4.2 and the solutions to them (section 4.3). In section 4.4 the lessons from loading HGMD are then summarized. Chapter 5 then proceeds to explain the process of loading dbSNP, in which the various sections describe how the conceptual schema was improved with the introduction of the Single Nucleotide Concept (section 5.1 and section 5.2), describe dbSNP (section 5.3), and detail the encountered problems (section 5.4). In section 5.5 the outcomes of loading dbSNP are summarized. In a very similar manner as the two previous cases, the BIC database is discussed in chapter 6. Section 6.1 describes the BIC database in general terms. In section 6.2 the problems encountered in the data loading process are listed. In section 6.3 we summarize the lessons learned from loading the BIC database. These lessons learned from these data sources combined have lead to the creation of a data loading application, the "Genoma Data Loader", which will be covered in depth by chapter 7, where its structure will be discussed in section 7.1, what is being loaded and to what extend by section 7.2.

Part three consists of a chapter 8 that covers conclusion and suggests further research in chapter 9.

1.1 Domain description

Considering the particularities of the problem domain, before introducing and explaining the basic components of the intended Conceptual Model of the Human Genome, we provide next a short but necessary summary of the main properties of the studied genomic domain, as these properties need to be understood for adequately representing them in the target Conceptual Model. Alberts et al. [8] provide a very complete guide to cell biology, and for a good understanding of this article some basic knowledge about genetics is recommended. This section serves in no case as an exhaustive guide to genetics, it however aims to provide a minimum of knowledge required to understand the rest of this article.

1.1.1 Genetics 101

The chemical structure holding this hereditary information is called deoxyribonucleic acid, or DNA. The syntax in which the genetic code is written, consists of 4 elements; A, C, T and G denoting particular chemicals, referred to as nucleotides, or bases. Each of these nucleotides comprises of 3 components; a phosphate group, a five-carbon sugar and a nucleobase. The phosphate group and the sugar form a backbone structure while the nucleobase defines the nucleotide denotation, or meaning. 4 different nucleobases exist; Adenine, Cytosine, Thymine and Guanine, hence the nucleotide identifiers. In DNA the phosphate groups of each nucleotide bond with the sugar of the next nucleotide

forming a sequence of nucleotides in that process. At the same time, nucleobases adhere to each other in a specific way: A only bonds to T and C only bonds to G. See figure 1.1 for details on the chemical structure of DNA.

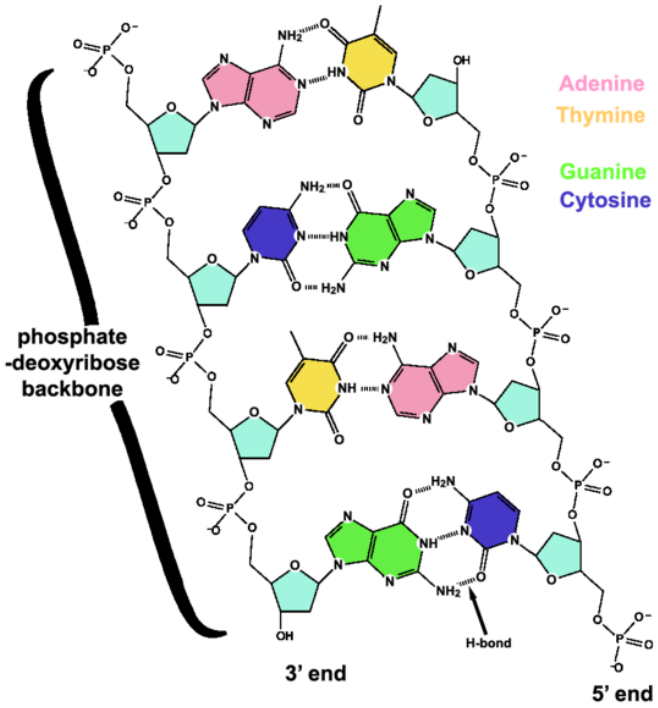

Figure 1.1: DNA chemical structure.

DNA as formed by these bonds consists of two deoxyribose phosphate backbones with pairs of nucleobases in between. The 3D structure of the molecule is what is known as the famous helix shape. The deoxyribose phosphate backbones are often referred to as strands and since the nucleotides adhere to each other in only one manner, the strand sequences are complementary to each other. This phenomenon is often referred to as one strand sequence being sense and the other anti-sense. The sequence on the strand that is transcribed to mRNA is called the 'sense' sequence, while the sequence on the opposite strand is called the 'anti-sense' sequence. The sense strand then is denoted with a '-' symbol, while the '+' symbol indicates a non-sense strand. Both sense and anti-sense sequences can exist on different parts of the same strand of DNA.

Since the nucleotides bond in an asymmetrical way, sugar to phosphate, DNA has a direction. In the helix, the direction of one strand is opposite to the direction of the other. The strand ends are referred to as the 5' and 3' ends, where the 5' corresponds to the end with a terminal phosphate group and the 3' to the end with a terminal sugar group. Nevertheless, genomes rarely exist

as one uninterrupted DNA string, in human beings for example the hereditary information is divided over 23 pairs of DNA strings, commonly referred to as chromosomes. Contrary to what would be expected however, only a small part of the total genome sequence codes for genes. About 95% of the human genome has been designated 'junk' and percentages representing coding parts of the genome range from 1.1% [9] to less then 5% [10]. The non-coding parts of the genome were considered for a long time to be evolutionary artifacts, serving no present day function. However, recent research shows the non-coding parts of the genome might actually be fulfilling functions, not yet well understood [11], [12] and [13].

Every individual of our species shares approximately 99% of DNA with all other individuals, meaning that the differences observed among individuals can be traced back to around 1% of our genetic sequence. Some of these individual differences, which we will call variations, can be retraced to relatively neutral aspects like hair and eye color, while others have more serious effects like susceptibility to disease. A clear example of the omnipresence of ambiguity in the domain is the lack of a clear distinction in concepts that distinguishes between these different manifestations of genetic variations. A large amount of recent genetic research has been focused on discovering the relationship that exists between genotype and phenotype. Genotype being the chemical ordering of the earlier mentioned bases and represented by a letter sequence "$ATGGGCCT$". When people speak of "sequencing" a genome, it is the process of determining the nucleotide sequence of one set of chromosomes of a given organism. Phenotype relates to the manifestation of all observable characteristics of an organism. Clearly for this research to take place successfully, data about genotype needs to be stored, exchanged and analyzed in large amounts.

1.1.2 Genome structure

The concept of genome refers to the collection of an organisms' hereditary information. In most organisms it is encoded in DNA while in others, mostly viruses, in RNA. Some common explanations for the origin of the term include the blending of the words **gen**e and chromos**ome**, or roots in the ancient Greek word for $\gamma\iota\nu o\mu\alpha\iota$; "I become, I am born, to come into being". The terms "genetic" and "genomic" are often used as synonyms. Semantically speaking, however, they are not exactly equal: the first usually refers to a gene-centered science, while the latter considers the genome as a whole. In the context of this work, they are often used interchangeably.

Genome size varies greatly per organism, from about 3600 base pairs in the *Bacteriophage MS2* virus, to around 3.2 billion base pairs in *Homo Sapiens*. It deserves mentioning here that size of genome does not appear to be directly related to the perceived complexity of the organism as there exists a fish, *Protopterus aethiopicus*, for which the genome contains a staggering 130 billion base pairs. If for a single human cell, all DNA would be connected end-to-end and straightened, it would stretch out to a length of approximately 2 meters. In order for this quit large molecule to fit in the nucleus of said cell, it is clear

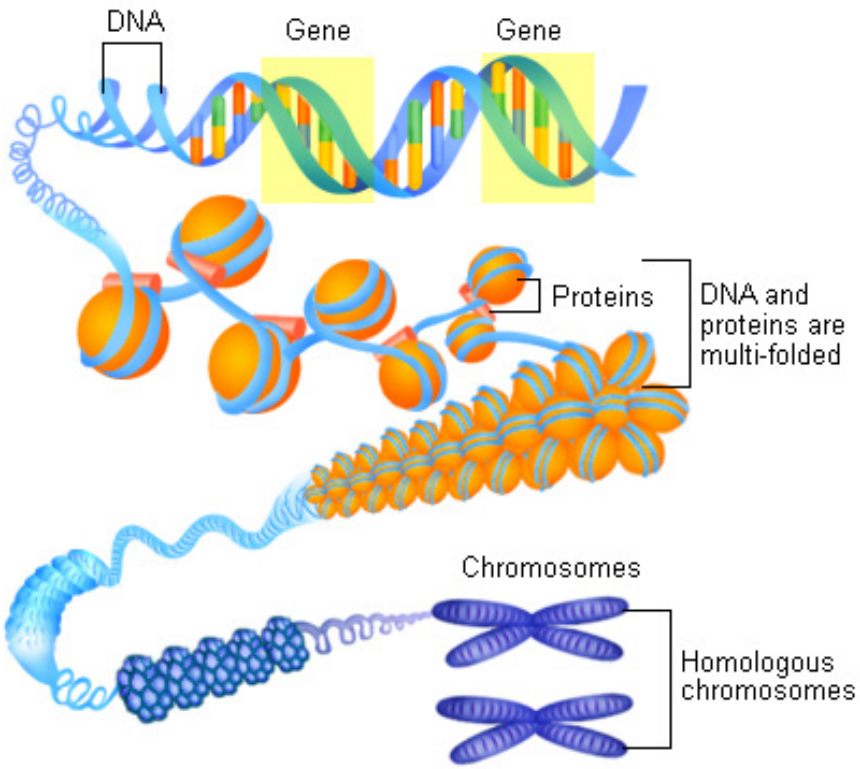

Figure 1.2: Genome structure.

some kind of ordering needs to take place.

Figure 1.2 depicts the various levels of organization that a genome undergoes. We will follow the image bottom-up seen from a biological point of view, starting at the top of the image with the parts of DNA being marked as genes. Stretches of DNA that code for a type of protein, or for an RNA chain that has a function in the organism are said to be genes. Genes hold the information that is required to build and maintain and organism's cells and pass genetic traits to offspring. Currently it is expected that the human genome contains ∼20.000 genes, these protein-coding sequences make up 1-2% of the human genome. Some, or all, of these genes are able to encode various different proteins depending on the way their genetic message is interpreted by the transcription and translation processes that will be discussed in section 1.1.3. It deserves mentioning here that, although not depicted in figure 1.2, genes can be further decomposed into parts of nucleotide sequences. This decomposition is based on whether these sequences are said to be coding, in which case we refer to them as *exons*, or non-coding, in which case they are said to be *introns*. The semantics of this particular difference will also be clarified in section 1.1.3.

Said genes are contained within a DNA sequence, the next level of organization, of which the helical shape already makes it quite space efficient. The DNA then folds around special proteins, the so-called histones, that make it's organization even more efficient and predictable. It is important for the DNA to be organized in an efficient manner as space is a limited resource within the cell nucleus. Predictable folding however is just as, if not more important. Many processes in which the DNA is involved require specific parts of it to be 'read' by specific proteins, the exact folding of the DNA might enable, or inhibit this exact process. Moving on to the last layer of organization considered in this example. Chromatin is the complex of tightly wound DNA and protein that packages chromosomes. The composition of chromatin varies through the various phases of the cell cycle, depending on which parts of the DNA need to be expressed. Chromosomes come in two varieties: autosomes and sex chromosomes. The autosomes exist in equal number of copies in both males and females, while this is not the case for sex chromosomes. In diploid organisms, like humans, cells have two homologous copies of each chromosome, one from the mother and one from the father. The number of chromosomes varies per organism. Humans for instance have 23 chromosome pairs, 22 autosomes and 1 sex chromosome, while the goldfish accounts for 100 to 104 chromosomes. Again, complexity can not be related directly to sheer quantity.

1.1.3 Genotype to phenotype

The process of transferring the information contained on the genome -the genotype- to a set of observable traits -the phenotype- is a complex intercommunication of organic, ánd inorganic compounds. The exact functioning of this mechanism is still under research, and many parts remain unknown. What is known can be decomposed into three major processes, namely *transcription*, *translation* and the interaction of the resulting compounds in what is known as *metabolic pathways*. Each of these processes will shortly be discussed separately in more detail, but we start by introducing how these relate to each other. For genetic information to leave the cell nucleus, and ultimately lead to observable changes in an organisms traits, the first step is to transfer the message from DNA to Ribonucleic Acid, or RNA. This RNA is then able to leave the cell nucleus, and enter the cell's cytoplasm where it connects to a ribosome. The ribosome is an internal component of a biological cell which function is to assemble proteins from RNA. The resulting proteins then interact with each other and other chemicals, or metabolites, in series of chemical reactions that ultimately lead to a specific trait.

Transcription

This is the process that transfers genetic information from the relatively stable DNA molecule, to a complementary RNA copy. DNA itself is unable to leave the cell nucleus, where it is protected from potentially dangerous factors. RNA however, due to its slightly different chemical make up, cn leave the nucleus.

During transcription, a DNA sequence is read by RNA polymerase, which results in a complementary RNA copy in which all instances of thymine have been replaced with uracil. Various types of RNA exist, the most aberrant one being messenger RNA, or mRNA, in which the enclosed message potentially encodes for a protein. Other examples include ribosomal RNA, transfer RNA and micro RNA.

Splicing is the process of modifying an RNA after the initial transcription process in which introns are removed and exons joined. Splicing rarely involves RNA that is not of the mRNA type, thus in the context of this work only this case will be considered. As explained earlier, in section 1.1.2, genes can be decomposed in coding sections, *exons*, and non-coding sections, *introns*. In order for the messenger RNA to produce a correct protein through translation, these sections need to be either eliminated or joined. Depending on various factors like in which tissue the particular cell is situated and cell age, the splicing process might also exclude specific exons, resulting in different mRNAs. The process of splicing allows for a 'run-time' reordering of genetic information that enables for one gene to produce a variety of products. The earlier mentioned organism complexity issues can largely be explained by exactly this; organisms considered to be complex, like eukaryotes, splice many protein-coding mRNAs and some non-coding RNAs. Prokaryotes, on the other hand, splice rarely and mostly non-coding RNAs.

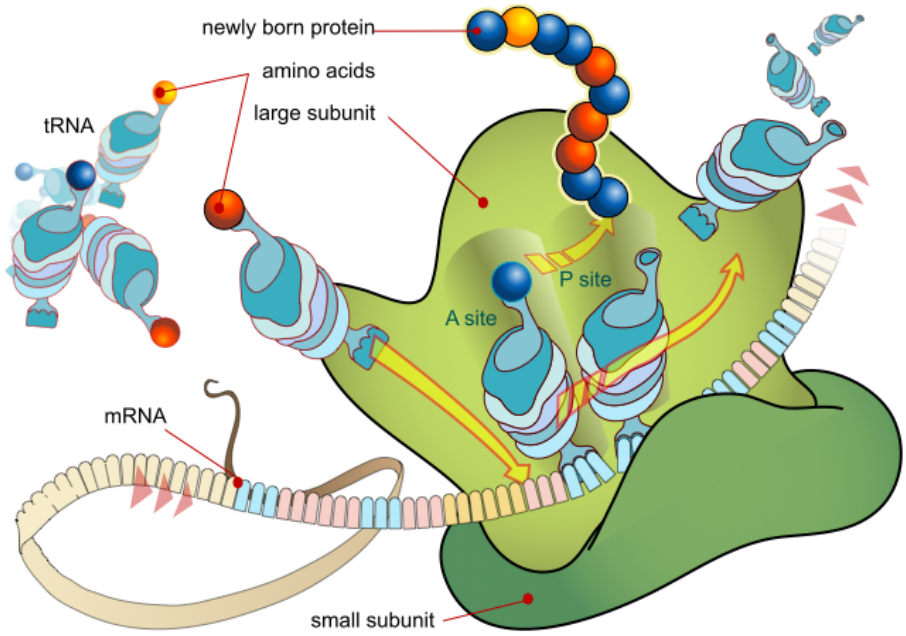

Figure 1.3: The translation of mRNA and the synthesis of proteins by a ribosome.

Translation

Once the genetic message has been transcribed from DNA onto an RNA molecule, it is able to leave the cell nucleus and enter the cell's cytoplasm. It is here that the ribosomes are located. These ribosomes act like 'protein-factories' in which mRNA is translated, or synthesized, to proteins. The mRNA sequence binds to the ribosome and is interpreted in codons, which means nucleotide triplets. As the former holds true, the way of breaking the sequence in three letter codons -the reading frame- defines how the sequence is read. There are three possible reading frames in an mRNA strand: each reading frame corresponding to starting at a different alignment. The reading frame that has a start codon, and a subsequent region usually containing a multiple of 3 nucleotides, but excluding the stop codons is called the open reading frame (ORF). Each possible combination of the mRNA bases -uracil (U), adenine (A), cytosine (C) and guanine (G)- represents a particular semantic value, as shown in figure 1.4, either relating the triplet to a specific amino acid or providing syntactical clues. The latter refers to start (AUG) and stop codons (UAA, UAG and UGA).

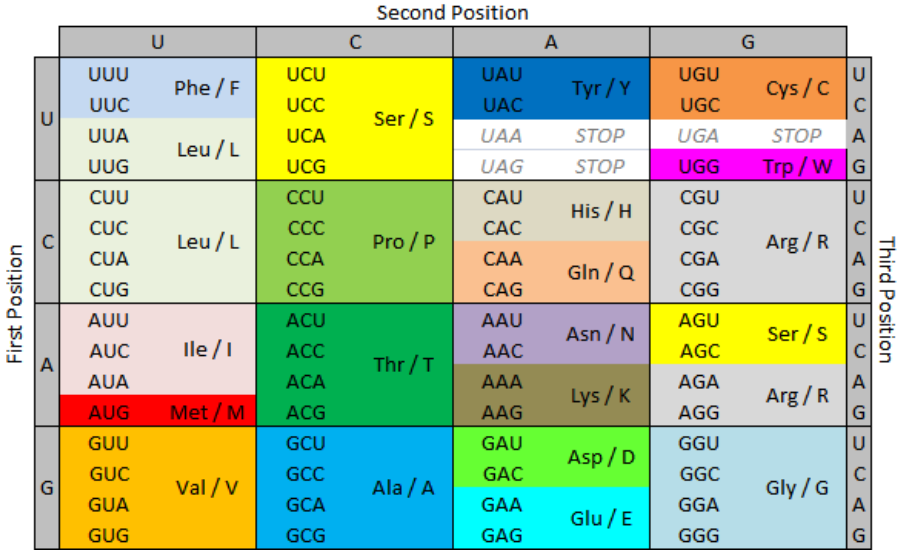

Figure 1.4: Semantics of the triplet codes for each amino acid.

Once the ribosome encounters a start codon, the translation is initiated and for each following codon the corresponding amino acid is connected to a growing chain of amino acids. This chain is what ultimately will lead to the finished protein product. Once the ribosome encounters one of the stop codons it initiates a process that induces the binding of a release factor protein which will release the created protein from the mRNA-ribosome assembly. As can be seen from the table, there exist more distinct codons (27) then amino acids (20). This allows for natural redundancy in which evolutionary speaking important

amino acids appear multiple times.

Metabolic pathways

The resulting products from transcription and then translation then interact in a series of chemical reactions known a metabolic pathways. In each pathway, a principal chemical is modified by these series of chemical reactions. These reactions are catalyzed by specific proteins, also known as enzymes, and often require additional chemicals like minerals, vitamins and other cofactors. The involved compounds are referred to as metabolites. Pathways may include other pathways, and thus create an intricate network known as the metabolic network, see figure 1.5 for an overview of how the major metabolic pathways interact with each other.

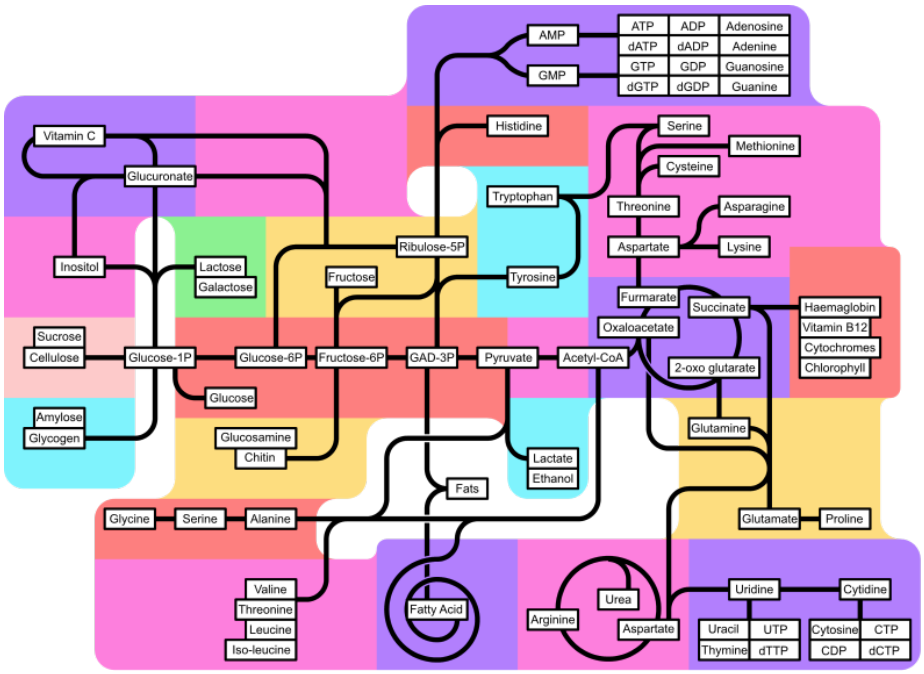

Figure 1.5: The major metabolic pathways.

Metabolic pathways describe the process in which an initial molecule, the substrate, is transformed through a series of chemical reactions to form another product. Pathways that break down compounds are said to be catabolic, while anabolic pathways often create new biomolecules as final end-products. Although technically speaking all chemical reactions are reversible, the conditions in the cell are often such that it is thermodynamically more favorable to flow in only one direction of a reaction.

1.1.4 Genetic variation

It is clear that the biological mechanism of life is very complicated, and knows many failure points. A common cause for both evolutionary advantages and disadvantages is genetic variation. Genetic variations occurs through mutation, a change in the nucleotide sequence of a gene. A variation will result in phenotypic variation if the change in DNA sequence affects the order of amino acids resulting from translation, and the resulting difference in amino acid sequence influences the shape, and thus function of the protein. This phenotypic change can then have either one of three distinct effects on the capability of the individual to reproduce; (i) positive, when for instance the change makes the individual resistant to certain disease, (ii) negative, when the change increases susceptibility of the individual to disease and (iii) neutral, in which case the individual perceives no altered effect. It deserves mentioning here that geographic variations largely influences the effect of genetic variation on the individual. One can imagine the case of a genetic variation that impedes the proper expression of the *EPAS1* gene, resulting in the translation of a less functional *Endothelial PAS domain-containing protein 1* transcription factor. The function of this protein is to switch on genes that produce proteins associated to oxygen binding. Someone living at high altitude in the Himalayas will perceive this particular genetic variation as having a negative effect, clearly posing an evolutionary disadvantage over 'healthy' individuals. But someone living at sea level, a situation in which this particular mutated *EPAS1* gene is probably never expressed, will classify the variation as being neutral.

Common ways to classify variations include (i) by effect on structure, (ii) by effect on function, (iii) by effect on fitness, (iv) by impact on protein sequence and (v) by inheritance ability. For the scope of this work we will confine the description to the first three classifications, namely (i) by effect on structure, (ii) by effect on function and (iii) by effect on fitness. Each of these will now be discussed in the following sections.

By effect on structure

When classifying genetic variation in terms of the effect on structure one emphasizes the alterations that occur in the genetic sequence of a given gene. Depending on where these variations occur, and whether they alter the function of the protein product will define the effects on health.

- Small-scale mutations

  Point mutations, these variations substitute one nucleotide for another, thereby slightly changing the sequence of the gene, and therefore the resulting gene product. Common subclassifications for this type of variations include (i) silent mutations, (ii) missense mutations and (iii) nonsense mutations.

  Insertions, variations that insert one or more nucleotides at a given position in the genetic sequence are considered insertions. Insertions, like

deletions, might trigger a frame-shift which is very likely to significantly alter the gene product.

Deletions, remove one or more nucleotides from the genetic sequence. Very much like insertions these variations might affect the reading frame in such a way that the gene product is significantly altered.

- Large-scale mutations

  Amplifications, duplicates certain areas of the chromosomal regions, altering the gene products significantly.

  Deletions, large-scale deletions differ from small-scale deletions such that in the former a small amount of nucleotides is involved while in the latter case multiple genes might simply disappear.

  Complex reorganizations, reorders pieces of DNA in such a way that gene products significantly change. Common subclassifications for this type of variations include (i) chromosomal translocations, (ii) interstitial deletions and (iii) chromosomal inversions.

  Loss of heterozygosity, in which case an entire allele is lost

By effect on function

Another way of looking at genetic variation, is classifying it according to how function is affected by the change. We distinguish five different types of variations in this particular case.

- Loss-of-function variations, cause the gene product to have less or no function. In case of complete loss of function this type of variation is often referred to as an amorphic mutation.
- Gain-of-function variations, modify the gene product in such a manner that the product acquires a functionality previously not present.
- Dominant negative variations, the result of this type of variations is that the gene product acts in an exact opposite manner to the 'healthy', or wild-type, variant.
- Lethal variations, cause the death of the individuals that carry the variation
- Back variation or reversion, this type of point variation cancels out variations restoring the original sequence and thereby the original phenotype.

By effect on fitness

Viewing genetic variations in terms of how they affect the fitness of the individual to cope with environmental factors and reproduce determines this classification. Seemingly, a variation has either one of two effects: harmful, or beneficial. In which case the former means a decrease in fitness of the individual, while the

latter refers to an increase of fitness of the individual. In reality these discrete categories turn out to be an oversimplification. The earlier mentioned case of *EPAS1* already illustrates this. Simply put, determining whether a variation has a harmful or beneficial effect is determined to large extend by the environment in which the particular individual resides.

1.2 Problem statement

The sequencing of the Human Genome in the year 2000 by Craig Venter and Francis Collins [9] came with tremendous promises. These effects are most probably not yet apparent and science still struggles to process the huge accomplishment into usable artifacts. Scientists now broadly agree that reading the sequence of DNA is the easy part of genome analysis; figuring out what the sequence actually means is the real challenge. So following these insights a new scientific line of research was opened as the marriage of Information Technology (IT) and biology: bioinformatics. It is here that bioinformaticians try to combine rigorous yet computationally powerful IT with the ambiguous and fuzzy biology. And as Venter and Collins efforts started to bear fruits, more and more sequencing experiments have been performed world-wide generating large amounts of data. These data are stored in countless locations, according to countless formats and structures, leading to the following problem statement: how do we manage the data chaos?

As technology advances the amount of data generated by sequencing experiments increases at an equally rapid pace. The task of biologists has traditionally been to analyze and interpret these data, effectively turning them into useful knowledge that eventually can be used in various applications. However, as data creation starts to greatly surpass the processing capability of a single human being, this analysis and interpretation activity becomes a daunting task. A possible solution to this data-overload would be the use of powerful computers, effectively pre-processing the data and separating the junk from useful data thereby allowing the expert to efficiently dedicate time to data worth looking at. For this to happen, the data needs both structuring into formats that allow for computational reasoning while at the same time computer programs that perform this act of reasoning need to be created.

We thus identify the problem in current genomics as being two fold: on the one hand we have large amount of data. These data are stored globally in a very fragmented manner, by countless institutes each of whom often uses its own, ad-hoc, standards [14]. The data are considered to be highly complex and often inconsistent with each other due to the rapidly progressing comprehension of the domain. We have come to refer to this particular phenomenon as the 'data chaos'. On the other hand, apart from this obvious issue there is also a more hidden case, where conceptually speaking the domain is very much ambiguous and fuzzy. This 'conceptual chaos' poses many barriers on promising reasoning technologies, that would otherwise enable exciting possibilities. These two pillars that define the main motivation behind this work will be discussed

separately in the following sections.

1.2.1 Data chaos

Genomic data is stored globally in a very fragmented manner. The lack of a consensus about these data, how to store and validate them, often results in inconsistencies and sometimes invalid information, as has been identified and confirmed by [15] and [16]. As an added problem, the fragmentation requires a high level of expertise for these data to be interpreted, which is the main task of a biologist. All in all, the core tasks of biologists become tedious, error prone and costly in terms of both time and money due to this lack of unity. In the context of this work we will now introduce some of the common repositories of genomic data, please notice that we follow a similar structure as above in section 1.1, starting with genomic structure subdomain, moving through the genotype to phenotype process and pathways towards genetic variation.

The National Center for Biotechnology Information[1] (NCBI), established in 1988 as a national resource for molecular biology information, NCBI creates public databases, conducts research in computational biology, develops software tools for analyzing genome data. Since then they have continued to strike a compromise between the convenience and simplicity required for the everyday use of human gene nomenclature and the need for adequate definition of the concepts involved. In short, NCBI is a U.S. government-funded national resource for molecular biology information, providing access to a variety of data including genetic variations [17] and genetic sequences [18]. See [19] for a detailed overview of all services and tools provided by the NCBI. A data repository similar in mission to NCBI, is Ensembl[2] [20]. Ensembl is a joint project between EMBL-EBI and the Sanger Centre to develop a software system which produces and maintains automatic annotation on genomes. The goal of Ensembl was initially to automatically annotate the genome, integrate this annotation with other available biological data and make all this publicly available via the web. Since the website's launch in July 2000, many more genomes have been added to Ensembl and the range of available data has also expanded to include comparative genomics, variation and regulatory data.

The Online Mendelian Inheritance in Man[3] (OMIM) [21] [22] project, also a project under the NCBI umbrella, consist of a semi-structured collection of diseases and genetic mutations linked to these diseases. The full-text, referenced overviews in OMIM contain information on all known mendelian disorders and over 12,000 genes. OMIM focuses on the relationship between phenotype and genotype. It is updated daily, and the entries contain copious links to other genetics resources. It is available as a single file download but due to it's low degree of structure very difficult to process automatically. It is for this latter reason that OMIM is not considered in this work.

[1] http://www.ncbi.nlm.nih.gov/
[2] http://www.ensembl.org/
[3] http://www.ncbi.nlm.nih.gov/omim

Bioinformatics resources that provide information on pathway data include the REACTOME[4] [23] [24] and KEGG[5] [25] [26] [27] databases. REACTOME is an open-source, open access, manually curated and peer-reviewed pathway database. Pathway annotations are authored by expert biologists, in collaboration with Reactome editorial staff and cross-referenced to many bioinformatics databases. These include NCBI Entrez Gene, Ensembl and UniProt databases, the UCSC and HapMap Genome Browsers, the KEGG Compound, PubMed, and Gene Ontology. KEGG (Kyoto Encyclopedia of Genes and Genomes) is a collection of online databases dealing with genomes, enzymatic pathways, and biological chemicals. The PATHWAY[6] database records networks of molecular interactions in the cells, and variants of them specific to particular organisms.

Apart from datasources that strive to provide structural information like genetic sequences of genes, chromosomes and entire genomes, there are also data repositories with the sole purpose of storing genetic variational data. The Human Gene Mutation Database[7] (HGMD) [28] provides a per-gene repository of mutations, or base changes that research has uncovered to be associated to disease. The full data set of HGMD can be acquired for a fee, while a less complete version is available for free in case of academic use. Given that it is curated by experts, reading scientific papers and extracting the mutational data from them, the source is considered to be highly reliable. At present day the HGMD is considered to be the primary source for obtaining genetic mutations among various genes, although alternatives exist. Many of these alternatives are non-profit based, meaning they provide open-access without limitations. Examples include the Catalogue of Somatic Mutations in Cancer[8] (COSMIC) [29], the HapMap[9] [1], the Breast Cancer Information Core (BIC) [30], dbSNP[10] [17] and many Locus Specific Databases (LSDBs). An LSDB aims to store only data for a specific locus, or region of the genome. Often these type of data sources store genetic variations for one gene, or a limited amount of genes. An interesting effort to standardize these LSDBs, is the Leiden Open Variation Project (LOVD) [31].Table 1.1 shows an overview of available data sources that contain data on the BRCA1 gene. It is generally assumed that the data overlaps at least to some extend, but the exact amount is unclear.

Adding to this already large list of data sources come the repositories that provide data on proteins. The mission of UniProt[11] [32] is to provide the scientific community with a comprehensive, high-quality and freely accessible resource of protein sequence and functional information. InterPro[12] [33] is an integrated database of predictive protein "signatures" used for the classification and automatic annotation of proteins and genomes. InterPro classifies sequences

[4] http://www.reactome.org/
[5] http://www.genome.jp/kegg/
[6] http://www.genome.jp/kegg/pathway.html
[7] http://www.hgmd.cf.ac.uk/
[8] http://www.sanger.ac.uk/genetics/CGP/cosmic/
[9] http://hapmap.ncbi.nlm.nih.gov/
[10] http://www.ncbi.nlm.nih.gov/projects/SNP/
[11] http://www.uniprot.org/
[12] http://www.ebi.ac.uk/interpro/

Table 1.1: Number of genes, and amount of mutations for the BRCA1 gene per genetic mutation source.

Source	Genes	BRCA1
HGMD (academic)	3132	1085
HGMD (commercial)	4122	1339
COSMIC	18647	15
LOVD BRCA1/2	2	502
BIC	2	1446

at superfamily, family and subfamily levels, predicting the occurrence of functional domains, repeats and important sites. InterPro adds in-depth annotation, including GO terms, to the protein signatures.

Clearly a large amount of databases exist, and every subdomain has at least two different repositories in use. The above only shows a fraction of what databases actually exists in the domain, which is estimated to be more then 1300. Some of those provide comprehensive ways to access their data, like Ensembl does with Biomart [34], while others provide no others means then either a manual copy-paste process or screen scraping based methods, like the HGMD. In general the data can be accessed through public FTP servers. The data are then provided in Tab Separated Files (TSV), Comma Separated Files (CSV) or XML. In other cases yet, the SQL-based database in use by the repository can be downloaded as a whole and installed as a local copy. It is clear that for each subdomain various data sources exist, apart from this each has its own manner of making the data accessible to third parties.

1.2.2 Conceptual chaos

The conceptual chaos, although largely related to, is fundamentally different from the above stated data chaos. In short, having a fragmented data storage is one thing, not knowing the precise semantics of those data is of a different order. In rapid evolving domains like genomics it comes as no surprise that the state of knowledge keeps progressing, maybe even on a daily basis. Although in principle a benevolent process, this also means data have a risk of becoming outdated; essentially rendering them invalid. Apart from this fact, it is difficult to appropriately store data in such a manner that it allows for applications later on that were not envisaged at the time of storage. In biology, basic concepts like a gene, or allele still have not been defined to a level of formality usually seen -and often necessary for a proper application- in Information Systems [2]. Fixing exactly this issue has been topic of research for over a decade, and the Gene Ontology[13] [4] has commonly been recognized to be the de-facto standard as a result. Conceptual modeling based solutions include [35] [36], some first efforts, and [37], which builds upon the former. A detailed state of the art will be discussed in section 2.2.

[13] http://www.geneontology.org/

Bringing order in the conceptual chaos has the potential benefit of allowing a computationally based data selection process through which relevant data can be sifted out and presented to a domain expert, often a biologist, for further interpretation. Clearly, the amount of data being generated currently -which in the future is only expected to increase [38]- is impossible to process by a single human being. It is therefore becoming increasingly difficult for biologists, whose core task is interpreting the various data, to assemble a 'complete' picture and act on that. A possible solution exists in the use of computers to alleviate the load, and pre-select potentially relevant data from the data chaos. But in order to do this, a proper semantic understanding of the domain is required, exactly something that the conceptual schema of the human genome aims for.

1.3 Contribution

First of all it is important to mention this work has been the result of a multi-disciplinary effort, combining conceptual modeling techniques to the domain of bioinformatics, and genomics specifically. It can be considered a case study for the application of Information Systems methodologies in new areas, thereby defining a clear contribution to the IS community. At the same time, the contribution to bioinformatics is clear in that our approach shows clear evidence of improving current problems.

Fully resolving the issues of chaos stated above represent an immense task, far outside the scope of this work. We do however aim to present a step in the direction we believe is right towards a bioinformatics utopia where both conceptually speaking, and in terms of data fragmentation the chaos is no long existing. Conceptual modeling as a methodology has been long recognized as a proven way to improve Information System quality, as will be discussed shortly in section 2.1. Bioinformatics in general is producing large amounts of complex data, that need to be interpreted by human experts. It is clear that to enable a proper management of this task, Information Systems can prove very useful. Seen from this perspective it makes perfect sense to use a well-defined conceptual schema as a basis.

The Conceptual Schema of the Human Genome presented in this work can be used as a type of blueprint schema from which generic Genomic Information Systems (GIS) can be created, both manually and automatically using well-proven MDE methodologies. It is expected that this effort will allow for easier creation of more consistent and correct GIS, as has been shown to happen in other areas when applying MDE [39]. Another interesting of the CSHG introduces itself as the implicit specification of the underlying ontology. By viewing a domain as a set of concepts with properties, interconnected through various types of relations the engineer, along with the domain expert, creates a specification of a conceptualization. The added advantage of using a visual representation -as is used by the conceptual schema- is that it is intuitive to both engineer and domain expert. This latter property enables the engineer to work much closer to the problem domain, effectively having a continuous and direct feedback loop

with the domain expert, eventually leading to a higher quality representation of reality. Adding to the ongoing discussion of how -and if- ontologies and conceptual models significantly differ is an important academic contribution of this work.

The work further shows that, although efforts are on the way, the earlier mentioned claims of data inconsistency and invalidity are still very much real. We have analyzed various data sources, and report here the many inconsistencies found among them. Having a sound data set of genetic variation, validated and consistent, in itself presents clear advantages to the domain. The effort as present in this work does not pretend to offer new biological knowledge to the domain, rather making the already available more consistent, valid and accessible. The current situation in which researchers, biologists and clinical staff need to navigate the many existing resources is time consuming, costly in terms of money and error prone. The effort that has gone into analyzing, interpreting and eventually loading the datasources mentioned in this work has lead to the establishment of a sound, reliable data set not just from a biological point of view but also from an Information Systems.

1.4 Methods and materials

For the purpose of this work the Conceptual Schema of the Human Genome has been used to create a database, the Human Genome Database or HGDB. Transforming the model into a database was done manually, although recent efforts have gone into automating this effort by using the Moskitt[14] tool [**gin:2009**]. The database has been created in Oracle Database 11g and runs on a virtual CentOS 4 installation. The various databases that have been analyzed and will be discussed in their respective chapters are as follows: (i) the Human Gene Mutation Database (HGMD), (ii) the dbSNP and (iii) the BIC. Currently work is being done on integrating the LOVD LSDB's that store data on the Usher syndrome[15] into HGDB using the same loading module as will be presented in this document. The preliminary results of loading these repositories in HGDB will be included in the overviews for informative purposes, but will not be discussed in detail. Most of the development took place on a workstation with the following specifications.

- Intel Core 2 Quad Q9550 @ 2.83GHz x 4 CPU
- 8Gb RAM
- Ubuntu 10.10 64-bits
- 500Gb HDD

As the current approach is gene centric, alleles and their reference sequences play a vital role in the creation of the HGDB. Their importance is obvious as

[14] http://www.moskitt.org
[15] https://grenada.lumc.nl/LOVD2/Usher_montpellier/USHbases.html

all data, like genetic variation, is located in reference to these sequences. Table 1.2 provides an overview which reference sequences have been used for each gene, including accession and gi number. The accession number identifies each reference sequence uniquely, while the gi number serves as a versioning number identifying the sequence through time.

Table 1.2: Genes and their reference allele accession number, as well as the gi number

Gene	Accession	Gi
BRCA1	NG_005905	262359905
BRCA2	NG_012772	256574794
CDH23	NG_008835	209977015
CLRN1	NG_009168	218505692
COL1A1	NG_007400	167830498
COL1A2	NG_007405	167860098
DFNB31	NG_016700	291049787
FBN1	NG_008805	283837777
GPR98	NG_007083	169790827
MYO7A	NG_009086	215983053
NF1	NG_009018	213385299
PCDH15	NG_009191	218749815
USH1C	NG_011883	226874844
USH1G	NG_007882	189011553
USH2A	NG_009407	222352133

Chapter 2

State of the art

This chapter will discuss some of the research that was previously done. As this work represents a multidisciplinary effort it is difficult to view the Information Systems and genomics parts separately. We will discuss how conceptual modeling is viewed from both an Information Systems point of view, as well as shedding light on the current state of affairs in the bioinformatics domain considering this issue. We will then proceed to discuss what is currently being done to resolve the data- and conceptual chaos earlier mentioned. We will focus on the ontology-based solutions, as it provides interesting issues to discuss about as the boundaries between ontologies and conceptual models need to be defined, and later fade away to uncover that they are actually very much alike. Showing how ontologies and conceptual models, rather then being mutual excluding entities, can be used in a collaborative way to deal with the chaos is one of the prime contributions of this work.

2.1 Conceptual modeling in Information Systems

Conceptual modeling (CM) is often suggested but not as often put to practice. CM is expected to improve Information Systems' requirements determination and analysis of the problem domain as well as other parts of the systems development lifecycle. The result of the conceptual modeling effort crystalizes in a conceptual model. Defining this base concept is not trivial, as many varying definitions exist. The one adopted in this work is as follows: "*A conceptual model is a simplified representation of reality, devised for a certain purpose and seen from a certain point of view*". The idea of a conceptual model is to have a way of simulating a part of reality in a manageable manner, in order to be able to reason over the creation process of artifacts in that reality. Seen from this perspective, conceptual models in Information System engineering serve the same purpose as scaled models of cars used in wind tunnel testing do in the car industry. Various other uses of the technique have been identified, including (i) a communication tool between engineers and domain experts [40],

(ii) a formal conceptual foundation for organizational Information Systems at various levels (a common accepted model of reality and a communication tool between IS engineers, and developers), (iii) a foundation for applications developed by end users, and (iv) an essential part of the system documentation for the maintenance of the system.

When CM is properly applied, the final Information System is expected to be functionally richer, less error-prone, more aligned with user requirements, as well as being more flexible to coping with changing user requirements and less expensive [41]. Another advantage of CM is that it allows the software engineer to work at higher levels of abstraction, as well as provide a comprehensive way of communicating with domain experts. With this latter we refer to the intuitive nature of visually represented models that convey the understanding of the domain in a much clearer way then mere descriptions, or formal text-based methods as confirmed by Larkin and Simon as early as 1987 [42]. This feature allows the engineer to work closely together with the domain expert, ultimately leading to a more precise understanding of the domain, the problem and how to solve it using the appropriate Information System. Working with conceptual models enables engineers to decompose design problems into parts that can then be further assigned to professional with various levels of expertise in different areas of systems development and implementation [43]. It is a well known fact in Information Systems engineering that the later an error is encountered, the more costly its resolution. Detecting errors in an early stage thus makes full sense. It is for the above reasons that conceptual modeling as a discipline has become an important area of research in Information Systems engineering as pointed out by Topi and Ramesh [44].

Pastor et al. advance on these efforts by presenting the OO-method, an Object-Oriented based formalism that allows for 'automated programming' [7] [39]. This latter refers to the process of viewing conceptual modeling a logical step in the constant evolution of programming levels towards higher levels of abstraction. When considering conceptual schemas like a high level programming language, which can be compiled into machine code through various automated steps, it becomes clear how the earlier mentioned advantages of CM provide significant contributions to the domain.

Currently, the EER and Object-Oriented (OO) formalisms are two to the most common CM techniques used in systems analysis and design [45]. The most widely employed modeling component is without a doubt the class diagram, well ahead of use cases and other elements [46]. Aguirre-Urreta and Marakas [47] critically review the EER and OO empirical studies. It was found that current empirical research of the domain happens without sufficient consideration for the particularities of the process that mediates between inputs and outputs. As an additional contribution of this work five dimensions on which models commonly differ have been identified. First, the level of ontological expressiveness among modeling techniques may vary. Interesting views on this topic are presented by Wand et al. [48] and more recently, Weber [49] and Wyssusek [50]. Currently the application of ontologies to the earlier mentioned conceptual chaos in bioinformatics is trending. For this reason, this point is of

particular interest in the context of this document, as will be further discussed in section 2.3. Second, the differences in degree to which engineers are experienced in employing certain modeling methods may affect the outcomes to a large extend.

Third, as Khatri et al. [51] show, the level of knowledge about the problem domain has a strong impact on the performance with usage of conceptual models. This finding can be stretched even further by stating that individual differences among subjects play a role on the performance of the task. Fourth, consideration of the informational and computational equivalence of conceptual models used in research may lead to an expectation about which types of differences may arise as a result of the comparison. And fifth, a growing amount of research has focused on the alternative modeling practices that can be used to map real- world phenomena to the constructs provided by the modeling technique. This research has highlighted the importance of considering which of the many methods afforded by a particular formalism were used to construct conceptual models, and what differences may occur as a result.

2.2 Conceptual modeling in bioinformatics

The first and most relevant contributions in this field are those made by [35], and it should be considered the starting point for the later efforts done by Pastor et al. [52] [53] [37] and this work. While conceptual modeling has shown in the past to significantly improve software quality [54] [39], the enticing idea flowing from this predicate is that modeling the human genome might very well improve the quality of life itself. Of course, if we want to reason about genetic elements, we need to know their conceptual significance, how they relate to each other and to the environment. It doesn't come as a surprise then, that various approaches to the issue have been undertaken in the past, amongst which natural language descriptions and ontology-based solutions. Natural language descriptions work very well and are recognized as such in small, controlled environments where all knowledge is already present in some form. However, the extremely large, highly complex and ambiguous area of genetics does not fit this description, and different solutions have to be explored.

Although the issue of ontologies and their applications will be discussed in more depth in the shortly in section 2.3, we strive here to provide a bird's-eye view of how conceptual modeling in bioinformatics has been applied in the past. As we accept that the philosophy behind conceptual modeling in many cases resembles that of creating an ontology, it is clear that this section would be incomplete without explicitly covering ontologies. Ontology is the philosophical study that deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences. Originally adopted by the information science as a means of describing a common vocabulary when designing and constructing an Information System, ontologies are now being used in many more domains, among which: knowledge engineering [55], [56],

[57], [58], knowledge representation [59], [60], [61], qualitative modeling [62], [63], [64], language engineering [65], [66], database design [67], [68]. In these contexts, ontologies are usually understood as dictionaries, taxonomies, categorization schemata or modeling languages. Schulze-Kremer [69] confirms the need for a solution to the conceptual chaos by stating: "*to eliminate semantic confusion... or to provide an exact, semantic specification of the concepts*". He further proceeds to describe two , well-known at the time, biological ontologies: *Cyc* [70] and *microKosmos* [71]. The Gene Ontology (GO) as presented by Ashburner et al. [4] in 2000 is nowadays considered the de-facto standard as the biology ontology and can be considered an evolution of the Ontology for Molecular Biology as initially presented by Schulze-Kremer [69]. Schwarzer et al. [72] further elaborate on GO by describing the application of the Gene Ontology to the SNP concept. Coulet et al. [73] proceeds to provide an ontology-based converter that allows for solving the notational problems associated to heterogeneous SNP descriptions.

Bard and Rhee [74] offer a review of ontologies in biology. Some examples of relevant ontologies include the *Cell Ontology (OBO)*[1], *Galen*[2], a management architecture for clinical information that includes an ontology for human anatomy and *MetaCyc*[3]. It is interesting to note that in the 32 "*principal biological ontologies and other web sites*" three entries can be directly linked to GO, and probably more when delving deeper into the separate entries. This confirms our earlier statement that the GO acts as a de-facto standard. Bard and Rhee proceed to clarify the use of ontologies in general, and applications for them in biology in particular as well as providing a future vision on the topic. Their main conclusion is that the key to the general use of ontologies will be access to the data in biological databases -the genetic variation data repositories discussed later in this work fall into this category- that are annotated with the knowledge in these ontologies.

An Information Systems approach to this specific biological problem space is not entirely new. Okayama [75] describes the conceptual schema of a DNA database using an extended entity-relationship model. Chen et al. [76] has indicated how an extended object data model can be used to capture the properties of scientific experiments, and [77] includes models for representing genomic sequence data. Paton et al. [35] advanced on this work by presenting a first effort in conceptually modeling the *S. cerevisiae* genome, which is a type of yeast, by proposing a collection of conceptual data models for genomic data. Among these conceptual models are a basic schema diagram for genomic data, a protein-protein interaction model, a model for transcriptome[4] data and a schema for modeling alleles. Whereas [53] provides a broader view by presenting conceptual models for describing both genome sequences and related functional data sets, [78] further elaborated on the basic schema diagram for genomic data thereby

[1] obo.sourceforge.net/list.shtml
[2] www.opengalen.org
[3] http://metacyc.org
[4] The transcriptome is the set of all RNA molecules, including mRNA, rRNA, tRNA, and other non-coding RNA produced in one or a population of cells

narrowing the focus and specializing it for the human genome.

Whereas Paton et al. provide a broader view by presenting conceptual models for describing both genome sequences and related functional data sets, [52] converged on the basic schema diagram for genomic data adapting it to the human genome and eventually produced a database, the human genome database (HGDB) corresponding to this model and following the standard rules of logical design. This database is now in the prototype phase and the first 2 genes, NF1 and BRCA1, have been partially loaded. Pastor et al. describes the evolution HGDB went through during the process of conceptually mapping HGDB and HGMD to each other in [53]. In [78] Pastor et al. describe the evolution of the model more in general and provide a descriptive overview of how the model came to be, and from where it evolved to what it is now.

Banning ambiguity in the genomics domain has been subject of many earlier attempts, including ontologies and formal descriptions in natural language. Looking from the computer scientist perspective, ambiguity is usually considered an undesirable and often avoidable feature. Indeed, in computer design the behavior of the system is always intended to be known. In biology, and especially genomics, this is simply not the case. Complexity derived from the randomness which created the conditions allowing life to emerge is today obscuring the processes driving this very same system.

2.3 Ontology based solutions

The sequencing of the Human Genome in the year 2000 by Craig Venter and Francis Collins [9] came with tremendous promises. These effects are most probably not yet apparent and science still struggles to process the huge accomplishment into knowledge artifacts. Scientists now broadly agree that reading the sequence of DNA was the relatively easy part of genome analysis; figuring out what the sequence actually means is the real challenge. Following these insights a new scientific line of research was opened as the marriage of IT and biology: bioinformatics. It is here that bioinformaticians try to combine rigorous yet computationally powerful IT with the ambiguous and fuzzy biology. And as Venter and Collins efforts start to bear fruits and technology rapidly advances, more and more sequencing experiments are being performed worldwide generating large amounts of data; leading to the following question: how do we manage the data chaos?

Current solutions are often based on ontologies, most notably the Gene Ontology (GO). Literally translated from ancient Greek, "$o\eta\tau o\sigma$" means "of that which is" and "-$\lambda o\gamma\iota\alpha$" science, study. The science of Ontology (uppercase "O") is diverse and dates back to the early Greeks, where it referred to the analytic philosophy of determining what categories of being are fundamental, and in what sense items in those categories can be said to "be". In modern times, an ontology (lowercase "o") is considered many things. Gruber is credited for introducing the first compact, yet complete description: "[an ontology is a] *specification of a conceptualization*" [55].

Ontologies enable us to build large, maintainable knowledge bases that can codify what we know about specific areas of practice in precise, unambiguous terms. Adding to this, it allows us to reason over these structured knowledge bases, with the purpose of deducing new knowledge. In this short description we identify two different applications; knowledge management and knowledge deduction. An ontology can be of varying level of rigor, where a lower level, and as such more ambiguous ontology will be fine to deliver the promise of maintaining knowledge bases, but unable to allow for automated reasoning necessary to deduce new knowledge. A higher level of rigor will allow for both accurate knowledge management, and deduction of new knowledge at the cost of increased complexity.

When the Gene Ontology was conceived in 2000 [4], it came with the promise of enabling a conceptual unification of biology by providing a dynamic, controlled vocabulary. Adding to this, it was hoped that the common vocabulary would result "*in the ability to query and retrieve gene and proteins based on their shared biology*", thus deducing new knowledge. Later research has shown that Gene Ontology in reality often lacks rigor to allow for this high level of ontology application. The early discussion started by [79] and [80], stating that "*It is unclear what kinds of reasoning are permissible on the basis of GOs hierarchies.*" and "*No procedures are offered by which GO can be validated.*"'. We now proceed to discuss the main points of their work.

2.3.1 Universals versus particulars

In metaphysics, a *universal* is a meta-concept. It defines what particular things have in common, namely characteristics or qualities.The idea is that universals can be instantiated by particular things, or *particulars* (also called individuals, exemplars, instances, tokens). For example, the species *E. coli*, with the function to boost insulin production is a universal. Instantiated as the particular E. coli bacterium now existing in the Petri dish, its function is to boost insulin production in specific cells in your pancreas. Why is it important to have a distinction between particulars and universals? Consider the following case in which we have modeled a universal, say "*gene*", that corresponds to the biological concept by the same name and has a few properties: it is known for instance that a gene has a promotor and a terminator while being located on a specific chromosome. Once we now establish during a biological experiment that a certain stretch of DNA must be a gene (effectively instantiating the universal to an instance), we are now able to deduce from this knowledge that this particular stretch of DNA must have a promotor, terminator and that it is positioned on a chromosome. Imagine this same case but now we have not modeled the universal, instead storing a list of stretches of DNA (instances) that we consider to be genes. When we try to make the same deduction as earlier in case we find a new stretch of DNA that corresponds to the biological concept of "gene", we can not deduce what other properties it has.

2.3.2 Continuants versus occurrents

Entities that continue to exist through time are often referred to as *continuants*. In the GO context, organisms, cells and chromosomes are all continuants, even while undergoing changes they do not cease to preserve their identity.

Occurrents on the other hand, are never said to exist in full for a single instant of time. Rather they they unfold during successive phases, like for example a viral infection unfolds itself over time. (Biological) processes usually are characterized by passing through different states: where the nucleus is part of the cell, mitosis is a part of the cellular process.

The continuant/occurrent opposition corresponds in the first place to the distinction between substances (objects, things) and processes. GOs cellular component ontology is in our terms an ontology of substance universals; its molecular function and biological process ontology are ontologies of function and process universals. But functions, too, are from the perspective of philosophical ontology continuants. For if an object has a given function which means a token function for a given interval of time, then this token function is present in full at every instant in this interval. It does not unfold itself in phases in the manner of an occurrent. If, however, the token function gets exercised, then the token process that results does indeed unfold itself in this manner. Each function thus gives rise, when it is exercised, to processes or activities of characteristic types.

2.3.3 GO's relations

The GO relation *isa* is referred to as meaning instance of, however in practice it is clearly used in such a way to indicate *is a kind of* or specialization between universals (e.g. "*p53 is a protein*"). Adding to this, sometimes the *isa* relation is used to indicate *part-of*, as in the definition of vacuolar proton-transporting V-type ATPase, V0 domain (GO:0000220), which identifies the concept as *isa* vacuolar part, rather than as a component part thereof.

The part-of relation as defined by GO indicates a transitive relation intended to conceptualize "can be part of, not is always part of". GO uses the part-of relation for representation of parts of both substances and processes, and of functions/activities. The part-of relation is ambiguous in that it does not provide clear means of distinguishing between the following cases:

- A part-of any B
- A part-of some B
- A part-of B, only when a condition holds

However useful as a controlled vocabulary, Gene Ontology all to often fails to deliver on the promise of allowing for deduction of new knowledge due to a lack of necessary conceptual rigor. Egaña Aranguren et al. [81] captures the essence of this issue by stating that "*The computers' understanding is determined by the semantics of the language*", thus if the semantics of the language are unclear, so

is the computers' understanding. It is difficult for humans to adequately reason over situations they don't understand, it is even more so for computers.

2.3.4 Conceptual modeling

Conceptual modeling is the practice of creating models for the purpose of either designing an artifact, or achieving a higher understanding of a certain domain. Often these two overlap in a process referred to as Model Driven Architecture (MDA). In MDA the main objective consists of creating high quality software, made possible by extensive use of conceptual modeling techniques. The idea is to create models of the domain and the to-be created application, after which the software can automatically be generated from these models, in some cases without human interference [39]. Clearly, for this process to succeed the model must be formal and unambiguous, i.e. the semantics of the language must be clear. MDA is most often used in an Information Systems (IS) context, but is it so strange to view the biological mechanism of life as an IS? A very complex and organically based, but an IS nonetheless. Replacing human made, silicon based, chips with organic proteins and processor instructions with DNA transcription and translation to proteins. It is often through making analogies with systems we already comprehend, that we come to understand otherwise difficult to grasp mechanisms. Currently, a very attractive, challenging line of research is the personalized medicine context (have a look for instance at [82], where concrete applications of the IS-based working environment defended in this paper are presented).

The first documented effort to combine the practice of conceptual modeling is [35] which proposes conceptual models for genomic data. Pastor et al. [83] and [37] then further elaborated on this initial work.

It is important to understand that using either approach; conceptual models (CM) or ontologies is really not that different, as a matter of fact an ontology is always present when creating a conceptual model. It is defined implicitly by the model itself, while the other way around is not always true: not every ontology has a visual representation allowing it to be named a conceptual model.

The universals versus particulars discussion is easily addressed by CMs: that which is reflected in the model must always be a universal, for instance a gene, having the following attributes: HUGO_id, chromosome. While its particulars, for instance a gene that has the HUGO assigned id "BRCA1", and which is located on the human chromosome 17. These particulars are instances of the CMs concepts, and usually exists only at runtime. Runtime being interpreted broadly: both an executing object and a database tuple, stored persistently, are considered runtime.

The GO expressiveness for conceptualizing relations among entities is not rich enough. By being unable to capture certain knowledge adequately, the uncontrolled use of non-standard operators becomes a tempting, ad-hoc, solution. Conceptual modeling offers a rich variety of relations: an aggregation is a weak whole-part relation where the members can exist without the containing or enclosing class, e.g. *Staphylococcus epidermidis* forms part of human skin

flora, but can survive without a human host. A composition is also a whole-part relation, but stronger than an aggregation such that the whole does not exist without its parts, e.g. a human individual can not exist without a brain. Further, inheritance allows in an intuitive way to identify hierarchies of concepts; a skin cell is a specified type of cell, thus inherits properties of its super type. The concept of a simple relation merely describes the abstract property of allowing communication between two entities: for instance the communication between neurons (this would be a reflective association). In case expressiveness is still not rich enough, the Object Constraint Language [84] allows for even finer specification by allowing the application of textual constraints to the models' behavior.

In general, ontologies and conceptual models are not all that different. We support the idea that in the creation of a conceptual model, an ontology is always present implicitly. The ontology as a formal representation of what is said to be in the modeled subdomain of reality can be inferred from the model itself. A main difference is the visual aspect of conceptual modeling, in which the domain expert can easily understand and therefore cooperate on the creation of the model. This latter implies that conceptual models can be created closer to the problem domain and therefore, we believe, lead to higher quality representations with a better fit with reality.

Chapter 3

The Conceptual Schema of the Human Genome

Initially an ideal model of the Conceptual Schema of the Human Genome (CSHG) was created, essentially describing how the genomic concepts should be according to the latest of knowledge. However, as data was matched from various external sources to this ideal model, it soon became clear a dichotomy existed between the ideal model and the way data was represented in the real world. A second model was created, logically named the real model. This real model serves as a practical tool of resolving the encountered limitations, it therefore compromises on the aspect of understanding the domain. The intention of the real model is to adapt the modeling elements included in the ideal model to the way in which we found that data are stored and managed in practical settings. As there is always a conceptual mapping between concepts in the ideal model and how they appear in real models, we focus in this work on the ideal model. To simplify the presentation of the schema, 6 conceptual views are considered: the structural view, the variational view, the phenotype view, the transcription view, the genome view and the bibliography/databank view. The joins between these views are pointed out by shaded boxes. We will now proceed to further detail these views in the following sections.

3.1 The structural view

A genome is defined as the entirety of an organism's hereditary information, in this case a human being. By this we mean the full sequence of base pairs that make up the genetic sequence, consult section 1.1 for an introduction to genomics. The genetic sequence is rarely one consecutive DNA molecule, but rather a set of smaller DNA molecules, or chromosomes. These chromosomes then are further decomposed into units of heredity commonly known as genes.

CSHG takes the concept of gene as a central point resulting in a gene centric conceptual schema. The structural view aims to capture exactly this by

conceptualizing common genetic concepts like *Gene* and *Allele*, figure 3.1. In terms of genetics these concepts are still surrounded with ambiguity, as shown by [2] and [3].

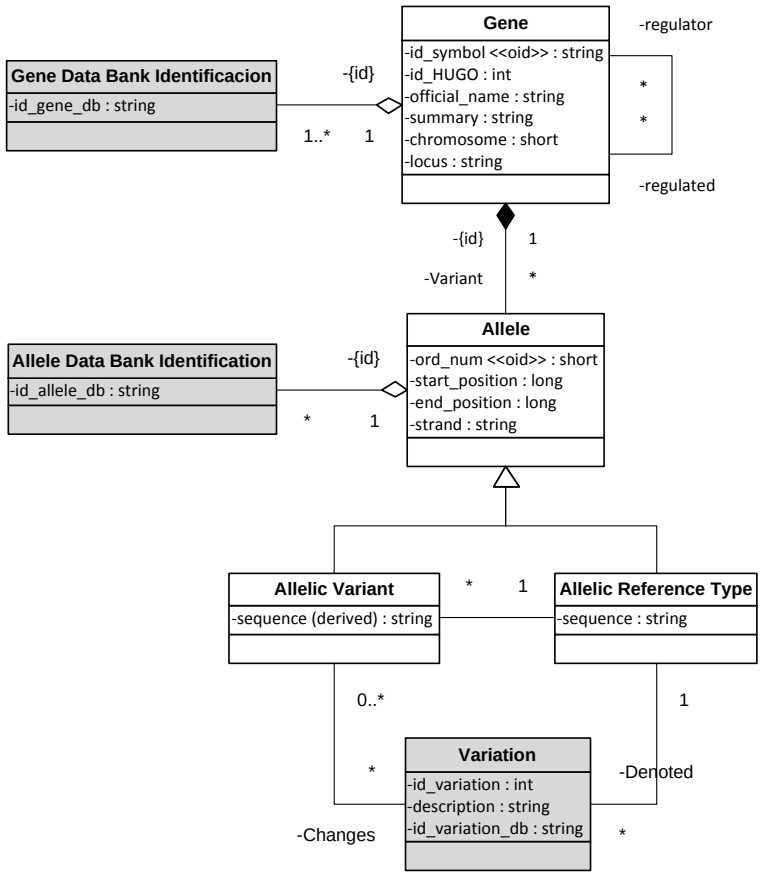

Figure 3.1: The structural view of the Conceptual Schema of the Human Genome

In CSHG, a gene is considered to be a collection of identifying properties and

attributes. *id_symbol* represents an alphanumeric code for the gene according to HGNC [85], it also functions as the primary key; *id_HUGO*, a numeric code assigned to the gene by HGNC; *official_name*, the full name of the gene; *summary*, a short description; *chromosome*, the chromosome on which the gene is located and *locus*, representing the location of the gene within the chromosome. Locus, also often referred to as cytoband, refers to stored information about the subregions of a chromosome that become visible with a microscope after staining during a specific cell cycle phase (Metaphase).

An *Allele* is the 'version' of a gene as encountered in organisms. By version we mean the exact sequence of base pairs that represent the unit of heredity we identify as gene. A particular gene might have one or many alleles associated. In some cases, different alleles lead to different phenotypes -of which some may have a negative or positive effect in terms of survivability of the individual- and in other cases the different alleles don't provoke differing phenotypes. In CSHG an *Allele* thus has a *start_position* and an *end_position*, relative to the beginning of the chromosome and a *strand* attribute to identify on which of the two DNA strands it is located, indicated by either 'plus' or 'minus'. We identify two types of Alleles: *Allelic Variants* and *Allelic Reference Type*, each of which has a single attribute *sequence*. The idea here is that a reference allele is said to exist and represents a 'standard' sequence of a particular gene, in the context of this work the so-called NCBI refSeq [18] is used. The *Allelic Reference Type* then can have various associated *Variations*, this latter concept will be discussed in detail in section 3.2. Each of these *Variations* triggers the possibility of a different *Allelic Variant*, which sequence is thus derived from the *Allelic Reference Type* and *Variation*.

The *Gene Data Bank Identification* and *Allele Data Bank Identification* entities will be discussed into more detail in section 3.6.

3.2 The variation view

Genetic variation is brought about by mutation, a change in the chemical structure of a gene. A variation can have either a negative, positive or neutral effect on phenotype. Chapter 5 will go into more detail exactly about this latter effect and how the conceptualization was introduced in CSHG. Natural selection avoids the reproduction of individuals with genetic variation that poses evolutionary disadvantages. Genetic variation with negative effect is thus expected to occur infrequently. Section 1.1.4 describes genetic variations in more detail.

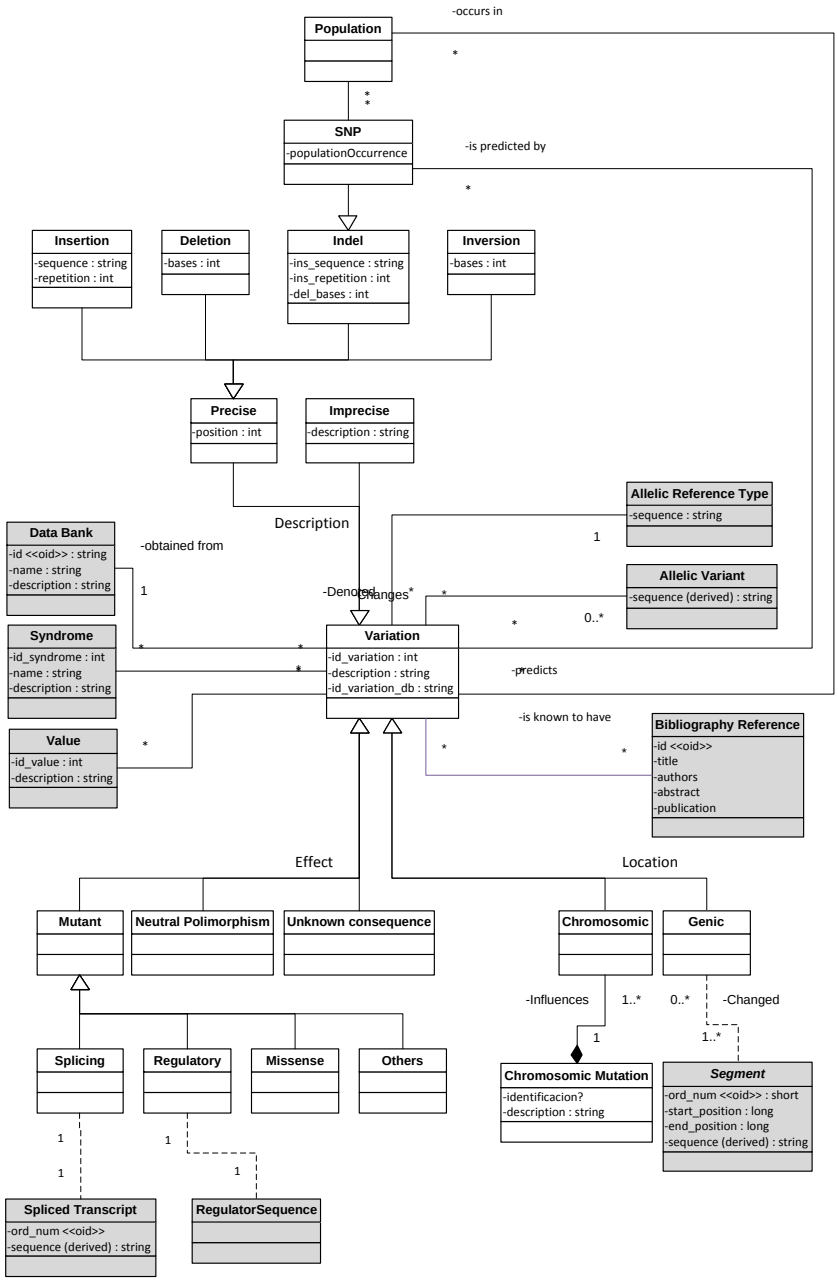

Figure 3.2: The variation view of the Conceptual Schema of the Human Genome

The variation view of the CSHG makes an effort to capture exactly these concepts, as displayed by figure 3.2. The *Variation* entity stores information about changes in a certain allele in respect to the reference, the *Allelic Reference Type*. It has an *id_variation* attribute for internal identifying purposes. *id_variation_db* refers to the identification used in the external source from which the *Variation* instance was extracted. It further holds a description, meant to store a small natural language description about the variation.

The entities specifying the *Variation* concept through generalization can then be classified into three categories; (i) location, (ii) effect and (iii) description, each representing a specific type of polymorphism. The location classes store information about whether the variation affects one or more genes. In the case of a *genic* location, only one gene is affected, while in the case of *chromosomal*, multiple genes might be influenced. The effect entities specify the variations effect on phenotype. This can either be *mutant* and thus influence phenotype in a negative way, a *neutral polymorphism* or the effect might be *unknown*. The *splicing*, *regulatory*, *missense* and *others* concepts are considered to be mutations since they have a negative effect, hence they are a specialization of the mutant concept. Ultimately, the description classes include descriptive information about the variation. Depending on the degree to which the data on the variation is precise, it falls into either the *precise* or *imprecise* class. When imprecise, the entity only stores a general description. In the case of precise data, it stores the position of the variation and further specifies the nature of the variation into four classes: *insertion* which corresponds to the insertion variation as defined in section 1.1.4, *deletion* which corresponds to the deletion variation as defined in section 1.1.4, *indel* which corresponds to the point variation as defined in section 1.1.4 and *inversion*. Each of these concepts store information about this specific type of variation and the exact attributes vary from type to type.

The *Data Bank* and *Bibliography References* entities will be discussed in section 3.6. *Syndrome* and *Value* will be explained in the following section, while *Allelic Reference Type* and *Allelic Variant* have just been clarified in section 3.1. *Segment*, *Spliced Transcript* and *RegulatorSequence* belong to the transcript view and will be discussed shortly in section 3.4

3.3 The phenotype view

The *Category*, *Feature*, *Value*, *Measurable* and *Syndrome* classes associate a variation to a phenotype. The *Syndrome* class corresponds most to the general concept of disease; Neurofibromatosis and Huntingtons are examples of instances of this class. A syndrome can be provoked by one or several variations; and a variation can have multiple diseases associated to it. Usually, syndromes are characterized by various features, instances of the *Feature* entity; in the case of Neurofibromatosis, this includes the so-called café-au-lait spot features. These features in turn, are classified by categories, which have a recursive property indicated by the self-referencing relationship. Adding to this, each feature has

an associated value, which is the measurable effect on phenotype (*Measurable*). In the case of Huntingtons syndrome, this corresponds to the blood markers used to detect tumors. It is important to note that not every variation is associated to a specific phenotype; typically, variations characterized as polymorphisms do not cause pathological phenotypes.

The *Variation* classes has been discussed in section 3.2.

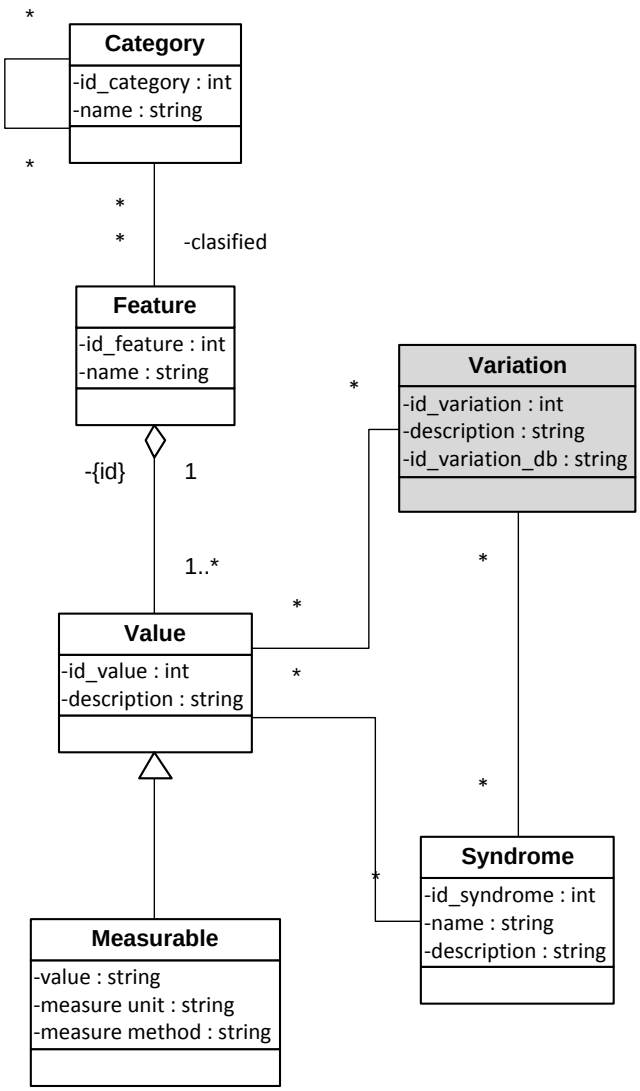

Figure 3.3: the phenotype view of the Conceptual Schema of the Human Genome

3.4 The transcription view

This view, figure 3.4, models the allele segmentation for the transcription process -the part of the conceptual model associated to the transcription process itself is explained below. The *Segment* class represents a segment of the allele, its attributes are: ord_num (identifies a certain segment among all the allele segments), start_position and end_position (initial and end position of the segment in the chromosome), and sequence (DNA sequence between start_position and end_position). The *Segment* entity has four specialized entities classified by their function in the transcription process: *Promoter* (DNA sequence region that facilitates the initiation of the transcription process); *TranscribedSequence* (DNA sequence transcribed by the RNA polymerase II); *Terminator* (DNA sequence that signals the end of the transcription process); and *RegulatorSequence* (DNA sequence that regulates one or many transcription units). The *Transcription Unit* class models -as its name indicates- the biological concept of a transcription unit; the attribute ord_num identifies a specific transcription unit in the system. This class is defined as a composition of a *Promoter* segment, many *TranscribedSequence* segments (many transcribed sequences may exist in the same transcription unit, all starting at the same position), many *Terminator* segments (a transcription unit may have more than one terminator segment), and many *RegulatorSequence* segments (a transcription unit may have many regulatory segments, shared by different transcription units belonging to several genes in the most general case). It is interesting to note here that regulator sequences do no necessarily have to reside within the gene they regulate, even more they can reside on entirely different chromosomes [86].

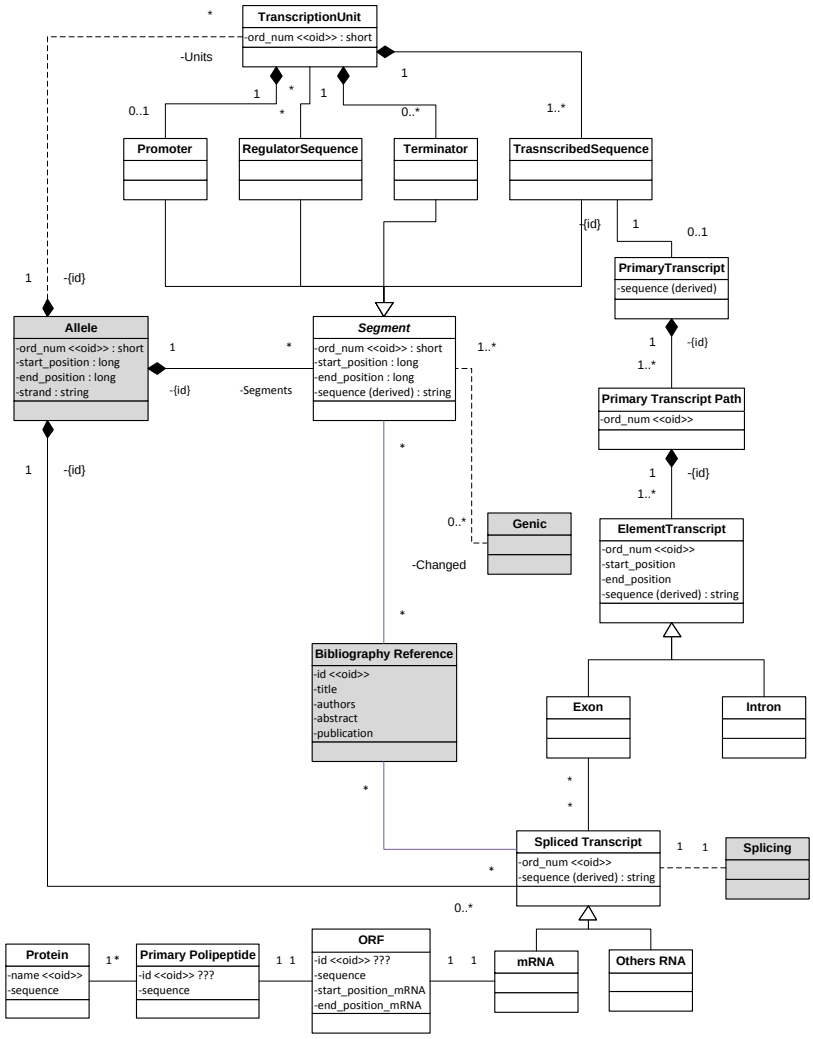

Figure 3.4: The transcription view of the Conceptual Schema of the Human Genome

Transcription is the biological process in which the genetic sequence in DNA is transferred, or transcribed onto RNA, a molecule very similar to DNA. This allows the genetic sequence to leave the cell nucleus, in order to reach other biological systems that will eventually lead to the construction of complex proteins. A gene can be thought of a as a set of separate blocks of nucleotides. The coding blocks, or exons, eventually form the transcript, although which exons are included in which version of the transcript may vary. The process that regulates which exons are included, and discards all of the introns is called splicing, see [2] for more information. The Transcription view in [fig. 2], models the basic steps in protein synthesis. The *Primary Transcript* class represents the transcribed copy from DNA to RNA of the *TranscribedSequence*. In the biological process of transcription, the primary transcript is an RNA molecule, containing a literal copy of the DNA sequence of a gene, including all coding (exons) and non-coding (introns) fragments. Its sequence attribute is a derived attribute from the *Segment* class. The *PrimaryTranscriptPath* class models the different splicing factor-driven partitions of the *Primary Transcript*, its attribute ord_num identifies a partition from the complete set of partitions of a *Primary Transcript*. In the *ElementTranscript* class, the ord_num attribute identifies a specific fragment within the partition. The *Exon* and *Intron* classes specialize the type of partition fragments. The *Spliced Transcript* class represents different exon combinations (sequences) of a *Primary Transcript*, its ord_num attribute identifies it among all the allele spliced transcripts. The result of these combinations will be the *mRNA* and *others RNA* types (specialized classes from *Spliced Transcript*).

The *mRNA* contains a nucleotide sequence that could potentially encode a protein, this is known as *ORF* (Open Reading Frame). The id attribute of an *ORF* identifies it in the system, and the sequence attribute stores the codifying sequence. The *Primary Polypeptide* class describes the protein primary structure: the amino acid chain obtained as a result of the translation of an *ORF*. This amino acid chain undergoes chemical transformations and the final result is a functional protein, represented in the model as the *Protein* class. A protein can consist of one or more *Primary Polypeptides*. In the *Protein*, its name attribute represents the name of the resulting protein, and its sequence attribute the amino acid sequence.

The *ChromosomeSegment* class represents the segments that comprise the chromosome. This class has a sequence attribute which stores the corresponding DNA sequence delimited by start_position and end_position attributes. A chromosome has two main types of segments: coding-related segments (*GenicSegment*) and non coding-related segments (*NonGenicSegment*). Two classes specialize *NonGenicSegment*: the *IntergenicRegion* class (the regions between genes) and *ChromosomalElement* class. The last one has three specializing classes that describe other elements of the chromosomes (*Centromere*, *Telomere* and *ORI*) whose function is to keep the chromosome functional and are not involved in protein production.

3.5 The genome view

The Genome view, figure 3.5, models individual human genomes. This view is interesting for future applications, since massive parallel sequencing technologies will allow the complete sequencing of individual genomes at a very low price in the near future [87].

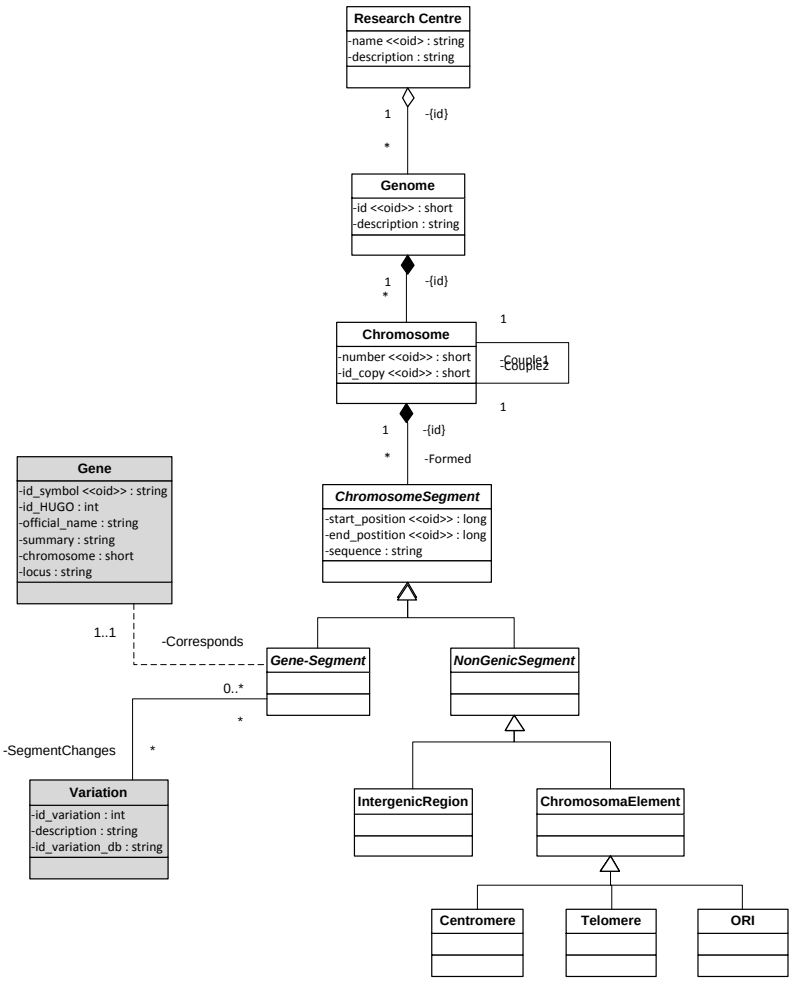

Figure 3.5: The genome view of the Conceptual Schema of the Human Genome

The class *Research Centre* represents the labs or research centers where an individuals' human genome was sequenced. A genome (*Genome*) is considered a set of chromosomes (*Chromosome*). The number attribute identifies a chromosome in a genome. The couple relation on the *Chromosome* class represents the concept of homologue pairing, i.e. every human cell will carry two equivalent chromosomes -one from the father and one from the mother- with the same genes but different alleles for each gene.

3.6 The bibliography/databank view

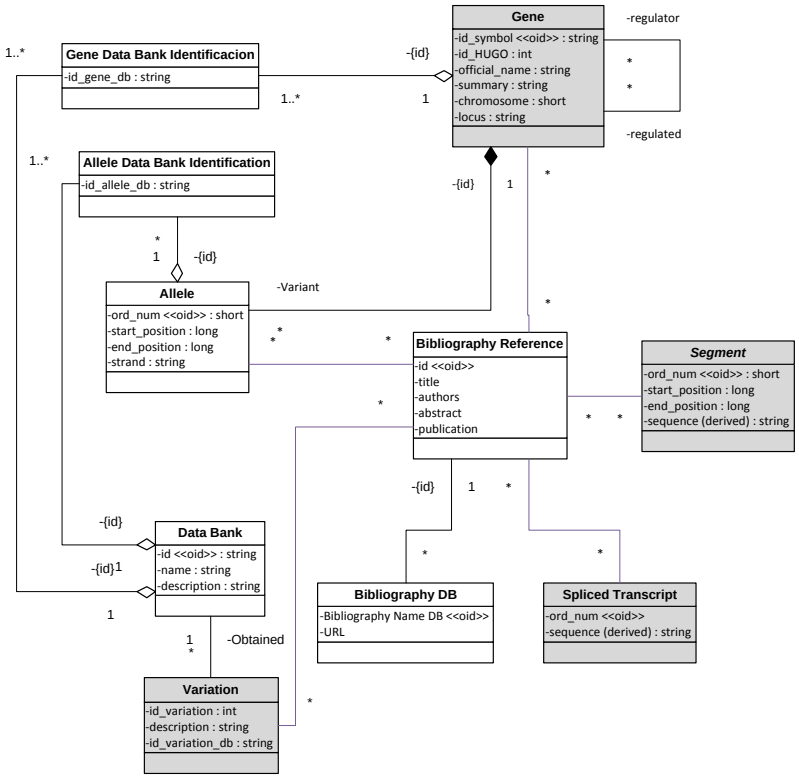

Figure 3.6: The bibliography/databank view of the Conceptual Schema of the Human Genome

Chapter 4

Solving the HGMD case

The first part of this work centers on emphasizing the importance of applying conceptual modeling techniques before constructing any Information System, and Genomic Information Systems in particular. The second part starts here and describes the results of loading the BRCA1 gene data from HGMD into the database that results from the Conceptual Schema of the Human Genome. The encountered problems serve as examples and a case proof of why an Information System without a conceptual modeling sound backbone results in myriad difficulties and undesired behavior, thereby enforcing our earlier mentioned statement.

[83] reports a study of comparing the HGMD to the CSHG, in order to identify a conceptual mapping between the two. It is this mapping that is followed in this document, and the following section will report the encountered problems for actually loading the information from the HGMD into the HGDB for the BRCA1 gene. Roughly problems can be separated in two categories; intrinsic data properties and data representation, with the final intention of providing a concrete report on what type of problems the correct load of a conceptual model-based genome data base must face and solve. This is important because the main benefits of applying conceptual modeling principles to the understanding of the human genome are located in both having the right data structure and the right data contents. Identifying and classifying these data loading problems provide general solutions that can help in solving the same problems in other biological data loading contexts. Verifiable incorrect, inconsistent or incomplete data (tuples) are examples of these encountered mishaps with the actual data, or intrinsic data properties. Difficulties associated to physically extracting the data from the external source and ambiguously description of mutation properties are typical examples of data representation problems. Naturally, the division between the two categories is not strict and thus some overlap exists, it is however useful to keep in mind that intrinsic data property problems tend to be affecting the entire genetics domain, while the data representation difficulties are restricted to HGMD.

4.1 HGMD description

The Human Gene Mutation Database (HGMD) represents an attempt to collate known (published) gene lesions responsible for human inherited disease. This database, whilst originally established for the study of mutational mechanisms in human genes [28], has now acquired a much broader utility in that it embodies an up-to-date and comprehensive reference source to the spectrum of inherited human gene lesions. Thus, HGMD provides information of practical diagnostic importance to (i) researchers and diagnosticians in human molecular genetics, (ii) physicians interested in a particular inherited condition in a given patient or family, and (iii) genetic counsellors. The HGMD can be accessed either publicly - having less entries but no fee is charged- or professionally -in which case a fee is charged for access to the full data set.

The HGMD provides an overview of mutations causing or associated with human inherited disease. Adding to that, the repository includes disease-associated polymorphisms reported in literature. In general, these data consist of various types of genetic variation. In the current version of HGMD, both somatic and mitochondrial variations are not included. All genetic variations in HGMD have been found as a result of DNA analysis, excluding thus variations inferred from amino acids sequencing. The latter is to avoid ambiguity issues that still surround DNA sequence changes. Genetic variation in coding regions that does not alter the encoded amino acid are also not included.

HGMD extracts its data from over 250 journals, which are not further specified. The papers are manually read and the data interpreted by experts, ensuring the data is in theory high quality, but vulnerable to human error. HGMD also includes some data from LSDB's. At the time of writing (20-07-2011), the HGMD contained 113 247 genetic variations, of which 82 808 are publicly accessible.

4.2 Encountered problems

HGMD distinguishes 10 mutation types: Missense/nonsense, Splicing, Regulatory, Small Deletions, Small Insertions, Small Indels, Gross Deletions, Gross Insertions, Complex Rearrangements and Repeat Variations. Roughly all the types can be mapped to the *Variation* and *Precise* concepts of the CSHG, except for the Gross Deletions, Gross Insertions, Complex Rearrangements and Repeat Variations. The latter are described in a very unstructured manner, almost natural language, and are thus considered impossible to process automatically. Appendix A.1 provides a screenshot of how precise mutations -the ones that have a precise description including position, change and phenotype- and appendix A.2 provides a screenshot of those earlier mentioned unstructured -or imprecise- mutations. The CSHG facilitates these tuples as *Imprecise*, which stores a description of the mutation.

4.2.1 Intrinsic data properties

In some cases the HGMD mutational data simply lacks entries. For instance, the splice mutations overview provided by HGMD mentions 5 mutations in intron 22, while [88] states at least 2 other mutations; IVS22+67(T>C) and IVS22+8 (T>A). Three concrete examples of this problem were encountered, all three in splicing site mutations. However, this particular type of problem is very difficult to detect, since finding them involves rereading the articles HGMD provides which is hard to automate. Thus, although only three concrete occurrences of this problem have been encountered, it is likely more exist.

Splicing site mutation CS961492 describes a C>T mutation, as a possible phenotype HGMD indicates Breast cancer. However, having the read the corresponding article [89], not once breast cancer is mentioned in combination with this mutation. The article does mention the mutation as being affiliated with men suffering from prostate cancer. Thus, deducing from the rather limited information made available by HGMD on this specific mutation, it is concluded HGMD made an error during data entry.

Splicing site mutations CS063247 and CS011027 should be located near intron 4, according to the HGMD splicing site mutations overview. However according to the splice junctions overview HGMD provides, there exists no intron 4, nor an exon 4. Since indeed both of the papers [90] and [91] state the mentioned mutations near intron 4, it would be logical to presume the problem is on HGMDs side. So at first, this specific problem instance was considered to be either a major flaw in HGMD due to inconsistent reference sequences, or a result of human error during the HGMD loading procedure. However, deeper research revealed a more subtle situation. splicing site mutations are located in HGMD by using a splice junctions overview, which provides an overview of intron- and exon borders in the gene. HGMD constructs this overview by using a NCBI reference sequence, in this case L78833. This reference provides a comment in natural language that explains the absence of an exon 4 as: "*Characterization of an aberrant BRCA1 cDNA clone in the original report [92] led to the misidentification of an inserted Alu element as exon 4. Not normally found in BRCA1 transcripts, insertion of this Alu would lead to introduction of a STOP codon. Hence, BRCA1 exons and introns are numbered 1a, 1b, 2, 3, 5, 6, etc.*".

Splicing site mutation CS012667 indicates a G>A mutation in nucleotide +3 from the start of intron 2 . However neither the HGMD splice junction overview, nor the NCBI reference gene sequence indicates a G-nucleotide at this location. A very similar event happens with splicing site mutations CS001825 and CS991331. They both involve a mutation located 7 nucleotides upstream (+7) from the start of intron 22. However, the first mentions an A>G mutation, while the latter describes, for that exact same location, a T>C mutation. Since both the NCBI reference sequence and the HGMD splice junctions overview indicate an A nucleotide at the appropriate position, and the CS001825 reference article [93] indeed mentions an A>G mutation at the specific location, one could easily conclude the CS001825 A>G mutation in the correct one. However, the

CS991331 reference article [88] does indeed point out a T>G mutation at the location, so the truth might be slightly more subtle. Since [88] involves an African-American population, while [93] entails a Chinese population, a possible reason for the irregularity might be general genetic differences between those ethnic groups. A phenomenon referred to as Single Nucleotide Polymorphisms, or SNP's [94]. 5 Occurrences of this problem have been identified in splicing site mutations, 1 in Small Deletions and Small Insertions each, leading to a total of 7 occurrences. SNP's are currently included in the CSHG as a separate concept, facilitating the placement of these type of mutations. Finding these type of genetic variations, mixed in with another type of variations does emphasize the need for an unambiguous data representation.

4.2.2 Data representation

The HGMD public version, used for this research presents data through a website in HTML tables. This makes an automatic extraction procedure very difficult, but not impossible. Using a 'screen-scraping' approach including HTML parser technology, the individual tuples can be isolated and extracted. This approach however has several drawbacks, among which its inflexibility of coping with changing environments, which especially in this rapid evolving domain is undesirable. It deserves mentioning here that the professional license, available for a fee, does allow for acquisition of the HGMD database as a .sql file, making this extraction process a lot easier. Also, the public version has a delayed update cycle, while the professional license includes the entire up-to-date dataset.

Some data is provided in natural language. For instance the fact that the first two BRCA1 exons are alternative non-coding exons is only mentioned in the header of the Splice Junctions overview: '*The first 2 exons are alternative non-coding exons* and *The translation initiation codon is located within exon 2*, this mainly affects locating splicing site mutations (1 instance). Adding to this, in Small Deletions (2 instances) and in Small Insertions (3 instances) some mutations are located through mouse-over tags, the information communicated by these tags is highly unstructured to a degree that we might call it natural language as well. Also, in the case of imprecise mutations (Gross Deletions, Gross Insertions, Complex Rearrangements and Repeat Variations), the greater part of the information presented by HGMD is in natural language, impeding an automated approach severely in the affected cases.

In some cases, the HGMD database uses different ways of locating mutations, within the same type of mutations. For instance, Small Insertion mutations CI030168, CI962219 and CI022582 happen in non-coding areas of the gene, just like the Small Deletions mutations CD991644 and CD994433. Since HGMD generally uses a cDNA codon referenced way of locating these types of mutations, and given that non-coding sequences simply not exist in the cDNA, HGMD locates these earlier mentioned mutations in a different way. In the case of Small Insertions, HGMD provides a Splice Junction reference, very much like the method used to locate splicing site mutations. In this case the CI030168, CI962219 and CI022582 mutations are located at IVS20+21, IVS20+48 and

IVS20+64 respectively. So IVS20 indicates the intron number, where +21 indicates the offset, however since no acceptor/ donor information is provided, it is unclear from which side of the intron the offset should be referenced. In the case of Small Deletion mutations CD991644 and CD994433 at first sight, no indication of how to locate them is provided. However, this information is provided through mouse-over tags in the Splice Junctions referenced form, described earlier. CD991644 is thus located by I7E8-24, aka IVS7 -15 del10. and CD994433 is located by I12+34 / polymorphism ?. This problem was thus encountered 3 times in Small Insertions and 2 times in Small Deletions, making a total of 5 occurrences.

As said, HGMD refers to codons in many cases, which are sets of three nucleotides. Since HGDB will be using a nucleotide referenced position to locate mutations, a transformation of HGMD provided data is necessary. In theory, acquiring the correct nucleotide would be a matter of multiplying the codon number by three, reality is slightly more complicated as will be discussed in the next paragraph. HGMD uses this way of locating mutations in the case of Missense/nonsense (320 instances), most of the Small Deletion mutations (288 instances), most of the Small Insertion mutations (98 instances) and Small Indels (11 instances), leading to a total of 715 instances of this problem. Retrieving the corresponding nucleotide in DNA would simply be a question of multiplying the codon number by 3, if not for the existence of introns and exons. cDNA only comprises of the genes' exons, thereby excluding the introns, contained in the DNA. Due to this fact and given that HGDB will be incorporating a DNA referenced scale, a linear transformation, by multiplying the codon number by 3, simply is not possible.

HGMD splicing site mutations are located by referring them to so-called splice-junctions. These splice junctions indicate the borders between exon and introns. HGMD thus indicates an intron border, by giving an intron number and specifying which border by providing either a donor- (ds) or acceptor- (as) site of the intron. The donor site corresponds to the side closest to the 5' end of the DNA strand, while the acceptor site corresponds to the side closest to the 3' end of the DNA strand. Then an offset is given, to indicate the amount of nucleotides between the indicated splice junction and the actual mutation. In the so-called splicing mutations overview HGMD then provides a sample sequence for each intron/exon-junction contained in the gene. This method of locating mutations is used primarily in splicing site mutations (80 instances), but in some exceptional cases HGMD also uses this notation to provide locational data for other types of mutations. For instance, In Small Deletions (2 instances) and in Small Insertions (3 instances).

In the HGMD data exists ambiguity; for instance, mutations may or may not result in a certain phenotype, this is indicated by a question mark following the supposed phenotype. However, no probability scores are stated and a mutation without a (noticeable) phenotype is considered to be a variation with neutral effect. Since variations and mutations are considered to be two different concepts in the HGDB data- model, this poses problems with loading the database correctly. 94 instances of this problem have been identified: missense/nonsense

mutations account for the most instances (73), splicing site mutations contains another 16, small deletion mutations 2 and small insertion mutations account for 3 instances.

In short to summarize the encountered problems in the data loading process. Nine distinct problems have been identified, each of which has been discussed separately in the above section. The following sections will provide useful insights on how to resolve future instances of these. The problems as encountered, will now be summarized in the following list.

Intrinsic data properties

1. Lacking data
2. Data entry errors
3. Reference to non-existing introns
4. General inconsistency

Data representation

1. Data difficult to access
2. Use of natural language
3. Inconsistent way of positioning mutations
4. Inconsistent with NCBI refSeq
5. General ambiguity

4.3 Data loading problem solutions

Inserting the 804 precise and 90 imprecise variations provided by the non-commercial version of HGMD manually seemed like a cumbersome and more importantly, error prone operation. For this reason a series of scripts was devised to automate the procedure, while at the same time providing useful experience and knowledge about how to further process the variational data automatically. A full technical report on this process, including the source code of these scripts can be found in [95]. The basic function of the software is to convert the HGMD provided mutational data into the format used by HGDB and detect inconsistencies in the data. The main tools are the cDNA- sequences provided by HGMD [Genebank, U14680] [Genebank, L78833], the NCBI DNA reference sequence [NCBI NG_005905.1] and the NCBI Coding Sequences (CDS, located in NCBI RefSeq NG_005905.1). The HGMD cDNA sequence is the aggregation of all coding sequences (exons) of the BRCA1 gene, thus excluding introns. The NCBI DNA sequence then is the complete sequence of the BRCA1

gene, including introns. The CDS information specifies what parts of the NCBI DNA sequence are coding and which are not. The scripts extract, and in some cases, calculate the variables which are to be inserted into the HGDB Variation, Precise and Imprecise tables. At the same time detecting inconsistencies in the HGMD database that can not be resolved in an automated way, indicating a manual approach is needed in those cases. Since dealing with the above mentioned problems was necessary to devise those scripts, they are considered to be the crystallized solutions to the earlier mentioned problems. The main disadvantage of this screen-scraping approach, is high vulnerability to changes in HGMD data structure. This means the scripts will require regular updating. Also, by depending on a medium like HGMD, trust is invested in the integrity of the source. However, at this point the exact reliability of HGMD has not been identified and some of the encountered problems during this project clearly suggest reasons to doubt this reliability.

4.3.1 Intrinsic data properties

HGMD lacks data entries for two reasons. HGMD might simply not be up-to-date with the latest information provided by scientific research on the subject, or HGMD missed entries during the manual loading of the database. A partial solution to this problem involves using the professional version of HGMD, which includes more and more up-to-date entries. However, it is also a characteristic of the immatureness of the field that no full coverage exists, simply not enough research has been performed to identify all existing mutations. Therefore the database will inherently be incomplete, and thus resolving falls outside the scope of this paper.

HGMD provides erroneous data for a variety of reasons, human error on either HGMD or the source paper side might be an issue. Inconsistencies between HGMD and source paper notation style might play a role. In either case, this is a very difficult problem category to detect since detection involves rereading the papers. A possible, but unsatisfactory to some degree, solution would be to perform a deep investigation on a limited amount of mutational data provided by HGMD, thereby uncovering error frequency. This error frequency, provided it is investigated according to scientific measures, can then be extrapolated to the rest of the database, thus providing a handle from which to calculate reliability of the data-set.

Since inconsistencies can be detected, a manual check of the apparent inconsistent data is possible. Indeed, the scripts indicate only a few inconsistencies and so the shear amount of papers to be read manually can be reduced drastically, making the manual approach possible in these cases. In case the source is locating a mutation within a non existing intron, the solution might be to manually complete the splice junctions overview, combining the information from the NCBI reference sequence and coding sequences information (CDS), however contradiction exists about whether an intron 4 actually exists. The implications of this, as has been discussed earlier that reference papers are using different reference sequences, are far more serious and indicate a structural flaw

in the HGMD splicing site mutations data-set since no facilities to indicate such differences exist within HGMD, neither indicate the involved reference papers exactly what reference sequence they are using. Therefore reasons exist to doubt the splicing site mutations integrity. However, detection depends on whether a given sample corresponds to the actual nucleotide occurrence at that position in the DNA sequence. Since HGMD only provides a single nucleotide sample for splicing site mutations, the odds of a nucleotide at any position in the DNA corresponding to it is 25%, reducing error detection reliability greatly. As has been mentioned, in some cases the HGMD mentioned mutation involves a nucleotide change where the to-be changed nucleotide is actually different in the reference DNA gene sequence, thereby indirectly suggesting an error on either the source paper side or the HGMD data entry side. Except for splicing site mutations, as has been discussed earlier, most of these inconsistencies can be detected with relatively high reliability. This is possible because HGMD gives a sample sequence surrounding the mutations, that can be used as a handle to see whether the mentioned nucleotide actually is the one on the calculated position in the DNA sequence. In case of an inconsistency between the reference, and the given sample sequence a manual approach is possible to investigate further.

4.3.2 Data representation

Solving the data extraction problems involves copy-pasting the HTML-tables with mutational data, provided by HGMD, into the programming logic of the scripts. By using the PHP explode-function the data is then cut into bite-size chunks and stored in an array, ready for further processing. Due to choosing this screen-scraping approach, information contained in HTML mouse-over tags is not captured. A simple solution to this problem would be to copy-paste the HTML source-code instead of the browsers rendering, effectively including the HTML tags and thus the mouse-over contained information. Then use a more elaborate algorithm to extract the pieces of information from the source. Also, since many biological information sources present their information in HTML tables as confirmed by [96], many solutions to this specific problem have been devised, although no silver bullet solution exists today. The information contained in these tags is highly unstructured to a degree where it is considered natural language. Also, the informations structure of the different occurrences of the problem differs highly. So a more generic solution has been chosen to solving this problem: instead of using the information HGMD provides to locate the mutation, the PHP script simply takes the sample string provided by HGMD and matches this to the entire BRCA1 gene sequence, given that the string is unique, a location will be found and presented.

Although solving this particular instance proved possible due to the fact the natural language contained non-vital information, future occurrences might be more difficult and no satisfying resolution exists, for coping with natural language inherently is a weakness in computer technology. HGMD has a tendency to disrupt it's own structure, by providing different ways of locating mutations within the same mutation type. This complicates an automated approach, how-

ever a generic solution to this problem has been devised. Since the main difficulty here is the uncertainty about whether HGMD might present locational data in yet other ways, it was decided not to use HGMDs locational data at all in these exceptional cases. Instead, the earlier mentioned sample sequence given by HGMD is extracted from the HTML table. The location of the mutation within this sample sequence is then found by detecting a character case change. The result of this detection is then considered to be the offset, very much alike the offset given by HGMD in for instance the missense/ nonsense mutations. The entire sample sequence is then matched against the entire BRCA1 reference gene sequence, after which the offset is added to the found location resulting in the absolute mutation position. The major flaw in this approach however, is the fact that the given sample sequence might be happening more then once in the BRCA1 gene, thereby compromising this methods reliability to some extend. For this reason, this approach is only considered usable in case the regular method, which is considered to be more reliable due to the more rigid structure, fails.

The way HGMD stores genetic variation positions, is very distinct from the format in which HGDB stores this information. Each of the HGMD variation types, undergoes a distinct transformation operation, depending on how HGMD indicates the location within the cDNA and will be discussed separately.

Missense/nonsense

HGMD indicates a missense/nonsense location by providing a codon number, referencing cDNA plus an offset. Since HGDB requires a DNA referenced nucleotide position, transformation of this data is required. First, the software composes its own cDNA sequence by extracting and merging the coding sequences from the NCBI DNA sequence using the NCBI CDS information. It then matches the composed cDNA sequence to the HGMD cDNA to detect inconsistencies. By matching the original and substituted codon (ATG > GTG) provided by HGMD the position of the mutated nucleotide in reference to the indicated codon is detected. Subsequently, it multiplies the codon number minus 1 by 3, in order to acquire a nucleotide referenced scale. It adds to this number the exact location of the mutation within the codon hereby obtaining the exact location of the mutation, referenced on a nucleotide cDNA scale. Ultimately, to acquire the correct location within the DNA, a calculation involving the NCBI CDS and nucleotide location of the mutation in the cDNA takes place, consult appendix C.1 for the algorithm used to perform this calculation. The software calculates the length of each coding sequence and each non-coding sequence in the DNA, according to the NCBI CDS. It then identifies the amount of non-coding nucleotides between the start of the NCBI DNA sequence and the mutation position, then adds this number to the cDNA referenced nucleotide location, resulting in the DNA referenced nucleotide position of the mutation.

Splicing

Solving the problems affiliated with the manner in which HGMD locates splicing site mutations, seems rather straightforward at first: by simply using the splice junctions overview provided by HGMD, every mutation should be located. However due to the discovery of HGMDs splicing site mutations overview poor correspondence to its own Splice Junctions overview, an alternative solution had to be devised. In this case the script uses a matching strategy in which it grabs the given nucleotide sequence from the Splice Junctions overview and matches this to the reference sequence provided by NCBI, thereby locating the location of the mutation more reliably.

Small Deletions / Small Insertions / Small Indels

These three types of variations are located by HGMD in exactly the same manner, therefore they are discussed together. HGMD uses a codon referenced cDNA scale to locate these variations, just like with missense/nonsense mutations. However, in this case, the mutation can involve up to 20 bps and therefore often happens outside the referenced codon, either to the 3' or the 5' side. In this case, the transformation software calculates the given codon location in very much the same way as with the missense/nonsense mutations, by matching the NCBI CDS data with the codon number, resulting in a nucleotide position on a DNA scale for the first codon base. The software then calculates the offset in nucleotides between the referenced codon, and the actual start of the mutation, adding (or subtracting, depending on where the mutation starts in relation to the reference codon, downstream or upstream the DNA strand) this quantity to the nucleotide position of the codon in the DNA, resulting in the DNA referenced nucleotide start position of the mutation.

Gross Deletions / Insertions / Complex rearrangements / Repeat Variations

Since these type of mutations are considered to be imprecise, no data about them needs processing. In this case, the script simply appends an identifier and applies the values to the corresponding cells in the HGDB Variation and Imprecise tables.

Absolutely resolving the data ambiguity would include rereading all the papers used to populate HGMD. However, this approach is impossible to automate due to the highly unstructured nature of scientific papers. Hence another solution is suggested, modifying the HGDB to account for the uncertainty by adding a certainty attribute to the variational table. HGMD presents on their background pages an indication of how to interpret this uncertainty by providing the inclusion criteria for Disease-Associated/Functional Polymorphism's.

4.4 Lessons learned

In this chapter we have shown the primary reason of existence of conceptual modeling techniques. The HGMD is considered an extremely useful source of data about genetic mutations in the field. For being curated by domain experts, it is also considered to be highly reliable. This document shows, that although we do not doubt the quality of the data in itself, a lot remains to be wished for. Some of the problems can be retraced to the relative youth of the genetic domain. The reference to the non-existing intron 4 exemplifies this nicely. The data are difficult to extract, but we do not consider this a real problem. It has been the HGMD decision to choose a more commercial route, and when respected one can obtain a more accessible means of retrieving the desired data. This being said, the apparent lack of a thorough conceptual modeling approach seems to bear its traces on the service.

Every tuple in the HGMD is supposed to represent a genetic variation, known to be associated to disease. This quite rigorous definition becomes endangered in cases where indicated variations 'might' be associated to disease, as indicated in the HGMD by the question mark. Indeed, a variation that is not associated to disease should not be considered a mutation and thus not enter the dataset as is. The CSHG handles these cases by providing the *neutral polymorphism* dimension, for the *Variation* concept.

Another point of improvement is the lack of a proper way of facilitating the various reference sequence in common use by research papers. For illustration, a certain mutation might be located in position 131 in reference sequence X, but correspond to position 125 in reference sequence Y. The HGMD provides its own cDNA sequence, from which it locates the majority of the mutations. However this cDNA sequence is based on an NCBI sequence, and can thus differ from it. For an optimal use of the data provided by HGMD, this means an expert in many cases still needs to evaluate and interpret the data. This is expensive in both time and money. Aligning the HGMD set of mutations to the NCBI reference sequence, which is considered to be the 'golden standard', thus seems a logical step and has been one of the merits of this work.

The results of resolving the problems encountered in HGMD are displayed in figure 4.1. It shows an overview of the total number of variations per gene -of the 15 genes considered in this work- and the number of entries finally loaded in the HGDB. Appendix B.4 provides an overview of how the amount of loaded variations relates to the total number of variations present in the repository for a given gene. In the case of HGMD around 90 % of variations that are considered are loaded, and in some cases -CLRN1, DFNB31, GPR98, PCDH15 and USH1G- a 100 % score has been achieved. The latter is probably the result of a relatively low number of variations that HGMD stores for these genes -in all of these cases less then 25- which reduces the odds of encountering problematic tuples. This is however not always the case, as shown by USH1C, accounting for 16 variations in HGMD but a success rate of only 75 %. Deviation from the average per gene is relatively small with a standard deviation of 8.37 %, but can serve as a guide to identify particularly irregular genes. For instance the NF1

(77.80 %) and USH1C (75.00 %) genes are first candidates to perform further research on. Research has already been performed on the NF1 gene but the necessity to incorporate the lessons learned has been confirmed by this work.

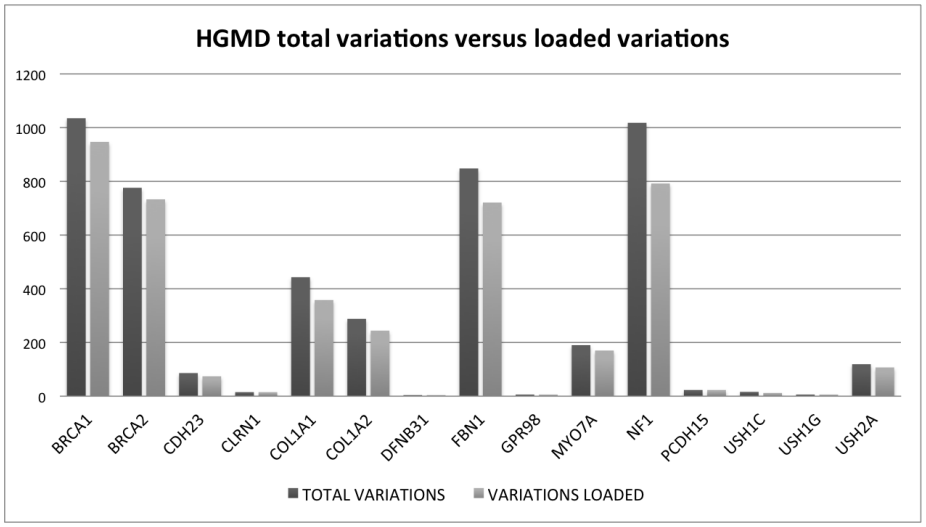

Figure 4.1: Total number of variations per gene, and the number of loaded variations per gene for the HGMD data repository.

Concretely, we suggest two major changes to the HGMD: (i) facilitate a more elaborate way of handling associated phenotype, perhaps link directly to the Online Mendelian Inheritance in Man (OMIM) database [22]. And (ii) add a new column, in which the reference sequence indicated by the source paper is also stored. This will allow for a much easier, and more efficient use of the HGMD data set. Considering data is acquired manually from the papers, adding this element of extracted data seems to be relatively low cost.

The CSHG aims to avoid the earlier mentioned problems, by applying the conceptual modeling approach. It is our strong belief that the only way of accurately representing any data, and perhaps genetic data in particular, can only be done by means of careful analysis of the domain and its peculiarities. Furthermore, following the selected conceptual modeling approach, the conceptual schema itself is open to incorporating new concepts and new discoveries in a domain whose constant evolution is without a doubt. A conceptual schema is ready to adapt to any new knowledge in a way that enables having a continuously updated whole, correct picture of the problem. The logical step of representing the acquired model appears to be in the form of a conceptual schema. When we look at the HGMD we can not help but notice that although very useful, a lot is still to be wished for from an Information Systems point of view. It is exactly this what characterizes the present day problems in the genetics domain; so many data are generated but no coherent, holistic view, of these data exists. The data are scattered around the globe in various databases, many

much like the HGMD, and hidden among them we possess over solutions to so many problems. It is in this haystack, that the needle will allow for exciting new possibilities and huge improvements in health care. We are now facing the choice, to either brute force our way through the haystack, sifting and sifting until we encounter the big prize. Or we structure the haystack, bring order in chaos and obtain an understanding of what exactly composes the mechanical processes that drive life. It is our belief that the only way to do this, is by applying the use of a conceptual modeling approach.

Chapter 5

Solving the dbSNP case

Explaining the various uses of the SNP concept, separating them into their appropriate conceptual context and presenting a conceptual modeling approach will allow solving this ambiguity. This is achieved by answering the main research question; *What makes a genetic variation, classified as a SNP different from those genetic variations not classified as SNP?*. The main contribution is to show how a model driven approach to the genomic domain can aid in the understanding and adequate representation of relevant concepts. Previous solutions applied to the problem space, including ontologies [72] and literal descriptions in natural language, often fail to comprehensively model the concept. As a result of the ambiguity intrinsically associated to natural language, defining the concept exhaustively this way has proven to be a daunting task. In addition to this, relating the concept to its context is equally as important as defining it intrinsically, and conventional methods often fail exactly at satisfying this requirement. The contribution of this work is twofold however, on the one hand fixing the semantics of the concept in a conceptual model while on the second hand enforcing the use of conceptual models in the bioinformatics domain. Bioinformatics has seen a rapid evolution in the past decade, starting with the sequencing of the human genome in 2000 by Venter and Collins [9], and the amount of data being generated is constantly growing. Managing these data has been done by ad-hoc solutions mostly until now. These solutions often do not cope well with changing environments –one that bioinformatics is without a doubt– scale poorly and do not support reusability well. Conceptual modeling, as the cornerstone of the Model Driven Architecture, has been proven to show significant improvements in engineering Information Systems [39]. Conventional solutions require a large amount of manual coding effort each time the domain changes in order to keep implementations in line. The conceptual modeling based solution allows for a process much closer to the problem space by allowing the engineer to create models that can be understood by the domain expert. From these models partial, or in some cases full, implementations can be generated following sets of rules and formal transformation processes, greatly reducing the effort required to maintain implementations consistent with a changing domain.

5.1 SNP definition

The dbSNP data repository [17] is an NCBI [97] driven effort to collect and integrate all available data on genetic polymorphisms. The name suggests emphasis on single nucleotide polymorphisms, but reality is slightly different. As is mentioned on the dbSNP website: "*This collection of polymorphisms includes single-base nucleotide substitutions (also known as single nucleotide polymorphisms or SNPs), small-scale multi-base deletions or insertions (also called deletion insertion polymorphisms or DIPs), and retroposable element insertions and microsatellite repeat variations (also called short tandem repeats or STRs)*". Clearly NCBI is stretching the definition of SNP to fit any type of polymorphism, thereby blurring the concept. Confusing as it may be, we believe dbSNP should actually be called dbVariation, as stated by NCBI itself in the Frequently Asked Questions section of their website. Yue and Moult [98] use a very similar definition of SNP, stating that a SNP can be either neutral or deleterious. This resource appears to be using the literal meaning of the label, simply indicating every polymorphism that influences a single nucleotide as being a SNP, regardless of its effect on phenotype.

A different definition originates from the Department of Biological Sciences, Oakland University, Rochester, MI, USA [99]: "*Single nucleotide polymorphism (SNP) is the simplest form of DNA variation among individuals. These simple changes can be of transition or transversion type and they occur throughout the genome at a frequency of about one in 1,000 bp. They may be responsible for the diversity among individuals, genome evolution, the most common familial traits such as curly hair, interindividual differences in drug response, and complex and common diseases such as diabetes, obesity, hypertension, and psychiatric disorders. SNPs may change the encoded amino acids (non-synonymous) or can be silent (synonymous) or simply occur in the noncoding regions. They may influence promoter activity (gene expression), messenger RNA (mRNA) conformation (stability), and subcellular localization of mRNAs and/or proteins and hence may produce disease*". It resembles the conventional definition loosely, as it mentions a frequency of occurrence in an individual (*about one in 1,000 bp*) and pathogenicity (*and complex and common diseases such as*). Depending on interpretation, one could state that also the size dimension is mentioned implicitly, considering the definition handles single nucleotide polymorphism.

5.1.1 SNP characteristics

During the investigation of various definitions of the SNP concept existing in todays genomic domain, the determinant characteristics, or dimensions, have been identified. They form the conceptual spectrum in which the SNP concept is positioned. Each concrete definition represents an instance of these abstract dimensions, thereby fixing a position within the gamut. It is expected that by incorporating these dimensions into a database, instead of fixing the concept on one definition, flexibility is acquired and data can be queried dynamically depending on the desired SNP interpretation. The commonly accepted defini-

tion for SNP is explicitly based on 2 dimensions: occurrence in the population and genotypic size. According to [100] a SNP is a single base change in a DNA sequence. It is considered that the least frequent allele should have a frequency of 1% or greater. The first dimension, occurrence in population, hides a third and fourth characteristic. As a direct result of natural selection, for a polymorphism to happen more than a given percentage in the population, it needs to have either a neutral or a positive phenotypic effect. A negative, or deleterious effect is impossible to happen in a substantial portion of the population as it eliminates itself. Therefore, SNPs considered according to the conventional definition have no direct relation to disease. For this very same reason, SNPs as defined in the conventional sense are relatively common among each individual. On average, ∼8.33 SNPs happen every 10kb [94].

Various literature sources use various definitions of the SNP concept. dbSNP forms one extremity of this spectrum by defining the concept very loosely. The conventional definition then describes the other side of the spectrum by being more rigorous. Following the various definitions leads to the identification of four dimensions on which SNPs are commonly defined: occurrence in population, occurrence in sequence, pathogenicity and size. Every dimension will now be clarified briefly.

Occurrence in population

This dimension entails the aspect of occurrence of a SNP in a given population. Examples of different populations include Asian, African or European. A higher population occurrence than a fixed low percentage, suggests a polymorphism with neutral effect on phenotype. This barrier is not defined rigorously, but commonly accepted percentages vary from 0.5% to 1%.

Occurrence in sequence

Due to natural selection, polymorphisms with deleterious or negative effects are usually filtered out of a population. This process conserves neutral polymorphisms without observable effect and polymorphisms with positive effect on phenotype. As a result of this mechanism, these type of polymorphisms stack up in the genome of an individual. According to conventional definition, in the human genome every ∼1200 base pairs a SNP is expected as stated by [94]. This dimension however, describes a characteristic of the SNP concept in general. Indeed, a specific SNP does not happen once every ∼1200 base pairs, rather a different SNP is expected to be occurring with this frequency. It is therefore a characteristic of a genome; stating the amount of SNPs happening in it, rather than being a defining criterium whether a specific polymorphism is considered to be a SNP or not.

Pathogenicity

The association of a SNP to hereditary disease is captured in this dimension. The general belief is that SNPs are not causatively associated to disease. Rather, due to a phenomenon called Linkage Disequilibrium [101] and by using SNPs as markers, susceptibility to hereditary disease can be detected without having to sequence an entire individuals genome. In short, the SNP itself has no relation to a pathogenic effect. SNPs used as marker, however, predict a different mutation, happening at another position. This second, predicted, mutation usually does have a direct pathogenic effect.

Size

Clearly, a single nucleotide polymorphism should involve only one nucleotide, hence the name. However, considering the definition in which the SNP label is used to identify non deleterious polymorphisms this description might be stretched. Considering that DNA is read in triplets (combinations of three nucleotides), the polymorphisms that are not a multiple of three often result in frame-shift, changing the transcribed mRNA and thus the resulting protein. Frame-shift often leads to an entirely different protein, and is thus very likely to result in negative phenotype. A non-deleterious polymorphism, in theory, can thus be any variation in which the affected nucleotides are a multiple of three. However, every changed triplet, if non-synonymous, also changes an amino-acid eventually leading to an increased chance of significantly changing the resulting protein. Thus, although in theory a non-malicious polymorphisms might include more then one nucleotide, it is most likely to involve a single nucleotide change.

5.1.2 Uses of the SNP concept

For a proper understanding of the genomic concept known as SNP, and to define it properly in terms of a conceptual model, it is necessary to know what purpose the term serves. Not only needs to be investigated what the domain exactly considers to be a SNP, but also how this concept is used in the domain. This exact understanding of the concept, will allow for an adequate application of the conceptual modeling approach. After applying the conceptual modeling perspective to the problem of properly specifying SNPs, the concepts behind them will less ambiguous and clearer. By more clear is meant that specific uses in particular contexts are labeled with an adequate term, while the SNP notion emerges clearly from the conceptual schema. Three distinct uses of the SNP concept have been identified. (i) dbSNP, (ii) distinguishing harmless polymorphisms from harmful variations and (iii) as a genetic marker.

dbSNP uses the term to cover a wide variety of polymorphisms known to happen in the human genome. It thereby stretches the conventional definition of the term, and should thus actually be called dbVariation, as it mentions itself. Another application of the term is identified when assuming SNP to be defined according to measures on the dimensions shown in 5.1.

When sequencing a human genome, various polymorphisms are usually encountered. Every ∼1200 base pairs, a mismatch happens between the sequence used as reference and the sample at hand. Most of these polymorphisms happen without negative effect on the phenotype, as has been explained earlier. Considering that DNA is read in triplets, or combinations of three nucleotides, the polymorphisms that are not a multiple of three often result in frame-shift, rendering the transcribed mRNA sequence senseless. This taken into account, the most probable size in which a polymorphism is expected to not alter the genetic sequence is one nucleotide, or a multiple of three. In the absence of negative effect, they are likely to happen in more than 0.5% of the population. Hidden among these relatively innocent polymorphism, lurks the one that is associated to the development of disease. More than one polymorphism with negative effect on phenotype, happening in any individual is very unlikely. Discriminating between polymorphisms with negative effects, and polymorphisms without is thus very important. The SNP concept, shaped in the way as is done in 5.1 serves this purpose elegantly.

Table 5.1: The SNP concept defined to discriminate polymorphisms with neutral effect on phenotype from polymorphisms with negative effect on phenotype.

Property	Value
Occurrence in population	>0.5 %
Pathogenicity	None
Size	1 nucleotide

SNPs are also commonly used as genetic markers. Due to relative high pricing of DNA sequencing, ways of using the available sequencing capacity efficiently had to be come up with. One approach to this cost-saving strategy, is making use of linkage disequilibrium. LD describes the nonrandom correlation that exists between a marker and the presence of polymorphisms at a locus [101]. Following LD it is possible to asses the probability of a polymorphism happening in a subjects genetic code, by knowing the state of a marker position. Vignal [100] describes the use of SNPs as molecular markers in animal genetics. The use of SNPs as genetic markers can thus be considered an imperfect solution to high costs associated to DNA sequencing.

5.2 SNP incorporation in the CSHG

Incorporating SNPs into the Conceptual Schema of the Human Genome is essential. Figure 5.1 demonstrates a representation of the extension. It provides means of discriminating various types of polymorphisms. The most important being the separation between harmful and harmless genotypic variations. In the schema, harmful variations are considered variations with mutant effect, while SNPs are considered to be variations of the type neutral polymorphism. It is for this reason that data on SNPs need to be stored in such a way, that it facilitates the earlier mentioned dimensions on which the concept is defined.

This will allow for dynamic creation of queries, extracting the correct tuples depending on which definition at runtime is chosen. To solve the ambiguity associated to the SNP concept, an often considered sufficient solution involves providing a concise, literal description of the term. Due to the various uses of the concept, a static definition is believed to be insufficient. Also, this approach fails to relate the concept to its context, an aspect considered very important in the complex reality of genomics. Rather, coping with the variability of the term seems appropriate and is facilitated by the extension to the conceptual model of the human genome. In practice, this means that the earlier identified dimensions of the definition needs to be represented in the schema, allowing for dynamic capture of the concept.

An instance of the concept can then be instantiated with different values, thereby allowing for the different uses and definitions that have been identified. Thus, by separating the instance of the concept from the structural definition, flexibility is obtained. First of all, the *SNP* concept is considered to be a specification of the *Indel* concept, which in turn is a specification of the *Variation* concept. It deserves mentioning this implementation does not rule out SNPs of size greater then 1, leaving room for flexibility on the size dimension. The occurrence in population (*populationOccurrence*) is regarded as a SNP specific attribute. Occurrences in populations might differ among various populations, it is therefore crucial to also store the investigated population that resulted in the discovery of the SNP at hand. Other polymorphisms, either mutant ones or with unknown effect, are expected to have different effects among populations as well. It is for this reason, the *Population* concept is related to both the *SNP* and Variation concepts. The pathogenicity dimension relates the neutral polymorphism concept in the conceptual model to the SNP concept, securing that a SNP is never associated to disease. According to the schema, a polymorphism associated to negative phenotypic effects is considered to be mutant and is thus captured already in the form of a *Variation*, the *SNP* super-type.

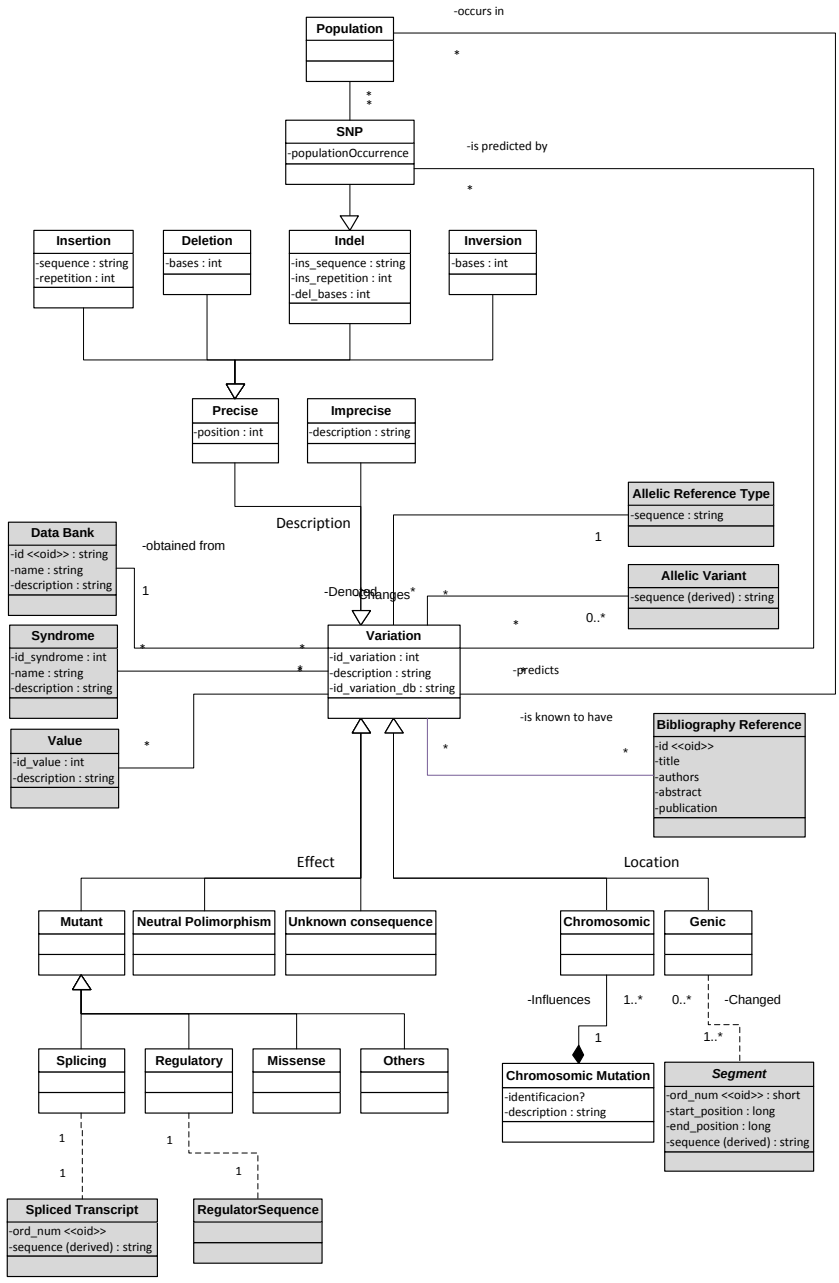

Figure 5.1: Conceptual model of the human genome including the *SNP* concept.

As one might notice, the dimension occurrence in sequence is not incorporated in the schema. The occurrence in sequence represents not a characteristic of a single SNP, rather a characteristic of a specific genome. Indeed, a specific SNP does not occur every 1200 base pairs, but rather every 1200 a SNP is found. It is therefore not associated to the SNP concept and thus left out of the schema. Also, it is considered unnecessary to be stored, for the reason that it can be derived for every genome from the database itself, simply combining the data on the total amount of nucleotides in an individuals genome, and the amount of found SNPs.

To incorporate the marker functionality of the SNP concept, a new relation would need to be made between *SNP* and *Variation*. This relation would consist of a probability score, indicating the probability to which the SNP at hand predicts the variation. This use of the concept is considered to be a result of the immatureness of the domain. Using SNPs as markers merely satisfies a cost-efficiency strategy, and is expected to be rendered useless in the near future by advances in DNA sequencing technology. Although a correlation exists between neutral SNP occurrence and malicious polymorphisms, there appears to be no causal relation. It is for this reason, that has been decided not to incorporate this use of the SNP concept in the conceptual model of the human genome.

5.3 dbSNP description

The Single Nucleotide Polymorphism database (dbSNP) [17] is a public-domain archive for a broad collection of simple genetic polymorphisms. This collection of polymorphisms includes single-base nucleotide substitutions (also known as single nucleotide polymorphisms or SNPs), small-scale multi-base deletions or insertions (also called deletion insertion polymorphisms or DIPs), and retroposable element insertions and microsatellite repeat variations (also called short tandem repeats or STR's). Each dbSNP entry includes the position, the sequence context of the polymorphism (i.e., the surrounding sequence), the occurrence frequency of the polymorphism (by population or individual), and the experimental method(s), protocols, and conditions used to assay the variation.

As established earlier, the dbSNP database contains much more then simply SNP's, making the extraction process described below significantly more difficult. dbSNP stores data for 100 species, with a total of 87 536 603 entries (Rs#'s), of which 29 813 700 have been validated. Currently (20-07-2011, dbSNP build 132) the dbSNP repository contains 30 442 771 entries -of which 19 727 605 have been validated- for the Human genome.

5.4 Encountered problems

The problems encountered when loading dbSNP into HGDB can roughly be divided in two categories: data quantity and semantic issues. The first refers to the intrinsic difficulties of processing the large files, 22Gb compressed using the

Gzip algorithm and approximately 120Gb when decompressed into XML files, dbSNP uses on a regular workstation with 8Gb RAM. The latter refers to the problems encountered associated to conceptual ambiguity or collision with the conceptual schema. Both will now be discussed separately.

5.4.1 Data quantity problems

The dbSNP data repository provides access to its files through an open FTP server[1], where the files are organized per organism and file type. The files can be downloaded as a SQL database dump, XML or ASN. The latter being a more space efficient representation of XML. For the purpose of this work the XML files are most appropriate. The already present solution for transforming data from outside sources to the CSHG format, which will be discussed in detail in chapter 7, is a Java application which makes it very suitable for integrating already existing XML parser libraries.

The XML files are organized as one per chromosome, including the Mitochondrial chromosome, and a few summary files. All together they account for 22Gb in a compressed state, and approximately 120Gb when uncompressed. The algorithm used to compress them is Gzip. Processing these file sizes pose obvious performance issues when done on a regular workstation PC. The relevant specifications of the PC from which was worked are described in section 1.4.

The file size for the individual files ranges from 56Mb compressed (591Mb uncompressed) in the case of the relatively small chromosome Y, to 1.7Gb compressed (18Gb uncompressed) for the largest chromosome on the human genome, chromosome 1. Thus simply opening one of these with a regular text editor - clearly a logical step in analysis of the files- poses serious difficulties, often 'freezing' the system. Fortunately, Ubuntu provides powerful tools to handle this type of files directly from the command line. Each file has a header section with general data about dbSNP version, organism, taxonomy id etcetera. The absolute majority of the file is taken up by the dbSNP entries, each entered as a so-called *Rs*-item, identified by the *<Rs>* opening tag and *</Rs>* closing tag. In between, all data for that that entry is stored, ranging from the various submitted samples that lead to the establishment of that particular *Rs* entry to the validated details like position, and observed alleles. A sample section of a typical *Rs* entry can be consulted in appendix A.3. For more details on the exact structure of the XML file, please consult the XSD file as provided by dbSNP on their FTP server[2].

In general the elements that are of interest to the HGDB are contained within these earlier mentioned *Rs* entries. The *Sequence* element stores data about the observed alleles in the sequence (the *Observed* element), and provides flanking sequences -both upstream *Seq5* and downstream *Seq3*-. Within each *Rs* entry there exist multiple *Ss* entries, that represent submitted samples which

[1] ftp://ftp.ncbi.nih.gov/snp/
[2] ftp://ftp.ncbi.nih.gov/snp/specs/docsum_3.3.xsd

lead to the identification of this *Rs* entry. Each *Rs* can then contain various *Assembly* elements, each representing an alignment to a specific build of dbSNP and a specific build of the genome. Within these *Assembly* elements the relevant entry of interest is the so called *Component* element, which tells us the reference sequence to which the alignment was made, this is indicated by the *accession* attribute along with the *gi* attribute. These latter two attributes correspond to the refSeq attributes with the same name.

The *Maploc* element has various attributes associated that provide information on the alignment itself and the associated quality: (i) *locType*, which describes whether the alignment is exact or not, (ii) *alnQuality*, which gives an align quality measure on a scale from 0 to 1, (iii) *orient*, which tells us on which strand the SNP has been observed, (iv) *leftFlankNeighborPos*, which tells us the starting position of the SNP. Table 5.2 provides an overview of how the various XML elements and their attributes map to the CSHG.

Table 5.2: dbSNP relevant XML elements.

XML element	XML attribute	CSHG Class:attribute
Rs		class:SNP
Rs	rsId	Variation:id_variation_db
Sequence		Insertion:sequence
Sequence		Deletion:bases
Sequence		Indel:ins_sequence
Sequence		Inversion:bases
Maploc	orient	Allele:strand
Maploc	leftFlankNeighborPos	Precise:position

The solution to the large file problems meant parsing the XML files in such a manner that limited computing resources would not present major obstacles. As a single uncompressed dbSNP file surpasses the available RAM memory the obvious solution of using a DOM based XML parsed was ruled out. Some investigation took place and eventually the StaX[3] based XML parser solution was chosen. StAX is a standard XML processing API that allows you to stream XML data from and to your application in an event pull manner, rather then in an event push manner used by conventional SaX[4] parsers. The implementation details, interesting as they might be, surpass the scope of this work and shall for that reason not be discussed in detail.

Another optimization to improve performance is related to the semantics conveyed by the individual names of the dbSNP files. Each filename follows the following convention; a string, "*ds_ch*" followed by the chromosome identifier, ending with the file extension. The latter is "*.xml*" when uncompressed and "*.xml.gz*" when in compressed state. This identifier corresponds to the chromosome number, 1 to 22, and in cases of Mitochondrial or the male/female sex chromosome "*MT*", "*X*", and "*Y*" respectively. So for example, the SNPs for chromosome 17 will be stored in a file by the name "*ds_ch17.xml.gz*". This

[3] http://stax.codehaus.org/
[4] http://www.saxproject.org/

naming convention allows us to optimize the SNP extraction process. As the current approach is gene centered, SNPs will have to be extracted, transformed and loaded gene by gene. So once we establish the chromosome to which the gene belongs, we can use that information to efficiently select the adequate file and avoid having to process the full 120Gb. Processing the full 120Gb requires around 70 minutes, while a single chromosome takes between 5 and 10 minutes.

5.4.2 Problems related to semantics

The semantic issues as identified in dbSNP deal with problems that arise when matching the data entries from dbSNP to data entries in HGDB. Roughly we identified three problems: (i) processing the 'hidden' variations, (ii) the separation of 'junk' data from valid data and (iii) correcting the observed alleles when differing from reference allele strand. Each of these problems will now be discussed separately in the following sections. Please consult appendix B for an overview of the full coverage data that GDL obtains.

Processing the hidden variations

It was soon found that one single *Rs* entry could actually convey more then one SNP. The *Observed* XML element provides a character String in which the observed alleles can be stored, separated by slash characters ("/"). Clearly, if more then two alleles have been observed, this particular *Rs* actually corresponds to multiple SNPs as conceptualized in the CSHG. In order to make this rather abstract statement more clear, let's have a look at an example. The *Rs* with id 245 -present in the Human genome on chromosome 11 at absolute position 276 447 63- has three known alleles, as taken literally from the *Observed* field: -/A/T. Interpreting this string brings us to the conclusion that in some samples no nucleotide was present at that position, while in others either an Adenine or a Thymine nucleotide was encountered at that specific position. These different 'instances' of the SNP need to be separated in order for them to be properly introduced in the HGDB database.

Doing so requires a matching strategy in which the reference sequence already in use by HGDB needs to be queried for the nucleotide present at that position, in order to determine which of the presented alleles is actually part of the reference and which is not. Effectively determining which of the multiple options actually is a *Variation*. The following pseudo code will take a String as reference sequence and a Map with key-value pairs (in this case as key an observed allele, and the value a Boolean identifying whether that observed allele is the variation (TRUE) or reference (FALSE)). The algorithm delivers the Map in which the single reference allele nucleotide is marked with TRUE, and for all the others different instances of the SNP class need to be created. Please consult appendix C.2.

```
String reference_sequence;
Map<String, boolean> observed_alleles;
int position;
```

```
for (int i = o; i < observed_alleles.size(); i++) {
        String observed_allele = observed_alleles.get(i);
        String reference_allele = reference_sequence.position;

        if (observed_allele EQUALS reference_allele)) {
                observed_alleles.put(observed_allele, false);
        }
}
```

Table 5.3: dbSNP total entries corrected for hidden variations.

Gene	Total SNPs	Corrected	Excluding/including
BRCA1	3091	1662	53.77 %
BRCA2	2712	998	36.80 %
CDH23	4954	4754	95.96 %
CLRN1	547	565	103.29 %
COL1A1	748	809	108.16 %
COL1A2	787	843	107.12 %
DFNB31	1289	1222	94.80 %
FBN1	2320	2399	103.41 %
GPR98	5409	5254	97.13 %
MYO7A	1205	1094	90.79 %
NF1	2558	2499	97.69 %
PCDH15	11 830	12 186	103.01 %
USH1C	647	661	102.16 %
USH1G	121	111	91.74 %
USH2A	7925	7965	100.50 %
TOTAL	**46 143**	**43 022**	**93.24 %**

Separating the junk from valid data

Identifying valid entries in the large amount of data as provided by dbSNP is essential to properly populate HGDB. The first step here is to determine what exactly can be considered a valid entry. It has been decided that for the purpose of HGDB only data entries that are known to be valid may be introduced in the data set. This automatically excludes all dbSNP entries that are not classified as "*locType=exact*", and where "*alnQuality<1.0*". Also the dbSNP data set is known to contain poorly defined entries -from a formal IT perspective that is- known as Satellite Tandem Repeats, or STR's, for which currently we provide no automated solution to interpret them. In order to have a coherent set of valid SNPs these latter need to be sifted out, which can only be done in a 'rough' manner. The *Observed* element of the XML structure conveys the various alleles as observed by the various submitted entries (*Ss* entries), separated by slash characters ("/"). Once the observed field contains a parenthesis -either "(" or ")"- it was found to describe an STR. Unfortunately the dbSNP data files

provide no native way of conveying mutation type. By this latter is meant, for instance, a '*SNP type*' element that would properly identify the nature of the variation.

Another example includes the determination whether a SNP inserts nucleotides, deletes nucleotides or both (Indel). In order to determine this distinction, the *Observed* fields needs to be scanned for hyphens ("-"), which in some sense represent a non-occurring nucleotide. When present at the beginning of the string one can safely deduce this particular SNP has been observed to insert one, or more nucleotides at the given position. In the case where the hyphen appears at the end it is safe to assume this concerns a deletion of a certain sequence of nucleotides. For the lack of those proper semantics, this particular piece of information needs to be deduced from the 'Observed' alleles string. This rather rigorous process of separating junk from valid data has led to the conclusion that around 63 % of the dbSNP entries per gene are actually considered valid. Figure 5.4 provides an overview per gene of how many SNPs are considered valid.

Table 5.4: dbSNP total entries as compared to valid entries per gene.

Gene	Total SNPs	Valid SNPs	Total/valids
BRCA1	3091	1140	36.88 %
BRCA2	2712	684	25.22 %
CDH23	4954	3273	66.07 %
CLRN1	547	370	67.64 %
COL1A1	748	575	76.87 %
COL1A2	787	595	75.60 %
DFNB31	1289	827	64.16 %
FBN1	2320	1638	70.60 %
GPR98	5409	3570	66.00 %
MYO7A	1205	734	60.91 %
NF1	2558	1747	68.30 %
PCDH15	11 830	8208	69.38 %
USH1C	647	455	70.32 %
USH1G	121	74	61.16 %
USH2A	7925	5445	68.71 %
TOTAL	**46 143**	**29 335**	**63.57 %**

Correcting the observed alleles

dbSNP provides an attribute -the *orient* attribute within the *Maploc* element- that identifies the DNA for which the alleles as provided have been observed. The value for this attribute needs to be taken into account when comparing with the reference sequence. Obviously, when comparing an observed allele classified with a 'reverse' orientation to the reference sequence classified as 'forward' has some implications. In short, the two strand DNA helix shape needs to be kept

in mind. When one speaks of orientation, it is actually a reference to the strand; either going from 3' towards 5' or the other way around. Consult section 1.1.1 for a more detailed overview on DNA strands and their directions. The strand that conveys a coding sequence -which might eventually lead to the successful translation of a protein- is often referred to as the 'sense' strand, denoted with a minus '-', while the other strand as 'anti-sense', denoted with a plus '+'. As nucleotides bond to each other in a complementary manner -Adenine to Thymine and Cytosine to Guanine- we now understand that when the reference allele and the observed allele orientations do not match we need to perform a transformation. This transformation takes the observed allele, for the sake of simplicity in this case a single nucleotide, as input and replaces it with the complementary equivalent, matching it to the orientation of the reference allele. In more complex cases, in which more then a single nucleotide are considered, one will also need to correct for the change of direction. This latter is done by a simple reverse operation that reverses the order of the observed alleles' sequence. Appendix C.3 provides the algorithm that was used in GDL to handle this exact issue.

5.5 Lessons learned

This chapter has contributed a conceptual modeling approach applied to the ambiguity associated to the SNP concept. The existence of ambiguity surrounding the term has been explained, and various existing solutions to it have been discussed. It has been clarified how and why a conceptual modeling approach is structurally different from existing solutions, like ontologies, and that only conceptual modeling techniques allow for providing a precise definition. By researching the uses and existing definitions of the term, a conceptualization of the domain was achieved. A clear understanding of the concept has thus been established, and modeled accordingly as an extension to the already existing conceptual model of the human genome. The dimensions on which the SNP concept is commonly defined have been identified and formalized in the conceptual model, allowing for dynamically querying the resulting data repository according for each type of definition. Coming back to the main research question: *What makes a genetic variation, classified as a SNP different from genetic variations, not classified as SNP?*, the answer is clear. A genetic variation identified as a SNP is a single nucleotide polymorphism, not causatively associated to disease and (therefore) happening in more then 0.5% of a population. At the same time, SNPs are common due to their characteristics and for this reason happen about once every 1200 base pairs. Therefore, the dbSNP data repository should not be considered a collection of SNPs, but rather one of genetic polymorphisms. The authors of this paper stress the importance of a name change to dbVariation, in order to avoid future confusion. This paper further demonstrates the capabilities of a conceptual modeling approach to solving ambiguity in a domain. Especially the Bioinformatics domain, where large amounts of data have to be properly managed, and where, unfortunately, too

often sound and well-known Information Systems concepts required to perform an effective and efficient data exploitation, are not correctly applied.

The results of resolving the problems encountered in dbSNP are displayed in figure 5.2. It shows an overview of the total number of variations per gene -of the 15 genes considered in this work- and the number of entries finally loaded in the HGDB. Appendix B.4 provides an overview of how the amount of loaded variations relates to the total number of variations present in the repository for a given gene. In the case of dbSNP an average success rate of 31.32 % was achieved, significantly lower then HGMD (90.32 %), but this is due to the relative high amount of variations present in dbSNP in general -dbSNP accounts for approximately 77 % of total loaded variations- but also to the high degree of ambiguity associated to the *Observed* field as discussed earlier. The deviation from the average per gene is also highest in dbSNP with a standard deviation of 13.47 %

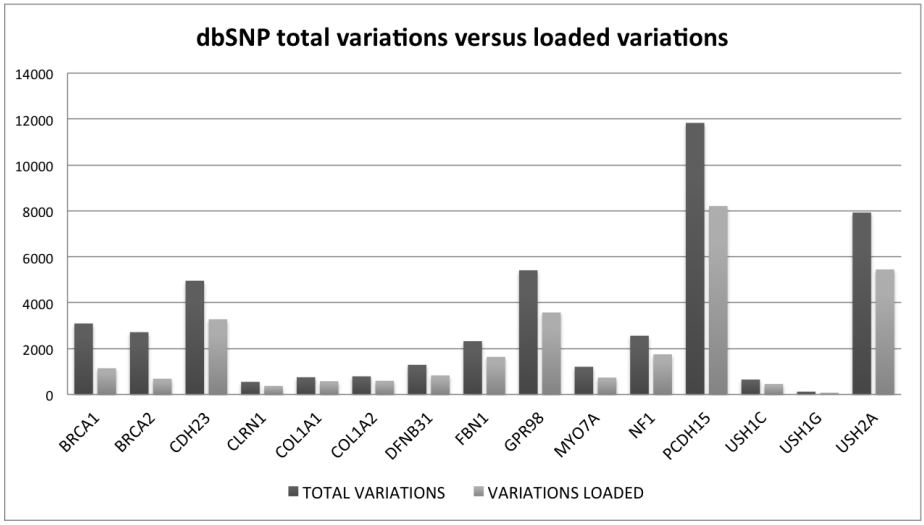

Figure 5.2: Total number of variations per gene, and the number of loaded variations per gene for the dbSNP data repository.

Early signs indicate that the dbSNP web interface provides much more data then currently loaded from 'raw' files as provided over FTP. Unfortunately there was not enough time to properly investigate this claim, but a very plausible explanation includes the number of variations that were eliminated during the loading process on account of being considered invalid. It is possible that the web interface shows each and every variation as submitted to dbSNP, without taking into account to which reference sequence they have been aligned, they quality of the alignment and whether the alignment was exact. This finding further nourishes the debate about quantity versus quality in genomic databases. A discussion in which this work attempts at all costs to remain on the 'quality' side, perhaps discarding some of the quantity originally worked with. It is however

the philosophy of this project to only store data that is valid and consistent, ultimately leading to the reliable data repository that is so much needed in the domain right now.

As a suggestion for improvement to the dbSNP database -beside the obvious name change from dbSNP to dbVariation to account for the conceptual ambiguity earlier mentioned-, the authors of this work suggest the addition of an extra element in the dbSNP XML files. This element should identify the type of each SNP; either being an insertion, deletion, indel or perhaps a more elaborate classification. An adequate, and equally obvious name for this element would be *Type*, and could be located within the *Component* element already present. This latter allows for expressiveness to cope with the same SNP having perhaps different observed behavior in different reference sequences.

Chapter 6

Solving the BIC case

The Breast Cancer Information Core (BIC)[1] [30] is an open access, on-line mutation database for breast cancer susceptibility genes. In addition to creating a catalogue of all mutations and polymorphisms in breast cancer susceptibility genes, a principle aim of the BIC is to facilitate the detection and characterization of these genes by providing technical support in the form of mutation detection protocols, primer sequences, and reagent access. Breast Cancer is the most common malignancy among women worldwide. Of these cancers, approximately 5-10% is attributable to inheritance of a mutation in one or more highly penetrant autosomal dominant susceptibility genes. Both *BRCA1* and *BRCA2* have been identified as exactly this type of genes. It was evident that scientific and clinical interest in *BRCA1* and *BRCA2* would lead to an effort to screen women at high risk of developing breast cancer for mutations in these genes. Building a knowledge base about cancer susceptibility genes is one of the steps towards realizing this goal. The BIC represents an effort aimed at exactly this. As of April 2000 the repository contained 3416 entries describing genetic variations in *BRCA1*, and 2292 entries for *BRCA2*.

Unfortunately the BIC data repository seems to have been inactive the last decade or so. Since the publication by C. Szabo in 2000 [30] it seems no new publications have been made. The data remains available but it is unclear when it was updated, or whether new data keeps being added. However, browsing through the data reveals that the latest date in which entries have been added is the 25th of November, 2004. The data set in itself represents a collection of curated tuples and is thus regarded by the community as reliable. An interesting, but as of yet unanswered question is whether the data set has been super seeded by other data sources like the Human Gene Mutation Database, or various LSDB's. The data set as is now has been found to accumulate 12 025 entries for the *BRCA1* gene, and 11 331 for the *BRCA2* gene. Of these entries, 1446 and 1692 have been recognized as valid and unique for *BRCA1* and *BRCA2* respectively.

[1] http://research.nhgri.nih.gov/bic/

6.1 BIC description

The BIC dataset can be viewed from the database website through a fee less membership account, the overall layout of the BIC web site is shown in figure 6.1. Browsing the data can be done per gene, and to even finer granularity per exon. The *BRCA1* gene is divided into 25 exons, in which exon 11 is subdivided into 11A until 11D, and no exon 1 and 4 exist (the numbering starts with exon 2). The absence of an exon 4 has been discussed in detail in chapter 4, section 4.3.1. The data can be downloaded as a Tab Separated File (TSV) either per gene or per chromosome. The data files consist of 32 columns and store a variety of data about each variation. The relevant ones are as follows: (i) *Accession Number*, identifying the variation internally, (ii) *NT*, locating the variation nucleotide precise within the cDNA, (ii) *Base Change*, describing which change was observed. Other columns of interest include *Reference*, which in some cases provides a literature reference for the variation, and *Clinically Important* which tells us whether the observed change either led to a phenotype change or not. The latter is strongly related to the SNP concept earlier discussed thoroughly in chapter 5. for a full overview of the BIC data format, consult appendix A.4.

Positioning in BIC is done in two ways, the most common one is done by aligning the variation with a transcript while in some cases the variation is located against genomic DNA. All variations have been aligned with the same -although different per gene, obviously- transcript reference sequences. The reference sequences used by BIC are as follows: *BRCA1* GenBank U14680[2] and *BRCA2* GenBank U43746[3]. Some variations have also been positioned according to their genomic DNA location, but this only occurs for the *BRCA1* gene, and *BRCA1* GenBank L78833[4] is used.

6.2 Encountered problems

The BIC data set turns out to be quite comprehensive and accessible when compared to the previous two cases, HGMD and dbSNP. The data are readily available, although one needs to create a membership account free of charge, and can be downloaded as a single file. Interpreting the data proved to be no problem because of the [30] publication which comprehensively clarifies how the data is structured, as well as assigning semantical meaning to each data property. The Breast Cancer Information Core website is quite sparse in providing information on how to interpret the data, and it does not provide a list of publications associated to the BIC. Searching the well known channels for publications led to the retrieval of the above document, but it is unclear whether this is the last sign of life of the BIC or whether more recent publications exist.

A more interesting issue encountered in the BIC repository, and one that is prone to reoccur, is the positioning problem. As stated above, BIC variations are

[2] http://www.ncbi.nlm.nih.gov/nuccore/U14680
[3] http://www.ncbi.nlm.nih.gov/nuccore/U43746
[4] http://www.ncbi.nlm.nih.gov/nuccore/L78833

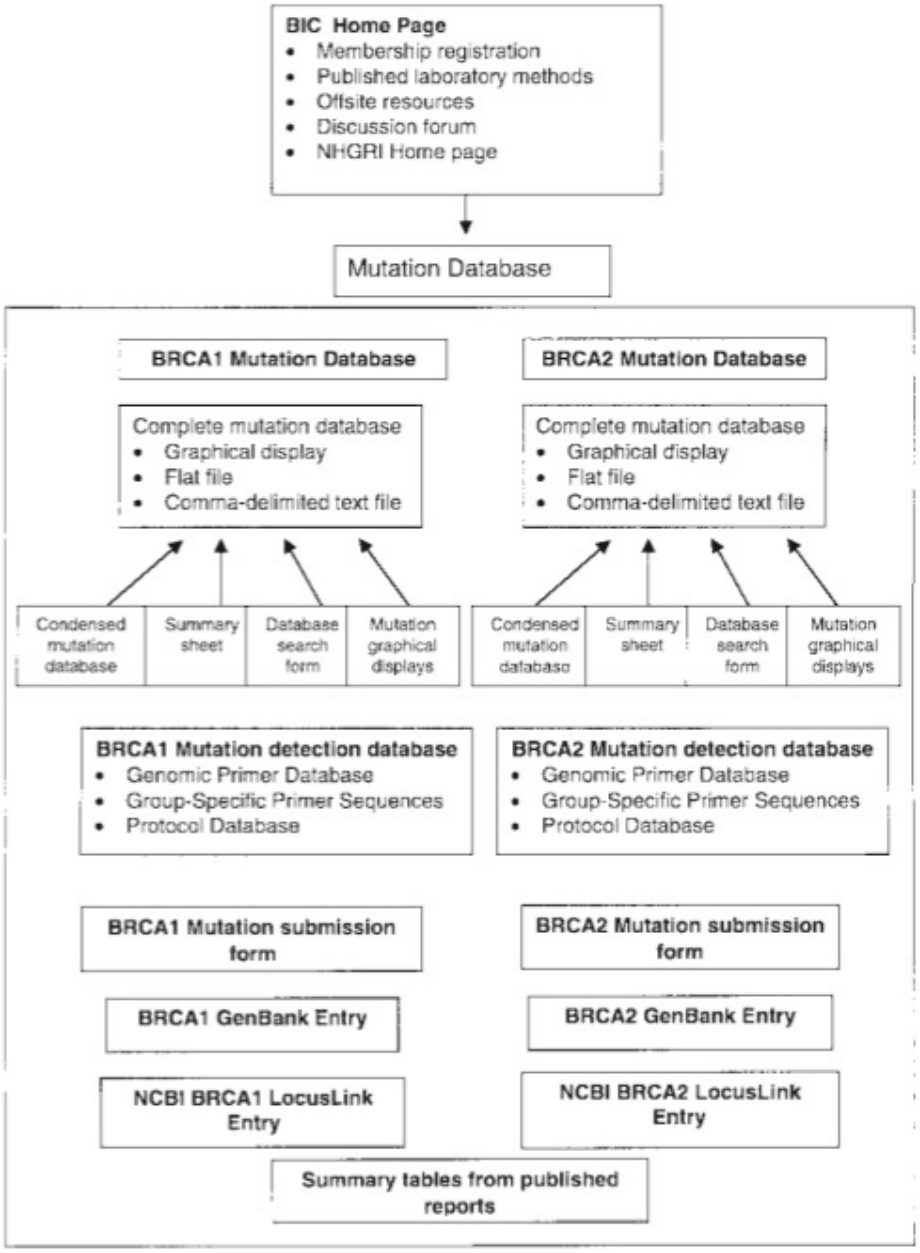

Figure 6.1: The Breast Cancer Information Core (BIC) website features. Overview and map of the BIC website. Details of the links contained in the Mutation Database section are shown.

positioned along well established reference sequences. These sequences however, are quite old and for this reason outdated. The *BRCA1* GenBank U14680 came to be as a result of combining the efforts by Miki [92] and Futreal [102] in 1994, the latest update as recorded by the NCBI dates back to the 10th of June 2002. The *BRCA2* GenBank U43746 as first presented by Teng [103] in 1996 has not seen any modifications since then. It is not uncommon for genomic data repositories to refer to outdated reference sequences; sometimes the effort of realigning 'old' data to new reference sequences is to costly or simply impossible. The challenge in the case of BIC consists of 'saving' these data by realigning them with the currently used reference sequence, see section 1.4 for more details on which reference sequences have been used in the creation of HGDB. This work reports on the process of manually aligning the sequences and deducing the relevant corrections values from it. Due to time constraints no automated solution can be presented but creating this would offer interesting future research lines. In the following section the procedure that was followed to retrieve the BIC datasets' positions will be elaborated on.

6.2.1 Aligning reference sequences

Aligning genetic sequences has been subject of research for a long time. The de-facto standard has become an algorithm known as BLAST as presented in 1990 by [104], which stands for Basic Local Alignment Search Tool. Nowadays, various versions of BLAST exist, some refined for a specific purpose while other provide performance enhancements. Another popular algorithm is the BLAT [105], the BLAST-Like Alignment Tool. All of these tools provide an algorithm capable of comparing primary biological sequence information, such as the amino acid sequences of different proteins or the nucleotides of DNA sequences. A BLAST search allows a researcher to compare a sequence against a library or database of sequences , and identify library sequences that resemble the query sequence above a certain threshold.

Because of the relatively uncomplicated nature of the required alignment, the conventional BLAST algorithm as provided by the NCBI was deemed sufficient. The NCBI implementation of BLAST[5]. For the purpose of this work not every observed change between the sequences is relevant. Structural differences that do not reflect a differing number of nucleotides between sequences can safely be ignored. Indeed, these latter differences do not change the positioning reference and are thus in the context of this particular application irrelevant. The differences that do represent a change between the number of nucleotides of both sequences, and thereby warp the positions accordingly need to be noted and taken into account when correcting for them. An example of the BLAST result file is shown in figure 6.2. The result of analyzing the BLAST result file lead to the identification of a series of structural differences, both in the *BRCA1* and *BRCA2* genes. Currently performed manually, there is clear advantage in devising an automated approach. These structural changes are then taken into

[5] http://blast.ncbi.nlm.nih.gov/Blast.cgi

account when creating a new, 'virtual', set of joins that represent the exon borders. These exon borders are required to calculate the absolute positions -genomic; referenced on a scale where both exons and introns are taken into account- of variations that are positioned in a relative style. The latter refers to positioning in which introns have been left out of the sequence, for a more detailed explanation refer to section 4.3.2 and appendix C.1.

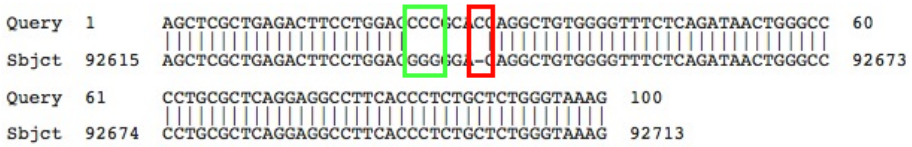

Figure 6.2: BLAST result file showing the alignment between GenBank U14680 and refSeq NG_005905. The green box identifies variations between the two sequences that did not affect structurally (length-wise) while the red box shows a structural difference between the two. In the latter case the query sequence, U14680, has a Cytosine base at a position where NG_009505 appears to have no nucleotide at all.

6.3 Lessons learned

The BIC data set represents a clear added value to the HGDB. Although very outdated in some aspects, the data was once well curated and of high quality. The character of the data, being directly related to common malignancies among women, makes it of clear interest both in both clinical and research aspects. It would clearly be a waste to simply discard it, and this work has presented an effort to recovering it. The method used to recollect the BIC data entries, by aligning the outdated reference sequences to the current ones, is expected to be of future use in other sources as well. As the genomics domain rapidly advances, data sets become increasingly fast outdated. Sequences once considered the latest and a reference, can quickly be discarded from this status once progressing understanding proves them erroneous. As presented an obvious solution is to calculate the differences between those old sequences and their new counterparts, allowing the calculation of correction values. These corrections values can then be put to use in establishing the positions of the BIC variations.

The results of resolving the problems encountered in BIC are displayed in figure 5.2. It shows an overview of the total number of variations per gene -of the 15 genes considered in this work- and the number of entries finally loaded in the HGDB. Appendix B.4 provides an overview of how the amount of loaded variations relates to the total number of variations present in the repository for a given gene. In the case of BIC, the percentages are relatively low with an average of 13.48 % and a standard deviation of 1.45 %. This is due to the high amount of double entries as discussed earlier. Indeed of the 12 025 and 11 331

variations for the BRCA1 and BRCA2 genes respectively only 1446 and 1692 are considered unique and valid.

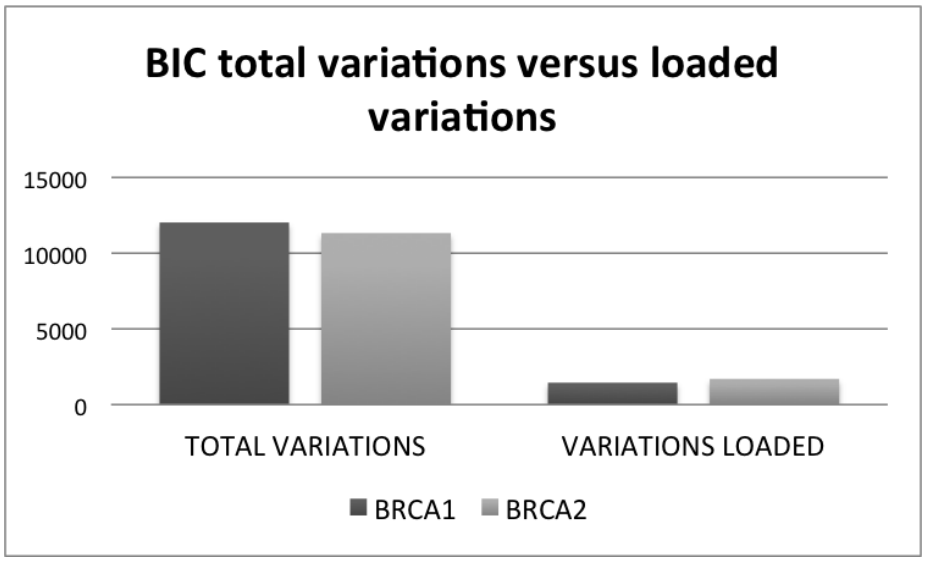

Figure 6.3: Total number of variations per gene, and the number of loaded variations per gene for the BIC data repository.

Chapter 7

Genoma Data Loader

The analysis, interpretation and loading of the above mentioned databases lead to the establishment of a variety of lessons learned about these data, and their typical problems. Considering the vast amount of genomic data, a manual approach to loading them into HGDB would be rather cumbersome and probably impossible within a reasonable time span. The solution was found in creating a custom Extraction Transformation and Load (ETL) framework, built in the Java programming language. ETL applications have been used in many different domains to integrate data from distinct data sources and integrating them in a single repository, however all current solutions had to be discarded for two main reasons. No ETL solution was found to provide the level of flexibility required to cope with the wide variety of data sources (HTML, raw text, XML) as well as present enough capability to incorporate custom algorithms to change the data. These latter algorithms, although not always very complicated, ofter require the access to data already present in the database, or 'external' files. These external files include cDNA sequences, that will after use not be loaded in the database. All in all, these latter requirements minimized the pool of possible ETL solutions to virtually none, however the applications that would maybe have offered the required functionality proved to exceed the budget in licensing fees.

Once opted for the 'from scratch' approach to build the ETL tool, the Java programming environment seemed the most appropriate tool to do it. Bioinformatics has traditionally favored Java for its multi-platform capacities and large amount of available libraries. For the purposes of this work, performance is subordinate to flexibility and availability. Indeed, it doesn't matter that much whether the database loading takes 1 hour or a whole day, as the actual process of introducing new databases and maintaining the already considered up to date is much more time consuming. The wide arrange of libraries available for Java made it a clear alternative, as this would minimize the need for reinventing the wheel in many cases. At the same time, as the project was currently set up to use Windows, Linux and Mac OS X firmly intertwined, it was reasoned that a future solution should in theory be capable of running on all these platforms,

as well as being developed on the various platforms.

The constant evolution of the Conceptual Schema, and thus the resulting database ánd the ETL solution, meant that for a useful solution the application needed to cope well with change. It was decided that a plugin-based solution would be most adequate, in which a framework would take care of the regular execution flow of the application, while plugin based execution blocks would allow for a precise definition of how each external data repository should be treated separately to transform its data to the appropriate format. Looking at the future, in which not only the input side of the application would change but also the storage side, and thus a "plugout" concept was introduced. Currently the only plugout that has been implemented is the Oracle database output interface, which allows for loading of the transformed tuples to the Oracle database. It is not difficult to imagine that various other plugouts might be created, like one that makes a flat file dump or an XML output. The use of plugins and plugouts enables the application to quickly evolve along with changes to existing data sources, as the required modifications to programming code are confined to as few classes as possible. At the same time, adding new data sources becomes a quick and easy process in which previous lessons can be taken advantage from as much as possible and requires a minimum amount of code creation; in the best case scenario only two classes need to be created, with relatively little programming complexity, to load a new data repository.

Saying that the Genoma Data Loader has known issues, yet to be resolved, can be considered an understatement. There are many known, and probably even more still unknown. This, however, should not be considered the result of a lack interest or effort as invested by both the author of this work, and many others. The GDL represents a step in the ambitious endeavor at unifying genomic data. In a domain where data is considered the main currency -characterized by complexity, large quantities and still surrounded by so much that is left unknown- this unification is exactly what is craved for. In this section we will introduce the successes of the GDL in section 7.2..

7.1 Structure

Roughly the Genoma Data Loader can be decomposed in three layers: extraction, transformation and loading. Figure 7.1 provides a conceptual schema of the main structural components of the Genoma Data Loader framework. This corresponds to the well established concept of so called ETL (Extract, Transform, Load) applications. The idea of the Genoma Data Loader is to provide a framework environment that handles the execution flow while at the same time providing great flexibility and efficiency in adding additional functionality. The latter is achieved by the use of a plugin based approach, in which so called *AGetter* plugins are used to obtain raw data from external sources. The *ATransformers* plugins are used to transform the raw data into their destination format, this includes semantic as well as syntactic changes. The *ALoader* plugins then define classes that perform the loading procedure of the prepared

data into the Human Genome Database. The high level plugin is defined by the *APlugin* abstract class. The *Tool* object is used as a wrapper around the actual plugins, instantiated by the Main class as it is executed and the command line arguments are parsed. It deserves mentioning here that the framework can be considered a pipeline where the Getters generate output -in the form of 'raw' data files- which then serve as input for the Transformers, whose output than -in the form of organized and 'clean' data files- serves as input for the Loaders.

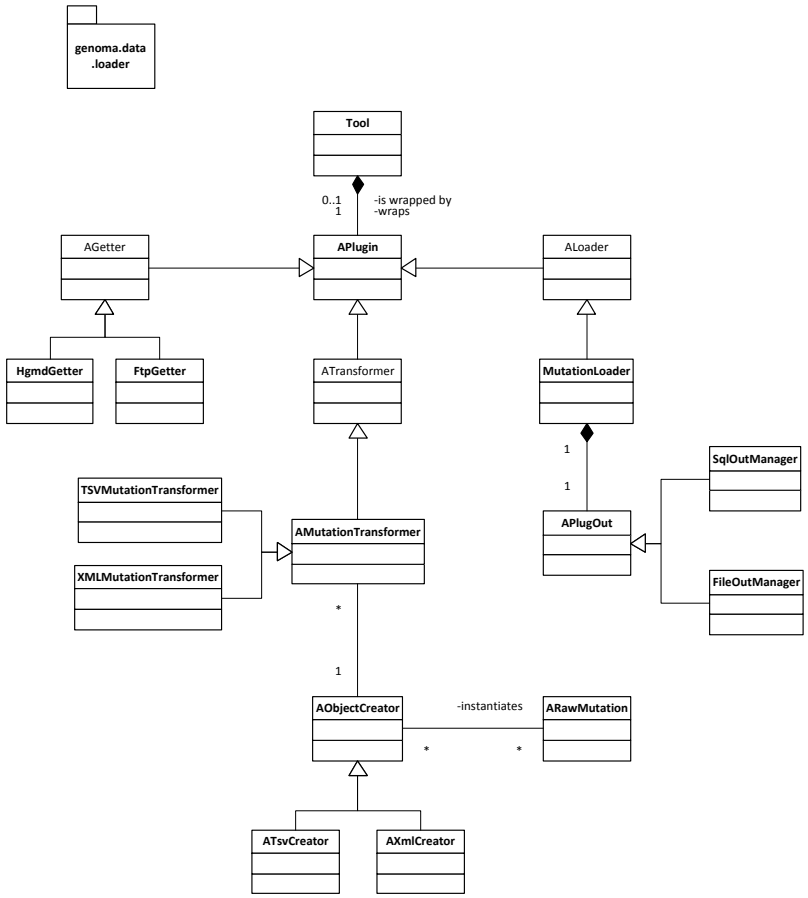

Figure 7.1: A conceptual schema that represents the structure of the Genoma Data Loader program.

Currently two Getters have been implemented: *HgmdGetter* and *FtpGetter*.

The first allows for an extraction of the HTML based HGMD data repository. As input the following needs to be specified: HGMD account user name, HGMD account password, HGMD account country and the gene for which the extraction needs to take place. The *HgmdGetter* then retrieves the Splice Junctions overview, as well as the cDNA and mutational records of that specified gene. The cDNA is currently not used in the transformation process, but might very well serve as a validation strategy in the future. It deserves mentioning here that the *HgmdGetter* should be used with precaution, as extensive use has proven to be detected by the HGMD as abuse. The used account is then blocked to deny further access. The author of this work advises approximately 5 minute gaps between each gene query.

The *FtpGetter* is more generic then the *HgmdGetter* in that it can serve to download data from various sources, including dbSNP. The reason for this higher level of genericity is plain: far more data sources provide access to their data over FTP then over HTML. As a matter of fact, currently the HGMD is the only data repository that requires an HTML-enabled data access layer. The input for the *FtpGetter* consists of a URL which defines the FTP server address, a path that defines the exact location of the file(s) within the FTP server file hierarchy and optionally log in credentials. These latter use as default an anonymous login. The *FtpGetter* then proceeds to download a single file, in case the path as specified in the parameters links a file, or the contents of a directory, in case the path refers to a directory.

The *ATranformer* currently further specializes into *AMutationTranformer* but it is conceivable that further additions are made that handle data that concerns the 'core' part of the schema -genic and allelic data, among others- which would then logically be named *ACoreTranformer*. The *AMutationTransformer* is an abstract class which bundles some common behavior for both Mutation Transformer implementations: *TSVMutationTransformer* and *XMLMutation-Transformer*. The first is an implementation that handles the input from tab separated values (TSV) -as provided by for instance BIC- and the latter handles input from XML files, as provided by dbSNP. These input files are most often generated by the earlier mentioned Getters, but for instance in the case of BIC these files need to be downloaded manually. Each *ATransformer* requires an association with an *AObjectCreator*. The latter takes an atomic tuple from the input file and instantiates the appropriate *ARawMutation* from it, for each individual data repository we have an implementation of the *ARawMutation*: e.g. *BicMutation*, *HgmdMutation*, *LovdMutation* etc.. For each external data repository we need to define the *ObjectCreator*; which can be either an *ATsvCreator*, in the case of tab separated values, or an *AXmlCreator*, in the case of XML files. For instance, in the case of the BIC we have two implementations, *BicCreator* and *BicMutation*. The BIC data files are in the TSV format and so the *BicCreator* extends *ATsvCreator*, while in our execution flow an instance of *TsvMutationTransformer* is executing. The *TsvMutationTransformer* takes an input file as parameter and uses the *BicCreator* to instantiate each atomic tuple as a *BicMutation*, which in this case corresponds to each individual line. A *BicMutation* represents the concept of what the BIC understands to be a

mutation, meaning that each field of the class corresponds to a column in the TSV input file. This allows us to capture the BIC mutation into our framework. Each *ARawMutation*, and thus extending implementations like *BicMutation*, defines a number of methods, among which a method called 'translate'. Calling this methods initiates the transformation process of this single 'raw' mutation to a 'clean' *Variation*. The latter corresponds exactly to the concept with the same name as described in section 3.2. In case of XML input files, the process is rather similar but instead of considering one line at a time -which is very convenient for TSV files- the entire file is considered by the *AXmlCreator*. The current implementation of *AXmlCreator* is thus an XML parser compliant with the requirements as described in section 5.5.

The final step in the pipeline, loading, is performed by *ALoader* plugins. For each 'part' of the conceptual schema a different *ALoader* can be created, and currently the only implementation handles the loading of mutational entries: *MutationLoader*. Future implementations might include a *CoreLoader*. As the Genoma Data Loader is expected to cope with changing environments, both input wise and output wise, an extra layer was introduced which we will refer to as *Plugouts*. The idea of a plugout is that it handles the interface with a specific output medium, like an SQL-based database, an XML file or a TSV file dump. Currently two implementations of the *APlugOut* exist: *SqlOutManager* and *FileOutManager*. The first refers to the communication layer with an SQL based database, like for instance the current Oracle database, while the latter refers to a solution that dumps the data in a TSV file format. The *SqlOutManager* uses Java PreparedStatements in batches to perform a quick and efficient load. It is a 'clean' load each time, as the affected tables are purged before actually reloading them with data.

7.2 Coverage

Currently the GDL has a coverage of 15 genes, for which various data repositories are loaded. Figure 7.2 provides a graphical overview on the distribution of the loaded data entries. It is interesting to note that in most cases dbSNP variations account for the vast majority of the tuples, except for the *BRCA1* and *BRCA2* genes. This is even more remarkable taken into account these two genes, due to their relation to highly common forms of both Breast and Ovarian cancer, have exhaustively been researched; probably more then the others. The data suggest, even when leaving the *BRCA1/2* specific BIC data set out, that far more genetic variations cause negative effects rather then being neutral on phenotype. This leads us to the interesting hypothesis that perhaps many of the variations in other, less researched genes, that are currently classified as SNP -and therefore having a neutral effect on phenotype- might actually represent a possible risk factor for susceptibility to disease. The large differences observed in the number of genetic variations among genes is also startling. Quite unexpected the *PCDH15* gene accounts for more then 8000 variations, while *BRCA1* and *BRCA2* together only account for around 3600 variations, even though these

genes are included in the BIC database and *PCDH15* is not. Possible explanations include gene size and the relative importance of the genetic message to survival of the individual. The *BRCA1* (193 689 bp) and *BRCA2* (91 193 bp) genes combined account for 284 288 base pairs while *PCDH15* alone is almost 1 million base pairs long (987 192 bp). The latter gene has been associated to hearing loss in case of malign genetic variations, which may be less crucial to the survival of the individual then susceptibility for cancer. Both facts provide interesting questions that can be considered interesting new lines of research.

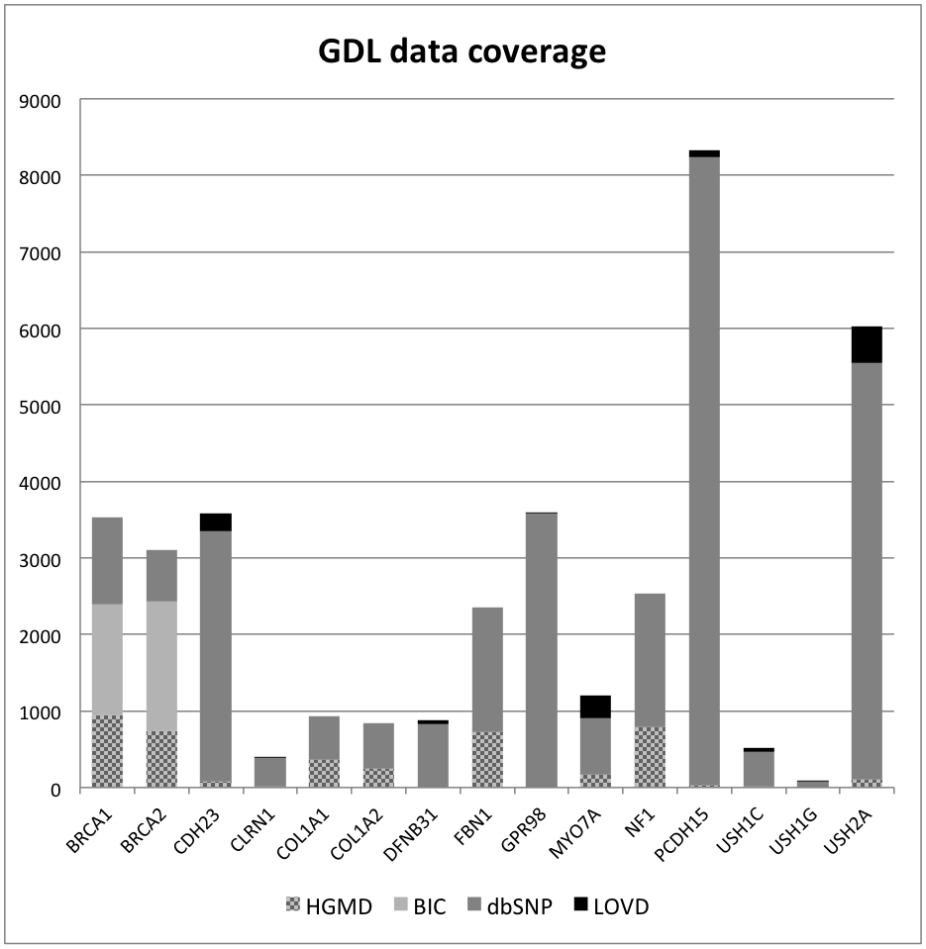

Figure 7.2: The current coverage of unique, successfully translated variations as obtained by the GDL. The data are organized per gene and per repository. It is clear that dbSNP represents the vast majority of variations in almost every gene. Only *BRCA1* and *BRCA2* appear to obtain more entries from other sources. This image might be distorted in that it is still unknown to what extend the tuples from the various data sources overlap.

Chapter 8

Conclusions

This work presents a multi disciplinary effort to combining the domains of bioinformatics and informations systems. Current bioinformatics poses a variety of challenges which makes it an interesting domain for the application of conceptual modeling techniques. These challenges include coping with (i) the large quantities of data, (ii) a high degree of ambiguity and (iii) the rapidly evolving knowledge. As technology progresses and high-throughput sequencing becomes more mainstream the problems as identified in the genomics area are expected to increase, rather than disappear. We decomposed the issue into two aspects, namely the data chaos and the conceptual chaos. The first deals with problems associated to data quantities and how they are globally stored in a fragmented manner. The latter refers to issues associated to the semantics of the data, or more specifically the ambiguity surrounding exactly these semantics in many cases.

Solving the current data related issues in genomics has the potential to drastically change the way we view health care. Craig Venter, one of the scientists that lead the 2000 effort to sequencing the first draft of the human genome, describes many of the exciting new possibilities in his book "The Language of Life" [82]. Personalized medicine is what current science believes to be the answer to many diseases, including cancers, Alzheimer's, diabetes, heart disease and many more. The idea here is that every human individual has a unique DNA fingerprint that makes it vulnerable to disease on the one hand, but more sensitive to certain medicinal compounds on the other. Charting these individual characteristics might prove a magnificent weapon in the fight against all disease. As a real world example, the cancer case is easy to grasp. Cancer can be considered in by far the most cases a mechanical defect of the genomic mechanisms. The DNA sequence contained in each cell of an individual accumulates variations throughout it's lifetime, by natural cell division -which although very accurate is not entirely error-free- or external factors like carcinogens. Each of these variations might affect phenotype slightly, like for instance increasing the speed with which the particular cell divides. Often these slight changes can be corrected by protection mechanisms present in the body, but sometimes

these accumulated variations reach a certain threshold after which the natural correction mechanisms can no longer correct the situation: a cancer is born. From these steps, it is easy to see how a each cancer is different in every individual, following the different variations one might accumulate over the years. What follows is a specific DNA fingerprint for each individual cancer which in turn makes it either vulnerable or resistant to specific chemical compounds, or agents. Please notice here that when we speak of individual cancer, we mean the specific cancer in each affected individual, not the various types of cancer like breast cancer, colon cancer, pancreatic cancer etcetera. Once we are able to relate these agents to the adequate sections of tumor 'DNA fingerprints', effective treatments are expected to emerge.

We have seen that current solutions to both the data chaos and conceptual chaos are insufficient. Ad-hoc databases and ontologies are to often to limited in their functionality, flexibility and extensibility to provide that what is needed: a holistic view of the genomic domain as an Information System. The studied data repositories (HGMD, BIC and dbSNP) each prove challenging in their own respect. The HGMD appears to be largely affected by conceptual chaos problems; here ambiguity and conceptual fuzziness makes it hard, even for experts, to determine what exactly is presented from a semantic point of view. Adding to this, clear data inconsistencies and errors have been detected. The results of the HGMD study have been published as a summary paper in the 2011 International Conference on Bioinformatics Models, Methods, and Algorithms proceedings by the title: "*Mutational data loading routines for human genome databases: the BRCA1 case*" [106] and as a full paper in the Journal of Computing Science and Engineering, in 2010 by the same title [16]. Sheer data size is also not an issue in the BIC, but rather interpretation of the data and correcting them for being outdated. In both sources, the algorithms as presented in this work aim to reduce inconsistencies by mapping data to a source deemed more reliable -the NCBI- thereby allowing for elimination of those entries considered invalid. Further more, the algorithms are able in some cases to restore outdated data tuples by mapping genetic locations among 'old' reference sequences and their newer counterparts. The results of this have been presented in section 7.2. Where the HGMD and the BIC are rather similar in their problem characteristics -being both largely conceptual chaos qualified- dbSNP is rather different.

In dbSNP the large amounts of data are what actually makes an efficient load difficult. Of course, conceptual mismatches also needed to be resolved and the data interpreted, but this was mostly a matter of elimination of invalid data, rather then transforming outdated data to a valid state. The real challenge has been to cope with the large quantities of XML-based data, as has been described in chapter 5. All in all, the conclusions of studying these data repositories is clear: the image of a fragmented genomic domain exists, both from a conceptual- ánd data-oriented perspective, is correct. The data are stored in various locations, following various schemas and standards while at the same time not always adhering to clear, common biological concepts and semantics. The solution to this problem as presented by the domain itself in the form of ontologies, although not extensively studied in this work but superficially touched

in the *State of the art* section -chapter 2-, is in our view not sufficient. It has been clarified that, in our perspective ontologies and conceptual models are not mutually exclusive but rather complementary. The broadly accepted standard, Gene Ontology, knows very specific flaws as was pointed out by Smith [79] and Kumar [80] and further reinforced in this work. A result of the latter is the acceptance of a paper in the ER2011 Onto-CoM workshop in Brussels with the title: "*Gene Ontology Based Automated Annotation: Why it isn't Working*" [107]. The GO lacks expressiveness, making it difficult for the creators to fully capture the semantics of the domain. The latter often results in inadequate use of specific constructs to compensate, thereby fading the formality of the resulting ontology. A result of the latter is the reduced ability to reason over the contents of the ontology, making it hard -if not impossible- to fulfill the promises that were originally made when GO was conceived.

Early attempts at applying a conceptual modeling approach by [35] and [52] have been extended into the Conceptual Schema of the Human Genome as reported in this work. The results of this extension have been published as a book chapter in the 2011 Handbook of Conceptual Modeling, by the title: "*A Conceptual Modeling Approach to Improve Human Genome Understanding*" [37]. The current state of data and conceptual chaos in bioinformatics in general, but genomics specifically calls for solutions. Conceptual modeling has proven to be a valuable tool in the creation of higher quality Information Systems in a variety of domains. By using a visual representation of concepts, their characteristics and relations among them we hold in hands a powerful reasoning tool which allows informations systems engineers and domain experts to closely work together. The visual aspect of conceptual schemas make it an intuitive tool for both the engineer and domain expert and enables a shared language. The fact that from sound conceptual representations nowadays full implementations can be generated in almost fully automated ways [39] further reinforces the argument of having a wider spread use of conceptual modeling techniques in genomics, where up until now most Information Systems prove to be victim of a variety of weaknesses, as has also been shown in this work. The fact that a conceptual schema implicitly defines an ontology through in conceptualization further emphasizes the complementary character of these sometimes thought to be very different technologies.

It is thus our vision that a conceptual modeling of the human genome can aid the current state of affairs in genomics in roughly two ways. First, data chaos as identified by earlier works [14] [16] [15] and confirmed by this work as a result of not applying or improperly applying well-known Information Systems principles, can largely be resolved by following a Model Driven Architecture approach. In this approach, conceptual schemas play a vital role as the backbone of the paradigm. They allow for communication between the domain experts and the Information Systems engineers, as pointed out earlier, while at the same time serving as design artifacts that allow for automated generation of (parts) of the Information System. Second, the creation of the conceptual schema of the human genome is a goal in itself. Slightly more abstract and ambitious then the former, but important nonetheless, the creation of a conceptual model

can be seen as the crystallization of a certain body of knowledge. Exactly as is being attempted by Gene Ontology -creating a tool for the unification of biology- conceptual schemas might very well fulfill the promise. As has been argued in this work and previously by Smith and Kumar [79] [80], among others, the current de-facto standard leaves a lot to wish for. We firmly believe that exactly conceptual modeling techniques can bring that what is needed to make the promise of the Gene Ontology come true: by offering a visual tool, along with improved and formally described expressiveness a powerful reasoning tool can be acquired.

One of our goals in this work was to start a discussion on the application of conceptual modeling techniques in genomics. Capturing biological understanding is a huge challenge we face, and the question is not whether we want to do it, but how do we do it. Specifications of conceptualizations are everywhere around us, and all of these correspond to the broad concept of an ontology. It is when we try and structure knowledge into a computationally sound structure and consider this to be the ontology, where not everything can stand that qualification anymore. Rigorous methods are needed, for a semantically sound ontology to emerge. The practice of conceptual modeling has been long around in Information System development, and has proven very suitable to capture domain knowledge in a controlled manner, both syntactically and semantically sound; ultimately even leading to the automated generation and validation of computer software. If we take the liberty of considering the biological system of life as analogous to an Information System it is not easy to miss the grand opportunities such a perspective provides. The analogy here is tempting and exciting; if we can create higher quality Information Systems by fully understanding their respective problem domains, perhaps we can improve life itself too by applying that very same method. After all, the mechanisms that underly the processes that drive life are not that different from our artificial, man-made Information Systems. Indeed, it requires only a small amount of imagination to see how information contained on long-term storage in DNA transfers to short-term storage RNA and is eventually 'executed' in an intricate chemical processing unit to form an organism.

Chapter 9

Future work

One of the first future research lines that comes to mind is evolution of conceptual data modeling to cope with immense data quantities. It is clear that in bioinformatics very large amounts of data are being managed, and that this is very likely to increase exponentially the coming decades as technology further develops. The current MDA techniques allow for the automated creation of conventional relational databases like MySQL, Oracle and PostgreSQL. The rules for transforming platform independent models (conceptual schemas) to implementations that follow this relational paradigm have been well established. These technologies, although very mature and performing exceptional in some applications, date back to the '70s. The current information age has changed a lot since then; data quantities have increased, hardware capabilities have changed, as well as user requirements, imposing entirely new challenges on storage technology. Current popular solutions include the so-called NoSQL-based solutions [108], in which conventional relational algebra is replaced with different paradigms that allow for far larger data quantities, faster access times while in some cases trading in some consistency in real time applications [109]. Currently NoSQL solutions are put to practice by large multinationals like Facebook [110], Google [111], Amazon and Twitter. Although often referred to as NoSQL, we prefer the term non-relational databases as it covers the semantics better. It would be a very interesting line of research to see if and how conceptual modeling can be applied to the application to these new technologies. In particular, discovering the set of rules and transformations required to translate a conventional conceptual schema (platform independent model) to a non-relational database, thereby combining the reduced development effort of model driven engineering with the ability to cope with today's high data quantity requirements.

Next, an interesting thought is the classification of different classes of loading problems. The Genoma Data Loader program is able to identify certain types of problems, like inconsistencies among data sources and format errors. The latter referring to for instance, typing errors in which a letter is introduced in the 'location' attribute of a mutation. Classifying these problems into separate categories and determining how runtime encountered problems can be assigned

to them is expected to spark well directed research lines. By applying statistical methods it is then possible to determine 'out of the ordinary' data sets within data sources, and among them. To illustrate we will take the example of the HGMD, here we might assume that the average format error -for instance a letter in an integer column- affects around 1 in a 100 mutations, per gene. We now know, through applying statistics, if a certain average format error for a newly loaded gene is significantly higher then the average overall that something might be off; either the Genoma Data Loader code does not cope well with this particular gene, or this is a new kind of formatting not yet implemented.

Through this work we have used a gene-centric approach. The conceptual schema takes the concept of a gene as a central hub, an connects additional properties to this main artifact. Reality might be slightly different as will be illustrated with an example. Currently all variations in the Human Genome Database are associated to genes and their respective result on phenotype. The idea here is that if a gene is the atomic element of heredity, only those variations that affect a gene will have an effect on phenotype. This however turns out not be the case, as is now commonly believed among geneticists that also variations in non-coding parts of the genome can affect phenotype. This latter renders, at least to some extend, the currently chosen perspective invalid. A better approach, and one that is being investigated right now, is to use the chromosome centric approach. This allows for connecting variations to chromosomes, rather then genes. This is in turn makes full sense, as we take into account that the earlier gene centric solution was based on the shortcomings of technology at that time, being unable to sequence entire genomes at reasonable prices. Now that we possess over the short term prospect of acquiring cheap full genome sequencing, these DNA stretches between the genes will be analyzed in much more detail. The current evolution of the conceptual model, which took place in parallel with this work, is already exploring these ideas.

One last interesting thought deals with data redundancy. It is currently unknown to what degree common genomic data sources like the BIC, LOVD and HGMD overlap. They each store very similar data, but in varying data formats that make a comparison non-trivial. The result of this work, basically the integration of these data in a common format, has made this comparison a very feasible research line. It would be particularly interesting to see how the entries in the relatively old and abandoned BIC made it, or not, to the other data sources. The difficulty will be to determine what are the basic properties of a variation that determine its identity. The authors of this work suggest that the actual change and position are all that is needed. Some discussion can exist about whether the phenotype should also be added. After all, it is still very unknown if one variation can have just one, or various phenotypes. And if a variation leads to various phenotypes, it must mean that other genetic changes also affect the process; shouldn't they be part of the concept of *Variation* too, if that is what we want to indicate the genotypic blocks that lead to different phenotypes? Indeed, following this line of thought the *Variation* concept should not be restricted to a single change when it is connected to the *Phenotype* concept, but rather a new concept in between. This new concept could be

called *Haplotype*, following the genetic concept of variations being transmitted together.

Acknowledgements

The contents of this work are a result of the two and half years I have had the pleasure to work as a junior researcher at the Universidad Politécnica de Valencia, Spain. Working together with so many wonderful people, and being able to reside in such a great environment -both from an academic ánd a general point of view- is what sparked my interest for the genomics domain, where so much is still left to be done. Naturally, a work of this size comes to be as a result of the collaboration of many and I take mere credits, if any, for assembling those lessons learned from so many into a single body of work. In the following section I will proceed to thank and credit those of you who have influenced me for the better, and stood by me when times were rough. Naturally, although all merits of this work ought to be shared with those mentioned here, I take sole responsibility for any errors you as a reader might discover.

Where would we be, without our parents? Simply nowhere. This holds true from both an existential point of view for we would not exist, and a moral perspective because without them, we would not know how to act. Existential when we talk about genetics and the messages conveyed by it -some good and some maybe not so good- and moral when we refer to who we are as a human being and how we treat others. Parenthood only deals with the first, you see, while good parenthood is associated with both. In the end, we need our parents to shape us while later we need them for guidance and support, as much as we need them for the petty act of existence. It is for this reason, that although perhaps not much of the work presented here has been influenced directly by my parents, the mere existence is fully indebted to them. For this, and everything else I thank both of them. I, on the other hand, feel that this work represents my humblest effort of contributing to science in such a manner that maybe one day problems we face today, like cancers, may be overcome.

Further, I want to pay explicit gratitude to my mentor and good friend Óscar Pastor for creating so many opportunities and accepting me as one of his students. It has been an extremely educational and fun experience to work so closely with one of the founders of the Model Driven Architecture paradigm. Although -or perhaps because- we disagreed on many occasions it has been trough the resulting discussions that new insights were born, many of which made it to this work. And isn't that what science is all about? Special thanks also go out to Ana Levin, who I think of as the biology conscience of my work, as well as a very close friend. If it weren't for her, I would probably not have been

where I am now. Apart from monitoring my scientific development, correcting and teaching me when necessary she was also there to support me whenever necessary.

My thanks go out to the entire ProS institute, with all it's great members for providing such a pleasant environment to work in and welcoming a stranger with so much warmth. Special mentioning deserve Ainoha Martin Mayordomo for her contribution to the analysis of the BIC data repository, Veronica Burriel Coll for sharing her results of analyzing the LOVD data repositories and Arthur Baars for enlightening me with his great wealth of knowledge on Information Systems design and development. I also want to thank the full Genoma team, apart from the members already mentioned, for sharing so much valuable knowledge in a variety of domains.

Bibliography

[1] The International HapMap Consortium. "The International HapMap Project". In: *Nature* 426.6968 (2003), pp. 789–796.

[2] M.B. Gerstein et al. "What is a gene, post-ENCODE?" In: *Genome Research* 17.6 (2007), pp. 669–681.

[3] H. Pearson. "Genetics: What is a gene?" In: *Nature* 441.7092 (2006), pp. 298–402.

[4] M. Ashburner, C.A. Ball, and J.A. Blake. "Gene Ontology: Tool for the Unification of Biology". In: *Nature genetics* 25.1 (2000), pp. 25–30.

[5] E.J. Chikofsky and J.I. Cross. "Reverse Engineering and Design Recovery: a Taxonomy". In: *IEEE software* 7.1 (1990), pp. 13–18.

[6] G. Canfora and M. Di Penta. "Frontiers of reverse engineering: a conceptual model". In: *Proceedings of Frontiers of Software Maintenance (FoSM 2008)*. 2008, pp. 38–47.

[7] O. Pastor et al. "The OO-method Approach for Information Systems Modeling: From Object-Oriented Conceptual Modeling to Automated Programming". In: *Information Systems* 26.7 (2001), pp. 507–534.

[8] B. Alberts et al. *Essential Cell Biology*. Ed. by E. Zayatz and E. Lawrence. 2nd. Garland Science USA, 2003.

[9] J.C. Venter et al. "The sequence of the human genome". In: *Science* 291 (2001), pp. 1304–1351.

[10] E.S. Lander and et al. "Initial sequencing and analysis of the human genome". In: *Nature* 409 (2001), pp. 860–921.

[11] J.S. Mattick. "Challenging the dogma: the hidden layer of non-protein-coding RNAs in complex organisms". In: *Bioessays* 25.10 (2003), pp. 930–939. ISSN: 1521-1878.

[12] J.S. Mattick. "RNA regulation: a new genetics?" In: *Nature Reviews Genetics* 5.4 (2004), pp. 316–323. ISSN: 1471-0056.

[13] S.E. Ahnert, T. Fink, and A. Zinovyev. "How much non-coding DNA do eukaryotes require?" In: *Journal of theoretical biology* 252.4 (2008), pp. 587–592. ISSN: 0022-5193.

[14] E. Baralis and A. Fiori. "Exploring Heterogeneous Biological Data Sources". In: *19th International Conference on Database and Expert Systems Applications*. 2008, pp. 647–651.

[15] H. Macauley, N. Wang, and N. Goodman. "A model system for studying the integration of molecular biology databases". In: *Bioinformatics* 14 (1998), pp. 575–585.

[16] M. van der Kroon et al. "Mutational data loading routines for human genome databases: the BRCA1 case". In: *Journal of Computing Science and Engineering* 4.4 (2010), pp. 291–312.

[17] S.T Sherry et al. "dbSNP: the NCBI database of genetic variation". In: *Nucleic acids research* 29.1 (2001), p. 308.

[18] K.D. Pruit and D.R. Maglott. "RefSeq and LocusLink: NCBI gene-centered resources". In: *Nucleic Acids Research* 29.1 (2001), pp. 137–140.

[19] T. Tatusova. "Methods in Molecular Biology". In: ed. by C. Oliviero and F. Eisenhaber. Vol. 609. Biomedical and Life Sciences. Springer-Verlag, 2010. Chap. 2 - Genomic databases and resources at the National Center for Biotechnology Information, pp. 17–44.

[20] T. Hubbard et al. "The Ensembl genome database project". In: *Nucleic Acids Research* 30.1 (2002), pp. 38–41.

[21] A. Hamosh et al. "Online Mendelian Inheritance in Man (OMIM)". In: *Human Mutation* 15 (2000), pp. 57–61.

[22] J. Amberger et al. "McKusick's Online Mendelian Inheritance in Man (OMIM (R))". In: *Nucleic Acids Research* (2008).

[23] G. Joshi-Tope et al. "Reactome: a knowledgebase of biological pathways". In: *Nucleic Acids Research* 33.issue suppl. 1 (2005), pp. D428–D432.

[24] Lisa Matthews et al. "Reactome knowledgebase of human biological pathways and processes". In: *Nucleic Acids Research* 37.suppl 1 (2009), pp. D619–D622.

[25] M. Kanehisa and S. Goto. "KEGG: Kyoto Encyclopedia of Genes and Genomes". In: *Nucleic Acids Research* 28.1 (2000), pp. 27–30.

[26] M. Kanehisa et al. "From genomics to chemical genomics: new developments in KEGG". In: *Nucleic Acids Research* 34.issue suppl. 1 (2006), pp. D354–D357.

[27] M. Kanehisa et al. "KEGG for representation and analysis of molecular networks involving diseases and drugs". In: *Nucleic Acids Research* 38.issue suppl. 1 (2010), pp. D355–D360.

[28] P.D. Stenson et al. "The Human Gene Mutation Database (HGMD®): 2003 update". In: *Human Mutation* 21.6 (2003), pp. 577–581.

[29] S. Bamford et al. "The COSMIC (Catalogue of Somatic Mutations in Cancer) database and website". In: *British journal of cancer* 91.2 (2004), pp. 355–358.

[30] C. Szabo et al. "The Breast Cancer Information Core: Database Design, Structure, and Scope". In: *Human Mutation* 16 (2000), pp. 123–131.

[31] I.F.A.C. Fokkema, J.T. den Dunnen, and P.E.M. Taschner. "LOVD: Easy creation of a locus-specific sequence variation database using an "LSDB-in-a-box" approach". In: *Human mutation* 26.2 (2005), pp. 63–68.

[32] Amos Bairoch et al. "The Universal Protein Resource (UniProt)". In: *Nucleic Acids Research* 33.suppl 1 (2005), pp. D154–D159.

[33] Sarah Hunter et al. "InterPro: the integrative protein signature database". In: *Nucleic Acids Research* 37.suppl 1 (2009), pp. D211–D215.

[34] Damian Smedley et al. "BioMart - biological queries made easy". In: *BMC Genomics* 10.1 (2009), p. 22.

[35] N.W. Paton et al. "Conceptual modeling of genomic information". In: *Bioinformatics* 16.6 (2000), pp. 548–557.

[36] E.W. Bornberg-Bauer and N.W. Paton. "Conceptual Data Modeling for Bioinformatics". In: *Briefings in bioinformatics* 3.2 (2002), pp. 166–188.

[37] O. Pastor et al. "A Conceptual Modeling Approach to Improve Human Genome Understanding". In: ed. by D.W. Embley and B. Thalheim. 1st. Handbook of Conceptual Modeling. Springer-Verlag, 2011. Chap. 16, p. 25.

[38] F. Christen. "Global Sequencing: A Review of Current Molecular Data and New Methods Available to Assess Microbial Diversity". In: *Microbes and Environments* 23.4 (2008), pp. 253–268.

[39] O. Pastor and J.C. Molina. *Model-driven architecture in practice: a software production environment based on conceptual modeling*. Springer-Verlag. Berlin-Heidelberg, 2007.

[40] D. Batra, J.A. Hoffler, and Bostrom R.P. "Comparing Representation with Relational and EER models". In: *Communications of the ACM* 33.2 (1990), pp. 126–139.

[41] D. Batra and G.M. Marakas. "Conceptual Data Modeling in Theory and Practice". In: *European Journal of Information Systems* 4.3 (1995), pp. 185–194.

[42] J. Larkin and H. Simon. "When a Diagram is (Sometimes) Worth Ten Thousand Words". In: *Cognitive Science* 11.1 (1987), pp. 65–100.

[43] B. Hailpern and P. Tarr. "Model Driven Development: The Good, the Bad, and the Ugly". In: *IBM Systems Journal* 45.3 (2006), pp. 451–462.

[44] H. Topi and V. Ramesh. "Human Factors Research on Data Modeling: A Review of Prior Research, An Extended Framework and Future Research Directions". In: *Journal of Database Management* 13.2 (2002), pp. 3–19.

[45] I. Davies et al. "How Do Practitioners Use Conceptual Modeling in Practice". In: *Data and Knowledge Engineering* 58 (2006), pp. 358–380.

[46] B. Dobing and J. Parsons. "How UML is used". In: *Communications of the ACM* 49.5 (2006), pp. 109–114.

[47] M.I. Aguirre-Urreta and G.M. Marakas. "Comparing Conceptual Modeling Techniques: A Critical Review of the EER vs. OO Empirical Literature". In: *The DATA BASE for Advances in Information Systems* 39.2 (2008), pp. 9–32.

[48] Y. Wand and R. Weber. "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars". In: *Journal of Information Systems* 13.4 (1993), pp. 363–376.

[49] R. Weber. "Conceptual Modeling and Ontology: Possibilities and Pitfalls". In: *Journal of Database Management* 14.3 (2003), pp. 1–20.

[50] B. Wyssusek. "On Ontological Foundations of Conceptual Modeling". In: *Scandinavian Journal of Information Systems* 18.1 (2006), pp. 63–80.

[51] V. Khatri et al. "Understanding Conceptual Schemas: Exploring the Role of Application and IS Domain Knowledge". In: *Information Systems Research* 17.1 (2006), pp. 81–99.

[52] O. Pastor. "Conceptual modeling meets the human genome". In: *Conceptual modeling - ER 2008*. Vol. 5231. Lecture Notes in Computer Science. Springer-Verlag Berling Heidelberg, 2008, pp. 1–11.

[53] O. Pastor et al. "The Evolution of Conceptual Modeling". In: ed. by H. Mayr. Springer-Verlag, 2009. Chap. Model driven-based engineering applied to the interpretation of the human genome.

[54] A. Olive. *Conceptual modeling of Information Systems*. Springer-Verlag. Berlin-Heidelberg, 2007.

[55] T.R. Gruber. "A translation approach to portable ontology specification". In: *Knowledge Acquisition* 5 (1993), pp. 199–220.

[56] M. Uschold and M. Gruninger. "Ontologies: Principles, Methods and Applications". In: *The Knowledge Engineering Review* 11.2 (1996), pp. 93–136.

[57] B. Gaines. "Editorial: Using Explicit Ontologies in Knowledge-based System Development". In: *International Journal of Human-Computer System* 46 (1996), p. 181.

[58] A.K. Gomez-Perez. "The Handbook on Expert Systems". In: CRC Press, 1997. Chap. Knowledge Sharing and Reuse.

[59] N. Guarino. "Formal Ontology, Conceptual Analysis and Knowledge Representation". In: *International Journal of Human and Computer Studies* 43.5/6 (1995), pp. 625–640.

[60] A. Artale et al. "Part-Whole Relations in Object-Centered Systems: an Overview". In: *Data and Knowledge Engineering* 20.3 (1996), pp. 347–383.

[61] J.F. Sowa. "Formal Ontology, Conceptual Analysis and Knowledge Representation". In: IEEE Press, 2010. Chap. Basic Tools of Formal Ontology.

[62] N.M. Gotts, J.M. Gooday, and A.G. Cohn. "A Connection Based Approach to Commonsense Topological Description and Reasoning". In: *The Monist: An International Journal of General Philosophical Inquiry* 79.1 (1996).

[63] S. Borgo, N. Guarino, and C. Masolo. "An Ontological Theory of Physical Objects". In: *In Proceedings of Qualitative Reasoning 11th International Workshop* (1997), pp. 223–231.

[64] R. Casati and A. Varzi. "Spatial and Temporal Reasoning". In: Kluwer, Dordrecht, 1997. Chap. Spatial Entities.

[65] E. Lang. "Text Understanding in LILOG". In: Springer-Verlag, Berlin, 1991. Chap. The LILOG Ontology from a Linguistic Point of View.

[66] J.A. Bateman. "On the Relationship Between Ontology Construction and Natural Language: a Socio-Semiotic View". In: *International Journal of Human-Computer Studies* 43 (1995), pp. 929–944.

[67] J.F.M Burg. *Linguistic Instruments in Requirements Engineering*. IOS Press, 1997.

[68] R. van der Riet, H. Burg, and F. Dehne. "Formal Ontology in Information Systems". In: IOS Press, 1998. Chap. Linguistic Issues in Information Systems Design.

[69] S. Schulze-Kremer. "Ontologies for molecular biology". In: *Proceedings of the Third Pacific Symposium on Biocomputing* (1998), pp. 705–716.

[70] D.B. Lenat et al. "Cyc: Toward Programs with Common Sense". In: *Communications of the ACM* 33.8 (1990), pp. 30–49.

[71] K. Mahesh and S. Nirenburg. "Meaning Representation for Knowledge Sharing in Practical Machine Translation". In: *Proceedings of the FLAIRS-96 track on Information Interchange*. 1996.

[72] D.F. Schwarz et al. "SNPtoGO: characterizing SNPs by enriched GO terms". In: *Bioinformatics* 24.1 (2008), p. 146.

[73] A. Coulet et al. "SNP-Converter: An Ontology-Based Solution to Reconcile Heterogeneous SNP Descriptions for Pharmacogenomic Studies." In: *DILS'06* (2006).

[74] J.B.L. Bard and S.Y. Rhee. "Ontologies in Biology: Design, Applications and Future Challenges". In: *Nature Genetics* 5 (2004), pp. 213–222.

[75] T. Okayama et al. "Formal design and implementation of an improved DDBJ DNA database with a new schema and object-oriented library". In: *Bioinformatics* 14.6 (1998), p. 472.

[76] I.-M.A Chen and V. Markovitz. "Modeling scientific experiments with an object data modeling approach". In: *Proceedings of the SSDBM*. IEEE Press, 1995.

[77] C. Medigue et al. "Imagene: an integrated computer environment for sequence annotation and analysis". In: *Bioinformatics* 15 (1999), pp. 2–15.

[78] O. Pastor et al. "Proceedings of the IVth Int. Conference on Research Challenges in Information Science". In: IEEE Press, 2010. Chap. Enforcing Conceptual Modeling to Improve the Understanding of Human Genome.

[79] B. Smith, J. Williams, and S. Schulze-Kremer. "The Ontology of the Gene Ontology". In: *American Medical Informatics Association Annual Symposium Proceedings* (2003), pp. 609–613.

[80] A. Kumar and B. Smith. "Controlled vocabularies in bioinformatics: a case study in the Gene Ontology". In: *Drug Discovery Today:: BIOSILICO* 2.6 (2004), pp. 246–252.

[81] M. Egaña Aranguren et al. "Understanding and using the meaning of statements in bio-ontology: recasting the Gene Ontology in OWL". In: *BMC Bioinformatics* 8.57 (2007).

[82] F.S. Collins. *The Language of Life: DNA and the Revolution in Personalized Medicine*. Profile Books Ltd., 2010.

[83] O. Pastor, M.A. Pastor, and V. Burriel. "Conceptual modeling of human genome mutations: a dichotomy between what we have and what we should have". In: *Proceedings of Bioinformatics 2010*. BIOSTEC Bioinformatics. 2010, pp. 160–166.

[84] J. Warmer and A. Kleppe. *Object Constraint Language: Getting Your Models Ready for MDA*. 2nd. Addison-Wesley Longman Publishing Co., 2003.

[85] HGNC. *HUGO Gene Nomenclature Committee, http://www.genenames.org/*. 2010. URL: http://www.genenames.org/.

[86] C.G Spilianakis et al. "Interchromosomal associations between alternatively expressed loci". In: *Nature* 435 (2005), pp. 637–645.

[87] A. Giovane, A. Balestrieri, and C. Napoli. "New insights into cardiovascular and lipid metabolomics". In: *Journal of Cellular Biochemistry* 105.648-654 (2008).

[88] R. Panguluri et al. "BRCA1 mutations in African Americans". In: *Human Genetics* 105.1-2 (1999), pp. 28–31.

[89] A. Langston et al. "Germ-line BRCA1 mutations in selected men with prostate cancer". In: *American Journal of Human Genetics* 58 (1996), pp. 881–885.

[90] A. van der Hout and et al. "A DGGE system for comprehensive mutation screening of BRCA1 and BRCA2: application in a Dutch cancer clinic". In: *Human Mutation* 27.7 (2006), pp. 654–666.

[91] D. Shattuck-Eidens and et al. "BRCA1 sequence analysis in women at high risk for susceptibility mutations". In: *The Journal of the American Medical Association* 278.15 (2009), pp. 1242–1250.

[92] Y. Miki et al. "A Strong Candidate for the Breast and Ovarian Cancer Susceptibility Gene BRCA1". In: *Science* 266.5182 (1994), pp. 66–71.

[93] U. Khoo et al. "Mutational analysis of BRCA1 and BRCA2 genes in Chinese ovarian cancer identifies 6 novel germline mutations". In: *Human Mutation* 16.1 (2000), pp. 88–89.

[94] Z. Zhao et al. "Investigating single nucleotide polymorphism (SNP) density in the human genome and its implications for molecular evolution". In: *Gene* 312 (2003), pp. 207–213.

[95] M. van der Kroon et al. *Mutational data loading routines for human genome databases: the BRCA1 case*. Report UUCS2009020. Utrecht University, 2009.

[96] L. Stein. "Creating a bioinformatics nation". In: *Nature* 417.6885 (2002), pp. 119–121.

[97] NCBI. *The National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov/*. 2010. URL: http://www.ncbi.nlm.nih.gov/.

[98] P. Yue and J. Moult. "Identification and analysis of deleterious human SNPs". In: *Journal of Molecular Biology* 356.5 (2006), pp. 1263–1274.

[99] B.S. Shastry. "SNPs: Impact on gene function and phenotype". In: *Methods in Molecular Biology* 578 (2009), pp. 3–22.

[100] A. Vignal et al. "A review on SNP and other types of molecular markers and their use in animal genetics". In: *Genetics, selection, evolution* 34.3 (2002), p. 275.

[101] B. Devlin and N. Risch. "A comparison of Linkage Disequilibrium measures for fine-scale mapping." In: *Genomics* 29.2 (1995), pp. 311–322.

[102] P.A. Futreal et al. "BRCA1 mutations in primary breast and ovarian carcinomas". In: *Science* 266.5182 (1994), pp. 120–122.

[103] D.H. Teng et al. "Low Incidence of BRCA2 Mutations in Breast Carcinoma and Other Cancers". In: *Nature Genetics* 13.2 (1996), pp. 241–244.

[104] S.F. Altschul et al. "Basic Local Alignment Search Tool". In: *Journal of Molecular Biology* 215.3 (1990), pp. 403–410.

[105] W.J. Kent. "BLAT the BLAST like alignment tool". In: *Genome Research* 12 (2002), pp. 656–664.

[106] M. van der Kroon et al. "Mutational data loading routines for human genome databases: the BRCA1 case". In: *The International Conference on Bioinformatics Models, Methods and Algorithms 2011 Proceedings*. 2011, p. 4.

[107] M. van der Kroon and A.M. Levin. "Gene Ontology Based Automated Annotation: Why it isn't Working". In: *ER2011 Workshops*. Ed. by O. De Troyer et al. Vol. 6999. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2011, pp. 203–209.

[108] N. Leavitt. "Will NoSQL Databases Live Up to Their Promise?" In: *Computer* 43.2 (2010), pp. 12–14.

[109] P. Xiang, R. Hou, and Z. Zhou. "Cache and Consistency in NoSQL". In: *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*. 2010, pp. 117–120.

[110] A. Lakshman and P. Malik. "Cassandra: a decentralized structured storage system". In: *ACM SIGOPS Operating Systems Review* 44.2 (2010), pp. 35–40.

[111] F. Chang et al. "Bigtable: a Distributed Storage System for Structured Data". In: *ACM Transactions of Computer Systems* 26.2 (2008), pp. 1–26.

Appendices

Appendix A

Data format samples

A.1 HGMD precise mutations format sample

This screenshot provides a typical example of how HGMD precise mutations are presented to the user. In this particular case we see the first 11 missense/nonsense mutations of the *BRCA1* gene. The other precise mutations -small deletions, small indels and small insertions- have a very similar layout. The presentation is consistent among genes.

Accession Number	Codon change	Amino acid change	Codon number	Phenotype	Reference	Comments
CM021503	aATG-GTG	Met-Val	1	Breast and/or ovarian cancer	Meindl (2002) Int J Cancer **97,** 472	
CM041678	ATG-ACG	Met-Thr	1	Breast and/or ovarian cancer ?	Abkevich (2004) J Med Genet **41,** 492	
CM014520	ATG-AGG	Met-Arg	1	Ovarian cancer	Sekine (2001) Clin Cancer Res **7,** 3144	
CM960163	ATGg-ATT	Met-Ile	1	Breast cancer	Couch (1996) Hum Mutat **8,** 8	
CM940170	GTA-GCA	Val-Ala	11	Breast cancer	Castilla (1994) Nat Genet **8,** 387	
CM031646	aCAA-TAA	Gln-Term	12	Breast cancer	Adem (2003) Cancer **97,** 1	
CM041679	ATT-ACT	Ile-Thr	15	Breast and/or ovarian cancer ?	Abkevich (2004) J Med Genet **41,** 492	
CM012906	ATG-AAG	Met-Lys	18	Ovarian cancer	Machackova (2001) Hum Mutat **18,** 545	
CM004187	ATG-ACG	Met-Thr	18	Breast cancer	Malone (1998) JAMA **279,** 922 Additional report available to subscribers Additional report available to subscribers	
CM081536	gCAG-TAG	Gln-Term	19	Breast and/or ovarian cancer	Machackova (2008) BMC Cancer **8,** 140	
CM940171	aATC-GTC	Ile-Val	21	Breast cancer	Castilla (1994) Nat Genet **8,** 387	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Katagiri (1998) J Hum Genet **43,** 42	

A.2 HGMD imprecise mutations format sample

This screenshot provides a typical example of how HGMD presents imprecise mutations to the user. In this particular case the first few entries of the gross deletions overview of the *BRCA1* gene are shown. The other imprecise mutations -gross insertions, gross indels, regulatory and complex rearrangements- have a similar layout.

Accession Number	Description	Phenotype	Reference	Comments
CG973413	1 kb incl. ex. 17 (described at genomic DNA level)	Breast cancer	Puget (1997) Cancer Res **57,** 828 Additional report available to subscribers Additional characterisation report available to subscribers	
CG072660	10410 bp incl. ex. 13-15 (described at genomic DNA level)	Ovarian cancer	Lim (2007) Clin Genet **71,** 331 Additional characterisation report available to subscribers	
CG075769	106 bp c.442-547 (described at cDNA level)	Breast and/or ovarian cancer	Laki (2007) Cancer **109,** 1784	
CG994820	11 bp nt. 188 (described at genomic DNA level)	Breast cancer	Hopper (1999) Cancer Epidemiol Biomarkers Prev **8,** 741	
CG015438	11.6 kb ex. 13-15 (described at cDNA level)	Breast cancer	Gad (2001) J Med Genet **38,** 388 Additional characterisation report available to subscribers Additional characterisation report available to subscribers	
CG084357	11.6 kb incl. ex. 13-15 (described at genomic DNA level)	Breast cancer	Engert (2008) Hum Mutat **29,** 948 Additional characterisation report available to subscribers	Breakpoint 2.....
CG065883	11357 bp incl. ex. 20-22 (described at genomic DNA level)	Breast cancer	Walsh (2006) JAMA **295,** 1379 Additional characterisation report available to subscribers	
CG015437	119 bp nt. 5083 (described at genomic DNA level)	Breast and/or ovarian cancer	Martin (2001) J Clin Oncol **19,** 2247	
CG073414	14 kb incl. ex. 13-16 (described at genomic DNA	Breast cancer	Petrij-Bosch (1997) Nat Genet **17,** 341	

A.3 dbSNP XML format sample

A sample of the XML file used by dbSNP to transfer data. What is shown is a single Rs entry, which represents a single Single Nucleotide Polymorphism. The data of this particular entry has been manipulated in order to fit on the page, this includes semantical changes. The entry as shown on this page therefore does not represent a correct dbSNP entry.

```
<Rs rsId="3894" snpClass="snp" snpType="notwithdrawn"
molType="genomic" genotype="true"
bitField="0500000000010105000020100" taxId="9606">

  <Het type="est" value="0" stdError="0"/>
  <Validation byCluster="true" byOtherPop="true">
    <otherPopBatchId>7179</otherPopBatchId>
  </Validation>
  <Create build="36" date="2000-09-19 17:02"/>
  <Update build="132" date="2011-01-11 12:25"/>

  <Sequence exemplarSs="3931" ancestralAllele="C">
```

```
    <Seq5>TAATCAGTCTCCTCCCAGCAAGTGATAT</Seq5>
    <Observed>C/T</Observed>
    <Seq3>AATTAGGAAGAGCTGGTACCTAAAAT</Seq3>
  </Sequence>

 <Ss ssId="3931" handle="OEFNER" batchId="489" locSnpId="M3"
  subSnpClass="snp" orient="forward" strand="bottom"
  molType="genomic" buildId="36" methodClass="DHPLC"
  validated="by-cluster">

    <Sequence>
      <Seq5>TAATCAGTCTCCTCCCAGCAAGTGATATG</Seq5>
      <Observed>C/T</Observed>
      <Seq3>AATTAGGAAGAGCTGGTACCTAAAATGATTTT</Seq3>
    </Sequence>
  </Ss>

  <Assembly dbSnpBuild="132" genomeBuild="37_1"
  groupLabel="GRCh37" current="true" reference="true">

    <Component componentType="contig" ctgId="224514809"
    accession="NT_011875.12" name="NT_011875.12" chromosome="Y"
    start="13798578" end="23901427" orientation="fwd"
    gi="224514809" groupTerm="GRCh37" contigLabel="GRCh37">

      <MapLoc asnFrom="5297784" asnTo="5297784" locType="exact"
      alnQuality="1" orient="reverse" physMapInt="19096362"
      leftFlankNeighborPos="179" rightFlankNeighborPos="181"
      leftContigNeighborPos="5297783" rightContigNeighborPos="5297785"
      numberOfMismatches="1" numberOfDeletions="0"
      numberOfInsertions="0"/>
    </Component>

    <SnpStat mapWeight="unique-in-contig" chromCount="1"
    placedContigCount="1" unplacedContigCount="0" seqlocCount="1"
    hapCount="0"/>
  </Assembly>

  <RsLinkout resourceId="4" linkValue="60936"/>
  <MergeHistory rsId="56546490" buildId="130"/>
  <hgvs>NT_011875.12:g.5297785G&amp;gt;A</hgvs>
</Rs>
```

A.4 BIC data format sample

Accession Number	Unique identifier generated at the time mutation is entered is entered into database
Exon	BRCA1 or BRCA2 exon in which mutation has been identified
cDNA nucleotide	Nucleotide # in the transcript at which mutation occurs; Reference sequences: BRCA1 GenBank U14680; BRCA2 GenBank U43746
gDNA nucleotide	Nucleotide # in genomic DNA at which mutation occurs; Reference sequences: BRCA1 GenBank L78833
Codon	Triplet codon # (ATG is +1) in which mutation occurs
Base change	Description of nucleotide difference compared to reference sequence
Amino acid change	Description of resulting change in the encoded amino acid sequence
Mutation designation	Designation of described mutation according to nomenclature guidelines
Mutation type	Frameshift, nonsense, missense, splice, in frame deletion
Mutation effect	Frameshift, nonsense, missense, splice, unclassified variant, polymorphism
Depositor	Contributor of mutation report
Patient sample source	Hospital or research institute from which patient sample originated
Patient ID number	Patient identifier designation in publications, or used by the hospital or research institute
Number of times reported	Number of family members carrying mutation
Germline or Somatic mutation	
Detection method used	SSCP, direct sequencing, DGGE etc.
Proband tumor type	Breast, ovarian, other
Number of chromosomes screened	Number of normal control chromosomes screened
Frequency of polymorphism (A/C/T/G)	Information on the frequency of variants in normal control chromosomes, if known
Literature reference	Literature citation in which mutation was first reported
Contact person	Email address of individual to whom inquiries should be addressed
Notes	Additional information describing the mutation, family history, patient series, age of onset, etc.
Creation date	Date on which mutation is entered into the public database
Ethnicity	Information on patient ethnicity, if known
Nationality	Information on patient nationality, if known

Appendix B

Genoma Data Loader coverage

B.1 Variations per gene, per data repository

Gene	HGMD	BIC	dbSNP	LOVD	**TOTAL**
BRCA1	1035	12 025	3091	0	**16 151**
BRCA2	776	11 331	2712	0	**14 819**
CDH23	86	0	4954	1119	**6159**
CLRN1	15	0	547	276	**838**
COL1A1	443	0	748	0	**1191**
COL1A2	288	0	787	0	**1075**
DFNB31	4	0	1289	132	**1425**
FBN1	848	0	2320	0	**3168**
GPR98	6	0	5409	41	**5456**
MYO7A	190	0	1205	2136	**3531**
NF1	1018	0	2558	0	**3576**
PCDH15	23	0	11 830	504	**12 357**
USH1C	16	0	647	223	**886**
USH1G	6	0	121	50	**177**
USH2A	119	0	7925	2212	**10 256**
TOTAL	**4873**	**23 356**	**46 143**	**6693**	**81 065**

B.2 Translated variations per gene and data repository

Gene	HGMD	BIC	dbSNP	LOVD	**TOTAL**
BRCA1	1035	11 507	1662	0	**14 204**
BRCA2	776	11 169	998	0	**12 943**
CDH23	86	0	4754	1108	**5948**
CLRN1	15	0	565	276	**856**
COL1A1	443	0	809	0	**1252**
COL1A2	288	0	843	0	**1131**
DFNB31	4	0	1222	132	**1358**
FBN1	848	0	2399	0	**3247**
GPR98	6	0	5254	41	**5301**
MYO7A	190	0	1094	2090	**3374**
NF1	1018	0	2499	0	**3517**
PCDH15	23	0	12 186	489	**12 698**
USH1C	16	0	661	200	**877**
USH1G	6	0	111	50	**167**
USH2A	119	0	7965	2209	**10 293**
TOTAL	**4873**	**22 676**	**43 022**	**6595**	**77 166**

B.3 Translated unique variations per gene and data repository

Gene	HGMD	BIC	dbSNP	LOVD	**TOTAL**
BRCA1	947	1446	1140	0	**3533**
BRCA2	733	1692	684	0	**3109**
CDH23	74	0	3273	241	**3588**
CLRN1	15	0	370	16	**401**
COL1A1	358	0	575	0	**933**
COL1A2	244	0	595	0	**839**
DFNB31	4	0	827	54	**885**
FBN1	721	0	1638	0	**2359**
GPR98	6	0	3570	13	**3589**
MYO7A	170	0	734	299	**1203**
NF1	792	0	1747	0	**2539**
PCDH15	23	0	8208	91	**8322**
USH1C	12	0	455	54	**521**
USH1G	6	0	74	7	**87**
USH2A	107	0	5445	471	**6023**
TOTAL	**4212**	**3138**	**29 335**	**1246**	**37 931**

B.4 Translated unique variations versus total number of variations per gene and data repository

Gene	HGMD	BIC	dbSNP	LOVD	**AVG**	**S.DEV.**
BRCA1	91.50 %	12.02 %	36.88 %		**46.80 %**	**33.19 %**
BRCA2	94.46 %	14.93 %	25.22 %		**44.87 %**	**35.31 %**
CDH23	86.05 %		66.07 %	21.54 %	**57.88 %**	**26.96 %**
CLRN1	100.00 %		67.64 %	5.80 %	**57.81 %**	**39.08 %**
COL1A1	80.81 %		76.87 %		**78.84 %**	**1.97 %**
COL1A2	84.72 %		75.60 %		**80.16 %**	**4.56 %**
DFNB31	100.00 %		64.16 %	40.91 %	**68.36 %**	**24.31 %**
FBN1	85.02 %		70.60 %		**77.81 %**	**7.21 %**
GPR98	100.00 %		66.00 %	31.71 %	**65.90 %**	**27.88 %**
MYO7A	89.47 %		60.91 %	14.00 %	**54.79 %**	**31.11 %**
NF1	77.80 %		68.30 %		**73.05 %**	**4.75 %**
PCDH15	100.00 %		69.38 %	18.06 %	**62.48 %**	**33.81 %**
USH1C	75.00 %		70.32 %	24.22 %	**56.51 %**	**22.92 %**
USH1G	100.00 %		61.16 %	14.00 %	**58.39 %**	**35.16 %**
USH2A	89.92 %		68.71 %	21.29 %	**59.97**	**28.69 %**
AVG	**90.32 %**	**13.48 %**	**31.32 %**	**21.28 %**	**47.07 %**	
S.DEV.	**8.36 %**	**1.45 %**	**13.47 %**	**9.77 %**	**10.61 %**	**12.31 %**

Appendix C

Algorithms

C.1 Calculating absolute position

```
public static int getAbsolutePos(List<Exon> exons, int cDNAPos) {
        int position = 0;
        int length = 0;
        for (Exon e : exons) {
                int exonLength = (e.end_exon - e.start_exon) + 1;

                if (exonLength + length < cDNAPos) {
                        length += exonLength;
                } else {
                        position = (e.start_exon + (cDNAPos -
                            length - 1));
                        return position;
                }
        }
        return -1;
}
```

C.2 Separating reference from variation

```
/**
* in some cases dbSNP "hides" multiple variations
* in one <RS> entry; this can be identified by looking
* at the ''observed'' field, which provides observed
* alleles separated by backslashes (/). Each observed
* allele that doesn't correspond to the nucleotide at
* that position in refSeq means a unique variation
* according to CSHG.
*
* @return
*       If no observed allele corresponds with refSeq,
*       0 is returned
*/
private List<DbSnpMutation> parseObservedAlleles() {
        List<DbSnpMutation> variations = new ArrayList<
            DbSnpMutation>();
```

```
        Pattern pattern = Pattern.compile("[ACTG]+?");
        List<String> observedAlleles = normalizeSnpObservedAlleles(
            observed, (String) attributes.get("orient"));
        for (String oa : observedAlleles) {
                Matcher matcher = pattern.matcher(oa);
                DbSnpMutation dsm = new DbSnpMutation(this);

                if (!observedAlleles.contains("-")) {
                        dsm.type = Precise.Type.INDEL;
                        if (matcher.matches() && !
                            observedAlleleIsInRefSeq(oa)) {
                                dsm.change = oa;
                                variations.add(dsm);
                        }
                } else if (observedAlleles.indexOf("-") == 0) {
                        dsm.type = Precise.Type.INSERTION;
                        if (matcher.matches() && !
                            observedAlleleIsInRefSeq(oa)) {
                                dsm.change = oa;
                                variations.add(dsm);
                        }
                } else if (observedAlleles.indexOf("-") ==
                    observedAlleles.size()) {
                        dsm.type = Precise.Type.DELETION;
                        if (matcher.matches() &&
                            observedAlleleIsInRefSeq(oa)) {
                                dsm.change = oa;
                                variations.add(dsm);
                        }
                }
        }
        return variations;
}
```

C.3 Normalizing variations

```
/**
* normalizes an observed genetic change,
* thus taking into account the orientation of
* the change (either "reverse" or "forward").
*
* @param observedBase
*            the observed change
* @param orientation
*            orientation of the change, must be a
*           string that equals either ''revers'' or '
*           'forward''
* @return a normalized SNP change
*/
public static String normalizeSNP(String observedBase, String
    orientation) {

        BasePairComplement bpc = new BasePairComplement();

        String returnvalue = "";
        if (orientation.equals("reverse")) {
                char[] letters = observedBase.toCharArray();
                for (int i = 0; i < letters.length; i++) {
                        char letter = letters[i];
                        letter = bpc.basePairs.get(letter);
                        returnvalue = returnvalue.concat("" +
                            letter);
                }
                returnvalue = new StringBuffer(returnvalue).reverse
                    ().toString();
        } else {
                returnvalue = observedBase;
        }
        return returnvalue;
}

/**
* this convenience class makes it
* easier to retrieve for each
* nucleotide its complementary
* equivalent.
*/
public class BasePairComplement {
        public Map<Character, Character> basePairs;

        public BasePairComplement() {
                basePairs = new HashMap<Character, Character>();
                basePairs.put('A', 'T');
                basePairs.put('T', 'A');
                basePairs.put('G', 'C');
                basePairs.put('C', 'G');
        }
}
```